Høgskulen på Vestlandet

Masteroppgave i læring og undervisning - MAS3-307

MAS3-307-O-2021-VÅR-FLOWassign

Predefinert informasjon

Startdato:	30-04-2021 09:00	**Termin:**	2021 VÅR
Sluttdato:	14-05-2021 14:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Masteroppgave	**Studiepoeng:**	45
Flowkode:	203 MAS3-307 1 O 2021 VÅR		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Ole Einar Rebni
Kandidatnr.:	234
HVL-id:	051211@hvl.no

Informasjon fra deltaker

Antall ord *:	21172

Egenerklæring *: Ja

Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *: Ja

Jeg godkjenner avtalen om publisering av masteroppgaven min *

Ja

Er masteroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Er masteroppgaven skrevet ved bedrift/virksomhet i næringsliv eller offentlig sektor? *

Nei

Høgskulen på Vestlandet

MASTEROPPGÅVE

Plugged og unplugged programmering i matematikk – eit kvasieksperiment

Plugged and unplugged programming in mathematics – a quasi-experiment

Ole Einar Rebni

Masteroppgåve i læring og undervisning
Fakultet for lærarutdanning, kultur og idrett/Institutt for pedagogikk, religion og samfunnsfag
Rettleiarar: Sigve Høgheim og Kristin Myhra Sæterdal
Innleveringsdato: 28.05.21

Eg stadfestar at arbeidet er sjølvstendig utarbeida, og at referansar/kjeldetilvisingar til alle kjelder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Samandrag

Føremålet med denne masteroppgåva var å undersøke korleis programmering analogt (unplugged), kontra digitalt (plugged) påverkar elevar si evne til algoritmisk tenking og problemløysing i matematikk. Prosjektet har eit kvasi-eksperimentelt design der utvalet er 17 elevar på niande trinn. Deltakarane deltok i ein intervensjon, der det vart gjennomført eit undervisningsopplegg over to veker. Før intervensjonen vart elevane delte i to randomiserte grupper, der den eine gruppa jobba med programmering med digitale verktøy (plugged-gruppa), medan den andre gruppa jobba med programmering utan digitale verktøy (unplugged-gruppa). I forkant og i etterkant av intervensjonen gjennomførte elevane ein pre- og post-test, samt ein pre- og post-survey.

Med utgangspunkt i forskingsspørsmålet: *Korleis påverkar programmering analogt (unplugged) kontra digitalt (plugged) elevane si evne til algoritmisk tenking og problemløysing i matematikk?,* vart kvantitative data innhenta utifrå ulike variablar som er sentrale innanfor teori kring algoritmisk tenking og problemløysing. Det vart også gjennomført observasjonar undervegs i intervensjonen som dannar eit grunnlag for drøftinga saman med talmateriale frå testane og survey.

Utvalet i dette prosjektet er lite, og det vart difor vanskeleg å konkludere endeleg med dei funna som kom fram gjennom prosjektet. Ein kan likevel sjå nokre tendensar som kan vere spennande å forske vidare på. Dei mest sentrale tendensane er at elevane som programmerte plugged, hadde større framgang kring elementa *kreativitet, samarbeid og logisk tenking.*

Abstract

The means with this master thesis was to investigate how programming unplugged versus plugged influences the pupil's ability of computational thinking and problem solving in the subject of mathematics. The project has a quasi-experimental design where the participants are 17 pupils in the 9th grade. The participants took part of an intervention, were the pupils conducted a lesson plan which lasted for two weeks. The pupils were divided into two randomized groups before the intervention started, where one of the groups worked on programming with digital tools (plugged) and the other group worked on programming without digital tools (unplugged). Before and after the intervention the students carried out a pre- and post-test, as well as a pre- post-survey.

Based on the research question: How does programming plugged versus unplugged influence the pupil's ability of computational thinking and problem solving in the subject of mathematics? The quantitative data was gathered from different variables which are essential regarding computational thinking and problem solving. It was also performed observations during the intervention which forms a foundation for the discussion including the data from the tests and the survey.

The number of participants in this project is low, and it is therefore hard to conclude with the findings which came out of this project. There is still possible to spot tendencies which can be exciting to explore further. The most interesting tendencies are that the pupils who programmed plugged, had more increase in the elements *creativity, collaboration and logic thinking.*

Føreord

To år med hardt arbeid er endeleg over. Prosessen med å skrive ei masteroppgåve er ei berg og dal-bane med kjensler. Eg har stanga hovudet i veggen, dansa av glede og vore i ein tilstand der ein kjenner seg heilt kjenslelaus. Det går ikkje ein einaste dag der ein ikkje tenkjer på kva ein skal skrive om eller kva ein skal gjere vidare i prosessen, det vert noko som tek over livet ditt. At eg skulle klare å produsere ei masteroppgåve er ikkje ei sjølvfølge, då struktur og planlegging ikkje er mine sterkaste sider. Men med gode hjelparar og ei indre lyst til å gjennomføre, har eg no kome til vegs ende.

Det er mange som skal takkast for at denne reisa no er over. Fyrst vil eg takke min gode kompanjong, Joey Tempest som har gitt meg mentalt-påfyll med sin flotte puddelrock som har strøyma gjennom øyra mine gjennom heile prosessen. Når musikken har vorte sett på pause, har dei to gode rettleiarane mine Kristin og Sigve forsøkt på beste måte å gjere meg til ein forskar, noko som ikkje har vore ei enkel oppgåve. Eg vil vidare takke den nydelelege arbeidsplassen min og dei gode kollegaene mine der for hjelp. Leiinga ved skulen har vore særs hjelpsame med å gi meg fleksibilitet i arbeidskvardagen, slik at eg har kunna fullføre denne oppgåva. Elles vil eg takke Bjarte og Frank for gode programmerings- og matematikk-diskusjonar.

Heilt til slutt vil eg takke mi flotte kone, Linda, som har vore støttande og sexy gjennom heile prosessen. Utan deg og resten av familien hadde nok oppgåva stoppa etter dette føreordet.

Ole Einar Rebni

Kaupanger, mai 2021

Innhald

1 Innleiing og forskingsspørsmål

Dette kapitelet vil ta for seg bakgrunn og grunngjeving for val av tema, formål og fagleg relevans. Forskingsspørsmålet for prosjektet vert presentert, samt definisjon av omgrep og avgrensingar som er føremålstenlege for dette prosjektet. Det vil også bli presentert ein kort skildring av oppgåva sin struktur.

1.1 Fagleg relevans

Frå august 2020 er det Fagfornyinga, det fornya Kunnskapsløftet (LK20) som gjeld i grunnskule og vidaregåande opplæring. Programmering som metode er vektlagt i fleire av fagplanane i LK20. I matematikkfaget skal programmering innførast på dei aller fleste trinn. Utdanningsdirektoratet (Udir) (2019a) trekkjer fram at algoritmisk tenking og programmering er ein del av det fornya Kunnskapsløftet, og at algoritmisk tenking er vektlagd fordi det er ein viktig problemløysingsstrategi. Vidare peikar Udir på at når elevane nyttar programmering til å utforske og løyse problem, kan det vere eit godt verktøy for å utvikle matematisk forståing (Udir, 2019a). Ved å knytte programmering opp mot matematikkfaget kan ein kombinere fagspesifikke ferdigheiter med ferdigheiter i programmering. Dette gir moglegheit for å knytte teoretiske matematiske omgrep opp mot ein praktisk kontekst der elevane får ei forståing for kunnskap i ein større samanheng (Senter for IKT i utdanningen, 2016).

Argumenter for programmering i skulen vert ofte knytt til nødvendige ferdigheiter for det 21. århundre, kompetanse for framtida i næringslivet og evne til å forstå korleis eit meir og meir digitalisert samfunn fungerer. I 2013 vart det sett ned eit utval (Ludvigsenutvalet) som skulle gjere greie for grunnopplæringa sine fag opp mot krava til kompetanse i eit framtidig samfunns- og arbeidsliv. Rapporten frå denne utgreiinga: «Elevenes læring i fremtidens skole – et kunnskapsgrunnlag», har vore sentral i utarbeidinga av LK20. Eitt kompetanseområde som vert vektlagt i rapporten er; kompetanse i å utforske og skape. Dette kompetanseområdet inkluderer kritisk tenking og problemløysing. Problemløysinga vert tett kopla saman med kreativitet og innovasjon, og særskild det å vere fantasifull i problemløysing både åleine og ikkje minst i samarbeid med andre. For at elevane i framtida skal kunne bidra til nytenking, innovasjon og omstilling i arbeidslivet, må skulen legge til rette for at utviklar evner til å utforske, sjå nye moglegheiter og utvikle nye løysingar. Rapporten peikar på øving i problemløysing som ein metode til å nå desse måla (NOU 2014:7).

Disessa (2018) peikar på at programmeringa som går føre seg i dagens skule er som ei famling i blinde. Den beste måten å nytte programmering på er å verte godt kjende med dei verktøya ein vel å nytte, og kople dei til matematiske tema på ein slik måte at det gir elevane ein ny dimensjon innanfor dette emnet. Det skal auke deira forståing på ein måte som ikkje den vanlege matematikkundervisninga kan gi dei (Disessa, 2018). Utifrå undersøkinga Monitor 2019, er det 15,4% av elevane som nyttar seg av datamaskin til programmering/koding. Slik som føringane til LK20 seier, skal dette talet vere 100% allereie frå august 2020 (Monitor, 2019). Disessa (2018) peikar også på at det er mange lærarar tykkjer at dei har for lite kompetanse for å innføre programmering i skulen (Disessa, 2018).

Caeli og Bundsgaard (2019) peikar på tilstanden til programmering og algoritmisk tenking i skulen, gjennom ei undersøking, der danske skuleleiarar vart spurde om den profesjonelle utviklinga av algoritmisk tenking i grunnskulen. Informantane peika på at kompetansen algoritmisk tenking er viktig for framtida, men også at lærarar slit med kva algoritmisk tenking i ein utdannings-kontekst er, og korleis dei skal undervise i temaet. Det kom også fram at det er mange som syner initiativ for å innføre algoritmisk tenking i undervisninga, men det er likevel mange lærarar som manglar kompetanse for å undervise i temaet. Det vart også peika på at programmering ikkje er ein kompetanse som elevane treng berre for å verte databehandlarar, men også ein kompetanse som ein treng for å utvikle seg som menneske i eit fritt og demokratisk samfunn (Caeli og Bundsgaard, 2019).

Undervisningsmetodar som drama, gruppeprosjekt, diskusjonar og utspørjing aukar elevane sin kunnskap innanfor problemløysing og algoritmisk tenking. Det er ikkje gitt at plugged-programmering (digitalt) gir betre kompetanse innanfor algoritmisk tenking enn unplugged-aktivitetar (analogt). Det er den varierte tilnærminga som gir resultatet, ikkje verktøyet som vert brukt (Kalelioğlu og Gülbahar, 2019). Øvingar med spelutvikling og matematiske tilnærmingar utviklar elevane si evne til reflektiv-tenking. Programmeringsprogram som Scratch aukar motivasjonen til elevane meir innanfor programmering, enn det gjer dei til betre problemløysarar og algoritmiske tenkarar. Lærar som gjennomfører kurs med Scratch, tykkjer oppgåvene er morosame (Erümit, 2019).

1.2 Bakgrunn for val av tema

Bakgrunn for val av tema har for min del fleire årsaker. Medan eg gjennomførte vidareutdanning i faget programmering for lærarar, gjorde eg meg fleire refleksjonar som har hatt innverknad på tankesettet mitt og på min praksisutøving i klasserommet. Gjennom kursa i dette emnet vart det

arbeidd mykje med ulike programmerings-språk og -verktøy. Emnet gav meg også innsikt og kompetanse på programmering med og utan datamaskin, altså plugged og unplugged. I dette emnet vart også omgrepet algoritmisk tenking presentert. Etter denne utdanningskombinasjonen fekk eg ei openbaring kring linken mellom programmering, algoritmisk tenking og problemløysing. Eg har arbeidd mykje med programmering i skulen og har sett korleis det kan engasjere elevar som elles ikkje syner mykje engasjement i matematikktimane i skulen. Eg har også gjennom programmeringa i klasserommet danna meg eit bilete av at i mange av programmeringsoppgåvene støyter elevane på problem som dei kan relatere seg til, og på den måten skapar desse problema engasjement hjå elevane til å løyse dei. Noko som etter min oppfatning og erfaring ikkje alltid er tilfelle med mange ordinere problemløysingsoppgåver.

Programmering og algoritmisk tenking vert no ein del av matematikkfaget, og det er forventa at elevane skal lære programmering for å løyse matematiske utfordringar og problem. På bakgrunn av dette er eg nysgjerrig på programmering digitalt kontra analogt og kva det gjer med elevane sine evner til å tenkje algoritmisk og til å løyse problem. Dette er interessant i både eit økonomisk perspektiv, og også i eit lærarperspektiv i høve digital kompetanse og kompetanse om programmering. I det økonomiske aspektet er det snakk om kva verktøy skulane treng for å lære elevane programmering og kva desse kostar. Om ein skal programmere analogt treng ein berre blyant og papir. I høve lærarperspektivet er det etter min oppfatning mange lærarar som tykkjer det er vanskeleg og kvir seg for å ta i bruk programmering digitalt, då kan analog programmering vere ein alternativ metode som for desse lærarane.

1.3 Forskingsspørsmål

Programmering i skulen er svært aktuelt og hyppig omtala i høve Fagfornyinga (LK20). Det er forventa at programmering skal nyttast for å løyse matematiske problem, blant anna med å nytte algoritmisk tenking. Programmeringa skal hjelpe elevane til å verte klare for det samfunnet som møter dei når utdanninga er fullført, dei skal ha kompetanse som trengs i det 21. århundre. Er det datamaskina som skal gi dei denne kompetansen, eller kan elevane danne seg kompetanse innanfor programmering, algoritmisk tenking og problemløysing utan datamaskin? Dette bakteppet førte til at eg valte dette forskingsspørsmålet i mitt masterprosjekt:

Korleis påverkar programmering analogt (unplugged) kontra digitalt (plugged) elevane si evne til algoritmisk tenking og problemløysing i matematikk?

For å undersøke forskingsspørsmålet vil eg ved å gjennomføre eit intensivt undervisningsopplegg, i to grupper, prøve å finne ut av korleis elevane aukar kompetansen sin innanfor algoritmisk tenking og problemløysing. Metoden min for å prøve svare på dette forskingsspørsmålet, er ein survey og ein pre- post-test som kartlegg elevane sin kompetanse innanfor dei to tilnærmingane; plugged og unplugged. Eg vil elles nytte eit kvasi-eksperimentelt design på intervensjonen. I dette prosjektet er det intervensjonen som vert det mest sentrale og vert også omtala mest. Dette fordi fokuset mitt var å lage ein god intervensjon med godt fagleg innhald kring dei tre hovudemna programmering, algoritmisk tenking og problemløysing.

1.4 Definisjon av omgrep og avgrensingar

Programmering, algoritmisk tenking og problemløysing er tre omgrep som i læreplanen og i skulesamanheng er linka tett saman. Det er derfor naturleg å belyse alle desse tre omgrepa i dette prosjektet. I litteratur og forsking om programmering, algoritmisk tenking og problemløysing finn ein mange ulike definisjonar. I denne delen vil eg presentere dei definisjonane eg vil nytte for desse tre omgrepa.

Innanfor programmering finns det mange ulike programmeringsspråk, desse språka er konstruerte språk som berre nyttast til å skrive program til datamaskiner for å kontrollere og styre dei. Døme på desse ulike språka er Python og Scratch. Desse språka nyttast for å bestemme kva datamaskina skal gjere og for å få dei til å kommunisere med kvarandre. I matematikkfaget og skulesamanheng er det ikkje viktig kva for eit av desse programmeringsspråka som vert nytta, det er koplinga mellom programmering og algoritmisk tenking som er viktig. Denne problemløysingsmetoden og tankegangen som ligg bak, gjer at elevane tenkjer meir systematisk for å løyse matematiske problem. Programmeringsprosessen vert kjenneteikna som ei utforming og implementering av instruksjonar for dataprogram, også kjent som koding. Denne prosessen gjer det mogleg for datamaskiner å utføre spesifikke oppgåver, løyse problem og støtte menneskeleg interaksjon (Forsström & Kaufmann, 2018, s. 19). Ein må også finne ut kva ein skal gjere og korleis ein kan gjere det på ein presis måte (Lynnebakken, 2018). På ein forenkla måte kan ein seie at når ein programmerar, set ein kodar saman til ei oppskrift som datamaskina utfører akkurat slik den er skriven, utan form for tolking. Difor må oppskrifta vere presis, slik at datamaskina gjer det du vil den skal gjere.

I dette prosjektet vert omgrepet problemløysing definert som om at eit problem vert sett på som ei matematisk oppgåve som skal utførast, og i den innleiande fasen skal det vere uklart for problemløysaren kva løysingsmetodar som skal nyttast. Det er altså snakk om ein individrelatert definisjon. Det inneber at ei oppgåve som er problem for ein person, treng ikkje vere det for ein annan (Grevholm, 2003, s. 54).

Algoritmisk tenking vert definert som tankeprosessen knytt til å formulere eit problem og løysingane til problemet på ein slik måte at det kan løysast av eit menneske, ei maskin eller i ein kombinasjon av dei to (Wing, 2008) Algoritmisk tenking er den norske omsettinga av omgrepet computational thinking.

1.5 Oppgåva sin struktur

I neste del av oppgåva vil det teoretiske grunnlaget for prosjektet bli presentert. Vidare vil metoden for prosjektet lagt fram, samt argumentasjon for val av metodisk tilnærming, forskingsdesign og prosedyre for innsamling av data. Deretter vil det i resultatdelen bli presentert data som er samla inn under intervensjonen. Vidare i drøftingsdelen, vil dei aktuelle resultata bli drøfta i lys av teori og det vil bli gjort greie for aktuelle sider ved forskingsspørsmålet. Til slutt i oppgåva vil svakheiter og styrkar ved prosjektet bli satt i fokus.

2 Teoretisk forankring

I denne delen vert det teoretiske rammeverket for oppgåva lagt fram. Først vert det fokusert på programmering, algoritmisk tenking og problemløysing sin posisjon i skulen før og no. Det vert trekt fram læreplanar, NOU-ar, stortingsmeldingar og andre dokument som er relevante for dei tre hovudemna. Dette for å syne relevansen for desse emna i dagens skule og kvifor det er aktuelt å forske på dette temaet. Deretter vert det presentert teori kring dei tre hovudemna; programmering, algoritmisk tenking og problemløysing. Det er naturleg å inkludere teori om programmering, då intervensjonen i stor grad dreia seg om at programmering vert nytta som metode for algoritmisk tenking og problemløysing.

2.1 Programmering, algoritmisk tenking og problemløysing i skulen

I forbindelse med fagfornyinga (LK20) får matematikkfaget ei viktig rolle i opplæringa i programmering. Programmering, algoritmisk tenking og problemløysing er også tett kopla saman i tidlegare læreplanar, og det å kople programmering til matematikkfaget er heller ingen ny idé. Allereie i mønsterplanen for grunnskule (M87) vart datalærarar knytt til matematikkfaget, og at ein skulle ta utgangspunkt i algoritmeomgrepet. Det vart også peika på at algoritmeomgrepet skulle koplast opp mot problemløysing. Mønsterplanen nemnde at arealberekningar, løysing av likningar, tilnærmingsmetodar og simuleringar passa godt for datamaskina. Sjølv om det var gode intensjonar med å implementere programmering i matematikken gjennom mønsterplanen, skjedde ikkje dette. I L97 og K06 vart programmering lite vektlagt i matematikkfaget (Bueie, 2019, s. 23-28).

Programmering er komen inn i LK20 i fleire fag som musikk, naturfag, kunst og handverk og matematikk. I matematikkfaget finn ein programmering først i læreplanen under den grunnleggjande ferdigheita digitale ferdigheiter. Der heiter det at digitale ferdigheiter i matematikk inneber å kunne bruke programmering til å utforske og løyse matematiske problem. Det heiter vidare at utviklinga av digitale ferdigheiter inneber i aukande grad å bruke å velje formålstenlege digitale verktøy som hjelpemiddel for å utforske, løyse og presentere matematiske problem (Utdanningsdirektoratet, 2020). Det er kjernen av dette eg vil nytte forskinga mi til å finne ut av. Korleis kan ein nytte programmering som eit verktøy i det å løyse problem og å få elevane til å tenkje algoritmisk. I kompetansemåla i læreplanen for matematikk er det berre eit kompetansemål som går direkte på programmering for det aktuelle forskingsprosjektet som vert gjennomført på 9. trinn: «Målet for opplæringa er at eleven skal kunne simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering» (Utdanningsdirektoratet, 2020). Tanken bak det å innføre programmering i matematikklæreplanen verkar å vere at elevane skal verte kjende med programmering, slik at dei har dei har det som eit verktøy i verktøykassa si når dei skal løyse problem eller framstille data eller ei simulering. Det verkar som om at elevane skal nytte programmering dersom det er det beste verktøyet i den gitte situasjonen eller i det gitte problemet.

Algoritmisk tenking er ikkje nemnt eksplisitt i læreplanen for matematikk. Grunnen til dette kan vere korleis Udir har definert dette omgrepet. I artikkelen på Udir side nettsider om algoritmisk tenking, er algoritmisk tenking definert som ein problemløysingsmetode (Udir, 2019b). Om ein då har denne definisjonen til grunn når ein les læreplanen, vert omgrepet algoritmisk tenking innbakt i problemløysings-omgrepet som ein metode for å løyse problem. På denne måten er algoritmisk tenking indirekte nemnt i læreplanen som ein arbeidsmetode under omgrepet problemløysing.

Vidare har Udir utvikla kompetansepakkar der lærarane kan arbeide seg gjennom modular for å lære seg å tenkje algoritmisk, slik at dei kan utvide sin eigen kompetanse før ein lærer elevane denne metoden.

I 2013 vart det sett ned eit utval som skulle utreie grunnopplæringa sine fag opp mot krava til kompetanse i eit framtidig samfunns- og arbeidsliv. Kva treng elevane av kompetansar for å takle eit samfunns- og yrkesliv i framtida? Dette utvalet skreiv ein rapport som fekk tittelen: «Elevenes læring i fremtidens skole – et kunnskapsgrunnlag». Denne rapporten har vore særs deltakande i utarbeidinga av den nye læreplanen me har i skulen i dag. I denne rapporten kjem blant anna fram at problemløysing er ein viktig kompetanse elevane treng i framtida (NOU 2014:7). Utvalet kjem gjennom rapporten fram til fire kompetanseområde som grunnlag for fornying av skulen sitt innhald. Av desse fire er det eit kompetanseområde som er relevant i denne samanheng; kompetanse i å utforske og skape. Dette kompetanseområdet inkluderer kritisk tenking og problemløysing. Problemløysinga vert tett kopla saman med kreativitet og innovasjon, og særskild det å vere fantasifull i problemløysing både åleine og ikkje minst i samarbeid med andre. For at elevane i framtida skal kunne bidra til nytenking, innovasjon og omstilling i arbeidslivet, må skulen legge til rette for at utviklar evner til å utforske, sjå nye moglegheiter og utvikle nye løysingar. Og det er akkurat her rapporten peikar på øving i problemløysing som ein metode til å nå desse måla (NOU 2014:7). Så allereie i 2013 vart problemløysing sett på som ein kompetanse elevane treng for å vere kvalifiserte deltakarar i samfunns- og arbeidsliv når dei har fullført grunnskulen.

