Høgskulen på Vestlandet

MASTEROPPGAVE

Programmeringsverktøys undervisningsmateriell og dets applikasjon av algoritmisk tenkning

Programming tools' teaching materials and its application of computational thinking

Henriette Pettersen Holmøy

Master i matematikk i Grunnskolelærerutdanningen 1-7

Fakultet for lærerutdanning, kultur og idrett

Veileder: Magni Elen Hope Lossius

Innleveringsdato: 15. September

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Forord

Denne masteroppgaven ble til på grunn av et kritisk spørsmål jeg stilte meg selv i en undervisningsøkt i praksis. Muligheten til å forske på programmering i skolen er en jeg ikke tar for gitt. Dette har vært en utrolig frustrerende og lærerik prosess som jeg kommer til å se tilbake på med et smil. Jeg er stolt over eget arbeid og jeg gleder meg til å se hvordan jeg kan ta denne kompetansen i bruk.

Jeg vil takke veilederen min for å være så tålmodig og støttende. Dette har vært en krevende prosess og jeg setter stor pris på all hjelpen.

Jeg vil også takke vennene mine, som har vært gode støttespillere når det har vært vanskelig, for en nydelig heiagjeng jeg har! Takk til gjengen på C, det har vært lærerikt og gøy å tilbringe så mye tid med medstudenter som går gjennom den samme emosjonelle berg og dalbanen. Takk til samboeren min, for støtten du har gitt det siste året. Jeg setter stor pris på alt du har hjulpet meg med, du har stilt opp med både med motiverende ord og middag de dagene tiden ikke har strekt til. Takk til mams, for at du trøster når det trengs og presser meg til å stå på når motivasjonen har vært laber. Så sist, men ikke minst, en spesiell takk til Helene. Hadde det ikke vært for deg så hadde denne oppgaven faktisk ikke blitt til, du har virkelig vært den beste støtten jeg kunne hatt gjennom hele denne prosessen.

Jeg setter stor pris på dere!

Abstract

The purpose of this thesis is to search for similarities between the teaching materials published by programming tools that can be used to apply computational thinking in schools. I consider this research valuable since the teachers' competency on the field has been frequently disputed, especially since the new curriculum was released in 2020. I hope my analysis can help other teachers or educational institutions in the process of evaluating and choosing the most suitable tool to implement in their classrooms.

My thesis is: What are the characteristics of the teaching material of programming tools in mathematics and how do they apply computational thinking?

To answer my thesis, I have chosen to analyze nine teaching materials belonging to three separate tools. I will do so by using a framework based on Benton et al. (2016) and Brennan and Resnick (2012). These two frameworks combined with Charalambous et al. (2010) makes it so that I can analyze the tool both widely and deeply. This helps me with getting the point of the framework to come across in an appropriate way.

The tools I am looking in to is Kodeskolen, KUBO and Sphero. My findings show a lot of variation between them, but there are some common grounds considering computational thinking. All of them apply social cultural learning, even though they show different approaches. They are similar in the way that they can be used as an introduction to computational thinking, but they differ when it comes to how versatile they are, in the sense that some include several mathematical topics. The teaching materials also differ when it comes to how and to what extent they convey information and the teacher's role.

Sammendrag

Denne masteroppgavens hensikt er å undersøke kjennetegn i undervisningsmateriell tilhørende programmeringsverktøy som kan bli anvendt i skolen for å anvende algoritmisk tenkning. Jeg anser denne forskningen som vesentlig da det er mye omdiskutert hvor stor kompetanse lærere har innen algoritmisk tenkning og programmering, spesielt fordi det ble en del av læreplanen først i 2020. Jeg håper analyseverktøyet kan hjelpe lærere eller utdanningsinstitusjoner når de skal vurdere og velge hvilke programmeringsverktøy som vil egne seg til deres klasseromsundervisning.

Jeg tar for meg problemstillingen: Hva kjennetegner undervisningsmateriellet til programmeringsverktøy i matematikk og hvordan legger disse opp til anvendelse av algoritmisk tenkning?

For å svare på problemstillingen analyserte jeg ni undervisningsmateriell, hentet fra tre ulike verktøy, i lys av Benton et al. (2016) og Brennan og Resnick (2012) teori om algoritmisk tenkning, sammensatt i Charalambous et al. (2010) rammeverk. Denne sammensetningen gjorde at verktøyenes helhet kunne belyses, samtidig som jeg kunne dykke dypere inn i enkelte oppgaver og belyse tema på en hensiktsmessig måte. Verktøyene jeg har tatt for meg er Kodeskolen, KUBO og Sphero.

Funnene viser til stor variasjon mellom verktøyene, men at de har noen fellesnevnere innenfor algoritmisk tenkning. De legger alle opp til arbeid gjennom sosiokulturell læring, selv om de har ulik tilnærming til dette. De er like i at de kan anvendes som en introduksjon for algoritmisk tenkning, men de er ulike i hvor allsidig de er i form av at noen undervisningsmateriell inkluderer flere matematiske fagområder. De er også ulike i hvor stor grad de formidler informasjon og inkluderer lærer i undervisningsmateriellene.

Innholdsfortegnelse

1 Innledning

1.1 Bakgrunn for problemet

Tiden vi er i nå beskriver Bocconi et al. (2017) som en teknisk revolusjon. Dolonen et al. (2019) hevder at samfunnets utvikling har skapt et behov for å utvikle og utvide den digitale kompetansen i befolkningen. Forsström og Kaufmann (2018), samt Bocconi et al. (2017) hevder at det er vesentlig at neste generasjon får tilgang på slik kompetanse da det er de som skal møte nye, moderne problemstillinger. Videre skriver Bocconi et al. (2017) at over 20 land i Europa har tatt tak i læreplanen for å tilpasse den til de nye kompetansene og ferdighetene som kreves.

Den europeiske kommisjonen presenterte en ny digital utdanningsplan, 17. Januar, 2018 (Bocconi et al., 2018). Den var ment for å hjelpe undervisningsinstitusjoner og andre utdanningssystem med å tilpasse seg den digitale utviklingen som vi er vitne til i dagens samfunn. Bocconi et al. (2018) beskriver planens tre hovedpoeng: Å bedre utnytte digital teknologi for å undervise og lære; utvikle relevant digital kompetanse og ferdigheter for den digitale transformasjonen; forbedre utdanningen gjennom bedre data-analyse og fremsyn. Disse tre hovedpoengene innebærer å innføre algoritmisk tenkning i de europeiske klasserommene. Med denne oppgaven ønsker jeg å bidra med forskning til det ene hovedpoenget: Å bedre utnytte digital teknologi for å undervise og lære.

I Norden begrunnes integreringen av algoritmisk tenkning og programmering i læreplanen med forankring i at det vil bygge et ideelt grunnlag for problemløsning, logiske ferdigheter og digital kompetanse (Bocconi et al., 2018). Wing (2010) skriver at algoritmisk tenkning er et godt virkemiddel i alle fagfelt, da denne kompetansen forsterker intellektuelle egenskaper, som kommunikasjon. I Norge er hovedtyngden av kompetansen omhandlende programmering og algoritmisk tenkning likevel tillagt matematikkfaget. Det er dog uenigheter rundt overføringsverdien programmering har i matematikkundervisningen (Dolonen et al., 2019; Popat og Starkey, 2018). Dolonen et al. (2019) skriver at det er få argumenter for at programmering skal være en del av matematikkfaget, men at programmering uten tvil hører hjemme i læreplanen da forskning har vist at programmering har positiv påvirkning på elevenes digitale ferdigheter og forståelse.

I den nye læreplanen i matematikk er digitale ferdigheter nevnt som en av de grunnleggende ferdighetene. Programmering er nevnt under digitale ferdigheter. Der står det at

programmering skal anvendes som et verktøy til å utforske og løse matematiske problem (Utdanningsdirektoratet, 2020). Forsström og Kaufmann (2018) definerer programmering som prosessen som skjer når en gir instrukser til et program som gjør at datamaskinen kan gjennomføre spesifikke oppgaver, løse problemer og støtte menneskelig interaksjon. Det er denne definisjonen av programmering jeg forholder meg til i denne oppgaven.