Som tidlegare nemnt var denne NOU-rapporten noko som låg sterkt til grunn når den nye læreplanen vart utarbeidd og teken i bruk i 2020. Grunnen til at dei tre hovudtema er interessant å forske på no, er plassen dei tre hovudemna får i den nye læreplanen. I læreplanen for matematikk står det i fagets relevans og verdiar at matematikk skal bidra til at elevane utviklar eit presist språk for resonnering, kritisk tenking og kommunikasjon gjennom abstraksjon og generalisering. Vidare skal elevane utvikle kompetanse i utforsking og problemløysing. Dei skal utvikle kreativitet og skapartrong. Dei skal jobbe sjølvstendig og samarbeide med andre gjennom utforsking og problemløysing. Dei skal utvikle kreativitet og skapartrong. Fagets relevans og sentrale verdiar vert også avrunda med ein påstand om at når elevane får høve til å løyse problem og meistre utfordringar på eiga hand, bidreg det til å utvikle uthald og sjølvstende (Utdanningsdirektoratet, 2020). Her ser ein som i NOU-rapporten at utforsking og problemløysing vert tett linka saman, akkurat slik som tidlegare er poengtert med læreplanen.

2.2 Programmering

> «Programming a computer means nothing more or less than communicating to it in a language that it and the human user can both "understand". And learning language is one of the things children do best. Every children learns to talk. Why then should a child not learn to "talk" to a computer?" (Papert, 1993, s. 5-6).

Slik presenterer Papert (1993) viktigheita med det at elevar lærer seg programmering. Innlæringa vert samanlikna med det å snakke og korleis elevane også bør lære seg å kommunisere med datamaskina på den sitt språk på lik linje som dei kommuniserer med menneske rundt seg. I nyare forsking har omgrepet om å snakke med datamaskiner fått eit namn; c*omputational literacy.* I forskinga til Disessa (2018) kjem det fram at den beste måten å nytte c*omputational literacy* er å verte kjent med dei dataverktøya som ein vel å bruke, og kople dei til matematiske tema på ein slik måte at det gir elevane ein ny dimensjon innanfor det gitte emnet. Slik at det aukar deira forståing, og dei får ei forståing som den vanlege matematikkundervisninga ikkje kan gi dei (Disessa, 2018, s.22).

Det ligg uendelege moglegheiter i datamaskinene og elevane er dei som er best egna til å få oppfylt desse moglegheitene. Mange born har ein modell for læring, som går ut på at enten får ein det til, elles får ein det ikkje til. Når ein programmerar ei datamaskin, får ein det nesten aldri til på fyrste forsøk. Fokuset vert då ikkje på om ein får det til eller ikkje får det til, men ved å isolere og rette opp i «bugs» får ein eit fokus som ikkje går ut på om programmet er rett eller feil, men på om det er mogleg å fikse. Arbeid med «debugging» kan endre kulturen for læring, slik at ein ikkje engstar seg for mykje for å gjere feil (Papert, 1993, s. 23). Det er mange situasjonar innanfor utdanningsløpet der det er formålstenleg for elevane å tenkje som ei datamaskin. Innanfor emne som grammatikk og matematikk kan det ha positiv effekt å nytte denne stilen ved at elevane tenkjer steg for steg på ein mekanisk måte som ei datamaskin. Denne måten å tenkje på vert kalla *mekanisk tenking*. Ein annan fordel ved å tenkje mekanisk er at elevane vert bevisste på at det er noko som heiter forskjellige måtar å tenkje på, tenkjestilar. Ved å gjere elevane medvitne på dette, vert dei meir klare på korleis dei tenkjer, kva tenkjestil dei vel å nytte i ulike situasjonar og korleis dei kan veksle mellom desse tenkjestilane ut i frå kva problem dei skal løyse. Dette gjer at elevane lærer seg å tenkje om korleis dei tenkjer (Papert, 1993, s. 27). Denne mekaniske tenkinga kan koplast til omgrepet algoritmisk tenking som er ein problemløysingsmetode der ein skal tenkje systematisk som ein informatikar (Udir, 2019b). Systematisk og mekanisk er to omgrep som ligg nærme kvarandre med tanke på meining. Så koplinga mellom Papert (1993) sin mekaniske tenking og omgrepet algoritmisk tenking

som vert nærare utgreia i delen om algoritmisk tenking, er ei kopling som koplar programmering og algoritmisk tenking saman.

Programmering er ikkje berre positivt med tanke på korleis elevane kan lære om sine eigne tenkjestilar, det kan også vere eit godt verktøy for elevar. Turtle Geometry, som er eit teikneprogram der ein programmerer ei skjelpadde, der av namnet Turtle, til å teikne etter instruksjonar. I gjennomgangen av dei pedagogiske vinstane i bruken av Turtle i skulesamanheng, koplar Papert inn Pólya og has heuristikkar. Pólya vert nærare presentert og introdusert i delen om problemløysing. Når ein nærmar seg eit problem, skal ein gå gjennom ei mental sjekkliste bestående av heuristiske spørsmål som: Kan dette problemet delast inn i fleire enklare problem? Kan dette problemet relaterast til eit problem som ein har løyst tidlegare? Turtle er midt i blinken for å øve seg på slike spørsmål. Det manglar gode situasjonar i skulen der enkle heuristiske modellar kan innlærast. Men gjennom programmering og Turtle vert abstrakte utsegn frå Pólya som: For å løyse eit problem, sjå etter noko liknande som du allereie har forstått, konkretiserte. Dette kan ein gjer ved å spele Turtle for så å gjere det sjølv. På engelsk vert det kalla *Play Turtle. Do it yourself.* Då programmerer ein og teiknar først med Turtle, for så å teikne med sin eigen kropp. Programmering og særskild Turtle Geometry vert som ei bru til Pólya. Elevar som har jobba lenge med Turtle vert overtydde over viktigheita i å «sjå etter noko som liknar eit tidlegare problem» fordi dette rådet har gitt dei resultat ved tidlegare høve. Elevar som jobbar med programmering og Turtle lærer seg korleis dei kan nytte dette prinsippet i røynda, noko som Papert meiner er ei elles vanskeleg øving innanfor skulematematikken. Han meiner vidare at skulematematikk og det aritmetiske innhaldet vert ganske avansert for elevane å øve på Pólya sine prinsipp. *(Papert, 1993, s. 64).* Ein ser i teorien at programmering vert linka som ein god metode for elevar å lære seg Pólya sine fire fasar innafor problemløysing, som er ein teori som seinare vert presentert i teoridelen.

Ulike programmeringsspråk som gir deg kontroll over ei datamaskin, kan ha effekt på korleis me skildrar oss sjølv og læringa vår i framtida. Eit eksempel på dette er ved å skildre korleis programmerings-konsept som rammeverk kan hjelpe ein person å lære seg ein fysisk kompetanse, sjonglering. I dette dømet vert det lagt vekt på korleis programmeringa hjelp til med å styrke språket. Først laga Papert (1993) ein modell som gjorde sjonglering til ein menneskeleg prosedyre, som så vart omgjort til ein læringsstrategi for korleis å lære seg sjonglering. Denne læringsstrategien bestod av fem enkle steg. Ved å gå gjennom desse stega kronologisk, ser ein fort kvar det er *bug* i programmet/strategien. På denne måten kan ein rette opp i *buggen* som i sjonglerings-kompetanse ofte er kvar ein held blikket og korleis kasteteknikken er. Om ein skal lære seg sjonglering utan denne

tilnærmingsmetoden, hopar *buggane* seg opp og det kjem mange *buggar* samtidig. Då kan det vere veldig vanskeleg å feilsøke og innlæringa av sjonglering vert mykje meir tidkrevjande. Slike programmeringsstrategiar gjer ikkje til at ein kan lære seg alle fysiske kompetansar i ein rasande fart, men ved å nytte dei kan ein effektivisere innlæringa og spare seg masse tid og frustrasjon (Papert, 1993, s. 95-115).

Vidare peikar Papert (1993) på at elevar som har drive ein del med programmering vert flinkare til ein del ting enn dei som ikkje har vore borti programmering. I programmering lærer elevane seg at dei ikkje vert vurderte frå ein standard som; riktig svar – du får ein god karakter eller - feil svar, du får ein dårleg karakter. Ved å jobbe med programmering, vert ein kjend med *feil* og *feilsøking.* Ein forventar ikkje at noko skal fungere på første forsøk. Elevane vert meir fokuserte på spørsmål som typen – korleis kan eg fikse det? Ein skaffar seg uthald og ein vert meir løysingsorientert (Papert, 1993, s. 101).

Sentralt for dette prosjektet å ta med seg vidare frå denne delen, er at programmering kan få elevane til å verte meir medvitne på at det ikkje er rett eller feil svar på alt, men korleis ein skal fikse det aktuelle programmet eller problemet. Det er også viktig å ta med seg vidare at programmering kan gjere elevane meir medvitne på korleis dei tenkjer, og korleis dei kan veksle mellom desse tenkjestilane for å løyse ulike problem. Linken mellom programmering og problemløysing vert også belyst i denne delen, og då særleg linken mellom programmering og prinsippa til Pólya innanfor problemløysing og korleis desse prinsippa vert enklare å jobbe med gjennom programmering enn ved vanleg skulematematikk.

2.3 Algoritmisk tenking/ computational thinking

Algoritmisk tenking er eit relativt nytt omgrep i den norske skulen, men det å kombinere datamaskiner og den menneskelege kunnskap i skulesamanheng er ikkje like nytt. Seymour Papert (1993) var blant dei første til å tenke på å bruke datamaskiner innan utdanning. Han ville sjå på korleis datamaskinene har innverknad på korleis folk tenkjer og lærer. Han ville særskild sjå på korleis datamaskinene sitt inntog i samfunnet bidrog til ein mental prosess i tillegg til den instrumentale, korleis folk tenkjer også når dei ikkje har ei datamaskin tilgjengeleg. Språk og matematikk er veldig samanliknbart, elevar må lære seg å snakke matematikk nett som ein snakkar norsk, fransk eller spansk. Skuletilnærminga til å lære seg språk fungerer ikkje, men om ein elev flyttar til Frankrike og bur der i nokre år vert eleven god i fransk. Språk er noko av det ungar er best på, så kvifor skal ikkje

ungane lære seg språket til ei datamaskin? Språket er tankesettet til datamaskina. Det er mogleg å designe datamaskiner som gjer at prosessen med å lære språket til datamaskinene blir som å lære fransk ved å bu i Frankrike, og ikkje slik som språkinnlæring elles vert gjort i skulesamanheng. Papert (1993) er vidare fokusert på at det å lære seg å kommunisere med ei datamaskin kanskje endrar korleis all anna læring skjer. Han grunngjev dette med at datamaskina snakkar to språk; matematikkspråk og alfabetisk-språk. Me lagar datamaskiner som elevane elskar å kommunisere med. Når denne kommunikasjonen skjer, lærer elevane matematikk som eit levande språk. Han generaliserer så det å snakke matematikkspråk til ei datamaskin som å lære matematikk i Matteland på same måte som ein ville lært seg fransk ved å bu i Frankrike (Papert, 1993, s. 6).

Papert (1993) peika også på at sjølv om teknologi og datamaskiner vil spele ei viktig rolle når ein ser på korleis framtidig utdanning vil gå føre seg, er det ikkje det hovudfokuset, men heller på sinnet/hjernen til individet. Datamaskina har rolla som ein berar av kulturelle bakteriar eller frø som skal vekse i sinnet og hjernen til elevane, som sidan ikkje treng støtte frå teknologi når dei slår rot og byrjar å vekse i ein aktiv og voksande hjerne (Papert, 1993, s. 9).

I følgje Wing (2006) er algoritmisk tenking ein måte som menneske tenkjer på, ikkje ein måte som datamaskiner tenkjer på. Computational thinking/algoritmisk tenking er ein problemløysingsmetode; det er ikkje eit forsøk på å få menneske til å tenkje som datamaskiner. Datamaskiner er keisame; menneske er kloke og fantasifulle, det er derfor oss menneske som gjer datamaskinene spanande. Utstyrte med teknologiske instrument, nyttar me smartheita vår til å takle problem me aldri ville ha teke fatt på før datamaskinene sitt inntog i samfunnet. Me bygger no system med funksjonalitet som berre er avgrensa av vår eigen kreativitet. Algoritmisk tenking er ein fundamental ferdigheit som alle barn bør ha, sidestilt med lesing, skriving og aritmetikk (Wing, 2006).

Som Wing poengterte for 15 år sidan, er samanhengen mellom den menneskelege fantasien og teknologien blitt ein kombinasjon som kan hjelpe menneske med å løyse problem som tidlegare ikkje var løyselege. Denne kombinasjonen mellom kreativitet og bruken av teknologiske verktøy som problemløysingsmetode, er noko ho ser som særs nyttig for elevar i framtida (Wing, 2006).

Figur 1 Figur inspirert av The Computational thinker: Attitudes and Skills, henta frå: https://thinkspace.csu.edu.au/learning2learn/2016/06/01/computational-thinking/

Figuren ovanfor syner handlingar og kompetansar som er viktige når ein skal nytte algoritmisk tenking når ein løyser problem. Det *å gjere feil* er ein del av problemløysinga, men ein må ha ein strategi for å lære av feila sine slik at ein ikkje gjer dei same feile fleire gongar utan å endre tilnærmingsmetode. *Uthald* er ein viktig del av det å tenkje algoritmisk. Den lærande må bygge seg opp eit kognitivt uthald som gjer at ein held fram med problemet sjølv om det vert forvirrande og løysinga ikkje vert klar med ein gong. *Kreativitet* er også ein viktig del av det å tenkje algoritmisk. Ein må sjå på ting frå forskjellige perspektiv, ein må ikkje avskriv e det umoglege og ein må ha evna til å legge frå seg eit problem ei stund og la det godgjere seg, for så å ta det fram at ved eit seinare høve og kanskje då er det kome til nye element som gjer at det let seg løyse. For å nytte algoritmisk tenking som ein metode for å løyse problem, er *samarbeid* viktig. Ein må både nytte andre sine idear og kunne dele sine eigne idear med andre. Ein må gjennom å snakke om problemet med medelevar kunne lære av andre elevar og dele eigen lærdom med dei. *Å kjenne att mønster* er ein av kompetansane som er viktige for å tenkje algoritmisk. I det utrykket ligg det å forstå korleis delane av problemet heng saman. Ein må sjå på problemet om det er noko likt ved dette problemet samanlikna med eit problem som er løyst tidlegare, og om ein kan nytte noko av den kjennskapen til det tidlegare problemet for å løyse det nye problemet. Ein annan kompetanse som er essensiell er *dekomponering.* Det går på å bryte ned problemet i mindre delar, slik at det lettare let seg løyse. Det går også på å snakke matematisk ved å kunne forklare dei ulike delane av problemet og også løysinga

når den er klar. *Å tenkje i algoritmar* går ut på å kartlegge kva steg som må gjerast for å løyse problemet. Ein må lage seg ei stegvis oppskrift som gjer at ein kan løyse eit problem. *Abstraksjon og generalisering* går ut på å plukke ut den informasjonen som er viktig for å løyse problemet og å fjerne den informasjonen som er overflødig. Ein må også tenkje på kor overførbar løysingsstrategien er ovanfor andre problem. Er det mogleg å nytte same strategi for å løyse andre problem, og når er det mogleg? (Wing, 2006)

Utdanningsdirektoratet (2019) skriv at algoritmisk tenking er ein problemløysingsmetode der problemløysaren må tenke systematisk som ein informatikar. Det er også naudsynt at problemløysaren har oversikt over kva delar av problemet som kan løysast av menneske og kva som kan løysast av datamaskiner eller andre digitale verktøy. Å tenkje algoritmisk er å vurdere kva steg som trengs for å løyse eit problem, og å kunne nytte sin teknologiske kompetanse for å få ei datamaskin til å løyse problemet eller deler av det. I omgrepet ligg også ei forståing av kva type problem eller oppgåver som kan løysast med teknologi og kva som bør løysast av menneske (Udir, 2019).

Algoritmisk tenking er ein problemløysingsmetode der ein jobbar med å bryte ned komplekse problem til mindre problem som vert lettare å forhalda seg til og å løyse, *dekomposisjon*. Det går vidare ut på å organisere og analysere informasjon på ein logisk måte, *logikk.* Eit anna viktig element er å lage framgangsmåtar som algoritmar for å kome fram til løysingar, *algoritmar.* Å lage *abstraksjonar* og modellar av den verkelege verda ved å fjerne unødvendige detaljar og fokusere på det som har relevans for den aktuelle problemstillinga og løysinga, er også ein viktig del av det å tenkje algoritmisk. Ei løysing for eitt spesifikt problem kan ofte generaliserast, slik at den kan nyttast til å løyse andre og liknande problem. Løysingar på fleire delproblem kan vidare kombinerast for å løyse større og meir komplekse problem. Når ein tenkjer algoritmisk må ein vere systematisk og analytisk, ein må vere *skapande* og eksperimenterande for å halde moglegheitene opne for fleire alternative løysingar på ulike problem. Tilnærminga til problema bør vere nysgjerrig og utforskande for å formulere og løyse problem. Det å gjere feil undervegs er ein viktig del av prosessen, og den som tenkjer algoritmisk må ha strategiar for å oppdage at noko er feil og rette dei feila. Det krev ein kognitiv kondis for å ikkje gi opp, og denne kondisen må trenast ved å jobbe med kontinuerleg forbetring undervegs i prosessen, *holde ut. Samarbeid* og deling er sentrale arbeidsmetodar for den algoritmiske tenkjaren (Udir, 2019b)

2.4 Problemløysing

Det er mange ulike definisjonar på problemløysing i ein matematisk skulesamanheng. I denne oppgåva vil eg ta utgangspunkt i at eit problem vert sett på som ei matematisk oppgåve som skal utførast, og i den innleiande fasen skal det vere uklart for problemløysaren kva løysingsmetodar som skal nyttast. Det er altså snakk om ein individrelatert definisjon. Det inneber at ei oppgåve som er problem for ein person, treng ikkje vere det for ein annan (Grevholm, 2003, s. 54)

George Pólya (2014) var ein av dei første som såg på problemløysing i eit pedagogisk perspektiv. Han laga ein problemløysingsteori som bestod av fire fasar; *forstå problemet, utforme ein plan, utføre planen* og *sjå seg tilbake*. Denne problemløysingsteorien vart utgiven for første gong i 1945. For å løyse eit problem må ein *forstå kva problemet er*. Kva er det ukjende? Kva data finns? Kva er føresetnadane? Er føresetnadane gode nok til å definere det ukjende? I denne fasen kjem Pólya med tips om eleven kan teikne ein figur til hjelp, og om ein kan dele opp dei ulike delane av føresetnadane og skrive dei ned? (Pólya & Conway, 2014, s. xvi).

Den neste fasen i problemløysingsteorien er å *utarbeide ein plan.* I denne fasen stiller problemløysaren seg sjølv ei rekkje spørsmål; Har du sett problemet før? Har du sett same problemet i ei litt anna form? Kjenner du til eit liknande problem? Kjenner du til eit teorem/ein læresetning som kan vere til nytte? Eleven skal tenkje om det ukjende er noko han/ho kjenner att i ei liknande form frå eit anna problem. Om ein har løyst eit liknande problem tidlegare, kan ein nytte den informasjonen til å løyse dette problemet? Kan ein omformulere problemet? Dersom ein ikkje kan løyse det aktuelle problemet, bør ein prøve å løyse eit liknande problem. Finns det eit problem som er enklare eller meir generelt men likevel liknande som er lettare å løyse? Kan ein endre nokre element som t.d. data eller det ukjende for så å løyse problemet og relatere det til det aktuelle problemet? Sjekk om ein har nytta alle data, alle føresetnadane og all anna informasjon som er ein del av problemet (Pólya & Conway, 2014, s. xvi-xvii).

Er du heilt sikker på at kvart einskild steg er korrekt? Kan du bevise at det er korrekt? Det siste steget i problemløysingsteorien er å *sjå seg tilbake.* I dette steget er også framgangsmåten for problemløysaren å stille seg sjølv spørsmål: Kan du sjekke resultatet? Kan du sjekke argumenta dine? Kan du derivere resultatet på ein annan måte? Tenk igjennom om du kan nytte resultatet eller metoden til andre problem? (Pólya & Conway, 2014, s. xvii)

Forholdet mellom lærar og elev er viktig innanfor problemløysing. Korleis læraren opptrer i klasserommet, har mykje å seie for å skape eit godt læringsmiljø som gjer elevar til betre problemløysarar. Elevane skal lærast opp til å arbeide mest mogleg sjølvstendig. Ein skal likevel ikkje la eleven vert heilt overlaten til seg sjølv med problemet, dette er ein balansegong. Om læraren hjelper for mykje vert det ikkje att noko av problemet til eleven sjølv, medan om ein hjelp for lite vert eleven ofte ståande fast i problemløysingsprosessen. Det er viktig at hjelpa som læraren gir i denne prosessen er diskret, slik at eleven får ei kjensle av at han jobbar sjølvstendig sjølv om han vert hjelpt. Læraren må setje seg inn i elevane sin situasjon, gå inn i tankesettet til eleven slik at spørsmåla ein stiller som lærar reflekterer situasjonar eller steg som kunne ha skjedd med eleven sjølv (Pólya & Conway, 2014, s. 1)

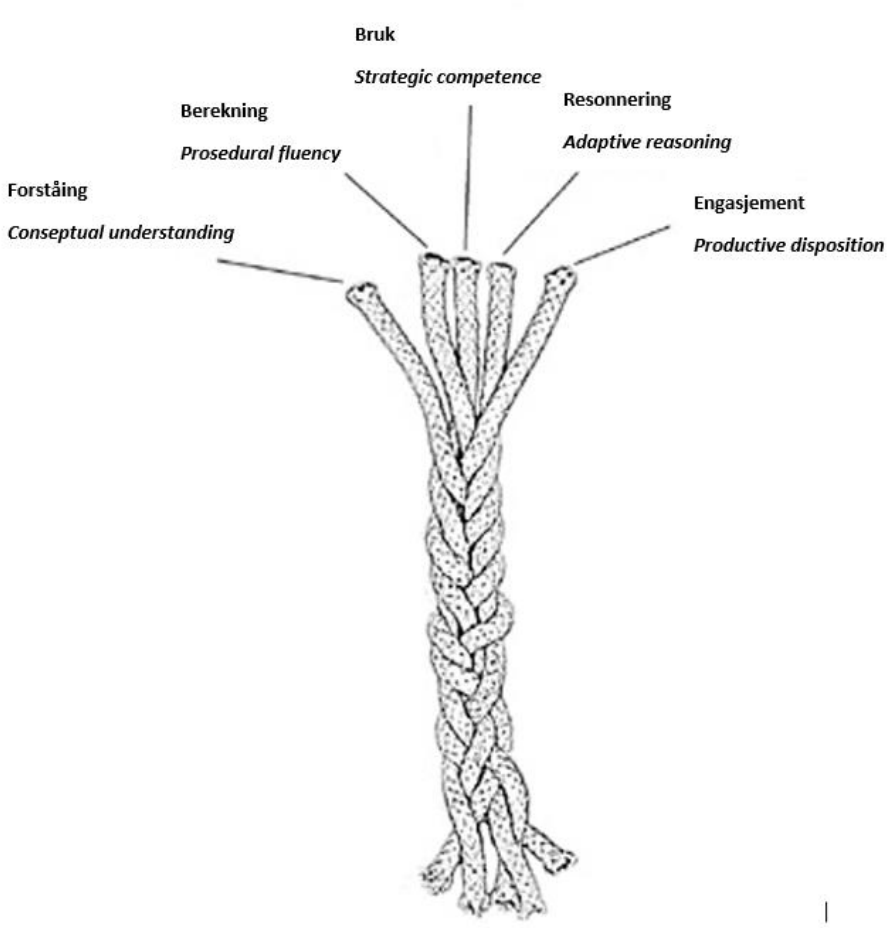
Det er også essensielt korleis ein lærar kjem med spørsmål og forslag til elevane. Om ein vil at eleven skal fokusere på eller finne ut på kva som er det ukjende i eit problem, må ein vere lur og tenkje ulike tilnærmingar når ein kjem med spørsmål og forslag til eleven. Still spørsmålet på mange ulike måtar og kom med mange ulike forslag, men målet må alltid vere: Kva er det ukjende? Forslaga og spørsmåla som læraren kjem med til elevane tek utgangspunkt i sunn fornuft. Forslag som; sjå på det ukjende og tenk på eit problem som har liknande ukjend, er eit forslag som han meiner elevane hadde gjort for seg sjølv uansett, fordi det er sunn fornuft. Lærar og elev, imitasjon og praksis, går ut på at læraren skal ha to mål når han kjem med spørsmål og forslag til eleven i problemløysingsprosessen: 1. Å hjelpe eleven å løyse det aktuelle problemet. 2. Å utvikle eleven si evne til å løyse framtidige problem sjølv (Pólya & Conway, 2014, s. 1-4).