Under kjerneelementet utforsking og problemløsing finner en algoritmisk tenkning. Dette begrepet er den norske oversettelsen fra computational thinking (Torkildsen & Gjøvik, 2019). Begrepet får stadig mer oppmerksomhet og det er mye diskutert hva begrepet egentlig innebærer (Brennan & Resnick, 2012). Jeg har valgt å ta utgangspunkt i Wings (2010) definisjon av algoritmisk tenkning i denne oppgaven. Wing (2010, s. 1) har formulert denne definisjonen:

> *«Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent»*

Slik jeg forstår Wings (2010) definisjon er algoritmisk tenkning en tankeprosess som er involvert i formuleringen og løsningen av et problem, og slik jeg forstår Forsström og Kaufmann (2018) er programmering å gi instrukser til et program for å løse/gjennomføre oppgaver og problem.

Det er en gjentagende problemstilling innenfor forskningsfeltet at lærere er avhengig av å besitte kompetanse om programmering og algoritmisk tenkning for å kunne implementere disse på en hensiktsmessig måte i sin undervisning. Dette kommer frem i forskningen til blant annet Benton et al. (2019), Benton et al. (2017), Bocconi et al. (2019), Dolonen et al. (2019), Forsström og Kaufmann (2018) og Sentance et al. (2019).

At lærerne har mangel på kompetanse kan føre til at det blir utfordrende å vurdere og velge ut riktig verktøy og undervisningsmateriell for undervisningen (Bråting et al., 2020). Formålet med denne oppgaven er å kunne gjøre prosessen i å finne et passende verktøy for klassen noe enklere.

Ifølge Bocconi et al. (2018) er det et behov for å se på forskning relatert til pedagogikk, verktøy og vurdering. Med denne oppgaven ønsker jeg å bidra med forskning på hva

utgiveren formidler, og hvordan ulike verktøy legger til rette for arbeid med algoritmisk tenkning i klasserommet. Dette samler jeg under problemstillingen:

> *Hva kjennetegner undervisningsmateriellet til ulike programmeringsverktøy i matematikk og hvordan legger disse opp til anvendelse av algoritmisk tenkning?*

Kompetansen blant lærere når det kommer til algoritmisk tenkning kan være noe mangelfull da dette er relativt nytt i læreplanen og lærerne har mangel på tid og ressurser til å kunne tilnærme seg denne kompetansen. Jeg mener at dette forskningsprosjektet kan bidra med å belyse flere programmeringsverktøy og hvordan disse kan legge opp til arbeid med algoritmisk tenkning. Målet er ikke å finne hvilket verktøy som er best, da det ikke er en fasit på hva som er best for alle klasser i Norge, men heller å skape et rammeverk som kan anvendes for å analysere undervisningsmateriell og som kan hjelpe meg med å besvare problemstillingen. Jeg håper rammeverket kan assistere andre i vurderingsprosessen for hvilke programmeringsverktøy som passer best for ulike lærere med ulike klasserom.

1.2 Begrunnelse for valg av verktøy

Da den nye læreplanen tredde i kraft, hadde jeg en vikartime som assistent hos første klasse. Læreren skulle undervise i programmering og skulle observere hvordan hen skulle undervise i dette tema. Læreren viste elevene hvordan de kom seg inn på kodeskolen, Salaby, og lot dem arbeide individuelt med oppgavene. Elevene så ut til å ha det gøy med oppgavene. Mens elevene jobbet med verktøyet, begynte jeg å lure på hvilket utbytte elevene får. Hvorfor jobbet elevene individuelt og hvilken variasjon tilbydde verktøyet? Dette førte til at jeg ble bevisstgjort rundt mangelen på egen kompetanse innen dette fagområdet, for å kunne vurdere undervisningsøkten.

Etter denne observasjonen har jeg lekt litt rundt på ulike plattformer for å få en viss forståelse for programmering. I følge Fokides (2017) har Scratch blitt forsket mye på, blant annet gjennom forskningsprosjektet ScratchMath som Benton et al. (2017) har skrevet artikkel om. Jeg har funnet lite litteratur om andre verktøy gjennom mitt litteratursøk, dette har ført til at jeg ønsker å undersøke andre verktøy som er tilgjengelig for lærere.

Jeg tok kontakt med læringslaben på høyskolen og fikk muligheten til å teste ut ulike roboter. Etter samtale med læringslabben og Bergen kommune endte jeg opp med å ta med KUBO og Sphero hjem for å undersøke litt hvordan de fungerer. Dette førte til at jeg ble interessert i å undersøke KUBO og Sphero. Etter flere samtaler med læringslabben og Bergen kommune

konkluderte jeg med at disse to sammen med Kodeskolen ble et godt utgangspunkt for hvilke programmeringsverktøy som tas i bruk i Bergen og Norge.

Flere skoler i Bergen har KUBO klassesett (Se vedlegg 2), Kodeskolen og Sphero blir mye brukt i norsk skole (Se vedlegg 1 og 3). KUBO er et fysisk verktøy, Sphero er også et fysisk verktøy, men krever bruk av en app på telefon/nettbrett for selve programmeringen, og Kodeskolen foregår bare på datamaskin. De er altså ulike i utgangspunktet på noen punkt, men alle har en felles hensikt, nemlig å bidra med læring av programmering og algoritmisk tenkning i skolen.

1.3 Læreplanen og algoritmisk tenkning

Videre skal jeg gjøre rede for algoritmisk tenkning og programmering i læreplanen i matematikk.

Læreplanen består av flere komponenter, læreplanen i matematikk består blant annet av fem grunnleggende ferdigheter. En av disse er digitale ferdigheter, hvor jeg finner programmering:

> *«Digitale ferdigheter i matematikk innebærer å kunne bruke graftegner, regneark, CAS, dynamisk geometriprogram og programmering til å utforske og løse matematiske problemer …».*
> (Udir, 2019)

Læreplanen inneholder også av kjerneelement, hvorav et av dem som eksplisitt nevner algoritmisk tenkning. Kjerneelementet utforsking og problemløsing går som følger:

> *Problemløsing i matematikk handler om at elevene utvikler en metode for å løse et problem de ikke kjenner fra før. Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer og innebærer å bryte ned et problem i delproblemer som kan løses systematisk. Videre innebærer det å vurdere om delproblemene best kan løses med eller uten digitale verktøy. Problemløsing handler også om å analysere og omforme kjente og ukjente problemer, løse dem og vurdere om løsningene er gyldige.*
> (Udir, 2019)

I læreplanen kan en krysse av for å se progresjon til hvert kompetansemål, altså kan en se hvilke kompetansemål som er koblet til hvilke kjerneelementer. I læreplanen i matematikk

finner vi kompetansemål knyttet til programmering allerede i andre trinn (Torkildsen & Gjøvik, 2019). Dersom en velger kjerneelementet utforsking og problemløsing får en opp disse kompetansemålene:

Tabell 1

Trinn	Kompetansemål
2.	lage og følge regler og trinnvise instruksjoner i lek og spill
3.	lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet
4.	lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker
5.	lage og programmere algoritmer med bruk av variabler, vilkår og løkker
6.	bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre
7.	bruke programmering til å utforske data i tabeller og datasett

Dersom en velger digitale ferdigheter, den grunnleggende ferdigheten som tar for seg programmering, får en opp de samme kompetansemålene, med unntak av kompetansemålet for 2. trinn.

1.4 Disposisjon

I kapittel to ønsker jeg å redegjøre for relevant teori. Jeg går da inn i hva algoritmisk tenkning og programmering er. Jeg tar for meg teori om programmeringsspråk og redegjør for det teoretiske grunnlaget for rammeverket mitt.

I kapittel tre går jeg inn i forskningsmetoden som er anvendt i oppgaven. Der gjør jeg rede for hvilken metode jeg har anvendt, prosessen med å skape et passende rammeverk for analysen og utvalget. Jeg gjør også rede for forskningsetiske betraktninger, reliabilitet og validitet.

Kapittel fire er analysen min. Den horisontale analysen skal gi et overblikk over verktøyet før jeg går i dybden på et og et undervisningsmateriell i den vertikale analysen.

I kapittel fem tar jeg for meg funnene i analysen og drøfter disse. Jeg søker da etter svar på min problemstilling og leter etter kjennetegn mellom de ulike undervisningsmateriellene og verktøyene. Jeg diskuterer programmeringsspråkene som blir anvendt og jeg ser på verktøyene i et sosiokulturelt perspektiv.

I kapittel seks oppsummerer jeg funnene i undersøkelsen og kommer med forslag for videre forskning.

2 Teori

2.1 Programmering

I dette kapittelet skal jeg redegjøre for hva programmering er og litt om hva forskning sier om hvordan dette kan læres.

Programmering er prosessen som skjer når en gir instrukser til program som gjør at pcen kan gjennomføre spesifikke oppgaver, løse problemer og støtte menneskelig interaksjon (Forsström & Kaufmann, 2018).

Papert forsket på programmering i skolen allerede i 1980. Han tok i bruk programmeringsspråket LOGO og brukte det til å programmere en skilpadderobot på en pc til å tegne geometriske former (Papert, 1993). Blokkprogrammering baserer seg på anvendelse av objekt i stedet for tekstbaserte kodespråk (Plaza, 2020). Eksempelvis har KUBOs TagTiles illustrasjoner som viser hva brikkene innebærer, i stedet for at elevene må skrive ut koder i tekstbaserte programmeringsspråk. Blokkprogrammering anvendes på barneskolen om en introduksjon til programmering (Fokides, 2017), selv om dette ikke er et programmeringsspråk som brukes profesjonelt (Kölling, 2015). Elevene går ofte over til tekstbasert programmering i ungdomsskolen (Kölling, 2015) og ifølge Dolonen et al. (2019) kan elever med kunnskap om blokkbasert programmering ha lettere for å forstå tekstbasert programmering da de har blitt introdusert til flere grunnleggende prinsipp fra før.

Papert argumenterte, ifølge Resnick et al. (2009), for at programmeringsspråk burde ha lav inngangsterskel, høyt tak og breie vegger. Slik jeg forstår det burde språket altså være brukervennlig så en enkelt kan komme i gang med programmering, ha muligheten til å skape og/eller løse gradvis mer komplekse prosjekt og/eller problem, og det burde kunne støtte mange ulike prosjekt for å engasjere mennesker med ulike interesser.

Slik jeg forstår Sung et al. (2017) har læringsmiljøet mer å si for elevens læring enn hvilket verktøy hen bruker. Læringsutbyttet til eleven avhenger av å forstå konseptet programmering, og å arbeide med passende aktiviteter som støtter algoritmisk tenkning har et større læringsutbytte enn å arbeide med digitale verktøy med mindre passende aktiviteter. Sung et al. (2017) refererer til metoden embodiment. Dette innebærer å utføre koder selv eller å kontrollere en annen person, det viser elevene hvordan kodene deres kan bli tolket og hvor nøyaktig de må være i språket. Flø (2021) skriver også om hvordan det han kaller «unplugged programmering» er et godt verktøy for å lære om konseptet programmering. Denne metoden

finner jeg altså i forskningen til Sung et al. (2017), Dolonen et al. (2019), Popat og Starkey (2018), Bocconi et al. (2018) og Torkildsen og Gjøvik (2019). Metoden får også kritikk fra Grover og Pea (2013). De skriver at denne metoden kan være en verdifull introduksjon, men at den også kan føre til at elevene ikke eksponeres for kritiske aspekt av algoritmisk tenkning.

2.2 Algoritmisk tenkning

Bocconi et al. (2018), Sung (2017) og Brennan og Resnick (2012) skriver alle at det er usikkerhet i hva computational thinking egentlig innebærer og at det blir diskutert i forskningen. Jeannette Wing har publisert flere artikler der hun diskuterer hva det innebærer og kommer med en definisjon på begrepet, som referert til i innledningen.

Hun forklarer at Computational thinking omhandler å kombinere det menneskelige med informatikk, slik at vi sammen med de ulike verktøyene kan ta fatt på utfordrende oppgaver (Wing, 2006). Ifølge Sung (2017) er det fremdeles ingen fast definisjon på hva computational thinking innebærer, men Bocconi et al. (2018) skriver at selv om det er mange ulike definisjoner i litteraturen har de funnet noen felles nevnere i de ulike definisjonene: Abstraksjon, algoritmisk behandling, automasjon, dekomposisjon og generalisering.

I Norge har vi som nevnt i kapittel … tatt i bruk begrepet algoritmisk tenkning. Gjøvik og Torkildsen (2019) nevner at denne oversettelsen kan være noe uheldig, da algoritmisk tenkning fort blir assosiert med algoritmer. Algoritmisk tenkning innebærer, som vi ser i Wings (2010) definisjon, mye mer enn bare algoritmer. Ifølge Udir (2019) er algoritmisk tenkning en tenkemåte for å tilnærme seg problemer på en systematisk måte. Torkildsen og Gjøvik (2019) skriver at det i tillegg handler om å ha kompetansen til å løse problemene med bruk av ulike hjelpemidler og metoder. Algoritmisk tenkning er altså, slik jeg tolket Bocconi et al. (2018), en problemløsningsmetode og tenkemåte. Bocconi et al. (2018) skriver at arbeid med algoritmisk tenkning kan inkludere teknologisk utstyr, men det er ikke avhengig av teknologi. Brennan og Resnick (2012) påpeker at selv om en kan arbeide med algoritmisk tenkning uten bruk av teknologi er det fordelaktig å arbeide med programmering gjennom teknologi for å oppnå denne kompetansen.

Embodied approach eller unplugged programming er to begrep som blir brukt om hverandre og handler om å programmere uten teknologisk utstyr. Dette er en metode en kan bruke for å arbeide med algoritmisk tenkning uten å ta i bruk teknologisk utstyr. Unplugged programming kan arbeides med gjennom å leke leker der koder blir brukt (Sung et al., 2017).

Sung et al. (2017), Dolonen et al. (2019), Popat & Starkey (2018), Bocconi et al. (2018) og Torkildsen (2019) skriver om hvordan det å programmere medelevene sine kan hjelpe dem med å utvikle elevenes kommunikative ferdigheter og samarbeid, da de må være bevisst i hvordan de bruker språket og være nøyaktig for at beskjedene ikke skal kunne misforstås.

Brennan & Resnick (2012) har, gjennom flere år med forskning, utviklet tre dimensjoner innenfor algoritmisk tenkning. «Computational concepts», «Computational practices» og «Computational perspectives». Disse blir en del av mitt rammeverk for analyse, da de går godt inn i hva algoritmisk tenkning faktisk består av, jeg har valgt å oversette disse til: algoritmisk konsept, algoritmisk praksis og algoritmisk perspektiv. Jeg tar utgangspunkt i at algoritmisk er det tilsvarende begrepet til computational, da algoritmisk tenkning tilsvarer computatinal thinking.

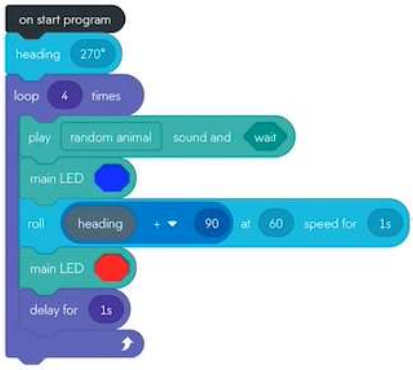
2.2.1 Algoritmiske konsept

Brennan og Resnick (2012) har funnet syv algoritmiske konsept gjennom bruk av Scratch: «Sequences, loops, parallelism, events, conditionals, operators and data». Videre vil jeg referere til disse gjennom de norske begrepene: *Sekvens, løkke, parallell, hendelse, betingelser, operatører og data*. Videre kommer en beskrivelse på disse begrepene, oversatt fra Brennan og Resnicks (2012) artikkel.

En *sekvens* er et nøkkelkonsept i programmering og beskriver en liste med individuelle instruksjoner. Den gir en spesifikk beskrivelse av hva som skal skje dersom en hendelse tar sted. En *sekvens* kan være en kode som blir kjørt dersom brukeren trykker på start-knappen.

En *løkke* er en instruks om å gjenta sekvensen i et gitt antall repetisjoner. Så dersom du har en sekvens kan du be programmet gjenta denne sekvensen med å kode en løkke inn i sekvensen. Du kan be KUBO om å ta fem steg med å sette gå funksjonen inn i en løkke med fem

repetisjoner. Nedenfor er et eksempel på en løkke i en sekvens.