Problemløysing er ein praktisk ferdigheit på lik linje med det å symje. Når ein skal lære seg å symje, ser ein på ein som kan symje og ein imiterer det den røynde symjaren gjer, slik at ein til slutt lærer seg å symje sjølv. Innan problemløysing er prosessen den same, ein lærar som underviser innan problemløysing må syne elevane sine korleis han løyser problem, og då særleg med vekt på å stille dei spørsmåla og forslaga som nemnt ovanfor nok gongar, slik at elevane til slutt nyttar seg av spørsmåla og forslaga sjølve når dei skal løyse problem på eiga hand (Pólya & Conway, 2014, s. 4-5).

I følgje Kilpatrick (2001) er det fem trådar som utgjer matematisk kompetanse. Desse kompetansane utgjer i lag eit *tau* som er fletta eller tvinna saman. Dette krev samarbeid, tid og ein tråd i *tauet* forutsett utvikling av ein annan tråd. Desse elementa heng saman. Dei fem trådane er;

1. Forståing

2. Berekning
3. Bruk
4. Resonnering
5. Engasjement



Figur 2 Dei fem "trådane", modell inspirert frå (Kilpatrick, 2001)

Kilpatrick (2001) peikar på at f*orståing*, eller omgrepsforståing handlar om evne til å forstå einskildomgrep, operasjonar om samanheng mellom omgrep i matematikken. Elevane har mest kontroll på isolerte fakta, dei må lærast til å sjå heilskapen og å verbalisere denne forståinga. Elevane treng hjelp til å sjå sine eigen feil. *Berekning*, eller prosedyrekunnskap handlar om ferdigheiter i å gjennomføre prosedyrar hensiktsmessig, effektivt, nøyaktig og fleksibelt. Dette krev at elevane må kunne bruke algoritmar. Dei må kunne bruke skriftlege berekningar, slik at dette hjelp dei til å skape ei forståing. *Bruk,* eller strategisk kompetanse handlar om evne til å representere, formulere, løyse og vurdere løysinga av matematiske problem. For at elevane skal klare dette bør dei kjenne til varierte løysingsstrategiar og ulike presentasjonsformer. *Resonnering*, eller fleksibel tenking handlar om evne til å tenkje logisk, reflektere, forklare og vurdere sanningsverdien av ein påstand, eit

argument eller eit resultat. For å få til dette må elevane bygge seg opp ein kunnskapsbase. Dei må også øvast i å grunngje og å tenkje logisk. Læraren må då lage oppgåver dei forstår og som motiverer. Konteksten elevane jobbar i, må vere kjend og trygg. *Engasjement*, eller produktiv haldning handlar om å sjå matematikken som fornuftig, nyttig og verdifull. Då må elevane oppleve matematikken som noko dei forstår, og læraren sitt engasjement må smitte over på elevane, slik at normene vert positivt påverka (Kilpatrick, 2001).

3 Metode og forskingsdesign

I hovudsak ligg det til grunn ei kvantitativ tilnærming i dette prosjektet, der surveymetode vart nytta, samt eit kvasi-eksperimentelt design for å svare på forskingsspørsmålet: Korleis påverkar programmering analogt (unplugged) kontra digitalt (plugged) elevane si evne til algoritmisk tenking og problemløysing i matematikk? I tillegg er det aktuelt å nytte observasjonar frå intervensjonen som eit grunnlag til drøftingar for å belyse ulike sider ved forskingsspørsmålet, sjølv om det ikkje er gjennomført systematisk observasjon av gjennomføringa av intervensjonen.

Det vil vidare i denne delen bli presentert og argumentert for val av metode, design og utforming og gjennomføring av intervensjonen. Vidare vil det kome ein presentasjon av utvalet og det vil også bli presentert val av analysemetode og bakgrunn for val av variablar til analysen. Til sist kjem det ein del om validitet og reliabilitet, samt kva forskingsetiske omsyn som er tatt i vare i prosjektet.

3.1 Metodisk tilnærming

Kvantitativ forsking er studiar der problemfeltet ofte vert definert ved hjelp av variablar og det vert nytta standardiserte metodar for datainnsamling. Ordet kvantitativ kjem frå verbet å kvantifisere, som tyder å talfeste, og det er dette som skal gjerast med denne studia sine variablar. Variablane skal kunne talfestast, og det skal vere mogleg å analysere desse resultata ved bruk av statiske metodar (Befring, 2014). I samfunnsvitskapleg forsking er surveymetoden den vanlegaste metoden blant dei mange metodane som finns innafor kvantitative metodar (Creswell & Guetterman, 2019).

Surveymetoden er ein framgangsmåte som nyttar spørjeskjema/survey for å samle inn data frå eit utval menneske om deira haldningar, meiningar, åtferd opplevingar eller andre karakteristikkar for populasjonen (Creswell & Guetterman, 2019). Ein får ein utdelt eit spørjeskjema der ein som regel skal markere kor einig eller ueinig ein er i ei rekke påstandar. Dette er ein effektiv metode for å samle inn store mengder data frå mange menneske samtidig på ein systematisk måte. Metoden er

systematisk fordi ein stiller identiske spørsmål til ei mengd personar. Surveymetoden vert elles rekna som lite fleksibel, då ein ikkje kan endre på spørsmåla eller stille oppfølgingsspørsmål når spørjeskjemaet er distribuert (Høgheim, 2020).

Det er relevant for å belyse forskingsspørsmålet, kva elevane sjølv tenkjer om eigne kunnskapar kring algoritmisk tenking og problemløysing. Derfor vart det gjennomført ein pre- post – survey, der elevane vart spurde om eigne kunnskapar innanfor ulike element kring algoritmisk tenking. Resultatet frå den gjennomførte surveyen, vil kunne talfestast og analyserast ved bruk av statistiske metodar. Dette vil vere ein systematisk tilnærming i den form at ein stiller same spørsmåla til ei mengd menneske, i dette tilfellet, elevar som tek del i intervensjonen.

3.2 Kvasi-eksperimentelt design

Eksperimentelle design inkluderer ikkje berre ekte eksperimentelle design, men også kvasi-eksperimentelle design. Den sistnemnde typen design vert kjenneteikna ved at kriteriet om intervensjonen er oppfylt (derav namnet eksperimentell), men ikkje kriteriet om tilfeldig personfordeling (derav namnet kvasi). Difor vil kontrollen av irrelevante faktorar, og difor den indre validiteten, generelt verte svakare enn ved ekte eksperimentelle design (Lund, Fønnebø & Haugen, 2006, s. 132). Lund (2002) peiker på sju ulike kvasi-design, men eg tykkjer ingen av desse designa høver godt til mitt eksperiment. Eg vil seie at mitt design er ei blanding av dei to designa som Lund kallar; *subjektiv-estimering-design* og *pretest-posttest-design med ikkje-ekvivalente grupper* (Lund, 2002, s. 231 og 253-254) Eg kallar difor designet mitt for eit kvasi-eksperiment med to grupper (unplugged-gruppe og plugged-gruppe) som gjennomfører pre- og post- test og ein pre og post-survey.

Høgheim (2020) peikar på at eksperimentelle design egnar seg godt i studiar som vil undersøke effekten av eitt eller fleire tiltak på eitt fenomen, for eksempel eit undervisningsopplegg. Vidare peikar han på at dette vert kalla ein *intervensjon*, fordi forskaren griper inn (*intervenerer)* og endrar noko, for å kunne undersøke det (Høgheim, 2020, s. 112-113). I dette prosjektet har det blitt gjennomført ein intervensjon, der eg ynskja å sjå på om det er skilnad mellom å programmere analogt (unplugged), kontra digitalt (plugged), med tanke på korleis evnene til elevane innanfor algoritmisk tenking og problemløysing i matematikk utviklar seg. Intervensjonen baserer seg på to undervisningsopplegg, der det eine er analogt (unplugged), og det andre er digitalt (plugged), med

mål om å kunne sjekke om det er forskjell i utviklinga mellom desse to gruppene. Resultata på dei gjennomførte testane (pre – post-test), vil på same måte som resultata på pre – post-survey, kunne talfestast og analyserast ved bruk av statistiske metodar.

Ein pre-test er ein måling ein føretek før intervensjonen vert gjennomført, medan ein post-test vert gjennomført etter intervensjonen. Føremålet med ein pre-test og ein post-test, er at ein kan samanlikne gruppene både før og etter intervensjonen for å kunne sjå om intervensjonen hadde effekt (Høgheim, 2020).

Forskingsspørsmålet i dette prosjektet er kausal, noko som fortel at den er basert på eit forhold mellom årsak og effekt. I kausale relasjonar er det ein variabel som skapar ein forandring i eit gitt fenomen. Variabelen er årsaka og fenomenet er effekten (Grønmo, 2004). Som nemnt tidlegare ynskjer forskingsspørsmålet i denne oppgåva å svare på om programmering unplugged eller programmering plugged (årsak) vil utvikle elevane sine evner til algoritmisk tenking og problemløysing (effekt). I kvasi-eksperimentelle design veit ein ikkje om dei gruppene som vert samanlikna, verkeleg er samanliknbare. Sjølv om gruppene er like på ein pre-test, er det ingen garanti for at det er like lett å produsere framgang i dei to gruppene. Difor vil det alltid vere ein del truslar mot indre validitet som kvasi eksperimentelle design ikkje kontrollerer for, og difor vil alternative årsakstolkingar vere moglege (Kleven & Hjardemaal, 2018, s. 128).

3.3 Observasjon av intervensjonen

Sjølv om intervensjonen i dette prosjektet er eit undervisningsopplegg med mål om å kunne sjekke om det er forskjell i utviklinga mellom unplugged-gruppa og plugged-gruppa, og at dette skal talfestast og analyserast ved bruk av statistiske metodar, er det sentralt og også inkludere observasjonar gjort av dei to gruppene. Det er interessant å inkludere desse observasjonane då det vert observert situasjonar som ikkje kan målast gjennom pre-post-testen eller pre-post-survey. Desse situasjonane er også viktige med tanke på utbetring av undervisningsopplegget, samt gjennomføringa. Det er også viktig å presisere at dette er eit kvasi-eksperiment med eit lite utval, som mest truleg vil gje resultat som ein ikkje kan dra endelege slutningar ut av. Observasjonsdelen av oppgåva vert difor viktig, og dei observasjonane som vart gjort, vil bli inkludert i datapresentasjonen i neste kapittel, samt drøftingsdelen.

Observasjon som forskingsmetode omhandlar registrering av informasjon om menneskeleg åtferd eller om stader i ulike situasjonar (Cresswell, 2019). Dette kan innebere sosiale relasjonar, kommunikasjon, dynamikk mellom og innad i grupper, åtferd og haldningar. Observasjon omhandlar systematisk innsamling og registrering (Cresswell, 2019). I denne samanheng er det naturleg å forstå observasjonssituasjonen som *simulert kontekst,* som er situasjonar ein «rigger» for å sjå kva som skjer (Høgheim, 2020). Simulerte kontekstar er vanleg når man skal observere korleis deltakarane blir påverka når ein setter opp eit bestemt opplegg, som i dette prosjektet, eit undervisningsopplegg med ein førehandsett agenda (Høgheim, 2020). Simulerte kontekster er truleg mest vanleg innanfor kvantitativ forsking, der ein kan gjennomføre ein intervensjon, der ein kan leggje til grunn at deltakarane kan verte påverka av intervensjonen (Høgheim, 2020).

Når det gjeld mi rolle som observatør, tok eg rolla som *passiv observatør*, ved at eg ikkje stilte spørsmål eller blanda meg inn i intervensjonen medan den føregjekk. Fordelen med passiv observasjon, er at ein lettare kan få overblikk over heile situasjonen og ein vert ikkje påverka av at ein sjølv deltek i aktiviteten (Høgheim, 2020). Undervegs i undervisningsopplegget nytta eg eit slags feltnotat, der eg noterte ned observasjonar eg tenkte var interessante og relevante for drøfting av forskingsspørsmålet i dette prosjektet.

3.4 Pre- og post-survey

I arbeidet med utforming av pre- og post-survey, var det relevant å nytte Wing (2006) sin modell som skildrar kva kompetanse den algoritmiske tenkjaren har. Desse kompetansane er; *å gjere feil*, *uthald*, *kreativitet, samarbeid, dekomponering, å tenkje i algoritmar, å kjenne att mønster, abstraksjon og generalisering.* Spørjeundersøkinga består av 18 påstandar (Vedlegg 5). Det er påstandar som omhandlar ni hovudkategoriar, det vil seie at kvar kategori hadde to påstandar. Desse påstandane var formulert slik: «Eg er flink til å…», og elevane skulle setje verdiar på desse påstandane frå 1-5. Under kjem eksempel på to påstandar frå pre-post-survey.

Eg er flink til å lære av feila mine

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

Eg er flink til å sjå på det å gjere feil som ein naturleg del av å løyse eit problem

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

SurveyXact vart nytta for utarbeiding, gjennomføring og innhenting av data. Spørjeundersøkinga vart gjennomført to gonger i løpet av intervensjonen. Ein gong etter den første undervisningsøkta og ein gong etter den nest siste undervisningsøkta. Dette vart gjort for å sjå om det var noko endring i kva elevane meinte om eigne eigenskapar kring algoritmisk tenking og problemløysing frå oppstarten av intervensjonen til slutten av intervensjonen. Spørjeundersøkinga var distribuert digitalt via lenkje på Teams til dei to ulike gruppene.

3.5 Pre- og post-test

Oppgåvene i pre- og post-testen (vedlegg 4) er henta frå Bebras-utfordringa (https://www.bebras.no/). Bebras-utfordringa er ein verdsomspennande konkurranse som har vore nytta i over 40 land, der elevane vert utfordra på oppgåver innanfor emna informatikk og algoritmisk tenking. Hovudmålet med Bebras-utfordringa er å motivere elevar til å verte interesserte i emne innanfor informatikk, og å fremje tenking som er algoritmisk, logisk, operasjonell og basert på det fundamentale innanfor informatikk. Utfordringa skal stimulere elevane si utvikling innanfor algoritmisk tenking (Dagienė & Stupuriene, 2016).

Dei tre hovudkarakteristikkane for oppgåvene i utfordringa er at; problemet er lett å forstå, oppgåva vert presentert på ei dataside og oppgåva er uavhengig av spesifikke system (Dagienė & Stupuriene, 2016). Hovudgrunnen til at eg har valt ut oppgåver frå Bebras-utfordringa til pre- og post-testen er at oppgåvene skal vere problem som er korte og enkle og forstå for elevane, dei skal ikkje ta lang tid å løyse. Bebras-oppgåvene går ut på å finne samanhengar og å sjå mønster og nytte dette for å finne den beste løysinga Oppgåvene kan ikkje koplast direkte til kompetansemål for niande trinn, men innan kjerneelementet utforsking og problemløysing er desse oppgåvene gode døme. I læreplanen står det at elevane skal utforske i matematikken ved å leite etter mønster og finne samanhengar (Udir, 2020). Elevane skal også utvikle ein metode for å løyse eit problem dei ikkje kjenner frå før

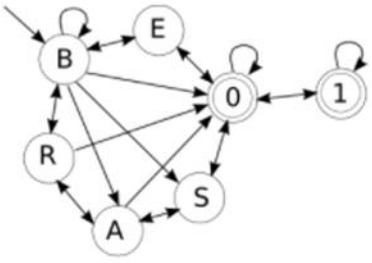
(Udir, 2020). Bebras-oppgåvene er ukjende problem for elevane og elevane vert utfordra til å utvikle sin eigen metode for å løyse desse oppgåvene.

Pre- post-testen bestod av fem oppgåver henta frå Bebras sine prøveoppgåver i 2016. Nokre av oppgåvene var berekna for mellomtrinnet medan nokre var berekna for ungdomstrinnet. Grunnen til dette var at oppgåvene skule vere progressivt vanskelegare, og at dei første oppgåvene hadde ein slik vanskegrad at dei fleste elevane kunne klare å svare på desse. Det var viktig for meg at den første oppgåva ikkje skulle vere så vanskeleg at nokre elevar mista motet og ville gi opp allereie på den første oppgåva. Det vart mange rundar med faglærar før me i lag vart einige om vanskegraden på oppgåvene. I lag med faglærar vart me også einige om å korte ned talet oppgåver på testen frå sju til fem. Me vart også einige om å ha ei tidsfrist på gjennomføring av testen på 15 minuttar. Under er eit eksempel på ein av oppgåvene i pre-post-testen.

Oppgåve 3

Beverane bygger flåtar. Ved trafikkontroll må alle flåtar vere registrerte. Dette betyr at kvar flåte skal ha eit nummerskilt med unik tekst.

Nummerskiltet må starte med bokstaven B og slutte med talet 0 eller 1.
Bruk figuren under til å finne ut kva nummerskilt som er gyldige. Følg pilene for å lage nummerskilt. 0 og 1 har to sirklar rundt seg, det betyr at ein kan velge om ein skal avslutte teksten eller fortsette



Spørsmål:
Hvilke to av nummerplatene kan ikke bli registrert?

BB0001	BBB100	BBB011	BB0100
BR00A0	BSA001	BE0S01	

Figur 4 Eksempel frå pre-post-test

Pre- og post-testen vart som sagt gjennomført før og etter intervensjonen. Elevane fekk utdelt testen i papirform, deretter vart testen samla inn og gjennomgått av meg. Elevane kunne få frå 1 -2 poeng på dei ulike oppgåvene, og kom ut med ein total poengsum som er grunnlaget for å sjå om det var endring frå pre- til post-testen. Resultata frå pre- post-testen vert i resultatdelen kategoriserte som logisk tenking.

3.6 Deltakarar i intervensjonen

Utvalet i dette prosjektet er 17 elevar på niande trinn. Alderen til utvalet er 14 eller 15 år. Desse vart valde fordi intervensjonen som vart planlagt inneheldt oppgåver som var høvande for denne aldersgruppa. I læreplanen i matematikk (LK20) er det to kompetansemål etter niande trinn der elevane skal berekne og vurdere statistikk i sannsyn og spel. Dei skal vidare simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å nytte programmering (LK20). Desse to kompetansemåla er logisk å kople opp mot programmeringsoppgåvene som elevane skal jobbe med under intervensjonen, og det er ein av grunnane til at akkurat denne aldersgruppa vart valt som deltakarar i dette prosjektet.

Elevane vart delt inn i to randomiserte grupper, der åtte av elevane vart plassert i unplugged-gruppa og ni av elevane i plugged-gruppa. Desse gruppene vart igjen delt inn i høvesvis to og tre grupper. Hovudskilnaden på dei to gruppene er at den eine gruppa jobbar med datamaskiner – plugged og den andre gruppa jobbar utan datamaskiner – unplugged. Det er to lærarar som er ansvarlege for undervisninga i kvar av dei to gruppene. Dei to lærarane har meldt seg frivillig. Dei er to matematikklærar som har fått instruksjon av meg om kva dei skal gjere. Tanken bak kven som skulle gjennomføre intervensjonen, var at dei begge var matematikklærarar i den aktuelle klassen. Deira kunnskapar om programmering var ikkje relevante i denne prosessen.

3.7 Intervensjonen

Intervensjonen i dette prosjektet er eit undervisningsopplegg som går over to veker. Kvar veke vert det gjennomført to økter, der den eine er ei økt med varigheit på 100 minuttar, og den andre er ei økt med varigheit på 40 minuttar. Elevane arbeider med ei større oppgåve per veke, det vil seie at dei byrjar i den lange økta og held fram med same oppgåve og fullfører den i den korte økta. Det første som vert gjort i intervensjonen er at den samla elevgruppa på 17 elevar gjennomfører pre-testen. Etter at pre-testen er gjennomført vert elevgruppa delt inn i plugged og unplugged-grupper. Dei to gruppene jobbar med oppgåvene i to ulike rom. Elevane jobbar i dei same gruppene gjennom heile

intervensjonen. Dei to hovudgruppene jobbar begge to med oppgåver som omhandlar tema innanfor matematikk, problemløysing, programmering og algoritmisk tenking. Under er ein figur som i korte trekk viser kva som er dei sentrale forskjellane og likskapane mellom dei to gruppene:

Figur 5 Oversikt over likskapar og ulikskapar ved plugged- og unplugged-gruppa

3.7.1 Plugged-gruppa

Gruppa startar med å gjennomføre pretesten. Plugged gruppa arbeider vidare med to oppgåver i Scratch, som er eit blokkbasert visuelt programmeringsspråk der ein kan programmere rett i nettlesaren, slik at ein ikkje treng å laste ned noko programvare. Ein treng berre ein brukarkonto på nettsida. Den første oppgåva plugged-gruppa jobbar med er oppgåva Soloball som er henta frå nettsida www.kidsakoder.no (Hjelle, 2018). Oppgåva er printa ut på papir til elevane, slik at dei slepp å byte mellom faner for å lese instruksjonane. I denne oppgåva skal elevane lage seg eit enkelt digitalt-ballspel ved å følgje ei oppskrift. Gruppene i plugged-gruppa vart oppfordra til å samarbeide mest med deltakarane på si gruppe, men det er ikkje forbode å hente tips eller inspirasjon frå dei andre gruppene. Soloball-oppgåva er bygd opp slik at elevane skal følgje ei oppskrift og kopiere ein

kode som dei får eit bilete av. Det vert forklart for elevane kva koden dei skal lage gjer med spelet, og det er laga ulike undringspausar i oppgåva. Undringspausane er som dette: «Fungerer bevegelsen bra? Flyttar katten seg når du rører musepeikaren?». Eller slik som dette: «Sjølv om vi ikkje endra programmet vårt, oppfører katten seg veldig annleis. Skjønar du kvifor?». I denne oppgåva er det programmeringstema som løkker, variablar, føresetnader og skript. Matematisk er oppgåva innom tema som geometri, vinklar og koordinatsystemet. På slutten av den første økta gjennomfører elevane surveyen for første gong (pre-survey). I den andre økta held dei fram med å lage spelet Soloball ved å følgje instruksjonar.

I den tredje økta byrjar elevane på oppgåva *terning* (Vedlegg 6). I denne oppgåva skal elevane nytte Scratch til å simulere terningkast. Dei skal få Scratch til å rekne sannsynet for å få ein gitt sum på kast med to terningar. Programmet skal byggast slik at ein simulerer 1000 terningkast med to terningar og reknar snittet av desse, slik at elevane kan finne sannsynet for å få ein gitt sum på dei to terningane. Oppgåva er bygd opp slik at elevane skal undre seg litt først kring sannsynet for dei ulike summane, vidare skal dei kopiere programmet og reflektere over korleis det verkar og om det kan utbetrast. Dei gjennomfører også spørjeundersøkinga for andre og siste gong (post-survey) på slutten av denne økta.

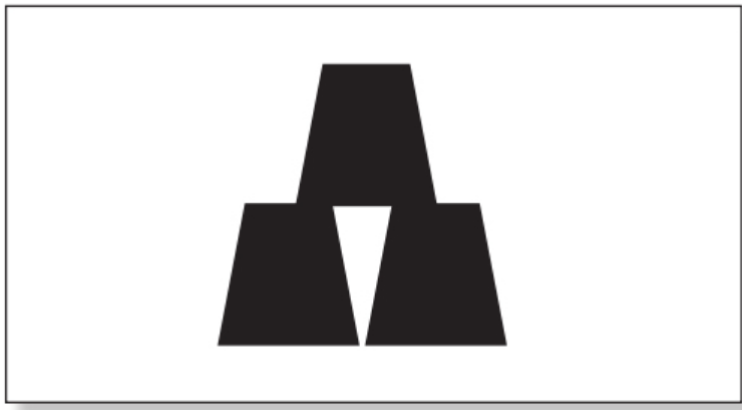
I fjerde og siste økta skal elevane førebu og ha ein presentasjon for dei andre plugged-gruppene, der dei presenterer utfordringar, oppdagingar, kva dei har lært og korleis ulike kodar/program fungerer ut i frå dei oppgåvene dei har jobba med. Her skal gruppene dele erfaringar og utfordringar med kvarandre, slik at det kan leie til ein fagleg diskusjon mellom lærar og elevgruppene. Tanken bak presentasjonen var at elevane skulle få forklare kodar og erfaringar, men også som ein oppsummering for det dei har jobba med gjennom heile intervensjonen. Dei skal også gjennomføre post-testen som ein avslutning av opplegget.