Figur 1

Parallell beskriver at et program kan kjøre flere sekvenser samtidig. For eksempel at en figur skal lage lyd samtidig som den beveger seg. Sphero kan både ha et mønster på skjermen og bevege seg samtidig, dette er da to parallelle sekvenser.

En *hendelse* beskriver at noe skal skje dersom en for eksempel trykker på start knappen på et spill. Hendelsen er da at en trykker på startknappen. Å trykke på en knapp er en hendelse som trigger en spesifikk sekvens. For eksempel kan hendelsen å trykke på knappen x trigge en sekvens som får figuren i spillet til å hoppe.

En *betingelse* er også kjent som en hvis-sekvens, hvis en spesifikk hendelse tar sted, så skal en annen sekvens iverksettes. For eksempel kan vi be Sphero om å snu seg dersom den bråstopper, for eksempel om den treffer en vegg etc. Hvis bråstopp = sant → snu 180◦.

Operatører støtter seg på matematikk, logikk og stringuttrykk. Dette tillater den som programmerer å gjennomføre manipulasjoner av numerisk og string type. Eksempel på dette er: velg vilkårlig tall mellom x og y, eller å tildele verdier til variabler. Dette åpner muligheten for å programmere skjermen til Sphero til å fungere som en terning. Den kan programmeres til å velge et tilfeldig tall mellom 1 og 6 dersom den blir ristet.

Data innebærer å lagre, hente og oppdatere verdier. Eksempler på data er variabler og lister. En variabel er noe som kan forandre verdi, for eksempel kan farger eller hvor mange ganger en kode skal kjøres være variabler. En liste kan være en sekvens bestående av lagrede verdier. En oppgave i Sphero går ut på å lage morsekoder, da får hver bokstav i alfabetet tildelt en kode som får Sphero til å lyse i morsekode. Eksempelvis vil tre lange lys tilsvare bokstaven S, etc. Disse verdiene blir lagret som data i programmet.

2.2.2 Algoritmisk praksis

I algoritmisk praksis er fokuset på hvordan en lærer, heller enn hva en lærer. Brennan og Resnick (2012) skriver at selv om elevene brukte mange metoder var det fire hovedmetoder de observerte: «being incremental and iterative, testing and debugging, reusing and remixing, and abstraction and modularizing». Disse har jeg oversatt til: å være inkrementell og iterativ, teste og «debugge», gjenbruk og remiksing, abstrahere og modularisere

At noe er *inkrementelt* betyr at det er økende, *iterativt* betyr at noe er gjentagende. Et godt eksempel på noe inkrementelt er klikkespill, hvor alt du gjør er å klikke på skjermen gjentatte ganger og tjener poeng på dette, som da kan brukes for å kjøpe bonuser som gjør at klikkene dine får høyere verdi. At noe er iterativt viser til å gjøre det samme flere ganger, men at det blir gradvis mer utfordrende. Den algoritmiske praksisen *inkrementell og iterativ* viser, enkelt forklart, til gjentagelse og økende vanskelighetsgrad.

Det er sjelden ting fungerer som en tenker med en gang. Den *algoritmiske praksisen testing og debugging* fremhever hvor vesentlig det er å utvikle strategier for å løse problemer underveis. Ulike metoder kan være: å identifisere problemet eller roten til problemet, lese gjennom «scripts», eksperimentere med «scripts», skrive «script» på nytt, finne eksempler på «scripts» som fungerer, fortell/spør noen andre om problemet, ta en pause.

Å *gjenbruke og remikse* koder handler om å ta nytte av andres koder, remiks henviser til å redigere disse kodene. Det kan være å bruke løsningen til en medelev, eller å gjøre egen kode tilgjengelig for andre. Det ligger mye kode ute på nettet og disse er det mulig å hente frem, redigere og anvende i passende anledninger. Dette åpner for å kunne skape mer komplekse koder, som en kanskje ikke hadde kompetansen til på egen hånd.

Å *abstrahere og modularisere* innebærer å bygge noe stort av å sette sammen flere små deler. Det kan være å lage flere små koder i stedet for en lang kode. For eksempel med å lage delmål dersom roboten/figuren skal bevege seg fra et sted til et annet. Dersom den skal bevege seg på en bestemt måte, snakke under visse omstendigheter og danse når den krysser mål så kan dette være en lang kode, eller tre små koder. Å ha flere mindre koder gjør feilsøkingsprosessen ryddigere da en kan ta for seg en liten kode av gangen og se bort fra forstyrrende detaljer som ville vært med i en lengre kode.

2.2.3 Algoritmisk perspektiv

Med algoritmiske perspektiv mener Brennan og Resnick (2012) perspektivene elevene utvikler når de arbeider med algoritmisk tenkning. Disse består av: Expressing, connecting og questioning. Dette har jeg ikke mulighet å undersøke i min oppgave, så dette går jeg ikke videre inn på i teorien min.

2.2.4 De fem e'ene

I tillegg til disse tilnærmingene til algoritmisk tenkning tar jeg for meg Benton et al. (2016) presenterer forskningsprosjektet ScratchMaths, som ble gjennomført i England. Der har de drevet med designbasert undervisning for elever i alderen 9-11. Ut fra dette prosjektet har de utviklet rammeverket de kaller de fem e'ene. Disse innebærer Explore, Explain, Envisage, Exchange og Bridge (Benton et al., 2016). Disse har jeg valgt å oversette til: *Utforske, forklare, å se for seg, utveksle og koble*.

Det første begrepet, *utforske*, tydeliggjør viktigheten av å utvikle og støtte aktiviteter som tillater elevene å utforske ideer, teste ting og debugge konseptuelle og tekniske feil, dersom det er nødvendig. Dette handler om å la elevene ta kontroll over egen læring og utforske hva som ligger bak de ulike utfallene.

Et vesentlig aspekt av å forstå ideer er å kunne *forklare* ens kunnskap og artikulere grunnene til den valgte tilnærmingen. Gjennom å svare på spørsmål fra medelever og uttrykke ideene eksplisitt utvikler elevene evnen til å avklare ideer. Dette begrepet uttrykker viktigheten av å legge opp til reflektive spørsmål og åpne for klasseromsdiskusjoner, eller diskusjoner med medelever og med læreren. Det kan være i form av lærerstyrte klassediskusjoner, samarbeid med en medelev eller å svare på refleksjonsspørsmål.

Et viktig punkt i programmering er å *se for seg* et klart mål når en bygger et program/kode, og å forutse hva det mulige utfallet blir før en kjører koden. Å generere et tenkt utfall gjennom koding kan være utfordrende. Dersom utfallet ikke blir som ønsket må elevene «debugge» koden. Begrepet å se for seg tydeliggjør behovet for at læringsaktiviteter gir elevene muligheten til å planlegge utfallet for programmet/koden og reflektere over dette. Hvorfor ble utfallet som planlagt, eller hvorfor ble utfallet ikke som planlagt? Det er viktig at dette konseptet er i balanse med konseptet utforsking, da det muliggjør å oppdage gjennom utforsking, samtidig som det gir anledningen til å se for seg mulige utfall før de tester programmet/koden.

Det fjerde begrepet, *utveksle*, fremhever viktigheten av meningsfulle muligheter til å dele og bygge på hverandres ideer. Hoyles (1985) foreslår at andres ideer potensielt kan resultere i modifiseringer av elevens tankerekke, det er spesielt hjelpsomt for å gjøre rede for og forklare ideer som ikke er fullstendig formet enda. Dette begrepet baserer seg på at elevene skal utveksle sine tanker og ideer, på denne måten blir oppdagelsene og utforskingen mindre lærerstyrt.

Det femte, og siste, begrepet er å *koble*. Det innebærer å koble sammen begrepene, som jeg nå har lagt ut om, med faglig kontekst. I artikkelen til Benton (2016) og i min masteroppgave er det matematikk, som er i fokus. Altså handler det femte begrepet (i denne oppgaven) om å koble sammen algoritmisk tenkning med matematikkfaget.

2.3 Sosiokulturell læring

Sosiokulturell læringsteori er en teori som legger stor vekt på sosial samhandling og språk (Christensen & Ulleberg, 2015). Abtahi et al. (2017) beskriver sosiokulturell læringsteori som en teori som omhandler hvor mye en kan få til med støtte fra en mer vitende andre. Sentance et al. (2019) og Abtahi (2018) beskriver at denne støtten kan være verktøy, for eksempel språk, lærere, medelever, en linjal eller bøker. Abtahi et al. (2017) definerer verktøy som et virkemiddel for ytre aktivitet, som kan anvendes av mennesker for å påvirke objekter. Sentance et al. (2019) diskuterer hvorvidt blokkprogrammering kan være et mulig verktøy for å lære programmering, da dette verktøyet minimerer muligheten for syntax-feil. Blokkprogrammering gjør det mulig for brukeren å fokusere på de andre delene av prosessen (Sentance et al., 2019).