3.7.2 Unplugged-gruppa

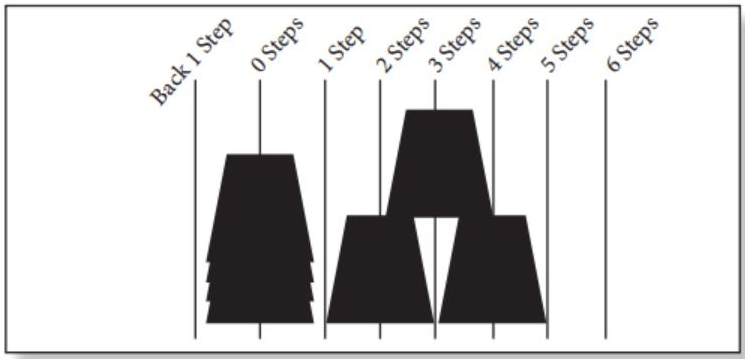
Gruppa startar med å gjennomføre pretesten. Vidare jobbar unplugged-gruppa med to oppgåver der dei skal programmere kvarandre og følgje kvarandre sine kodar for å løyse oppgåvene. Den første oppgåva unplugged-gruppa jobbar med er oppgåva *mine robotvener* som er henta frå www.kidsakoder.no (Thinkersmith, 2013). Oppgåva går ut på at deltakarane skal etter tur vere ein robot som skal stable koppar. Dei elevane som ikkje er robotar skal lage ein algoritme som skal omsetjast til piler som vidare fortel roboten korleis han skal stable koppane. I denne oppgåva skal

læraren utifrå mine instruksjonar forklare for dei to gruppene korleis oppgåva skal utførast. Det trengs ein del utstyr før oppgåva kan gjerast. Begge gruppene treng ein del koppar, eit ark med symbol og eit ark med ulike døme til koppestablar. Læraren skal vidare halde ei lita fellesøkt der han kartlegg elevane sine kunnskapar om robotar og korleis dei fungerer. Vidare går læraren gjennom symbolarket med dei seks pilene som er dei einaste symbola som kan nyttast i denne oppgåva. Elevane vert også introdusert for nokre omgrep og kva dei tyder: *Algoritme* – ein serie av instruksjonar som skildrar korleis ein skal ein kan nå eit mål, *Koding* – gjere handlingar om til eit symbolspråk, *Feilsøking* – Finne og rette feil og problem i koden, *Funksjon* – kode som kan brukast om og om igjen og *Parameter* – ekstra informasjon som kan leggast til ein funksjon for å tilpasse den.

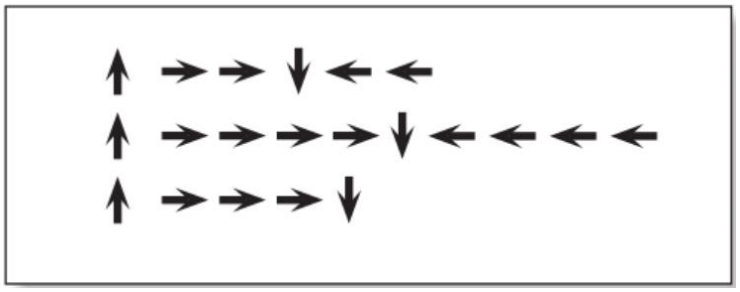
Etter dette går læraren gjennom ei eksempeloppgåve med elevane. Der elevane får sjå ein koppestabel og eit forslag til løysing på korleis ein kan skrive instruksjonane til roboten. Læraren forvandlar seg sjølv om til robot og gjer instruksjonane mekanisk, slik at elevane også kan leve seg inn i rolla som robot. Læraren legg vekt på at når ein først er robot, skal ein vere robot og ikkje menneske. Under følgjer nokre figurer for å vise eksempel på eksempeloppgåver.



Figur 6 Koppestabel til eksempeloppgåve

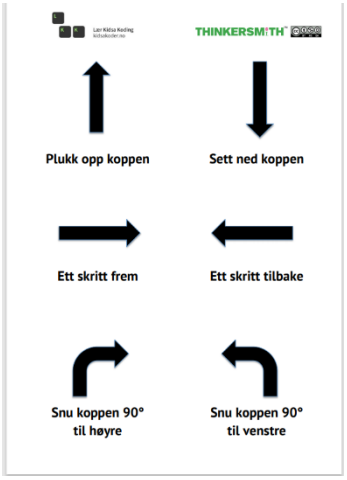


Figur 7 Oversikt over skritt, der elevane vert gjort merksame på at eitt skritt er ei halv koppelengd.



Figur 8 Mogleg løysing for eksempeloppgåva.

Etter introduksjonen går dei to gruppene til kvar sin plass i klasserommet for å avgjere kven som skal vere robot først. Roboten vert så sendt til robotstasjonen, der han ikkje kan sjå resten av gruppa si. På robotstasjonen er det ei mengd koppar og eit symbolark.



Figur 9 Symbolark

Dei resterande elevane som ikkje er robotar, vel seg ein koppestabel frå koppestabelfigurane. Dei har no berre tilgjengeleg dei enklaste koppestablane. Etter dei har valt seg ein koppestabel, lagar dei ei algoritme som dei omset til piler frå symbolarket. Når dei har koda ferdig programmet sitt og testa det ut, overleverer dei programmet til roboten. Når roboten skal utføre programmet er det ikkje lov å kommunisere, det einaste som er lov er å stoppe programmet om ein oppdagar ein feil, då stoppar roboten. Gruppa ser om roboten utfører programmet slik som dei tenkte, om den ikkje gjer det må dei tilbake til plassen sin, feilsøke og endre programmet slik at det vert riktig når roboten skal utføre det. Alle elevane skal byte på å vere robot og koppestabeloppgåvene vert vanskelegare etter kvart som dei vert løyste av roboten. Når elevane byrjar å verte gode på å lage program, kan dei som vil også få lage eigendesigna koppestablar som dei skriv program for og let roboten få løyse. På slutten av denne økta gjennomfører elevane pre-surveyen.

I den andre og kortare økta skal elevane halde fram med å programmere roboten, men no er hovudmålet å effektivisere programma så mykje som mogleg. Det vil seie at programma skal skrivast med så få symbol som mogleg, og roboten skal likevel kunne bygge den koppestabelen som er utgangspunktet. I denne delen av oppgåva får elevane ei oppfrisking av uttrykket *parameter* og dei vert presenterte for denne figuren:

Figur 10 Figur som skal følgjast med klokka, der x er parameteren

I den tredje økta skal elevane jobbe med oppgåva pikselprogrammering (Vedlegg 7). Dei får ein introduksjon om kva pikslar er og litt om korleis pikslar er ein del av deira teknologiske kvardag. Oppgåva går vidare ut på at elevane skal skrive kode til kvarandre og mottakar skal tolke koden og lage eit pikselbilete ut i frå den koden dei har motteke. Om bilete ikkje vert likt det som var planen til programmeraren, må elevane feilsøke og finne ut kvifor bilete ikkje vart likt. Elevane arbeider først med å programmere bilete frå fantasien, så får dei oppgåver der bilete skal vere til dømes eit siffer, ein figur eller ein bokstav. I den eine oppgåve er den nedste kodelinja gitt, slik at pikslane for den linja er «låst». Elevane gjennomfører post- surveyen på slutten av denne tredje økta.

På same måte som i plugged-gruppa, skal elevane i den fjerde og siste økta førebu og ha ein presentasjon for den andre unplugged-gruppa, der dei presenterer utfordringar, oppdagingar, kva dei har lært og korleis ulike kodar/program fungerer ut i frå dei oppgåvene dei har jobba med. Her skal gruppene dele erfaringar og utfordringar med kvarandre, slik at det kan leie til ein fagleg diskusjon mellom lærar og elevgruppene. Tanken bak presentasjonen var at elevane skulle få forklare kodar og erfaringar, men også som ein oppsummering for det dei har jobba med gjennom heile intervensjonen. Dei skal også gjennomføre post-testen som ein avslutning av opplegget.

3.7.3 Førebuing til intervensjonen/ samtale med faglærarar

I forkant av intervensjonen hadde eg og dei to lærarane fleire økter der me gjekk gjennom korleis opplegget skulle gjennomførast, og lærarane hadde høve til å stille spørsmål kring oppgåvene og strukturen. Eg gjekk gjennom skissa til opplegget og definerte deira rolle og mi rolle. Deira rolle var å

gjennomføre intervensjonen, medan mi rolle var å gjere klart og passe på at alt av utstyr var på plass før øktene skulle gjennomførast. Elles fekk dei to faglærarane vite at rolla mi i sjølve intervensjonen, var passiv observatør. Dette vart gjort veldig klart for dei to, slik at dei ikkje skulle forvente at eg skulle bidra med hjelp til dei under intervensjonen. Om dei stod fast i ei forklaring eller vart usikre på noko, var det ikkje hjelp i å støtte seg på meg. Dette gjorde eg for at dei skulle vite at det var dei som gjennomførte intervensjonen og at dei då visste at dei knipene dei hamna i, måtte dei kome seg ut av sjølve.

I forkant av intervensjonen kom lærarane også med mange spørsmål og problemstillingar som eg ikkje hadde teke høgde for i planlegginga mi. Desse spørsmåla førte til ein del endringar i opplegget, og hjelpte meg til å få eit endå betre planlagt opplegg. Som nemnd tidlegare vart vanskegraden og tal oppgåver på pre- post-testen endra. Her har faglærarane meir kunnskap enn meg om kva typar oppgåver elevane er kapable til å løyse og kva som vert for vanskeleg. Faglærarane var særs opptekne av at testen skulle starte med ei oppgåve som alle hadde føresetnadar for å løyse, slik at ikkje motivasjonen deira skulle forsvinne allereie ved den første oppgåva. Vidare hadde lærarane tankar om at oppgåvene skulle verte progressivt vanskelegare utover i testen.

Då det nærma seg starten på intervensjonen, hadde eg ein individuell samtale med kvar av dei to lærarane. I denne samtalen gjekk me heilt konkret gjennom den første oppgåva som skulle gjerast. Det vart gjeve konkrete instruksjonar til kvar lærar, slik at dei var heilt innforstått med korleis oppgåva skulle gjennomførast. Det vart også gjeve autonomifridom i form av korleis dei valde å forklare og introdusere oppgåva. Tanken bak dette var at sjølv om lærarane skulle gjennomføre min intervensjon, skulle dei få eit eigarskap til dei oppgåvene dei skulle gjennomføre med gruppene sine. I denne samtalen fekk lærarane høve til å teste oppgåvene og diskutere med meg korleis dei kunne tenkje seg å gjennomføre den

Vidare gjennom intervensjonen hadde eg individuelle- og felles-samtalar med dei to faglærarane før og etter kvar økt. Kva gjekk bra? Kva kunne vore gjort annleis? Er det me driv på med noko som høyrer heime i matematikken i grunnskulen? Dette var refleksjonsspørsmål som me diskuterte gjennom heile intervensjonen. Dei to lærarane var flinke til å kome med tankar om endringar og korleis ein skulle halde fram med oppgåver som ikkje var fullførte. Dette var veldig hjelpsamt for meg, då tida per oppgåva var noko av det som var vanskeleg å planlegge. Men gjennom god dialog og godt samarbeid, vart det veldig bra til slutt.

3.8 Analyse

Når ein jobbar med analyse av kvantitative data er statistikk det analytiske arbeidet ein gjer med taldata, og når ein trekk slutningar frå statistiske analysar, ser ein på heilskapsbiletet av arbeidet sitt (Høgheim, 2020). Datagrunnlaget er dei variablane som vert testa i pre- og post-testen og i surveyen som utvalet har svara på. Totalt er det 10 variablar som vert testa, der ni av variablane er frå surveyen og den eine variabelen er pre og post testen. Dei ni variablane som er testa gjennom surveyen er element som omhandlar algoritmisk tenking og er henta frå den algoritmiske tenkjaren (Wing, 2006). Desse variablane er *å gjere feil*, *uthald*, *kreativitet, samarbeid, dekomponering, å tenkje i algoritmar, å kjenne att mønster, abstraksjon og generalisering.* Den siste og tiande variabelen som vert testa er pre- post-testen som eg har gitt namnet *logisk tenking.* Eg nytta Microsoft Excel for å organisere data og å rekne gjennomsnitt og standardavvik. Vidare nytta eg ein online effektstorleik-kalkulator (Wilson, 2018) for å finne effektstorleik på dei ulike variablane. Som tidlegare peika på er dette eit kvasi-eksperiment med eit lite utval, som mest truleg vil gje resultat som ein ikkje kan dra endelege slutningar ut av. Likevel vil eg i presentasjon av talmateriale peike på dei ulike variablane frå analysen, for å kunne sjå om det er ein tendens, og også kan det vere av relevans knytt til dei observasjonane som er gjort.

Grunnen til at unplugged-gruppa står fram som hovudgruppa og plugged-gruppa står fram som kontrollgruppa i resultatdelen, er fordi eg måtte gjere det slik for effektstorleik-kalkulatoren trengte denne inndelinga for å rekne ut effekt. Det er ikkje noko anna grunn for at desse gruppene er framstilt slik i resultattabellen. Eg nytta ikkje ei hovudgruppe og ei kontrollgruppe i min intervensjon.

3.9 Validitet og reliabilitet

Når kvalitet på data skal vurderast, er empirien sin validitet og reliabilitet viktig. Validiteten refererer til kor gyldig den endelege datamaterialet er i forhold til forskingsspørsmålet og det forskaren vil ha svar på (Grønmo, 2004). Korleis gir empirien oss svar på det ein er ute etter å finne ut? Reliabilitet handlar om kor pålitelege data er , og i dette ligg det at undersøkinga sitt design og gjennomføring er viktig for å få høgast mogleg reliabilitet. Dette er to ulike omgrep, men det er likevel viktig å vite om samanhengane mellom dei. Høg reliabilitet er ein føresetnad for høg validitet, men ein treng ikkje ha høg validitet for å også ha høg reliabilitet (Grønmo, 2004). I dette prosjektet er den største trusselen mot validiteten og spesifikt den ytre og statistiske validiteten, det vesle utvalet som er i forskinga. I plugged-gruppa er n=9 og i unplugged-gruppa er n=8.

I kvasi-eksperimentelle design er ikkje kriteriet om tilfeldig personfordeling oppfylt, difor vil kontrollen av irrelevante faktorar, og difor den indre validiteten, generelt verte svakare enn ved ekte eksperimentelle design (Lund, Fønnebø & Haugen, 2006, s. 132). Ein veit heller ikkje om dei gruppene som vert samanlikna, verkeleg er samanliknbare. Sjølv om gruppene er like på ein pre-test, er det ingen garanti for at det er like lett å produsere framgang i dei to gruppene. Difor vil det alltid vere ein del truslar mot indre validitet som kvasi eksperimentelle design ikkje kontrollerer for, og difor vil alternative årsakstolkingar vere moglege (Kleven & Hjardemaal, 2018, s. 128).

3.10 Forskingsetiske omsyn

Lund og Haugen (2006) peikar på at det skal vere frivillig for informantar å delta i eit forskingsprosjekt. Vidare peikar dei på at informantane skal når som helst og utan grunn trekkje seg frå prosjektet, utan at dette får nokon negative konsekvensar for dei (Lund og Haugen, 2006). I dette prosjektet vart det både munnleg og gjennom samtykkeskjemaet informert om at deltakinga var heilt frivillig, og ein kunne trekkje seg når som helst frå prosjektet. Det vart også skissert eit alternativt undervisningsopplegg for dei som trekte seg, slik at det vart like enkelt å trekkje seg frå prosjektet som det var å samtykkje til det.

Ved godkjenning frå NSD sikrar ein at prosjektet tek i vare prinsippet om personvern, som er i tråd med lov om personopplysningar. Dette vil også vere ein tryggheit for deltakarane, då forskaren tek i vare sitt formelle forskingsetiske ansvar (Høgheim, 2020). I forkant av dette prosjektet, vart det søkt om tillating til å gjennomføre prosjektet til Norsk samfunnsvitskaplege datateneste (NSD). Når all data vart samla inn, vart det oppbevart på passordbeskytta PC. Alle data vert sletta etter prosjektet er fullført.

Alle moglege deltakarar til prosjektet skal ha full innsikt i forskinga sitt føremål, då skal dei ha moglegheita til å vurdere om dei er villige til å delta eller ikkje, og om dei vil trekkje deltakinga si når som helst i prosjektet (Grønmo, 2004). Alle deltakarar vart informerte om kva prosjektet gjekk ut på, korleis det skulle gjennomførast, at deltakinga var frivillig og at dei kunne trekke seg frå prosjektet når som helst utan å grunngje kvifor. Dette vart gjort munnleg og skriftleg for dei aktuelle deltakarane, medan dei føresette fekk det berre skriftleg gjennom samtykkeskjemaet.
Born og ungdommar kan som hovudregel sjølv samtykkje til deltaking i forsking når dei er fylt 15 år. For born under denne alderen bør foreldre samtykkje på vegne av barnet. Dersom det samlast inn særlege kategoriar av personopplysningar må ungdommen vere fylt 16 år for å samtykkje (NSD,

2021). I dette prosjektet var det ikkje samla inn personopplysningar som kan kategoriserast under særlege, men sjølv om alle forskingsobjekta var fylt 15 år, var det dei føresette som måtte signere informert samtykke (vedlegg 1).

4 Resultat

I denne delen av oppgåva vert funna som kom fram i datainnsamlinga presentert. Det vert presentert funn som er interessante å sjå nærare på i lys av tidlegare presentert teori og i høve forskingsspørsmålet; *Korleis påverkar programmering analogt (unplugged) kontra digitalt (plugged) elevane si evne til algoritmisk tenking og problemløysing i matematikk?* Det vil bli presentert talmateriale frå pre- og post-test og pre- og post-survey som elevane har gjennomført før og etter intervensjonen. I tillegg vil eg gå nærare inn i observasjonar som er gjort under intervensjonen. Det er igjen viktig for meg å igjen presisere at dette er eit kvasi-eksperiment med eit lite utval, som mest truleg vil gje resultat som ein ikkje kan dra endelege slutningar ut av.

4.1 Effektstorleik som målestorleik

Effektstorleiken til dei ulike variablane er faktoren som skil ut kva for variablar og funn som vert interessante for å sjå nærare på i resultatdelen av oppgåva. Dette vert gjort sjølv om eg framleis er medviten på at prosjektet har eit lite utval og ein kan ikkje dra konkluderande slutningar frå dei effekt-resultata ein får.

Effektstorleik syner kor markant ein skilnad eller samanheng er. Som forskar kan ein undersøke til dømes korleis det er å få ein behandling kontra det å ikkje få ein behandling. I dette forskingsprosjektet er «behandlinga» datamaskin eller ikkje datamaskin. Effektstorleiken vert berekna ut i frå gruppeskilnaden, men tek framleis variasjon innan gruppene med i betraktning. Heilt konkret skjer dette ved at gjennomsnittsskilnadene mellom gruppene vert dividert på standardavviket i gruppene. Jacob Cohen definerer effektstorleik som ”the degree to which a phenomenon exists”. Ein vanleg tolking av effektstorleik er .20=liten, .50=medium og .80=stor (Cohen, 2003).

Eg nytta effektstorleik for å finne samanhengen med dei ulike variablane sett opp mot forskingsspørsmålet. Dette gjorde eg for å finne ut kva for variablar som vart påverka av intervensjonen. Eg ville finne ut kva variablar innanfor algoritmisk tenking og problemløysing som

skilte seg ut med tanke på om effektstorleiken var liten, medium eller stor ut i frå Cohen's d. Under (Tabell 1) ser ein oversikt over variablane som vart testa. Som ein ser i samletabellen, er det tre variablar som skil seg ut; *kreativitet, samarbeid og logisk tenking.* Med skil seg ut, meiner eg at dei har ein Cohen's d på over .2, noko som seier at denne intervensjonen kan ha hatt ein effekt på dei nemnde variablane. Ein av desse aktuelle variablane har ut i frå skalaen liten effekt og dei to andre har medium effekt.

Tabell 1 Gjennomsnitt (M) og standardavvik (SD) for føresetnadane og effektstorleik (Cohen's d)

Variabel		Pre-test: M (SD)	Post-test: M (SD)	$M_{Differanse}$	Cohen's D
Å gjere feil	Unplugged (hovudgruppe)	2.81 (.53)	3.07 (1.27)	.26	.04
	Plugged (kontrollgruppe)	3.55 (.63)	3.78 (.75)	.23	
Uthald	Unplugged (hovudgruppe)	2.69 (1.22)	2.78 (1.38)	.09	.04
	Plugged (kontrollgruppe)	3.61 (.78)	3.66 (.83)	.05	
Kreativitet	Unplugged (hovudgruppe)	2.75 (.85)	2.78 (1.14)	.03	.22
	Plugged (kontrollgruppe)	3.11 (.69)	3.33 (.66)	.22	
Samarbeid	Unplugged (hovudgruppe)	2.5 (1.25)	2.14 (.98)	-.36	.66
	Plugged (kontrollgruppe)	3.83 (.79)	4.11 (.78)	.28	
Dekomponering	Unplugged (hovudgruppe)	2.18 (.92)	2.28 (.95)	.10	.15
	Plugged (kontrollgruppe)	3.00 (.70)	3.22 (.44)	.22	
Å tenkje i algoritmar	Unplugged (hovudgruppe)	2.38 (.83)	2.71 (1.14)	.33	.12
	Plugged (kontrollgruppe)	3.72 (.66)	4.16 (.90)	.44	

Å kjenne att mønster	Unplugged (hovudgruppe)	2.43 (1.05)	2.85 (.85)	.42	.12
	Plugged (kontrollgruppe)	3.33 (.50)	3.66 (.55)	.33	
Abstraksjon	Unplugged (hovudgruppe)	2.62 (1.18)	2.92 (1.23)	.30	.08
	Plugged (kontrollgruppe)	3.33 (.66)	3.77 (.65)	.22	
Generalisering	Unplugged (hovudgruppe)	2.62 (1.18)	3.00 (1.41)	.38	.05
	Plugged (kontrollgruppe)	3.33 (.66)	3.66 (.50)	.33	
Logisk tenking (pre- post-test)	Unplugged (hovudgruppe)	2.66 (1.50)	2.16 (1.69)	-.50	.64
	Plugged (kontrollgruppe)	2.88 (2.02)	3.55 (1.94)	.67	

4.2 Kreativitet

Den variabelen som har lågast effekt av dei tre som skil seg ut er *kreativitet*, som har ein Cohen's d-verdi på .22. Utifrå skalaen vil det seie at intervensjonen kan ha hatt ein effekt på *kreativiteten* i kontrollgruppa med data, men denne effekten er liten. Det ein ser i resultattabellen er at effekten som har oppstått innanfor dei ulike variablane, skjer i plugged-gruppa.

4.3 Samarbeid

Den neste variabelen som skil seg ut, er *samarbeid.* Innanfor samarbeid, er det også i plugged-gruppa ein ser auken i effekt. Variabelen samarbeid har ein Cohen's d på .66, noko som ut ifrå skalaen kan syne at intervensjonen har hatt ein medium effekt på variabelen samarbeid for plugged-gruppa.

4.4 Logisk tenking

Den tredje og siste variabelen der ein kan finne ein effekt, er *logisk tenking.* Som tidlegare nemnt i oppgåva, vart resultata til variabelen logisk tenking innhenta i form av ein pre- post-test, så resultata

for denne variabelen er resultata deltakarane skåra på pre- og post- testen. Også her ser ein at det er i plugged-gruppa det er framgang. Denne variabelen har ein Cohen's d på .64.

4.5 Observasjonar

Etter pre-testen vart elevane delte i to grupper som vidare vart delte i undergrupper. Dei to hovudgruppene vart fysisk delte, slik at dei arbeidde på kvart sitt rom. Ingen av gruppene fekk innsyn i kva den andre gruppa arbeidde med. Det verka for meg som om at elevane tykte programmeringsoppgåvene generelt var kjekke å jobbe med, då det var ingen elevar som kopla ut og ikkje jobba med dei oppgåvene dei vart tildelte. Elevane i dei to gruppene viste engasjement og vilje til å fullføre dei oppgåvene dei jobba med. Dei verka også undrande og løysingsorienterte når løysingane dei kom fram til ikkje verka slik som dei trudde dei skulle. Dei gav ikkje opp og vart skuffa fordi løysinga ikkje fungerte, men dei fokuserte heller på korleis dei kunne gjere endringar for at oppgåva kunne løysast på ein betre måte.

4.5.1 Plugged-gruppa

Under vil eg skildre prosessen til plugged-gruppa, samt trekke fram interessante observasjonar eg gjorde under dei ulike prosessane i intervensjonen.

Den første økta

I den første økta fekk gruppene i plugged-gruppa kvar si individuelle utskrift av oppgåva soloball, slik at dei kunne nytte datamaskina berre til programmering. Elevane skulle så registrere seg på www.scratch.mit.edu, noko som skapte litt problem og førte til frustrasjon. Dette var eigentleg noko elevane skulle ha gjort i forkant av intervensjonen i høve intervensjonsplanen, men det vart gløymd. Eit anna problem som dukka opp når elevane hadde byrja med oppgåva, var språkvalet i Scratch. Elevane på ein skule med nynorsk er godt opplærde og set språket til nynorsk, men ein finn då fort ut at mange av blokkene i oppskrifta som er på bokmål, ikkje høver med nettsida. Det virka som om elevane var ivrige til å kome i gang og dei var raske i gang med oppgåva. Då elevane hadde kome i gang med oppgåva, var det dei gruppene som snakka mest i lag som hadde størst progresjon. To av gruppene snakka mykje med kvarandre og hjelpte kvarandre ved å peike på skjermen og vise kva dei hadde gjort for å løyse ulike ting. Medan den eine gruppa var heilt stille, og dei sat med datamaskinene slik at dei ikkje kunne sjå kvarandre sine skjermar. På denne tause gruppa var progresjonen generelt treig, og det var også stor skilnad mellom medlemmane i denne gruppa kring progresjon. Læraren gjorde fleire val for å hjelpe denne gruppa. Det første han gjorde var at han

oppfordra gruppene til å dele progresjonen sin med dei andre gruppene ved å kople seg til den digitale tavla i klasserommet. Dette førte til at gruppene fekk høve til å dele idéar med dei andre gruppene. Dei fekk også høve til å setje ord på tankane, prosessane og ikkje minst programmeringa og matematikken i det dei hadde gjort. Etter denne delingsseansen skaut progresjonen hjå den treige gruppa til værs. Det førte også til at det vart lausare skilje mellom gruppene. Dei sat framleis i dei faste gruppene, men no var det opna opp for at elevane kunne hente informasjon og hjelp frå dei andre gruppene. Det vart også opna opp for at alle no hadde tilgang til den digitale tavla om dei hadde funne ut noko dei tykte var kult eller dei tykte dei andre gruppene burde få sjå.

Den andre økta

I den andre økta heldt elevane fram med Soloball-oppgåva. I denne økta kan det virke som om elevane er lite motiverte for å halde fram med oppgåva. Når læraren spør kvifor, seier dei at dei trudde dei var ferdige med oppgåva og at dei skulle få byrje med noko nytt. Denne mangelen på motivasjon held fram til dei kjem i gong i oppgåva att, då ser dei at dei ikkje var ferdige med oppgåva likevel. Etter at elevane er komne i gong og motivasjonen er tilbake, gjer læraren ein ny observasjon, han ser at spela til to elevar på same gruppe er ulike. Han spør så: «Oj, desse spela var ulike. Kvifor er spela ulike når de har nytta same oppskrifta? Korleis kan de finne ut kva som er ulikt?» Dette set i gong ein prosess i gruppa som fører til engasjement blant elevane. Her må elevane feilsøke for å finne ut om det er noko som er gale med programmet eller kanskje det er programvara? Alle elevane på gruppa er deltakande i denne prosessen, og dei tykkjer at det er merkeleg at spela er vortne så forskjellige. Spelet til den eine eleven står og hakkar, medan spelet til den andre eleven let seg spele fint. Elevane ser gjennom programma til begge elevane, studerer oppskrifta på nytt og diskuterer med kvarandre kva som har skjedd sidan det eine spelet fungerer optimalt medan det andre ikkje er funksjonelt.

Den tredje økta

I den tredje økta byrjar elevane på ei ny oppgåve som heiter terningkast. I denne oppgåva skulle elevane lage eit program som tel frekvens for ulike summar av terningkast med to terningar. Dette er ikkje ei oppgåve med oppskrift på same måte som soloball-oppgåva. I denne oppgåva byrjar læraren med ein introduksjon der han snakkar med elevane om sannsynet for ulike summar for terningkast med to terningar. Læraren viser så eit ferdiglaga program på den digitale tavla i klasserommet som elevane får i oppdrag å kopiere. I dette programmet er det tre robotar som vert programmerte til å simulere 1000 terningkast med to terningar og å rekne ut prosentdelen av dei ulike summane ein kan få på to terningkast. I denne oppgåva vert som tidlegare nemnt, oppdraget til elevane å forstå og forklare korleis programmet fungerer. Det virka som om elevane var særs motiverte for å starte med

ei ny oppgåve, då dei var ivrige med å kome i gang. Dei utrykte at det var kjekt at dei fekk lov å velje kva for figurar dei fekk nytte til å gjere dei ulike operasjonane i programmet. Sjølv om dei hadde stor fridom til å velje sjølve, vart det til at alle valte dei tre robotane som var i eksempel-programmet. Då dette var ei avskrifts-/kopierings-oppgåve, gjekk det forholdsvis fort for elevane å kopiere programmet. Når programmet var kopiert, sette læraren i gang diskusjonen om korleis programmet fungerer og kvifor det fungerte slik. Det første elevane la merke til ved programmet var at det var ein feil i programmet, noko som både eg og læraren visste då me jobba med programmet før undervisninga. Feilen var at første terningkast alltid gir sum null. Elevane vart då særs ivrige på å fikse denne buggen i programmet. Dette fekk dei sjølvsagt lov til å prøve, men dette var noko som eg og lærar visste at ikkje var nokon «quick-fix». Elevane fekk lov til å forske litt på problemstillinga og diskutere litt på gruppene om korleis den let seg fikse, men lærar var medviten på at dette var eit problem som dei med deira kompetanse mest sannsynleg ikkje klarte å løyse. Det var også eit tidsaspekt på intervensjonen i biletet, som førte til at ein ikkje kunne setje av mykje tid til å undre seg over dette problemet. Det læraren så gjorde, var å spørje elevane om kva denne feilen i programmet har å seie for matematikken som dei jobbar med i denne oppgåva. Her var elevane samstemte om at dersom eitt av 1000 terningkast gav feil verdi, hadde det veldig lite å seie for sannsynet for dei ulike utfalla. Likevel meinte dei at det var noko som ein måtte ha med seg i bakhovudet når ein køyrde programmet for dei ulike utfalla.

Ein annan situasjon som oppstod i programmet til nokre av gruppene, var at dei ulike karakterane/robotane i programmet oppførte seg ulikt frå gruppe til gruppe. På dei eine gruppa fann dei ikkje ut samanhengen mellom kvifor dei ulike karakterane/robotane i programmet ikkje fungerte på same måte som dei gjorde i dei andre gruppene sine program. Her også, tok læraren opp denne buggen i heil klasse og spurte kva som kunne vere gale og korleis ein kan fikse dette. I dette høvet er elevane sin løysningsmetode for debugging å samanlikne programma til dei forskjellige gruppene, for så og finne ut kva som er forskjellig og så finne ut om det rettar feilen ved å endre det som er ulikt. Løysinga på dette problemet var enkelt om ein har innsikt og erfaring med Scratch, men det hadde ikkje elevane på dette tidspunktet. I lag fann dei ut at hjå den gruppa som karakterane oppførte seg ulikt var blokkene heilt identiske, men det var nokre funksjonar som ikkje var huka av for. Den aktuelle funksjonen er som regel haka av som standard i Scratch, men akkurat på denne gruppa var ikkje funksjonen; *la variabelen gjelde for alle figurar,* huka av. Dette gjer at den aktuelle variabelen ein lagar for figurane ikkje gjeld for alle figurane, og då vil ikkje figurane oppføre seg likt i dei ulike programma. Det virka som om at elevane tykte det var interessant å jobbe med ei oppgåve på denne

måten ved at dei kopierte eit ferdig program, for så å gå djupare inn i kodane for å forstå korleis dei ulike elementa fungerer eller eventuelt ikkje fungerer.

Den siste økta

Den siste økta for plugged-gruppa vart i tillegg til post-test, nytta til å lage ein presentasjon for dei andre gruppene, der dei skulle fokusere på; kva som har vore utfordrande, kva dei tykte har vore interessant, kva dei er mest nøgde med/kva dei fekk best til og om det er noko spesielt dei har lært som dei vil vise til dei andre gruppene. I denne presentasjonsøkta fekk gruppene nokre minuttar på å presentere for dei andre gruppene.

På plugged-gruppene kom det fram under presentasjonane at dei følte sjølve at dei var betre å dele informasjon på terningoppgåva enn dei var på soloball-oppgåva. Elevane var usikre på om dette var fordi dei var vortne tryggare på programmeringa eller det var forma på oppgåva som gjorde det. Dei følte vertfall at delinga av informasjon mellom gruppene og det generelle samarbeidet var betre på terningoppgåva. Vidare under presentasjonen forklarte elevane kodane sine og kva funksjonar dei ulike blokkene i Scratch har. Det kom også fram under presentasjonen at nokre elevar tykte det var kjekt å berre programmere, dei såg på dette som noko anna enn dei gjorde til vanleg i matematikktimane. Det kom også fram at samarbeidet i oppgåvene var til tider to-delt; det var ein utfordring å få til samarbeidet, men på den andre sida var det kjekt når ein fekk samarbeidet til å fungere. Elevane fortalte også at dei hadde funne ut av at rekkefølgja av blokker hadde mykje å seie for korleis programmet køyrde. Det kom vidare fram at dei var merksame på at det dei kalla småfeil eller småting, kunne føre til problem når det ferdige programmet skulle køyrast. Heilt til slutt i presentasjonen konkluderte dei med at dei generelt fekk terningoppgåva betre til enn soloball-oppgåva. Om dette var fordi dei var vortne betre Scratch-brukarar eller om det var oppgåvene i seg sjølve, var elevane usikre på.

4.5.2 Unplugged-gruppa

Under vil eg på same måte som med plugged-gruppa, skildre prosessen til unplugged-gruppa, samt trekke fram interessante observasjonar eg gjorde under dei ulike prosessane i intervensjonen.

Den første økta

Den første økta for unplugged-gruppa starta med robotven-oppgåva etter at pre-testen var gjennomført. Under gjennomføringa av denne oppgåva vart det organisert slik at tre elevar var

programmerarar og ein elev var robot. I forkant av økta, har eg ordna ein robotstasjon til kvar av gruppene, der roboten har tilgjengeleg alt han treng av instruksjonar og rikeleg med koppar. Den første utfordringa ein ser med denne oppgåva under gjennomføringa, er kva roboten skal gjere medan dei andre medlemmane på gruppa programmerar. I oppgåva står det at roboten kan øve seg på å stable koppar medan han ventar. Dette i seg sjølv er truleg morosamt i starten, men når programmerarane skal skrive lengre og meir komplekse program, vert det særs lang ventetid for roboten som jo i dette tilfellet er ein robot, men i røynda er dette ein elev som finn på mykje rart når ventetida vert for lang. I starten av økta held læraren ein introduksjon, der han går gjennom ein del eksempel og korleis oppgåva kan løysast. I denne introduksjonen la eg merke til nokre beskjedar, den første beskjeden, eller mangel på beskjed, var eigenskapane til roboten. Det vart gitt ein uklar beskjed om at når roboten får eit program som den skal køyre, er dei menneskelege eigenskapane heilt vekke, eleven er no vorten ein robot. Dette virka som om at dette var vanskeleg for elevane å forstå. Dette førte til at roboten stilte spørsmål og fekk programmerarane til å rette på roboten undervegs i programmet, i staden for å stoppe programmet, for så å endre kodar i programmet og deretter køyre programmet på nytt, noko som ein ekte robot ville ha gjort.

Den andre økta

I den andre økta held elevane fram med robotven-oppgåva, og då ser eg ei utfordring som eg trur stammar til forklaringa i forkant av oppgåva. I introduksjonen vert elevane presentert eit døme på korleis dei kan forenkle/komprimere kodane sine i programmet, dette fører til at alle elevane kopierer komprimeringsdømet som vart gitt i introduksjonen. Dette var eigentleg eit ynskje frå meg at elevane skulle finne ut av sjølv, slik at ein kunne få fleire måtar å komprimere på og ut i frå dette få ein diskusjon kring kva for ein av metodane som er enklast og lettast å forstå for roboten.

Ein annan utfordring som eg observerte i den første økta, var lengd på eit steg, altså kor langt skal roboten flytte handa per à (pil i den retninga ein skal flytte). I oppgåva står det heilt klart at ei pil svarar til ei koppeleng. Når koppetårna er låge og lite komplekse var det ingen problem for elevane å finne ut kor langt elevane skulle flytte handa for kvar pil, men når koppetårna vart høgare og meir komplekse vart det vanskelegare. Dette er fordi når ein tel tal steg gjer ein det på grunnlinja av koppestabelen, og dette vert vanskeleg å gjere medan ein skal halde koppen tre, fire eller fem høgder oppe i lause lufta. Dette skapte ein del frustrasjon både hjå robot og programmerarane. Løysinga elevane kom fram til var å halde koppen i nivå med grunnlinja medan dei talte steg, for så å flytte koppen opp så mange steg dei skulle etter at dei hadde flytta stega sidelengs. Dette førte til at program som såg slik ut: ↑↑↑↑→→→→→→→→, vart gjennomført på denne måten av roboten: →→→→→→→↑↑↑↑.

I program-dømet som læraren viste på tavla, var roboten sine bevegelsar for å flytte armen tilbake til koppestabelen inkludert i programmet. Når elevane sette i gang med programmeringa, såg eg at begge gruppene ikkje inkluderte denne funksjonen i programmet sitt. Læraren observerte også dette og stilte då spørsmål om kvifor dei ikkje programmerte roboten til å flytte handa tilbake til koppestabelen. Elevane svarte då at når ei kodelinje var fullført, så måtte roboten tilbake å finne ny kopp uansett. Dette var då med føresetnad om at det ikkje var siste kodelinja. Dei sa også at koppestabelen alltid var plassert på same plass, så å kode roboten tilbake til koppestabelen vart difor overflødig.

Den tredje økta

I den tredje økta vart elevane presenterte for ei ny oppgåve, kalla piksel-programmering. Også denne økta starta med ein introduksjon frå læraren. Læraren gjekk gjennom korleis teknologien i dagens samfunn nyttar pikslar som til dømes ved sending av bilete via mobiltelefonar. Det vart også gjennomgått korleis reglane for piksel-programmeringsoppgåva var. Læraren viste også nokre dømer på korleis eit program vert overført til eit bilete. I eitt av døma valde læraren å vise eit program som gav eit bilete av første bokstaven i namnet has. Når elevane då vart slepte lause til å lage eigne program som dei andre elevane skulle gjere om til bilete, valde dei fleste av elevane å lage program som forma bilete av førebokstaven i namnet deira.

Elevane skulle i første del av oppgåva lage program til dei andre elevane på gruppa. Dette fungerte på den måten at elevane bretta arket sitt med kodefelt og biletfelt, slik at biletfeltet vart skjult for dei andre på gruppa. Når elevane laga program til dei andre elevane, teikna dei først biletet slik at dei kunne lage programmet ut i frå det biletet dei hadde teikna. Når dei då skulle overlevere oppgåva til eleven som skulle løyse den, var biletdelen bretta bakom slik at den eleven ikkje kunne sjå den. Når denne eleven då skulle løyse oppgåva, fann den seg eit blankt kodeark som han teikna biletet ut ifrå koden han skulle løyse. Etter at biletet er ferdig teikna, kan eleven snu rundt arket med den mottekne koden på, og voilà, der er fasiten. Dette fekk elevane på gruppene jobbe med ei stund og dei produserte mange oppgåver til kvarandre.

Siste økt

På slutten av intervensjonen fekk også unplugged-gruppa i oppdrag å halde ein presentasjon som hadde same struktur som for plugged-gruppa, der dei skulle fokusere på; kva som har vore utfordrande, kva dei tykte har vore interessant, kva dei er mest nøgde med/kva dei fekk best til og om det er noko spesielt dei har lært som dei vil vise til dei andre gruppene. I denne

presentasjonsøkta fekk gruppene nokre minuttar på å presentere for dei andre gruppene. Gruppene her hadde generelt kortare presentasjonar med mindre innhald enn gruppene som jobba plugged. Hjå unplugged-gruppene kom det fram at dei tykte det å lese oppskrifter og å bygge koppetårn var vanskeleg. Hjå unplugged-gruppene kom det fram at nokre fekk til alt dei jobba med, og at dei difor tykte det var kjekt. Ei anna gruppe sa at piksel-programmering var kjekt fordi det fekk det til. Medan den neste gruppa sa at piksel-kodane var vanskeleg å lage, og difor ikkje var kjekt å jobbe med. Det var ei felles meining på alle gruppene at det var vanskeleg å lese og utføre lange kodar, dette var generelt for både piksel-oppgåva og koppe-oppgåva.

5 Drøfting

I denne delen av oppgåva vert resultata diskutert og kopla til aktuell teori i lys av forskingsspørsmålet: *Korleis påverkar programmering analogt (unplugged) kontra digitalt (plugged) elevane si evne til algoritmisk tenking og problemløysing i matematikk?* Forskingsspørsmålet vart undersøkt gjennom ein intervensjon som bestod av to elevgrupper, der den eine gruppa programmerte utan datamaskiner (unplugged) medan den andre gruppa nytta datamaskiner (plugged) til å programmere. Dei to gruppene jobba med ulike oppgåver, men hovudinnhaldet i alle oppgåvene var algoritmisk tenking og problemløysing. Intervensjonen starta og avslutta med ein pre-post-test. Det var også to opphald i intervensjonen, der elevane svara på ein survey som handla om kva dei tykte om eiga evne til å tenkje algoritmisk. Heilt på slutten av intervensjonen vart det halde ein elev-presentasjon der elevane fekk legge fram kva utfordringar dei møtte på, kva dei tykte dei hadde lært og korleis dei programma dei hadde produsert fungerte.

I drøftingsdelen vert relevante tematikkar med bakgrunn i dei tre variablane og aktuelle funn frå observasjonane drøfta i saman. Funn både frå plugged-gruppa og unplugged-gruppa vil bli diskutert om kvarandre, for så til sist summere opp desse kvar for seg. Styrkar og svakheiter ved intervensjonen vert også diskutert, dette med tanke på kor replikerbar intervensjonen er for andre.

5.1 Kreativitet

Wing (2006) poengterte at kreativiteten er ein viktig del av det å tenkje algoritmisk. Samanhengen mellom den menneskelege fantasien og teknologi er ein kombinasjon som gjer at menneske i lag med teknologi kan løyse problem som tidlegare ikkje var mogleg å løyse. Denne kombinasjonen som problemløysingsmetode er særs nyttig for elevar i framtida (Wing, 2006). Effektstorleiken som kjem

fram under kategorien kreativitet har ein Cohen`s d på .22, noko som tilseier at elevane kan ha hatt ein liten utvikling innanfor kreativitet gjennom dette prosjektet. Dette utifrå Cohen´s d-skalaen der ein verdi frå .2 til .5 vert sett på som liten effekt. Gjennom intervensjonen kunne ein sjå at elevane nytta kreative tilnærmingar for å løyse ulike problem. Dette kan føre til at dei også aukar kompetansen sin til å tenkje algoritmisk sidan kreativitet er ein av faktorane som vert understreka som viktig i teorien for å tenkje algoritmisk (Wing, 2006).

Wing (2006) peikar på det å sjå på ting frå forskjellige perspektiv, evne til å ikkje avskrive det umoglege og det å kunne legge frå seg eit problem over tid for å late det godgjere seg, som viktige element innanfor algoritmisk tenking som problemløysingsmetode. Det å legge frå seg eit problem, for så å ta det fram att om ei stund, for å sjå om det har kome til nye element som kan hjelpe ein fram til ei løysing, vert sett på som ein viktig del av kreativiteten innanfor algoritmisk tenking (Wing, 2006). I dette prosjektet, var strukturen slik oppgåvene gjekk over fleire økter. Ei oppgåve vart ikkje avslutta etter økta, men vart sett på pause til neste økt starta. Denne pausen mellom øktene, som vara frå eitt døgn til seks døgn (sjå vedlegg 3, plan for intervensjonen), var eit slikt opphald der problemet eller programmeringsoppgåva kunne få godgjere seg i hovudet til elevane til neste økt. Etter godgjeringa kunne det hende elevane hadde fått nye element og nye tankar om løysingsmetoden til oppgåva.

Teorien til Wing (2006) om algoritmisk tenking peikar på samanhengen mellom teknologi og den menneskelege fantasien som essensen i algoritmisk tenking (Wing, 2006). Denne samanhengen er berre til stades i den eine gruppa i dette prosjektet. Unplugged-gruppa manglar det teknologiske aspektet, og vil derfor ikkje møte samanhengen mellom teknologi og fantasi som Wing (2006) peikar på i teorien. Plugged-gruppa har derimot truleg erfart denne samanhengen mellom teknologi og menneskeleg fantasi, då dei nyttar datamaskin til programmering under heile intervensjonen. Dette kan ha ført til at dei har utvikla ulike kunnskapar knytt til algoritmisk tenking.

Den menneskelege fantasien vert også sett på prøve når elevane i unplugged-gruppa skal vere robotar i oppgåva robotven. Elevane tykkjer det er vanskeleg å vere robot, dei klarar ikkje å leggje frå seg dei menneskelege tankerekkene og gå inn i ei rolle som robot. Dette kan det vere eit teikn på mangel av kreativitet hjå unplugged-gruppa. Her ser ein at unplugged-gruppa kanskje vantar både menneskeleg fantasi og teknologi som i høve Wing (2006) er essensen i algoritmisk tenking.