Abtahi et al. (2017) har gjennom sin forskning sett på verktøy som den mer vitende andre, og hvordan sosiokulturell læring kan være multidimensjonal. I artikkelen Abtahi et al. (2017) tar de for seg eksempelet om Mellony, Lila og en fjernkontroll (Abtahi et al., 2017). De diskuterer hvordan interaksjonen mellom fjernkontrollen og Lila skaper læring, men også hvordan Mellony kommer inn som en komponent i Lilas læring. Abtahi et al. (2017) argumenterer for at denne situasjonen er multidimensjonal da både fjernkontrollen og Lila bidrar til læringssituasjonen som en mer vitende andre.

3 Metode

I dette kapittelet skal jeg gjøre rede for metoden jeg anvender i oppgaven. Jeg går først inn i valg av metode, videre tar jeg for meg utviklingen av analyseverktøyet, utvalg av programmeringsverktøy samt undervisningsmateriell. Deretter gjør jeg rede for kildekritiske vurderinger og vurderinger knyttet til validitet, reliabilitet og forskningsetikk.

3.1 Kvalitativ metode

Hjerm og Lindgren (2011) skriver at kvalitativ metode ofte blir ansett som en mykere forskningsmetode enn kvantitativ metode. Dette argumenterer de for ved at det ikke brukes harde data i like stort omfang, og at en kvalitativ analyse er mer basert på subjektive tolkninger av data. Videre beskriver de kvalitativ metode som en håndfast metode, på tross av at den kan bli fremstilt på en uheldig måte i litteraturen. De påpeker at kvantitative undersøkelser også inneholder mye tolkningsarbeid og at kvalitative undersøkelser også kan bestå av harde data.

Hjerm og Lindgren (2011) samt Jacobsen (2015) presenterer kvalitativ metode som en forskningsmetode bestående av ord. Jacobsen (2015) skriver at kvalitativ metode også kan bestå av andre former for tekst, som bilder, film og lydopptak. Forskeren prøver å finne kategorier, beskrivelser eller modeller som beskriver en sammenheng i omverdenen (Olsson & Sörensen, 2001). Kvalitative data kan samles inn gjennom intervju, dokumentanalyse eller andre kvalitative metoder, og analyseres med bruk av ulike analyseverktøy.

En sorterer vanligvis kvalitativ metode under hermeneutisk forskningstradisjon (Krogtoft & Sjøvoll, 2018). Nils Gilje skriver, ifølge Asdal og Reinertsen (2020), helt kort at hermeneutisk tolkning «handler om å forsøke å gjøre noe forståelig som er uklart eller uforståelig» (Asdal & Reinertsen, 2020, s. 244). Asdal og Reinertsen (2020) skriver at å fortolke i hermeneutisk forstand handler om å finne en underliggende sammenheng, hva intensjonen til skaperen av verket egentlig er. Hermeneutikk er, ifølge Leseth og Tellmann (2014), en tolkningsmetode som ikke kan finne den «sanne» tolkningen av en tekst, da tolkningen alltid vil være påvirket av den samtidige og historiske konteksten tolkningen inngår i.

Krogh (2014) skriver om Gadamers forståelse av hermeneutikk og diskuterer den hermeneutiske sirkelen. Den hermeneutiske sirkelen beskriver hvordan leseren går inn i teksten med fordommer. Vi mennesker har alltid en førforståelse og fordommer, det kan være

at du har hørt om teksten før, har sett lignende tekster eller andre faktorer (Jacobsen, 2015). En tekst (eller andre objekt etc.) kan se veldig ulik ut for to personer med ulike vinklinger (Christoffersen & Johannessen, Forskningsmetode for lærerutdanningene, 2018). Gadamer beskriver, ifølge Krogh (2014), dette som en horisont. Den er i konstant utvikling, da våre forutsetninger og fordommer hele tiden vil forandre seg på grunn av nye inntrykk og ny forståelse (Krogh, 2014). Leseren av teksten hopper frem og tilbake mellom de ulike delene, noe som påvirker leserens syn og/eller forståelse (Jacobsen, 2015). Asdal og Reinertsen (2020) skriver: «Litt uavhengig av hvilken tekstlesnings- og dokumentstudietradisjon vi forholder oss til, vil dette nesten alltid være inkludert i metoden» (Asdal & Reinertsen, 2020, s. 247). Hermeneutikk vil altså inngå i min metode da min analyse handler om å tolke undervisningsmateriell og min forståelse vil utvikle seg underveis i skriveprosessen.

Fordommene jeg sitter med nå er ikke nødvendigvis de samme når jeg er ferdig, jeg kan finne ting jeg ikke hadde forestilt meg, noe som kan forandre min forståelse. Jeg vil stadig gå tilbake til analyseverktøyet mitt med nye øyne, så komme tilbake til undervisningsmateriellet med nye tanker og ideer. Mitt foreløpige analyseverktøy er derfor bare en mal. Frem til oppgaven er ferdig vil den være åpen for endring og selv etter det kan andre bruke og gjøre forandringer som passer deres tolkning og forståelse bedre.

3.2 Dokumentanalyse

Med hensyn til oppgavens vinkling, vil det i denne oppgaven være nærliggende å gjennomføre en dokumentanalyse. En dokumentanalyse gjøres ved å analysere utvalgte dokumenter i lys av relevant teori (Jacobsen, 2015). Undervisningsmateriellene jeg tar for meg i analysen er dokumenter, siden dokumenter kan defineres som: «ethvert konkret eller symbolsk tegn som har blitt bevart eller nedtegnet for et formål å representere, gjenskape eller påvise et fysisk eller intellektuelt fenomen» (Asdal & Reinertsen, 2020, s.15). Undervisningsmateriellene er nedskrevne dokumenter med formål om å formidle oppgaver som skal anvendes i en undervisnings- eller læringssituasjon.

I en dokumentanalyse henter forskeren inn sekundærdata, altså at dataene er hentet fra kilder som er skrevet av andre, som kan ha andre hensikter enn forskeren har (Jacobsen, 2015). Jacobsen (2015) skriver at sekundærdata kan gjøre at forskeren ikke får svar på alt hen ønsker å undersøke, da det er noen andre som har avgjort hva som er inkludert i teksten. Forskeren kan ønske informasjon om variabler, verdier og enheter som forfatteren ikke har tatt med.

Dette kan bety at skaperne av undervisningsmateriellet kan ha inkludert/ekskludert informasjon som kan være nyttig/unyttig for en gitt undersøkelse.

I denne undersøkelsen er det derfor hensiktsmessig å gjennomføre en kvalitativ dokumentanalyse. Dette åpner for å studere hvordan utvikleren klarer å formidle produktet til brukeren. Dette kan gjøres med å analysere undervisningsmateriellet med en vid vinkel sammen med dypdykk i enkeltoppgaver hos ulike utviklere.

Jacobsen (2015) skriver at den kvalitative analysen omhandler fire forhold: Dokumentere, utforske, systematisere og kategorisere og sammenbinde. Hjerm og Lindgren (2011) har undersøkt tidligere forskning og ser at en kvalitativ analyse ofte blir delt inn i tre eller fire trinn. De ser på de tredelte nivåene som mest hensiktsmessig og beskriver nivåene reduksjon, presentasjon og konklusjoner, samt verifisering av data.

Etter mye prøving og feiling konkluderte jeg med at dette ikke var en egnet metode for min forskning. Jeg vendte da tilbake til forskningen for å finne andre metoder som bedre kunne få frem hensikten med min masteroppgave. Jeg kom da over forskning på blant annet skolebøker. Skolebøker er analoge undervisningsmateriell og forskning på disse er derfor sammenlignbart med forskning på undervisningsmateriellene i oppgaven. Denne forskningen hjalp meg videre i prosessen med å finne passende rammeverk for min analyse.