Ein annan observasjon som vart gjort under robotven-oppgåva, for unplugged-gruppa, var at elevane vart presenterte for eit komprimeringsdøme når dei skulle forenkle og effektivisere kodane sine. Dette komprimeringsdømet førte til at begge gruppene nytta dette dømet, og ikkje at dei komprimerte kodane på sin eigen måte, slik at det skapte ein diskusjon i gruppene om kva komprimeringsmetode som var den mest effektive. Dette tykkjer eg også kan vere eit døme på korleis kreativiteten ikkje var heilt til stades hjå unplugged-gruppa. Men grunnane til at elevane ikkje synte kreativitet kan vere mange. Eg tenkjer at ein grunn kan vere korleis opplegget vart strukturert. I unplugged-gruppa, var det læraren som hadde oppgåvene, det var han som gjekk gjennom døma og instruksjonane. Medan på plugged-gruppa, vart oppgåvene delte ut til elevane i papirformat, og informasjonen om oppgåvene frå læraren var minimal. Då kan ein stille seg spørsmålet om det er læraren som hindrar kreativiteten, ved at han syner dømer. Pòlya & Conway (2014) peikar på at forholdet mellom lærar og elev er viktig innanfor problemløysing, og at elevane skal lærast opp til å arbeide mest mogleg sjølvstendig. Om læraren hjelper for mykje, vert det ikkje att noko av problemet til elevane sjølve (Pòlya & Conway, 2014). Dette kan tyde på at elevane sjølv burde ha sett seg inn i oppgåvene på eigenhand, slik at påverknaden frå læraren er minimal. Pòlya & Conway (2014) peikar også på at den hjelpa læraren gir i problemløysingsprosessen, bør vere diskret slik at elevane får ei kjensle av at dei jobbar sjølvstendig sjølv om dei vert hjelpte (Pòlya & Conway, 2014). Ein kan derfor mogleg anta at dette har vore eit hinder for kreativiteten og moglegheita for å vere fantasifulle for unplugged-gruppa. Ein såg også den same tendensen på pikseloppgåva, då læraren synte eit døme på ei oppgåve der han nytta den første bokstaven i namnet sitt til å lage eit pikselbilete. Det førte til at mange av elevane laga pikselbilete der dei nytta fyrste bokstaven i sitt namn, på same måten der læraren nytta sin forbokstav som eksempel for elevane.

Utifrå observasjonane kan ein sjå at rolla til lærarane var ulike i dei to gruppene med tanke på korleis oppgåvene vart presenterte. Dette kan mogleg ha hatt ein innverknad på korleis elevane i dei ulike gruppene opplever eigen kreativitet og korleis den vart stimulert. Det er derfor viktig i arbeidet med algoritmisk tenking og problemløysing at læraren har ei tilbaketrekt rolle, slik at elevane får ei kjensle av at dei løyser problema mest mogleg sjølve, uavhengig om ein har ein plugged- eller unplugged-tilnærming (Pòlya & Conway, 2014).

5.2 Samarbeid

Wing (2016) peikar på at om ein skal nytte algoritmisk tenking som ein metode for å løyse problem, er *samarbeid* viktig. Ein må både nytte andre sine idear og kunne dele sine eigne idear med andre.

Ein må gjennom å snakke om problemet med medelevar kunne lære av andre elevar og dele eigen lærdom med dei (Wing, 2006). Samarbeid er den andre kategorien der ein ser auke i effekten. Med ein Cohen`s d på .66 er det i den kategorien ein ser størst effekt. Gjennom heile prosjektet med unnatak av test og survey, samarbeida elvane med andre elevar. Det som overraskar meg med resultatet kring samarbeid, er at det også i denne kategorien er plugged-gruppa som syner seg som den med mest effekt. Sett utanfrå som ein observatør og med tanke på oppgåvetypane, var det for meg unplugged-gruppa som samarbeidde best og hadde oppgåver som oppfordra mest til samarbeid. Då særskild robotven-oppgåva der programmeraren og roboten utifrå min tankegong må samarbeide tett og godt for at programmet skal fungere slik det er tenkt. Det er også i denne oppgåva lagt til rette for eit samarbeid kring debugging; om roboten utfører programmet feil, må roboten samarbeide meg programmerarane for at programmet skal verte korrekt, slik at oppgåva vert rett utført. Som nemnd tidlegare er det vanskeleg å konkludere på bakgrunn av talmaterialet og dei observasjonane som er gjennomført, sjølv om ein observerer samarbeid i begge gruppene. Ein kan mogleg anta at samarbeidet i begge gruppene har ført til at dei lettare har løyst problema dei vart tildelte.

Udir (2019) peikar på samarbeid og deling er sentrale arbeidsmetodar for den algoritmiske tenkjaren (Udir, 2019b). Innanfor deling som ein del av samarbeid, såg ein stor skilnad på gruppene. På unplugged-gruppene skjedde delinga innad i gruppene, medan på plugged-gruppene vart ferdige og halvferdige program delt mellom gruppene. Det var også ein annan dynamikk i dei to klasseromma med tanke på deling. I unplugged-klasserommet heldt gruppene seg for seg sjølve, og dei delte ikkje noko med dei andre gruppene før i den siste økta då dei heldt presentasjonen sin. Medan i plugged-klasserommet gjekk elevane rundt og mingla med dei andre gruppene for å utveksle og dele både idéar og løysingar. Ein såg også at dei gruppene som snakka mest i lag, hadde størst progresjon. To av plugged-gruppene snakka mykje med kvarandre og hjelpte kvarandre med å peike på skjermen og å dele løysingane sine, medan den andre gruppa var heilt stille. Desse samtalane der elevar deler refleksjonar og erfaringar med kvarandre, kan koplast til det Kilpatrick (2001) kallar resonnering, der elevane blant anna skal reflektere, forklare og vurdere sanningsverdien av ein påstand, eit argument eller eit resultat (Kilpatrick, 2001). I desse delingssituasjonane kan elevane ha opparbeida seg evna til å resonnere ved at dei må forklare kvifor akkurat deira løysing gir det resultatet ein ynskjer. Grunnane til denne skilnaden i samarbeid og deling kan vere mange, men innleiingsvis hadde læraren i plugged-gruppa sagt at dei skulle samarbeide med elevane på si gruppe og ikkje på tvers av gruppene. Dette i tillegg til kanskje dårleg gruppedynamikk kan ha vore grunnen til at progresjonen var mindre på denne gruppa. Grunnen til at læraren ville at elevane berre skulle samarbeide innad på

gruppene, var at han trudde det var slik intervensjonen skulle vere. Han sa i evalueringa etter økta og at han ville ha ei slik organisering for å ikkje skape uro med masse bevegelse og vandring i klasserommet.

Samarbeid og då særskild delingsbiten av samarbeidet kjem ganske klart fram under den siste delen av intervensjonen, der elevane skal presentere det dei har jobba med gjennom heile intervensjonen. Dei skal då dele dei erfaringane dei har gjort seg, utfordringar dei støtte på, ting dei tykkjer dei har lært meir om og også konkrete løysingar på dei oppgåvene dei har jobba med. Wing (2006) peikar på at ein må gjennom å snakke om problemet med medelevar kunne lære av andre elevar og dele eigen lærdom med dei (Wing, 2006). Ei slik presentasjonsøkt er ein fin arena for å fremje denne delinga av lærdom, men ein ser gjennom observasjon at denne delinga og samtalane kring problema og konkrete løysingar, er ein prosess som skjer gjennom heile intervensjonen.

5.3 Logisk tenking

Dette er den siste kategorien der ein ser effekt i resultatdelen. Logisk tenking vart testa på pre- post-testen. Resultatet vart ein Cohen`s d på .64. Noko som kan tyde på at intervensjonen kan ha hatt ein effekt på elevane si evne til logisk tenking. Kilpatrick (2001) koplar logisk tenking til omgrepet r*esonnering*, eller fleksibel tenking handlar om evne til å tenkje logisk, reflektere, forklare og vurdere sanningsverdien av ein påstand, eit argument eller eit resultat. For å få til dette må elevane bygge seg opp ein kunnskapsbase. Dei må også øvast i å grunngje og å tenkje logisk. Læraren må då lage oppgåver dei forstår og som motiverer. Konteksten elevane jobbar i, må vere kjend og trygg. (Kilpatrick, 2001). Elementa med forståelege oppgåver og oppgåver som motiverer kan det sjå ut som i stor grad oppfylt gjennom intervensjonen. Det verka som om elevane forstod kva dei skulle gjere for å løyse oppgåvene dei vart tildelte, sjølv om løysinga og delar av framgangsmåten ikkje var gitt. Motivasjon er eit fagomgrep som eg ikkje skal gå nærare inn på, men med erfaring som lærar ein del år, ser ein vertfall når oppgåver ikkje motiverer. Desse oppgåvene som elevane jobba med under denne intervensjonen, såg ut til å motivere og ikkje det motsette. Ut i frå dette kan eg anta at elevane i begge gruppene har betra sine evner jamfør logisk tenking. Dette ser ein ut i frå tendensen i resultat frå pre- og posttesten samt deira tilbakemeldingar i presentasjonen etter intervensjonen.

Programmering kan vere ein fin metode for å øve seg på å tenkje logisk. Udir (2019b) seier at algoritmisk tenking går ut på å organisere og analysere informasjon på ein logisk måte (Udir, 2019). Når ein programmerer vert denne organiseringa og analysen særs viktig. Ein må som programmerar også tenkje logisk når ein skriv eit program. Kva operasjon skal gjerast først? Kva operasjon skal

gjerast til slutt? Då må programmerarane setje desse operasjonane i ei kronologisk og logisk rekkefølgje. Dette er noko som elevane i begge gruppene fekk prøve seg på i dei oppgåvene dei jobba med under intervensjonen.

Eg gjorde også ein observasjon med tanke på komprimering av program som eg tykte var interessant i unplugged-gruppa. I program-dømet som læraren viste på tavla, var roboten sine bevegelsar for å flytte armen tilbake til koppestabelen inkludert i programmet. Når elevane sette i gang med programmeringa, såg eg at begge gruppene ikkje inkluderte denne funksjonen i programmet sitt. Læraren observerte også dette og stilte då spørsmål om kvifor dei ikkje programmerte roboten til å flytte handa tilbake til koppestabelen. Elevane svarte då at når ei kodelinje var fullført, så måtte roboten tilbake å finne ny kopp uansett. Dette var då med føresetnad om at det ikkje var siste kodelinja. Dei sa også at koppestabelen alltid var plassert på same plass, så å kode roboten tilbake til koppestabelen vart difor overflødig. Slike forenklingar av kodar tykkjer eg er logisk å kople til logisk tenking i teorien kring algoritmisk tenking. Udir (2019b) peikar på at algoritmisk tenking går blant anna ut på organisere og analysere informasjon på ein logisk måte (Udir, 2019b). I dette døme er det akkurat det elevane gjer, dei analyserer informasjonen som er kodane, tek vekk dei overflødige elementa som er det same som å organisere informasjonen på ein logisk måte. Eg tykkjer det er spennande at eg gjer observasjonar på at det kan sjå ut til at elevane vert betre på logisk tenking gjennom prosjektet, og resultata ut i frå effektstorleik gir meg den same indikasjonen.

5.4 Å sjå seg tilbake

I presentasjonsøkta som skjedde heilt som ein avslutning av opplegget, fekk gruppene nokre minuttar på å presentere for dei andre gruppene. Tanken bak presentasjonen var at elevane skulle få forklare kodar og erfaringar, men også som ein oppsummering for det dei har jobba med gjennom heile intervensjonen. Denne øvinga er midt i blinken med tanke på Pòlya (2014) og has siste fase i problemløysingsteorien, *å sjå seg tilbake.* Der problemløysarane skal stille seg spørsmåla; kan du sjekke resultatet? Kan du sjekke argumenta dine? Kan du derivere resultatet på ein annan måte? Og tenk igjennom om du kan nytte resultatet eller metoden til andre problem (Pòlya & Conway, 2014). I teorien er det heilt klart den siste setninga som vert mest aktuell i denne konteksten. Kan elevane knyte noko dei har gjort med dei problema dei har løyst til andre problem? Er metoden noko dei kan nytte når dei dei møter på eit liknande problem, eller kan kanskje resultatet gi dei informasjon når dei skal løyse eit liknande, men nytt problem? Dette er spørsmål som elevane vart oppfordra til å tenkje over og å ta med seg vidare inn i den daglege matematikkundervisninga.

5.5 Uthald

Dette er eit element innanfor teorien til algoritmisk tenking og det elementet Udir (2019b) kallar for kognitiv kondis. Dette er ein av dei elementa der eg gjennom observasjon tykte elevane hadde stor utvikling. I starten av intervensjonen, gjorde elevane på begge gruppene eitt forsøk på ei oppgåve og vart oppgitt og litt leie når ting ikkje verka på første forsøk. Seinare ut i intervensjonen, observerte eg at ein del elevar tykte det var interessant at ting ikkje virka på første forsøk, dei synte ein iver på å finne ut kva som var gale. Dette i motsetning til at dei vart leie og oppgitte i starten av intervensjonen. Ein konkret observasjon her var då ein elev på plugged-gruppa sa: «Oj, desse spela var ulike.» Læraren spør då: «Kvifor er spela ulike når de har nytta same oppskrifta? Korleis kan de finne ut kva som er ulikt?» Dette set i gong ein prosess i gruppa som ein definerer som veldig god. Her må elevane feilsøke for å finne ut om det er noko som er gale med programmet eller kanskje det er programvara? Alle elevane på gruppa er deltakande i denne prosessen, og dei tykkjer at det er merkeleg at spela er vortne så forskjellige. Spelet til den eine eleven står og hakkar, medan spelet til den andre eleven let seg spele fint. Elevane ser gjennom programma til begge elevane, studerer oppskrifta på nytt og diskuterer med kvarandre kva som har skjedd sidan det eine spelet fungerer optimalt medan det andre ikkje er funksjonelt. Her ser ein kanskje at den kognitive kondisen (Udir, 2019b) er styrka, eller at dei har forstått at det å gjere feil er ein viktig del av det å løyse problem. Wing (2006) poengterte kring algoritmisk tenking som problemløysingsstrategi at det å gjere feil er ein del av problemløysinga, men ein må ha ein strategi for å lære av feila sine, slik at ein ikkje gjer dei same feila fleire gangar utan å endre tilnærmingsmetode (Wing, 2006).

Papert (1993) peika på at mange born har ein modell for læring, som går ut på at enten får ein det til, elles får ein det ikkje til. Denne tendensen observerte eg også i starten på intervensjonen i begge gruppene at elevane fort gav opp og vart leie når programmet ikkje gjorde det dei tenkte det skulle gjere, eller at programmet ikkje virka i det heile teke. Papert (1993) meinte at når ein programmerar ei datamaskin, får ein det nesten aldri til på fyrste forsøk. Fokuset vert då ikkje på om ein får det til eller ikkje får det til, men ved å isolere og rette opp i «bugs» får ein eit fokus som ikkje går ut på om programmet er rett eller feil, men på om det er mogleg å fikse (Papert, 1993). Dette fekk elevane oppleve og prøve seg på gjennom intervensjonen. Dei opparbeida seg ein vilje til å få programmet til å virke, og det vart dette som vart fokuset i staden for å fokusere på at det ikkje virka. Papert (1993) meinte vidare at arbeid med debugging kan endre kulturen for læring, slik at ein ikkje engstar seg for mykje for å gjere feil (Papert, 1993, s. 23). Det verka som at dette var noko som elevane lærte å akseptere gjennom intervensjonen, dei gjekk frå å vere irriterte over at ting ikkje virka, til å vere

ivrige etter å finne ut kva som var gale. Læraren sin rolle i dette arbeidet var også viktig, då dei støtta elevane sin vilje til å rette feilen i staden for å fokusere på at programmet ikkje virka. Som Pòlya & Conway (2014) peikar på, må læraren setje seg inn i elvane sin situasjon, gå inn i tankesettet til elevane slik at spørsmåla ein stiler som lærar reflekterer situasjonar eller steg som kunne ha skjedd med elven sjølv (Pòlya & Conway, 2014). Gjennom intervensjonen kom lærarane med gode spørsmål og forslag slik at elevane fekk passeleg støtte til å greie å løyse oppgåvene på eiga hand. Det at elevane fekk prøve seg på debugging og også at lærarane var flinke til å støtte elevane når dei møtte på problem, kan truleg ha ført til at elevane har fått betre uthald eller kognitiv kondis som er eit viktig element innanfor algoritmisk tenking (Udir, 2019). Papert (1993) peikar også på at om ein jobbar med programmering over tid, forventar ein ikkje at noko skal fungere på fyrste forsøk. Ein vert meir fokusert på korleis ein kan fikse det. Ein skaffar seg uthald og vert meir løysningsorientert (Papert, 1993). Dette også understøttar at elevane auka uthaldet og vart løysingsorienterte då dei aksepterte at problema dei støtte på under intervensjonen ikkje vart løyste på fyrste forsøk.

5.6 Unplugged kontra plugged

Som tidlegare nemnt peikar teorien til Wing (2006) om algoritmisk tenking på samanhengen mellom teknologi og den menneskelege fantasien som essensen i algoritmisk tenking (Wing, 2006). I unplugged-gruppa er elementet teknologi heilt fråverande, og då er det berre den menneskelege fantasien som er att. Korleis dette påverkar elevane med tanke på problemløysing og algoritmisk tenking har ikkje dette prosjektet kome med eit konkret svar på, men den gruppa har ikkje merkbar auke i nokon av dei måleelementa som vert gjort innanfor algoritmisk tenking. Så det kan sjå ut som at programmering utan datamaskiner ikkje gjorde elevane i dette prosjektet til betre algoritmiske tenkjarar. Ein ser vidare at tendensane særskild i dei talfesta resultata, er at det er klart mest framgang hjå plugged-gruppa som hadde denne samanhengen mellom menneskeleg fantasi og teknologi.

Disessa (2018) peikar på c*omputational literacy* som er å verte kjent med dei dataverktøya som ein vel å bruke, og kople dei til matematiske tema på ein slik måte at det gir elevane ein ny dimensjon innanfor det gitte emnet. Slik at det aukar deira forståing, og dei får ei forståing som den vanlege matematikkundervisninga ikkje kan gi dei (Disessa, 2018, s.22). Denne koplinga er det i dette prosjektet også berre gruppa som jobbar plugged som får kjennskap til. Det kan sjå ut som denne gruppa har ein fordel, då dei bygger seg opp ein kompetanse i det verktøyet dei nyttar gjennom intervensjonen, Scratch. Gjennom denne kjennskapen til dette verktøyet kan det hende dei får ei auka forståing kring dei emna dei jobbar innanfor, som er algoritmisk tenking og problemløysing.

I presentasjonsdelen tykkjer eg som observatør at eg ser den største skilnaden på dei to gruppene. Her kjem det tydeleg fram at plugged-gruppa har ein fordel, då dei hadde meir handfast å sjå tilbake på og vise. Dei kunne hente fram at dei konkrete programma sine og dra og flytte på blokker for å syne ulike fasar dei hadde vore gjennom i problemløysingsprosessen. Det verka som det vart lettare for dei å sjå seg tilbake og å dele. Presentasjonane på desse gruppene vart difor betre strukturerte og meir konkrete enn det som var tilfelle på unplugged-gruppene. Dette syner kanskje også ein fordel for unplugged gruppa kring Pòlya (2014) og has siste fase i problemløysingsteorien, *å sjå seg tilbake.* Der problemløysarane skal stille seg spørsmåla; kan du sjekke resultatet? Kan du sjekke argumenta dine? Kan du derivere resultatet på ein annan måte? Og tenk igjennom om du kan nytte resultatet eller metoden til andre problem (Pòlya & Conway, 2014). I dette prosjektet og slik det er strukturert, kan det sjå ut som om at elevane i plugged-gruppa kan lettare sjå seg tilbake og stille seg desse sentrale spørsmåla som kjem fram i Pòlya sin teori.

Presentasjonane hjå unplugged-gruppene var generelt kortare og med mindre innhald enn gruppene som jobba plugged. Hjå unplugged-gruppene var mange av presentasjonane fokusert på element dei tykte var vanskeleg, utfordrande eller element dei tykte dei hadde fått til. Sidan unplugged-gruppene hadde jobba ein del med blyant og papir, var det ikkje mange av gruppene som synte konkrete døme under presentasjonen, då desse arka ikkje vart skikkeleg teke vare på og kunne nyttast til presentasjonen. Hjå unplugged-gruppene kom det fram at dei tykte det å lese oppskrifter og å bygge koppetårn var vanskeleg. Det kom vidare fram i unplugged-gruppene at nokre fekk til alt dei jobba med, og at dei difor tykte det var kjekt. Ei anna av unplugged-gruppene sa at piksel-programmering var kjekt fordi det fekk det til. Medan den neste gruppa sa at piksel-kodane var vanskeleg å lage, og difor ikkje var kjekt å jobbe med. Det var ei felles meining på alle gruppene at det var vanskeleg å lese og utføre lange kodar, dette var generelt for både piksel-oppgåva og koppe-oppgåva. Dette kan vere ein indikasjon på at oppgåvene som unplugged-gruppa løyste var vanskelegare enn dei oppgåvene som plugged-gruppa løyste. Dette går ikkje direkte på skilnaden mellom unplugged og plugged, men det kan kanskje forklare litt kvifor dei ikkje synte den same framgangen. Dette går på det som Kilpatrick (2001) peika på når han koplar logisk tenking til omgrepet r*esonnering*, der han meiner at for at elevane skal øve seg i å tenkje logisk, må læraren lage oppgåver som elevane forstår og som motiverer (Kilpatrick, 2001). Det kan vere at i det aktuelle tilfellet i unplugged-gruppa var ikkje oppgåvene forståelege nok slik at dei motiverte elevane og på den måten hindra dei i å ha framgang kring logisk tenking.

6 Avslutning

I denne siste delen av oppgåva vil eg summere opp hovudfunna i dette prosjektet, både ut ifrå talmaterialet og observasjonane som er gjort gjennom intervensjonen. Avslutningsvis vil eg diskutere på styrker og svakheiter ved prosjektet.

6.1 Oppsummering

I løpet av ein skulekvardag er det mange ulike ting som skjer. Elevar forsvinn til tannlege, elevråd eller er fråverande grunna sjukdom. Plutseleg kjem ungdomsavdelinga innom for å informere om kva som skjer i vinterferien og plutseleg kjem det ein kulturell skulesekk dalande eller ein forfattar som vil fortelje om den nye boka si. Gjennomføring av eksperiment i ein skuleklasse kan planleggast etter punkt og prikke, men ein kan aldri få oversikt over alle dei uventa tinga som kan dukke opp i løpet av ein skuledag. Difor bør ei gjennomføring av ein intervensjon over eit slikt tidsrom som er tenkt i dette høvet, vere dynamisk slik at ein tek høgde for at økter må flyttast og personane som er involverte må vere fleksible og løysingsorienterte.

Ein intervensjon med ei så lite utval som denne, gjer at ein ikkje kan konkludere kring dei resultata som kjem fram. Når intervensjonen er så liten og resultata difor ikkje kan konkluderast med, har eg i etterkant tenkt på at dette prosjektet og då særskild intervensjonen kan sjåast på som ein grundig pilot. Eg har gjennom dette prosjektet gjort meg mange tankar om korleis dette kan forbetrast og gjerast på ny i ein større skala. Likevel vil eg trekkje fram nokre tal som kanskje syner nokre tendensar. Tendensane eg ser ut i frå datamaterialet er at den framgangen som skjer innanfor problemløysing og algoritmisk tenking, skjer i plugged-gruppa. I den gruppa ser ein auke i kreativitet, samarbeid og logisk tenking. Dette er som nemnd tidlegare viktige element innanfor algoritmisk tenking og problemløysing. Eg tykkjer det er interessant at gruppa som programmerer plugged er den gruppa som syner mest framgang. Ut i frå resultata som ligg til grunn både i talmaterialet og i observasjonane kan det sjå ut som om at datamaskina er ein viktig faktor med tanke på å verte ein betre algoritmisk tenkjar, og difor ein betre problemløysar.

I innleiinga av oppgåva vart det teke opp kva kompetansar elevane treng når dei skal ut i arbeidslivet og samfunnet. Dette er ein av argumenta som vart nytta for at programmering skal inkluderast i skulen. Elevane skal gjennom programmeringa skaffe seg nødvendige ferdigheiter for framtida i næringslivet og ei evne til å forstå korleis eit meir og meir digitalisert samfunn fungerer. Om denne kompetansen vert utvikla gjennom eit slikt intensivt programmeringskurs som er gjennomført i

denne intervensjonen er heller usikkert, men elevane har utan tvil fått ei innsikt i kva programmering er og korleis det kan nyttast i kvardagen, som pikseloppgåva der dei får innsikt i korleis telefonane deira prosesserer bilete. Programmering er implementert i den nye læreplanen og alle lærarar i dei faga som inkluderer programmering bør vere i gang med det. Gjennom programmering skaffar dei seg kompetanse kring algoritmisk tenking, problemløysing og andre ferdigheiter som dei treng når dei skal ut i arbeidslivet om nokre år. Om denne programmeringa skal skje digitalt på ei datamaskin eller med blyant på eit ark er eg framleis usikker på, men sjølv om resultata i denne oppgåva ikkje kan konkluderast ut i frå, trur eg datamaskina har ein plass i denne prosessen.