3.3 Utvikling av rammeverk

Gjennom å lese forskning på emnet bestemte jeg meg for å teste Charalambous et al. (2010) analyseverktøy. Charalambous et al. (2010) anvender et rammeverk som beskriver horisontal, vertikal og kontekstuell analyse av tekstbøker. Gjennom modifisering av Charalambous et.al (2010) analyseverktøy kom jeg frem til at dette kom til å fungere godt som et utgangspunkt for mitt analyseverktøy. Sammen med relevant teori kan dette gi meg muligheten til å få frem det overordnede med verktøyene, men også å gå dypt inn i de ulike undervisningsmateriellene.

I den horisontale analysen går Charalambous et al. (2010) inn tekstbokens helhet. Her får de frem et deskriptivt bilde av bøkene gjennom to kategorier: bakgrunnsinformasjon og den gjennomgående strukturen av bøkene. Bakgrunnsinformasjon har som hensikt å gi et beskrivende overblikk av læreboken og dens opprinnelse. Gjennomgående struktur skal få frem hvordan læreboken er organisert.

I min analyse vil dette være hensiktsmessig å anvende, da jeg ønsker å skape et overblikk over verktøyet. Jeg har valgt å se bort fra noen av punktene de har anvendt, da jeg for eksempel ikke anser antall sider som relevant for min analyse. Bakgrunnsinformasjonen jeg har valgt så etter er en beskrivelse av verktøyet, hvem som bruker det og hvem som har gitt det ut. I gjennomstående struktur har jeg valgt å inkludere hvordan en kan finne frem i verktøyet (oppsett), antall undervisningsmateriell og hjelpemidler. Under hjelpemidler ser jeg etter hvilke hjelpemidler verktøyene har tilgjengelig, men som jeg ikke har mulighet til å inkludere i den vertikale analysen grunnet omfanget av oppgaven.

I motsetning til horisontale analysen som tar for seg bokens helhet, så tar den vertikale analysen for seg et og et undervisningsmateriell. Charalambous et al. (2012) anvender tre hovedkategorier: Communicated to students, Required from students og Connections. Disse punktene vinkler seg mot elevenes interaksjoner. Dette passer ikke ukritisk opp mot oppgaven. Jeg har derfor valgt å beholde strukturen på analysen, men endret kategoriene. Analysen baserer seg på hva undervisningsmateriellene kommuniserer, og hvordan det formidler mål, rammer og vurderinger for lærere og elever, samt hvordan de legger opp til arbeid med algoritmisk tenkning. Mine to hovedpunkt under den vertikale analysen ble derfor: Informasjon og algoritmisk tenkning.

Under informasjon ser jeg på hvordan undervisningsmateriellet formidler målet for aktiviteten/oppgaven, rammene som blir satt og hvordan arbeidet vurderes. Under algoritmisk tenkning analyserer jeg oppgaven i lys av rammeverkene til Brennan og Resnick (2012) og Benton et al. (2016).

Charalambous et al. (2010) skriver også om kontekstuell analyse. Kontekstuell analyse baserer seg på intensjonen til forfatteren. Den overordnede intensjonen til verktøyene er grunnen til at de er valgt i utgangspunktet. Verktøyenes hensikt er å anvendes for å lære algoritmisk tenkning og programmering i klasserommet. Å gå dypere inn i hensikten til forfatteren/-e av undervisningsmateriellene er ikke relevant for oppgaven, da jeg tar for meg det som blir formidlet gjennom undervisningsmateriellene. Jeg ser derfor bort fra kontekstuell analyse i oppgaven.

Ifølge Brennan og Resnicks (2012) rammeverk baserer algoritmisk tenkning seg på tre dimensjoner: algoritmiske konsept, praksiser og perspektiv. Deres rammeverk virket interessant å anvende i oppgaven, da de bidrar med et teoretisk grunnlag for analysen.

Etter å ha analysert undervisningsmateriellene med deres teoretiske rammeverk i grunn var jeg fremdeles ikke tilfreds med analysen. Jeg oppdaget at aspekt av undervisningsmateriellet, som jeg ønsket å adressere, ikke kom frem. For eksempel hvordan de kobler algoritmisk tenkning til matematikkfaget.

Det ble altså behov for å tilføye mer teori. Med å søke videre kom jeg over rammeverket til Benton et al. (2017). Her går de inn på de fem e'ene – utforske, forklare, se for seg, utveksle og koble, som jeg har skrevet om i teorikapittelet. Med denne teoretiske utvidelsen av rammeverket ble analyseverktøyet vinklet for oppgaven på hensiktsmessig vis.

De fem e'ene blir presentert i analysen gjennom en tabell hvor de får en rangering fra 1 til 3 for å fremheve hvor godt de kommer frem i undervisningsmateriellet. 1 er lavest rangering og betyr at konseptet ikke ble funnet i analysen. 2 viser at konseptet kan oppstå, men at det ikke kommer tydelig frem eller ikke blir kommunisert av analyseverktøyet. 3 betyr at konseptet kom tydelig frem og ble kommunisert i undervisningsmateriellet. Under er et eksempel på en slik tabell.

Tabell 2

<table>
<tr><td colspan="6">Sammenfatning av valgt verktøy</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="2">Algoritmiske konsept</td><td colspan="3">Algoritmisk praksis</td></tr>
<tr><td colspan="2">-Hendelse
-Sekvens
-Løkke
-Data</td><td colspan="3">-Teste og debugge
-Abstrahering og modularisering</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td></tr>
</table>

Jeg har gjort en justering for konseptet å koble, da alle undervisningsmateriellene kobler algoritmisk tenkning til matematikk gjennom programmering. Konseptet koble får derfor 2 om undervisningsmateriellet kobler algoritmisk tenkning til programmering og 3 dersom det trekker inn flere matematiske kunnskapsområder. Under å koble ser jeg etter kompetansemål som omhandler programmering, disse har jeg gjort rede for i innledningen.

Jeg har gått frem og tilbake i analysen for å redigere analyseverktøyet, da nye funn har gjort det nødvendig å gjøre justeringer. Da jeg forsøkte å analysere igjen, med det nye rammeverket, fikk jeg problemer med å besvare spørsmålet om algoritmiske perspektiv på en hensiktsmessig måte. Jeg undersøkte da hvordan andre hadde anvendt rammeverket til Brennan og Resnick (2012). Bråting og Kilhamns (2021) artikkel om analyse av skolebøker beskriver hvordan de tar utgangspunkt i rammeverket til Brennan og Resnick (2012). De valgte å se bort fra algoritmiske perspektiv, da det ikke var hensiktsmessig i deres analyse av skolebøker i Sverige. Hvorvidt elevene tilegner seg ulike algoritmiske perspektiv, er ikke noe jeg kan måle ut fra undervisningsmateriellet. Noen tilegner seg trolig dette til en viss grad, men hvorvidt spesifikke undervisningsmateriell vil gi elevene et annerledes perspektiv på hvordan teknologi fungerer er ikke noe jeg kan konkludere med. Jeg innså da hvorfor jeg hadde problemer med å undersøke dette og konkluderte med at det ville være hensiktsmessig også for min analyse å se bort fra dette aspektet.

Å utrede et hensiktsmessig analyseverktøy for oppgaven har vært krevende. Etter mange runder med analyse og mange lite tilfredsstillende resultat har jeg til slutt utformet noe jeg mener belyser ulike aspekt ved algoritmisk tenkning, i undervisningsmateriell på hensiktsmessig vis.

Tabell 3

Horisontal analyse	
Bakgrunnsinformasjon	Gjennomgående struktur
Hvem har gitt ut verktøyet? Hva er verktøyet? Hvem bruker verktøyet?	Oppsett Antall undervisningsmateriell Hjelpemidler
Vertikal analyse	
Informasjon	Algoritmisk tenkning
Hvordan blir målet for aktiviteten/oppgaven presentert? Hvilke rammer blir satt for aktiviteten? Samarbeid, individuelt arbeid, tilpasninger for ulike elever, før- og etterarbeid, kompetanse, tidsramme etc. Hvordan vurderes arbeidet til elevene? *Rett/feil løsninger, tilbakemeldinger, løsningsmetoden etc.*	Dimensjonene av algoritmisk tenkning: Hvilke algoritmiske konsept kan en finne? *Sekvens, løkke, parallell, hendelse, betingelser, operatører og data* Hvilken algoritmisk praksis kan en finne? *å være inkrementell og iterativ, teste og «debugge», gjenbruk og remiksing, abstrahere og modularisere* De fem e'ene: Utforske Forklare Se for seg Utveksle Koble

3.4 Utvalg

Innenfor rammene til masteroppgaven er det ikke mulig å inkludere alle verktøyene skolene har tilgang på. Det er derfor inkludert tre undervisningsmateriell tihørende tre verktøy, i analysen. Følgende tre verktøy er Salabys kodeskole, KUBO og Spheroball. Dette er tre verktøy med fellesnevneren at de skal brukes for å arbeide med algoritmisk tenkning og programmering i skolen. De tre tar i bruk tre ulike former for blokkprogrammering, men det er bare Sphero som også har muligheten til å bruke tegning og JavaScript. JavaScript er et tekstbasert programmeringsspråk. KUBO er gitt ut i Danmark, Kodeskolen er gitt ut i Norge og Sphero Ball er gitt ut i USA, alle tre blir anvendt i norsk undervisningssammenheng. De er alle fra vestlige land, men jeg mener at de allikevel kan bidra med ulike tilnærminger til arbeid med algoritmisk tenkning i skolen.

3.4.1 Kodeskolen

Når jeg analyserer Kodeskolen, inkluderer jeg videoene tilhørende verdenen til hvert av brettene. Videoene i verktøyet blir brukt for å introdusere de nye verdenene, og da også det nye settet med brett og eventuelle nye blokker. Jeg ser derfor på disse som en overordnet oppgavetekst og en del av undervisningsmateriellet jeg har fremfor meg. Videoene vil ikke bare hjelpe med å få et bedre innsyn i verktøyet som helhet, men også med å få et mest mulig nøyaktig inntrykk av hva brettene innebærer og formidler. Jeg mener at analysen blir mer gyldig dersom jeg tar for meg hele undervisningsmateriellet, da dette vil gi et riktigere bilde av realiteten. Fokuset i analysen er på hvordan undervisningsmateriellet kommuniserer brettene, målene og det faglige innholdet, med fokus på algoritmisk tenkning.

Figur 2

Jeg har valgt å ta for meg tre undervisningsmateriell i Kodeskolen. Fabrikk – brett 1, som er det første undervisningsmateriellet i Kodeskolen. Kjøkken – brett 2, som er omtrent halvveis gjennom verktøyet. Kloakk – brett 6, som er det siste undervisningsmateriellet i Kodeskolen. Dette fordi elevene må arbeide fra start til slutt i verktøyet og jeg mener det er interessant å se hvordan undervisningsmateriellet utvikler seg etter hvert som elevene arbeider med det. Datamaterialet ble hentet 10.02.2022.

3.4.2 KUBO

KUBO har et nettbasert kurs for lærere, et verktøy for å lage egne kart og to apper. Appen MY KUBO brukes for å oppdatere programvaren til KUBO. Appen KUBO PLAY er en digital versjon av KUBOs robot. Dette er en digital plattform for læring av koding og kan anvendes både hjemme og på skolen. Jeg velger å ikke analysere KUBO PLAY da jeg har valgt å ta for meg KUBO på grunn av muligheten for å arbeide med roboten og brikkene fysisk.

Figur 3

I analysen tar jeg for meg læreplanen og lærerguiden, videre refereres disse to til som undervisningsmateriellet tilhørende KUBO. Disse kan en få på flere språk. Jeg ser på de norske versjonene, da det er dette som trolig er i bruk i norske klasserom.

Per 10.03.2022 har KUBO Coding to utvidelser som er lansert i Norge, KUBO Coding + og KUBO Coding ++. De har lansert en utvidelse til, men den har ennå ikke kommet til Norge, KUBO Math. Jeg anser denne utvidelsen som en god tilføyelse til oppgaven min da den spesifikt tar for seg matematikkfaget og selv om jeg ikke har tilgang på brikkene så har jeg tilgang på undervisningsmateriellet.

Jeg har valgt å ta for meg det første undervisningsmateriellet tilhørende KUBO for å se hvordan de introduserer brukeren/-e for verktøyet, KUBO Coding: Routes. Jeg ønsker å se videre på et undervisningsmateriell som fremdeles har koding i sentrum og valgte da KUBO Coding ++: Super Coders. Som nevnt over ønsker jeg også å se på utvidelsen KUBO Coding Math, der har jeg valgt det siste undervisningsmateriellet på nettsiden: Advancing. Datamaterialet ble hentet 11.02.2022.

3.4.3 Spheroball

Sphero har lansert flere roboter som er ment for arbeid med programmering i undervisningssammenheng. De har også en nettside med informasjon om hvilke produkt de har, samt undervisningsmateriell som kan anvendes med en eller flere av robotene deres. Roboten har en egen app, videre referert til som SpheroEdu-appen, som kan brukes både for å programmere ballen og for å gi ut oppgaver til klassen. Læreren oppretter en

Figur 4

lærerbruker i appen og legger inn klassen. Ballen kan tilkobles en telefon/nettbrett om gangen.

Det har vist seg at det er noe forskjell mellom undervisningsmateriellet når det er hentet fra nettsiden og når det er hentet fra appen. Nettsiden kan virke som oppgaveark en gir til elevene, mens SpheroEdu-appen tilbyr informasjon til læreren i tillegg. Jeg har derfor valgt å ta for meg undervisningsmateriell fra både appen og nettsiden. Det har vært litt problemer med språk i dette verktøyet, da det originalt er skrevet på engelsk. De norske oversettelsene har vært vage og fagbegrep har mistet mening etc. Utklippene fra nettsiden er på engelsk, dette gjøres for å ikke miste noe vesentlig innhold fra undervisningsmateriellet. Originalt tenkte jeg også å hente skjermbilder fra appen på engelsk, men denne får jeg bare i norsk versjon.

Når jeg har valgt undervisningsmateriellet fra Sphero har jeg begrenset søket mitt til at jeg ønsker å se på spesifikt på roboten Sphero. Sphero er utgiver av flere ulike roboter, men jeg har begrenset meg til en robot grunnet oppgavens begrensninger. Videre har jeg begrenset undervisningsmateriellene til bare de som er laget av Sphero og krysset av for at undervisningsmateriellet skal være knyttet til matematikk. Analysen av Sphero er noe begrenset grunnet betalingsmurer, jeg tar som forbehold at undervisningsmateriellene ikke kommer med ekstra kostnad.

Jeg velger å ta for meg både materiell som bygger på tegning som programmeringsspråk, blokkprogrammering og de som har tekstprogrammering. I kompetansemålene for programmering i matematikk står det ikke noe om tekstprogrammering eller tegning som programmeringsspråk, men jeg anser det som hensiktsmessig å inkludere disse da varierte undervisningsmateriell gjør det mulig å tilpasse undervisningen til elevene som mestrer programmering godt og mindre godt. Datamaterialet ble hentet 12.02.2022.

3.5 Reliabilitet og validitet

Reliabilitet og validitet er to begrep som omhandler kvaliteten av datamaterialet og danner forskerens troverdighet (Grønmo, 2016). I dette kapittelet skal jeg ta for meg reliabiliteten og validiteten av meg som forsker og oppgaven.

3.5.1 Reliabilitet

Grønmo (2016) skriver at reliabilitet beskriver hvor pålitelig datamaterialet er. Christoffersen et al. (2021) knytter reliabilitet til undersøkelsens data. For kvalitative studier skriver

Christoffersen et al. (2021) at forskeren må være åpen om fremgangsmåten gjennom hele forskningsprosessen, slik jeg har forsøkt å være i beskrivelsen av hvordan jeg har satt sammen rammeverket for analysen. Postholm og Jacobsen (2018) definerer reliabiliteten som forskningsresultatenes konsistens og at det skal være mulig å gjenskape resultatet av andre forskere.

Høy reliabilitet er krevende å oppnå i kvalitative studier (Larsen, 2007) da forsker, forskningsfelt og mennesker som deltar i studien vil variere og derfor skape ulike utfall (Postholm og Jacobsen, 2018). Reliabilitet setter spørsmål til om andre forskere vil få de samme resultatene dersom de anvender den samme metoden (Thagaard, 2018). Postholm og Jacobsen (2018) kobler reliabilitet til hvorvidt forskeren reflekterer rundt hvordan hen har påvirket undersøkelsen og resultatet. De beskriver to måter å gjøre dette på, gjennom at forskeren reflekterer over egen påvirkning og at hen gjør forskningsprosessen tydelig slik at andre kan reflektere rundt den.