Sjølv om taldata frå intervensjonen ikkje gir klare statistiske resultat, er dette eit prosjekt som har gitt meg mykje som lærar. Det å få lov til å vere ein forskande lærar i konkrete klasseromssituasjonar har sett i gong mange tankeprosessar hjå meg. Praksisdelen av læraryrket er særs viktig, og det å få moglegheita gjennom dette prosjektet å kople forsking til skulepraksis er noko som har gitt meg mange nye tankar som kjem til å gagne elevane i framtida.

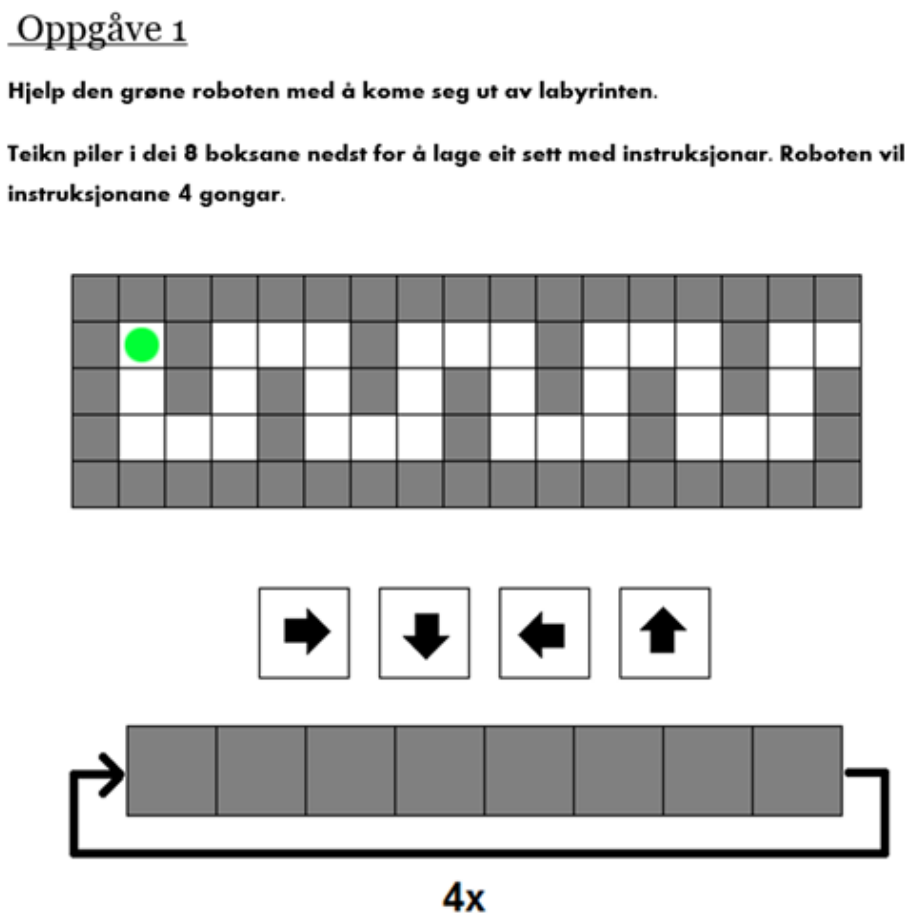
6.2 Styrker og svakheiter

Den største styrken med dette prosjektet, er etter mi meining at det er laga eit godt undervisningsopplegg som testar utviklinga til elevane innanfor algoritmisk tenking og problemløysing gjennom intensiv programmering. Det er også ein styrke at eg har nytta reelle klasseromssituasjonar, noko som gjer settinga autentisk for elevane og er bra med tanke på validitet. Sidan det er så få deltakarar i dette prosjektet er det vanskeleg å gjere nøyaktige statistiske analysar då det vert store feilmarginar og høg spreiing. Det vert også vanskeleg å nytte signifikanstesting for å løfte fram systematikk i observasjonane. Om nokon då replikerer dette ein gong i framtida med meir større utval og meir data, vil ein kanskje sjå resultat som er klårare og meir eintydige. Ein kan også sjå om dei tendensane som eg finn i mitt prosjekt er reelle.

Ein faktor eg tykkjer er ein svakheit med min intervensjon, er at dei to gruppene jobbar med ulike emne innanfor matematikken og ulike metodar innanfor problemløysing. Plugged-gruppa jobbar mykje med å følgje ei oppskrift eller eit sett med instruksjonar. Unplugged-gruppa vert meir utfordra innanfor problemløysing, då dei ikkje har nokon oppskrift dei skal følgje, dei har berre reglar og avgrensingar dei skal halde seg til. På denne måten er det vanskeleg for meg å finne ut om det er faktoren datamaskin eller ikkje som gir meg dei resultata eg får. Når dei to gruppene jobbar med så ulike oppgåver er det mange andre faktorar som kan påverke resultata. Det er ikkje berre oppgåvene som er ulike, men også korleis dei tek vare på det arbeidet dei har gjort gjennom intervensjonen.

Plugged-gruppa har alle program dei har jobba med lagra på datamaskinene sine, slik at dei kan lett ta dei fram at og diskutere løysingar og strategiar med andre. Dette kom tydeleg fram når dei skulle halde presentasjonen på slutten av intervensjonen. Medan unplugged-gruppa hadde mange ark som dei jobba med, og desse vart ikkje strukturerte og teke vare på slik at dei kunne nytte desse når dei hadde framføring.

Eg ser også eit problem, men også ein styrke ved at det ikkje vart stilt nokre spørsmål til påstandane i surveyen. Det vart heller ikkje stilt spørsmål til nokon av oppgåvene i pre- post-testen. Er grunnen til dette at eg har forma ein heilt perfekt survey og ein heilt perfekt test, eller sit elevane med mange spørsmål kring desse? Ein indikator som kan gi meg eit svar på dette, er den første oppgåva på pre-post-testen, der det ikkje er definert at ei pil svarar til eitt steg, dette ser eg ut i frå svara at mest truleg nokre elevar har misoppfatta (sjå figur 13 under). Dette kan vere eit teikn på at verken testen eller surveyen er perfekte, og at elevane sit att med mange spørsmål som dei ikkje har fått svar på.



Figur 11 Oppgåve 1 på pre-post-testen

Det som kunne vore gjort annleis er at eg kunne ha pilotert alle delane av intervensjonen. Då særleg pre- post-testen og pre- post-survey, der eg også hadde pilotert analyseprosessen med eit piloteringsmateriale. Dette ville ha styrka intervensjonen ved at eg kunne ha forbetra testen og surveyen. Grunnen til at eg ikkje gjennomførte pilotering er rett og slett mangel på tid, då eg jobbar fullt som lærar, og det gjer også dei lærarane som var så hyggjelege at dei slepte meg til i sin matematikklasse. Det kom også ein pandemi oppi det heile, som gjorde at opplegg der samarbeid var essensielt, vart ein stor utfordring. Dersom ein ser på lengda på intervensjonen samt før- og etterarbeid, ville det vore umogleg å pilotert intervensjonen for ein stakkars masterstudent/fulltids-lærar.

Ein annan svakheit ved opplegget er organiseringsvalet eg tok ved å la dei to gruppene gjennomføre intervensjonen simultant. Då fekk eg problem med å observere heile intervensjonen på dei to gruppene, eg måtte bevege meg i mellom dei to gruppene og eg mista ein del observasjonar. Tett dialog med dei to lærarane som gjennomførte intervensjonen gjorde til at eg fekk deira observasjonar og deira subjektive syn på dei situasjonane som oppstod. Dette er heilt klart ein svakheit med intervensjonen, men det var noko eg visste om på førehand. Det var heller ikkje noko kring denne problematikken som kunne løysast på ein enkel måte, då lærarane som gjennomførte intervensjonen har veldig strikte timeplanar og å legge intervensjonsgruppene sine gjennomføringar til to ulike tider var veldig vanskeleg, om ikkje umogleg.

Forskingsspørsmålet i prosjektet er som tidlegare nemnt kausal, der eg skulle finne ut om programmering plugged eller unplugged (årsak) ville utvikle elevane sine evner innanfor algoritmisk tenking og problemløysing (effekt). Variabelen i dette prosjektet er årsaka og fenomenet er effekten. Svakheitene innanfor kausaliteten i dette prosjektet kan vere at eg som forskar ikkje kan utelukka systematiske skilnader mellom gruppene som var til stades før intervensjonen.

Som sagt kan ein sjå på dette prosjektet som ein pilot, og difor ein open invitasjon til alle fantastiske forskarar der ute som vil teste intervensjonen i større skala. Eg bidreg meir enn gjerne med min formidable kunnskap om programmering, algoritmisk tenking og matematikk i skulen.

7 Litteraturliste

Befring, E. (2014). Kvantitativ metode. Henta frå: Kvantitativ metode | Forskningsetikk

Bueie, H. (2019). *Programmering for matematikklærere*. Oslo: Universitetsforlaget.

Bundsgaard, J. (2019). "Computational thinking in compulsory education: a survey study on initiatives and conceptions." Educational Technology, Research and Development: 1-23.

Cohen, J. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences.* New York: Routledge.

Creswell, J. W. & Guetterman, T. C. (2019). *Educational research : planning, conducting, and evaluating quantitative and qualitative research* (6th edition. utg.). Saddle River, New Jersey: Pearson.

Dagienė, V. & Stupuriene, G. (2016). Bebras - a Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education An International Journal, 15*(1), 25-44. https://doi.org/10.15388/infedu.2016.02

Disessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. Mathematical Thinking and Learning 20(1): 3-31. DOI: 10.1080/10986065.2018.1403544

Forsström, S. E. & Kaufmann, O. T. (2018). A systematic litterature review exploring the use of programming in mathematics education. Østfold University Collage, Faculty of Education, Norway, Østfold.

Grevholm, B. (Red.). (2003). *Matematikk for skolen*. Bergen: Fagbokforlaget.

Grønmo, S. (2004). Samfunnsvitenskapelige metoder. Bergen: Fagbokforlaget.

Hjelle, G. A. Soloball. Henta frå https://oppgaver.kidsakoder.no/scratch/soloball/soloball

Hsu, T.-C., et al. (2018). "How to learn and how to teach computational thinking: Suggestions based on a review of the literature." Computers & Education **126**: 296-310.

Høgheim, S. (2020). *Masteroppgaven i GLU* (1. utgave. utg.). Bergen: Fagbokforlaget.

Israel, M., et al. (2015). "Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis." Computers & Education **82**(C): 263-279.

Kalelioğlu, F. and Y. Gülbahar (2019). "A Holistic Approach for Computer Science Education in Secondary Schools." Informatics in Education **18**(1): 131-150.

Kilpatrick, J., Swafford, J. & Findell, B. (2001). Adding it up. Helping children learn Mathematics. Washington D.C.: National Academic Press.

Kleven, T. A. & Hjardemaal, F. (2018). *Innføring i pedagogisk forskningsmetode : en hjelp til kritisk tolking og vurdering* (3. utg. utg.). Bergen: Fagbokforl.

Lund, T. (2002). *Innføring i forskningsmetodologi*. Oslo: Unipub.

Lund, T., Fønnebø, B. & Haugen, R. (2006). *Forskningsprosessen*. Oslo: Unipub.

Lynnebakken, H. (2018, 11. oktober). Programmering er ikke det samme som koding: - Den største jobben er å finne ut hva du ønsker å gjøre. Henta frå: Programmering er ikke det samme som koding: – Den største jobben er å finne ut hva du ønsker å gjøre - Digi.no

Norsk samfunnsvitenskaplige datateneste. (2021). Henta frå: Samtykke og andre behandlingsgrunnlag | NSD

NOU 2014:7. (2014). *Elevenes læring i fremtidens skole - Et kunnskapsgrunnlag.* Henta frå NOU 2014: 7 - regjeringen.no

Papert, S. (1993). *Mindstorms : children, computers, and powerful ideas* (2nd ed. utg.). New York: Basic Books.

Pólya, G. & Conway, J. H. (2014). *How to solve it : a new aspect of mathematical method*. Princeton, NJ: Princeton University Press.

Thinkersmith. [Mine robotvenner]. Henta frå https://oppgaver.kidsakoder.no/uten_datamaskin/robotvenner/robotvenner

Utdanningsdirektoratet. (2019a, 13. februar). Hva er nytt i matematikk? (udir.no)

Utdanningsdirektoratet. (2019b, 27. mars). Algoritmisk tenkning. Henta frå https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2020). *Læreplan i matematikk (MAT01-05)*. Henta frå Kjerneelementer - Læreplan i matematikk 1.–10. trinn (MAT01-05) (udir.no)

Utdanningsdirektoratet. (2020). *Læreplan i matematikk (MAT01-05)*. Henta frå Fagets relevans og sentrale verdier - Læreplan i matematikk 1.–10. trinn (MAT01-05) (udir.no)

Utdanningsdirektoratet. (2020). *Læreplan i matematikk (MAT01-05)*. Henta frå Grunnleggende ferdigheter - Læreplan i matematikk 1.–10. trinn (MAT01-05) (udir.no)

Utdanningsdirektoratet. (2020). *Læreplan i matematikk (MAT01-05)*. Henta frå Kompetansemål etter 9. trinn - Læreplan i matematikk 1.–10. trinn (MAT01-05) (udir.no)

Wilson, David B. (2018). Practical Meta-Analysis Effect Size Calculator. Henta frå: https://www.campbellcollaboration.org/escalc/html/EffectSizeCalculator-SMD25.php

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing.

8 Vedlegg

Vedlegg 1

Samtykkeskjema

Vil du delta i forskingsprosjektet

Problemløysing, algoritmisk tenking og programmering i matematikk?

Dette er eit spørsmål til deg om å delta i eit forskingsprosjekt der formålet er å kartlegge elevane sin kompetanse kring problemløysing, algoritmisk tenking og programmering i matematikk. I dette skrivet får de informasjon om måla for prosjektet og kva deltakinga inneber for deg.

Formål
Prosjektet vil kartlegge elevane sin kompetanse i emna problemløysing, algoritmisk tenking og programmering. Prosjektet vil gå over to veker, der alle matematikktimar vert nytta til dette. Den totale varigheita til prosjektet vil difor vere 280 minuttar. Dette er eit studentprosjekt i form av ei masteroppgåve.

Kven er ansvarleg for forskingsprosjektet?
Høgskulen på Vestlandet er ansvarleg for prosjektet.

Kvifor får du spørsmål om å delta?
Det vert gjennomført eit undervisningsopplegg i klassen i matematikkfaget med varigheit på to veker. Elevane skal gjennom ei spørjegransking og ein før- og etter-test som skal kartlegge delar av elevane sin kompetanse i matematikk.

Kva inneber det for deg å delta?
Før og etter prosjektet skal elevane gjennomføre ein test. I tillegg skal elevane svare på eit elektronisk spørjeskjema.

- *Viss eleven vel å delta, inneber det at han/ho er deltakar i eit undervisningsopplegg innanfor tema matematikk, programmering, problemløysing og algoritmisk tenking. Varigheita på undervisningsopplegget er alle matematikktimane over to veker, som er til saman 280 minuttar.*
- *Viss eleven vel å delta i prosjektet, inneber det at han/ho fyller ut eit spørjeskjema. Det vil ta om lag 10 minuttar. Spørjeskjemaet inneheld spørsmål om problemløysing og matematikk. Svara frå spørjeskjemaet vert lagra elektronisk.*

Om nokon vil ha innsyn i testen eller det elektroniske spørjeskjemaet, er det berre å ta kontakt.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Dersom du vel å delta, kan du når som helst trekke samtykket

tilbake utan å gje nokon grunn. Alle dine personopplysningar vil då verte sletta. Det vil ikkje ha nokon negative konsekvensar for deg dersom du ikkje vil delta eller seinare vel å trekkje deg. Det vil ikkje ha nokon negative konsekvensar for forholdet mellom skulen og heimen eller mellom lærar og eleven om ein ikkje ynskjer å delta i forskingsprosjektet.

Om du vel å ikkje delta i prosjektet, vil du få ordinær matematikkundervisning i den perioden prosjektet går føre seg. Denne undervisninga vil ha same tematikken som ligg til grunn for intervensjonen; problemløysing, algoritmisk tenking og programmering i matematikk.

Ditt personvern – korleis me oppbevarer og brukar dine opplysningar

Vi vil berre nytte opplysningane om deg til formåla vi har fortalt om i dette skrivet. Vi behandlar opplysningane konfidensielt og i samsvar med personvernregelverket.

- *Det er berre student og rettleiar som har innsikt i opplysningane.*
- *Spørjeskjemaet vert laga i programmet SurveyXact og data vert analysert i programmet Jasp.*

Deltakarane vert ikkje mogleg å kjenne att når prosjektet vert publisert.

Kva skjer med opplysningane dine når vi avsluttar forskingsprosjektet?

Opplysningane vert anonymiserte når prosjektet avsluttast/oppgåva er godkjend, noko som etter planen er om lag 6. juli 2021.

Dine rettigheiter

Så lenge du kan identifiserast i datamaterialet, har du rett til:

- innsyn i kva personopplysningar som er registrert om deg, og å få utlevert ein kopi av opplysningane,
- å få retta personopplysningar om deg,
- å få sletta personopplysningar om deg, og
- å sende klage til Datatilsynet om behandlinga av dine personopplysningar.

Kva gir oss rett til å behandle personopplysningar om deg?

Vi behandlar opplysningar om deg basert på ditt samtykke.

På oppdrag frå *Høgskulen på Vestlandet* har NSD – Norsk senter for forskningsdata AS vurdert at behandlinga av personopplysningar i dette prosjektet er i samsvar med personvernregelverket.

Kvar kan eg finne ut meir?

Dersom du har spørsmål til studien, eller ynskjer å nytte deg av dine rettigheiter, ta kontakt med:

- *Høgskulen på Vestlandet ved Ole Einar Rebni*
  Mail: ole.einar.rebni@sogndal.kommune.no
  Telefon: 99279360
  Eller
  Høgskulen på Vestlandet ved Sigve Høgheim
  Mail: sigve.hogheim@hvl.no
  Telefon: 57676049
- Vårt personvernombud:
  Trine Anikken Larsen
  Telefon: 55587682
  Mail: trine.anikken.larsen@hvl.no

Dersom du har spørsmål knytta til NSD sin vurdering av prosjektet, kan du ta kontakt med:

- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennleg helsing

Prosjektansvarleg	*Student*
Sigve Høgheim	Ole Einar Rebni

Samtykkjeerklæring

Eg har motteke og forstått informasjon om prosjektet *Problemløysing, algoritmisk tenking og programmering i matematikk*, og har fått høve til å stille spørsmål. Eg samtykkjer til:

- ☐ å delta i *undervisningsopplegget*
- ☐ å delta i *spørjeundersøkinga*
- ☐ å delta i *før- og etter-testen*

Eg samtykkjer til at mine opplysningar vert behandla fram til prosjektet er avslutta

---

(Signert av prosjektdeltakar, dato)

Vedlegg 2

Godkjenning NSD

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Algoritmisk tenking, problemløysing og programmering i matematikk

Referansenummer

583770

Registrert

10.09.2020 av Ole Einar Rebni - 051211@stud.hvl.no

Behandlingsansvarlig institusjon

Høgskulen på Vestlandet / Fakultet for lærerutdanning, kultur og idrett / Institutt for pedagogikk, religion og samfunnsfag

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Sigve Høgheim, Sigve.hogheim@hvl.no, tlf: 4757676049

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Ole Einar Rebni, ole.einar.rebni@gmail.com, tlf: 99279360

Prosjektperiode

01.06.2020 - 06.07.2021

Status

19.10.2020 - Vurdert

Vurdering (1)

19.10.2020 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet 19.10.2020 med vedlegg, samt i meldingsdialogen mellom innmelder og NSD.

Behandlingen kan starte.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å

lese om hvilke type endringer det er nødvendig å melde:
https://nsd.no/personvernombud/meld_prosjekt/meld_endringer.html

Du må vente på svar fra NSD før endringen gjennomføres.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 06.07.2021.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra foresatte til behandlingen av personopplysninger om elevene. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som foresatte kan trekke tilbake. Barna/elevene vil også samtykke til deltakelse.

Lovlig grunnlag for behandlingen vil dermed være foresattes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
NSD vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:

- lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke viderebehandles til nye uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: åpenhet (art. 12), informasjon (art. 13), innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), underretning (art. 19), dataportabilitet (art. 20).

NSD vurderer at informasjonen som de registrerte og deres foresatte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert/foresatt tar kontakt om sine/barnets rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

FØLG DIN INSTITUSJONS RETNINGSLINJER
NSD legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

Jasp og SurveyXact er databehandlere i prosjektet. NSD legger til grunn at behandlingen oppfyller kravene til bruk av databehandler, jf. art 28 og 29.

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og eventuelt rådføre dere med behandlingsansvarlig institusjon.

OPPFØLGING AV PROSJEKTET
NSD vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Kontaktperson hos NSD: Gry Henriksen
Tlf. Personverntjenester: 55 58 21 17 (tast 1)

Vedlegg 3

Plan for intervensjonen

Før intervensjonen startar	- Deler gruppa i to like store og randomiserte grupper - Gruppene vert så inndelte i grupper på 3-4 elevar - Plugged-gruppa lagar seg brukarar på www.scratch.mit.edu - Alt materiale til unplugged-gruppa er printa ut og klargjort - Gjennomført samtale med lærarane, slik at dei veit kva som skal gjerast.
1. økt onsdag 100 min. 08:20 -10:00	Plugged - Pre test individuelt på ark - Jobbar med oppgåva i gruppa https://oppgaver.kidsakoder.no/scratch/soloball/soloball - Spørjeskjema på slutten av økta
	Unplugged - Pre test individuelt på ark - Jobbar med oppgåva i gruppa https://oppgaver.kidsakoder.no/uten_datamaskin/robotvenner/robotvenner - Spørjeskjema på slutten av økta
2. økt torsdag 40 min. 08:50-09:30	Plugged - Held fram med soloball-oppgåva om dei ikkje er ferdige - Byrjar på terningoppgåva om dei er ferdige med soloball
	Unplugged - Held fram med robotven-oppgåva - Startar på «tende» og «sløkte»-oppgåva om dei er ferdige med førre oppgåve
3. økt onsdag 100 min.	Plugged - Held fram med terningoppgåva - Øver seg til presentasjon av oppgåvene dei har løyst - Spørjeskjema på slutten av økta
	Unplugged - Held fram og gjer seg ferdige med «tende» og «sløkte» oppgåva - Spørjeskjema på slutten av økta
4. økt torsdag 40 min.	Plugged - Presenterer oppgåvene for dei andre gruppene - Gjennomfører post-testen
	Unplugged - Presenterer oppgåvene for dei andre gruppene - Gjennomfører post-testen

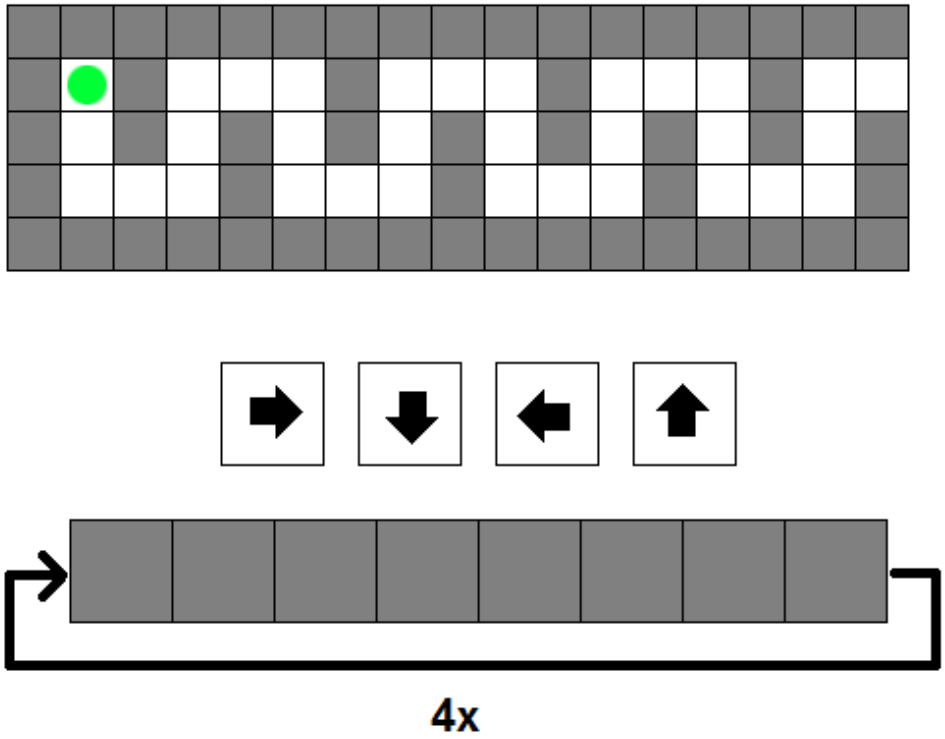
Vedlegg 4

Pre- og post-test

Oppgåve 1

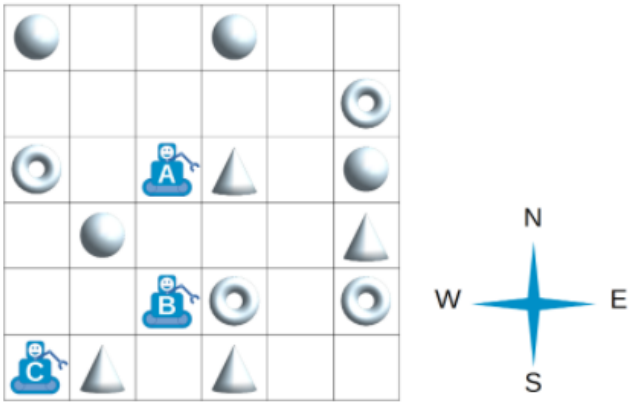
Hjelp den grøne roboten med å kome seg ut av labyrinten.

Teikn piler i dei 8 boksane nedst for å lage eit sett med instruksjonar. Roboten vil gjenta desse instruksjonane 4 gongar.



Oppgåve 2

På eit lager er det tre robotar som alltid jobbar som eit lag.
Når laget får ein retningsinstruksjon (N, S, E, W) vil alle robotane flytte seg i den gitte retninga, ei og ei rute i rutenettet samtidig.
Etter å ha motteke lista med instruksjonar vil robotane plukke opp objektet som er i den siste ruta.
Døme: Om me gir laget lista N, N, S, S, E vil robot A plukke opp ei kjegle, robot B vil plukke opp ein ring, og robot C vil plukke opp ei kjegle.



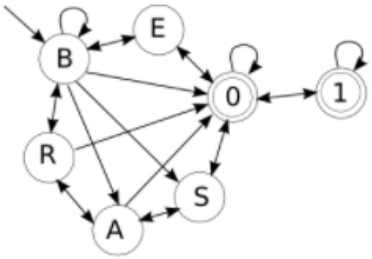
Oppgåve: Kva liste med instruksjonar kan me gi laget slik at dei plukkar opp ei kule, ei kjegle og ein ring? Set ring rundt den rette liste

N, E, E, E | N, E, E, S, E | N, N, S, E, N | N, E, E, S, W

Oppgåve 3

Beverane bygger flåtar. Ved trafikkontroll må alle flåtar vere registrerte. Dette betyr at kvar flåte skal ha eit nummerskilt med unik tekst.

Nummerskiltet må starte med bokstaven B og slutte med talet 0 eller 1.
Bruk figuren under til å finne ut kva nummerskilt som er gyldige. Følg pilene for å lage nummerskilt. 0 og 1 har to sirklar rundt seg, det betyr at ein kan velge om ein skal avslutte teksten eller fortsette



Oppgåve: To av desse skilta kan ikkje verte registrerte, set ring rundt dei to.

BB0001	BBB100	BBB011	BB0100
BR00A0	BSA001	BE0S01	

Oppgåve 4

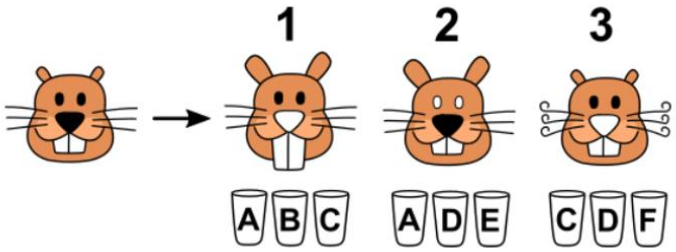
Beveren Bertil har oppdaga fem trylledrikkar.
Ein gjer øyrene lengre.
Ein annan gjer tennene lengre.
Ein annan gjer værhåra krøllete.
Ein annan gjer nasen kvit.
Den siste gjer augene kvite.
Bertil fyller ein av kvar trylledrikk i ein eigen kopp. Han fyller ein annan kopp med berre vatn, og har no seks koppar totalt. Koppane er merka frå A til F. Problemet er at han har gløymt kva trylledrikk som er i dei ulike koppane.

For å finne ut kva trylledrikk som er i dei ulike koppane, set Bertil opp følgande eksperiment:

Eksperiment 1: Ein bever drikk frå kopp A, B og C samtidig – og effekten er som vist på bilde 1

Eksperiment 2: Ein bever drikk frå kopp A, D og E samtidig – og effekten er som vist på bilde 2

Eksperiment 3: Ein bever drikk frå kopp C, D og F samtidig – og effekten er som vist på bilde 3



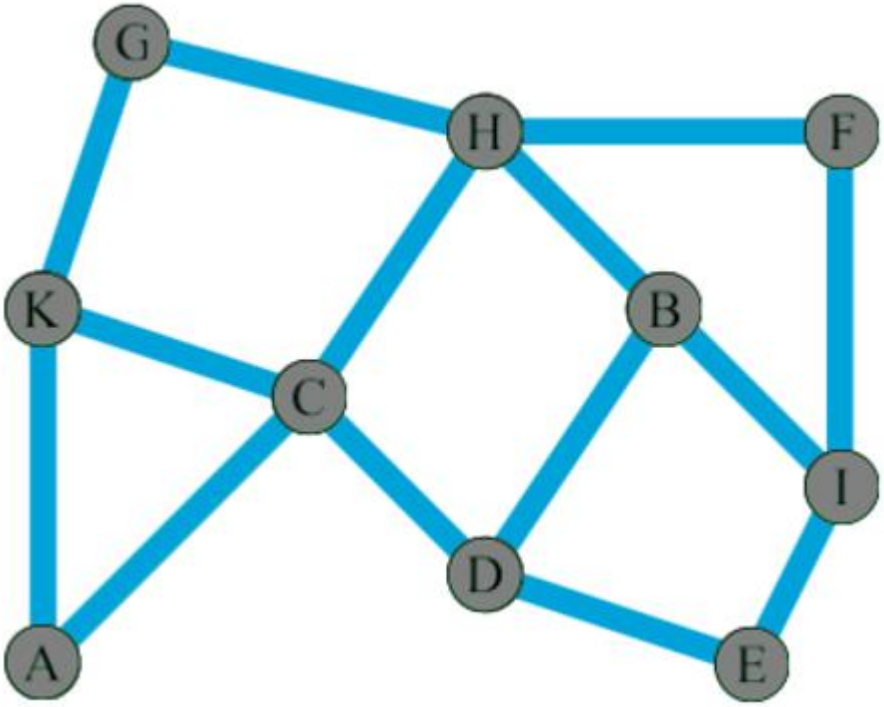
Oppgåve: Korleis kopp inneheld reint vatn? Set ring rundt riktig svar.

A	B	C	D	E	F

Oppgåve 5

Beverlegen ynskjer å bygge tre sjukehus for beverane.
Sjukehusa kan berre byggast på stadane (bokstavane) vist på kartet nedanfor.
Beverane kan ikkje symje over meir enn ei elv for å kome seg til eit sjukehus.

Oppgåve: Set ring rundt dei tre plassane der sjukehusa kan byggast.



Vedlegg 5

Survey konvertert til Word

Eg er flink til å lære av feila mine

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

Eg er flink til å sjå på det å gjere feil som ein naturleg del av å løyse eit problem

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

Eg er flink til å ikkje gje opp

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

Eg er flink til å halde fram, sjølv når ting vert forvirrande

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3
(4) ❑ 4
(5) ❑ 5

Eg er flink til å sjå på ting på uvanlege måtar

(1) ❑ 1
(2) ❑ 2
(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å legge frå meg eit problem, for at løysinga kan kome til meg om ei stund

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å dele mine idéar

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å snakke med andre for å løyse eit problem

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å dele problemet opp i fleire mindre problem

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å forklare løysingane mine

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å finne den stegvise oppskrifta for å løyse eit problem

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å bruke oppskrifta for å løyse problemet

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å kjenne att element frå tidlegare problem

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å skilje ulike problem frå kvarandre

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å plukke ut den informasjonen eg treng

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å ta vekk den informasjonen eg ikkje treng

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å finne den lettaste løysinga

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

Eg er flink til å sjå om løysingsmetoden kan brukast til andre problem

(1) ❑ 1

(2) ❑ 2

(3) ❑ 3

(4) ❑ 4

(5) ❑ 5

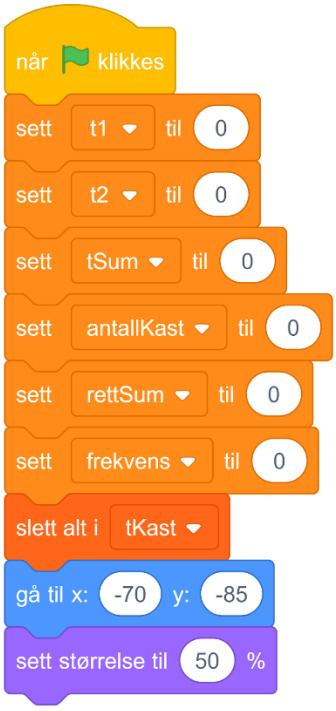
Vedlegg 6

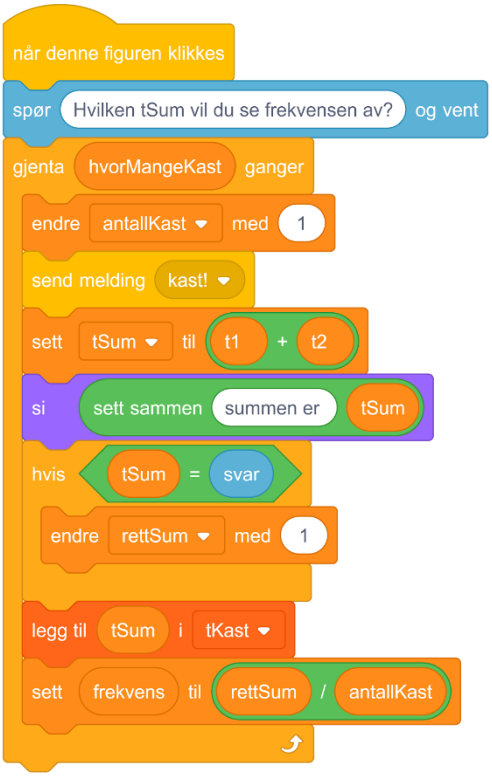
Terningoppgåve for plugged-gruppa

Vi skal lage et program som kaster to terninger for oss, og beregner frekvenstallet til poengsummene det er mulig å få (2-12). For å oppnå dette, bruker vi meldingssystemet i Scratch,

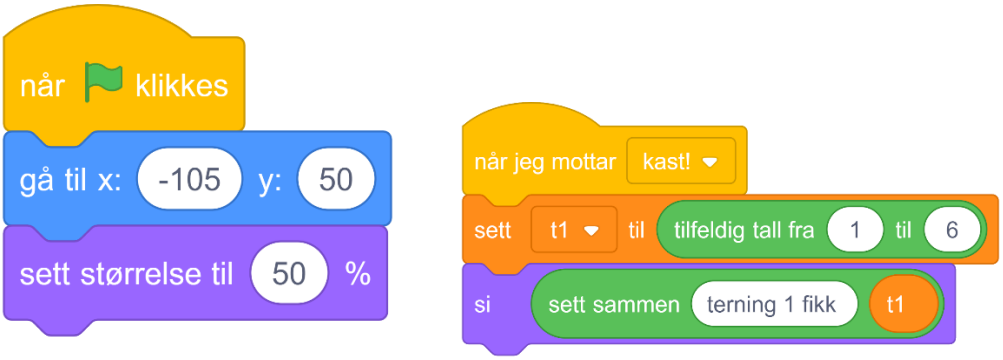
Før vi begynner selve kodingen, må vi opprette en rekke variabler og en liste som passer på tallene for oss. Under “Variabler”, trykk “lag en variabel” og kall den t1. Denne variabelen skal gi oss poengsummen fra terning 1. På samme måte oppretter du variablene t2 (terning2), tSum (summen av t1 og t2), antallKast (gir programmet beskjed om hvor mange kast vi ønsker å gjennomføre), rettSum (poengsummen vi vil ha frekvenstallet av) og frekvens (som skal vise er rettSum delt på antallKast). Til slutt lager vi listen tKast som passer på alle poengsummene vi får. Denne listen kan også eksporteres til et regneark om man vil utforske verdiene videre der.

Figur 1 gir vi navnet “MasterBrain”, og denne figuren har ansvaret for alt utenom selve terningkastene: Masterbrain trenger to algoritmer:





Vi må opprette en ny figur, som vi kaller “1D6”. 1D6 sitt eneste ansvar er å kaste terning nummer 1:



Videre lager vi en kopi av 1D6 (høyretrykk på figuren, “lag en kopi”) , som vi kaller 2D6. Gi 2D6 et annet koordinat enn 1D6, feks (103, 50), og bytt ut t1-variabelen med t2.

Vi kan skjule variablene t1, t2, tSum og rettsum fra scenen. Fjern den blå avhukingen ved variabelboblene i Variabel-menyen. Listen tKast behøver ikke være synlig, men den gir oss en oversikt over hvor mange kast som har blitt utført, og viser alle poengsummene fortløpende.

Høyretrykk på variabelen antallKast på scenen, og velg “skyveknapp”. Høyretrykk på den igjen og velg “endre verdiområde” for å bestemme minste og største mulige antall terningkast. Dra i skyveknappen for å sette verdien til antallKast-variabelen. Alle variabler, lister og figurer kan klikkes og dras rundt på scenen. Finn det oppsettet som passer best.

Dette programmet beregner altså hvor ofte en gitt terningsum forekommer, som kan sammenlignes med matematiske sannsynlighetsberegninger for terningkast med to terninger.

Følg lenken under for å teste et ferdigsnekret program. Legg merke til at vi her har lagt inn litt kode som lar MasterBrain “snakke” og fortelle brukeren hva hen skal gjøre. Denne snakkekoden er ikke nødvendig for en som har laget koden selv, men dersom programmet skal brukes av andre, er den kjekk å ha med!

https://bit.ly/kompmal9

Henta frå: https://www.kidsakoder.no/skole/#toggle-id-3 og https://docs.google.com/presentation/d/1vV3OYow3um3IoIFJMLkmKRoNDp6u3-D31PeKIZtuyxc/edit#slide=id.g479d84a5b0_0_150 (for 9. trinn)

Vedlegg 7

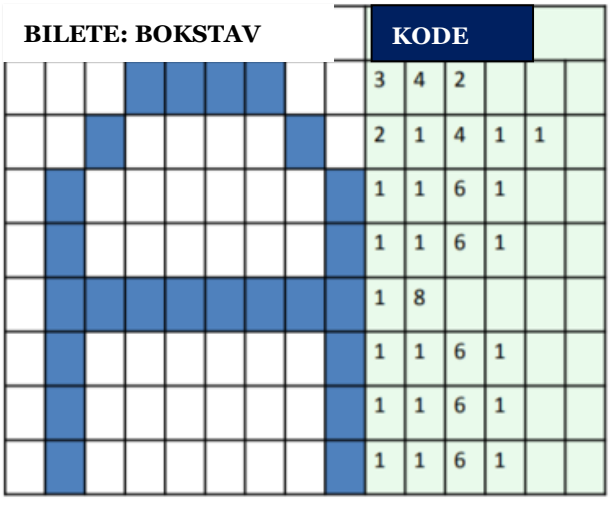
Pikseloppgåve for unplugged-gruppa

Pixelprogrammering

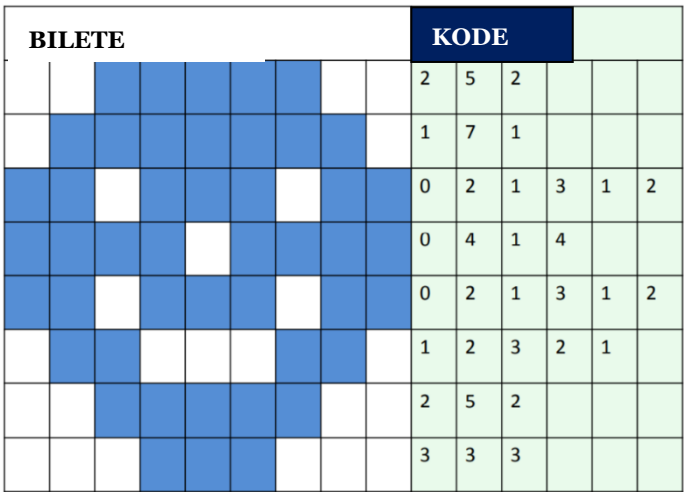
Korleis kan du sende bilete via ei datamaskin? Jo, du kan sende via mail, eller dele via sosiale medier. Men ei datamaskin kan berre lese dataspråk. Dataspråket består av einarar og nullar. Det vert kalla den binære koden. Datamaskina må fyrst gjere biletet om til pixlar (små kvadrat) for så å sende biletet. Datamaskina på mottakarsida omset pixlane tilbake til eit bilete. Same prinsippet gjeld også når du skal sende bilete med mobilen din. Du skal no lage ulike program ved å bruke pixlar.

1. **I denne oppgåva jobbar du med «sløkte» og «tende».** Det betyr at tomme ruter er dei som er «sløkte» og dei som skal fyllast er dei som er «tende».
2. **Bruk malen du har fått utdelt som utgangspunkt for ditt pixelprogram.**
3. **Du kan skrive seks kommandoar per rad**, desse skriv du i kodefeltet for kvar rad. Føl leseretninga. Start med «sløkte» (kvite) og bruk deretter annankvar-prinsippet.
4. **Tenk på korleis du skal skape pixelbiletet og den tilhøyrande koden.**
5. **La ein på gruppa teste programmet ved å få sjå koden og lage eit pixelbilete utifrå den.**
6. **Ser bileta like ut? Om ikkje, kva vart feil?** Er det ein feil i koden eller vart det feil når personen på gruppa tolka koden? Lag nye og meir komplekse kommandoar.

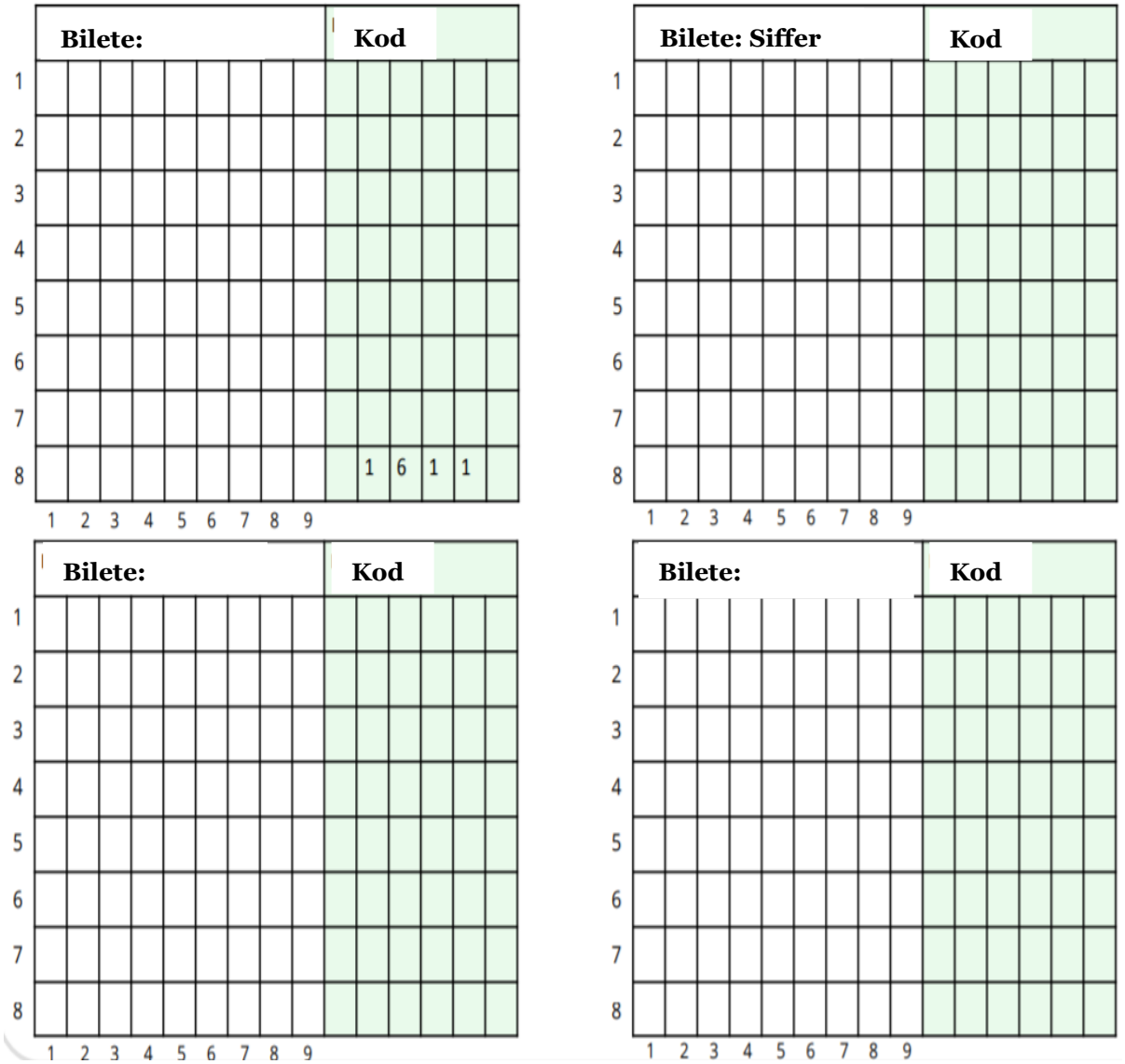
Døme 1



Døme 2



BILETE									KODE					

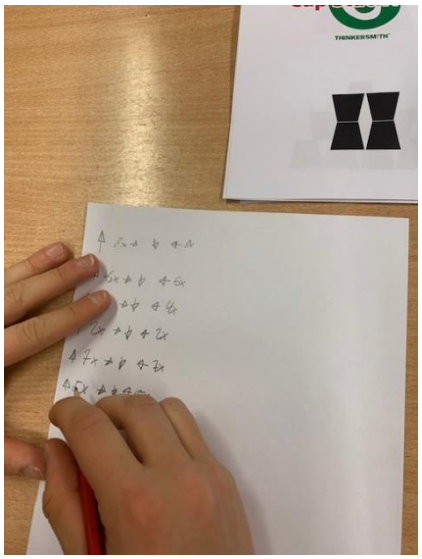


Henta frå:
https://www.skelleftea.se/Skol%20och%20kulturkontoret/Innehallssidor/Bifogat/Unplugged%20programmering%202018%20Skelleftea%20kommun.pdf

Vedlegg 8

Bilete frå intervensjonen

Figur 12 Mal til presentasjon for dei andre gruppene



Figur 13 Døme på komprimering av kode

Figur 14 Døme på komplekst koppetårn