I metode-kapittelet gikk jeg inn i hvordan hermeneutikken er relevant for meg som forsker, da mine fordommer og tolkninger vil påvirke ikke bare utfallet, men fremgangsmåten og datamaterialet mitt. Dette, i tillegg til å forsøke å tydeliggjøre forskningsprosessen min gjennom å beskrive forandringer jeg har gjort underveis i metode-kapittelet, er tiltak som er gjort for å fremme reliabilitet i min forskning.

3.5.2 Validitet

Ifølge Larsen (2007) har kvalitativ metode høyere validitet enn kvantitativ metode da en har mulighet til å gå tilbake i materiellet og gjøre korrigeringer. Validitet blir inndelt i tre kategorier av Christoffersen et al. (2021), intern validitet, ekstern validitet og objektivitet.

Grønmo (2016) skriver at intern validitet handler om at eksperimentet er gjennomført på en tilfredsstillende måte, Christoffersen et al. (2021) beskriver at dette omhandler hvilken grad fremgangsmåtene og funn gjenspeiler studiets hensikt og hvorvidt det representerer virkeligheten på en mest mulig riktig måte. I studiets gang har jeg vært bevisst i egne fordommer og arbeidet aktivt mot å fremstille analysen og funnene på en mest mulig rettferdig måte.

Christoffersen et al. (2021) beskriver ekstern validitet som om forskeren lykkes med å etablere beskrivelser, begreper, fortolkninger og forklaringer, og hvorvidt disse kan overføres til andre felt enn det som forskes på. Thagaard (2018) beskriver at overførbarheten til en

studie kommer an på hvorvidt tolkingen av det kan være relevant i andre sammenhenger. Forskeren kan styrke overførbarheten gjennom grundige beskrivelser som kan gjøre det enklere for utenforstående å bedømme om resultatene kan overføres til andre områder (Christoffersen et al., 2021). Forskningen min tar utgangspunkt i datamateriale som er anvendt i norsk skole, noen i større grad enn andre. Det kan være mulig å overføre beskrivelser, begreper, fortolkninger og forklaringer til andre områder, som andre verktøy eller lærebøker med tilsvarende bruksområde, men i hovedsak beskriver undersøkelsen utvalget jeg har tatt for meg.

Christoffersen et al. (2021) beskriver at objektivitet omhandler at forskeren skal stille seg mest mulig nøytral og upartisk til forskningen. De skriver at en kan styrke forskerens objektivitet med å beskrive beslutninger på hensiktsmessig vis slik at leseren kan vurdere disse. For eksempel da jeg tok beslutningen om å ikke inkludere algoritmiske perspektiv, da jeg ikke hadde passende datamateriale og undersøkte hvordan andre som hadde anvendt Brennan og Resnick (2012) løste det. Bråting og Kilhamn (2021) hadde konkludert med det samme som jeg. Dette styrker, slik jeg forstår Christoffersen et al. (2021), min objektivitet. Beslutningen var ikke basert på at noe var krevende, men heller at det ikke var hensiktsmessig for min eller andre tilsvarende undersøkelser.

3.5.3 Forskningsetiske betraktninger

Min oppgave kan aldri bli 100% objektiv. Jeg velger selv ut teorien jeg skal bruke, hvilke verktøy jeg skal analysere og hvordan dataene blir analysert. Som forsker er jeg forpliktet til å holde mine subjektive meninger tilbake og holde meg mest mulig saklig, kritisk og objektiv, men sannheten er at meningen min kommer til å påvirke formuleringen min (Christoffersen & Johannessen, Forskningsmetode for lærerutdanningene, 2018). Konsekvensen av anvendelse av hermeneutisk analyse av teksten er at analysen vil bære preg av den som står bak tolkningen (Leseth og Tellmann, 2014), altså meg. Det er ikke meningen å henge ut noen verktøy som mindre gode enn andre, målet er bare å se på hva som kjennetegner dem gjennom rammeverkene av Brennan og Resnick (2012) og Benton et al. (2016).

Innhentingen av data er, ifølge Christoffersen & Johannessen (2012), en selektiv prosess. Verktøyene jeg tar for meg har jeg valgt ut basert på egen kunnskap og egne fordommer. Utvalget er påvirket av forskningen jeg har lest, folkene jeg har snakket med og egne erfaringer. Fra mitt perspektiv er dette interessante ressurser å undersøke for en

masteroppgave. Utvalget mitt kan kritiseres da alle de tre verktøyene ble til i vestlige land. Det kan føre til at jeg finner nokså like vinklinger i undervisningsmateriellene. Det kunne vært interessant å ta for seg de mest brukte verktøyene fra flere ulike land og kontinenter, men dette valgte jeg å ikke gjøre da jeg ønsket å ta for meg noe av det som blir brukt i norsk skole etter den nye læreplanen ble tatt i bruk.

Som forsker har jeg et ansvar for å unngå skade (Christoffersen & Johannessen, 2018). I en dokumentanalyse som denne står det ikke om akutt liv og død, men jeg har et ansvar for å fremstille verktøyene på et rettferdig vis. Ingen individ blir identifisert, det kommer bare frem hvilke selskap som står bak verktøyene. Det kommer ikke frem at en person gjør en mye bedre jobb enn andre, så publisering av denne analysen vil trolig ikke påvirke noens karriere. Alle selskapene er informert om at jeg skriver om deres verktøy i masteroppgaven, da dette ble opplyst i e-posten jeg sendte for å hente data om antall brukere i Norge.

4 Analyse

4.1 Kodeskolen

Hensikten med dette kapittelet er å analysere et utvalg undervisningsmateriell fra kodeskolen. I den horisontale analysen er formålet å få en oversikt over hva verktøyet inneholder. Da ser jeg på de to kolonnene bakgrunnsinformasjon og gjennomgående struktur. I den vertikale analysen tar jeg for meg tre undervisningsmateriell, jeg går inn i et og et brett med de tilhørende videoene. Undervisningsmateriellet blir analysert i lys av rammeverket jeg har satt sammen av Benton et al. (2016) og Brennan og Resnick (2012).

4.1.1 Horisontal analyse

I dette kapittelet tar jeg for meg det overordnede rundt undervisningsmateriellet tilhørende kodeskolen. I bakgrunnsinformasjon ønsker jeg å få frem hvem verktøyet er laget av, hvem det er laget for og hvor mange brukere verktøyet har i Norge. I gjennomgående struktur ønsker jeg å ta for meg antall undervisningsmateriell, lærerveiledninger og verktøyets oppsett.

4.1.1.1 Bakgrunnsinformasjon

Utgiveren av kodeskolen er Gyldendal. Ifølge Salaby sine nettsider inneholder teamet deres pedagoger, designere, illustratører, programmerere og tekstforfattere (Gyldendal, 2022). Salaby arbeider med å skape innhold opp mot fagfornyelsen og nettsiden blir stadig oppdatert med aktuelle «temapakker».

Kodeskolen fra Salaby er en nettside, så elevene har behov for pc/nettbrett for å kunne anvende verktøyet. I kodeskolen skal elevene styre en robot med bruk av blokkprogrammering. Verktøyet inneholder en historie rundt Sirkusdirigent Salami, som ønsker å redde biene, roboten hans og en fortellerstemme. Verktøyet er ment for elever i 1.-7. trinn.

For å finne ut hvor stort dette verktøyet er tok jeg kontakt med Gyldendal. I en e-post kunne de fortelle at 84% av alle elever i 1.-7. trinn bruker Salaby. Kodeskolen er blant de topp tre mest brukte verktøyene de leverer (Se vedlegg 1). Så et nøyaktig antall brukere kan jeg ikke konkludere med, men at det er mye brukt i norsk skole kan man trygt si.

4.1.1.2 Gjennomgående struktur

Kodeskolen er en nettside hvor elevene kan løse brett gjennom anvendelse av blokkprogrammering. Brettene i kodeskolen har navn etter hvor langt i historien brukeren har komt. Hver verden har et navn, som fabrikk eller kjøkken og brettene har navn etter hvilken verden en er på, for eksempel: fabrikk – brett 2.

For å finne frem til kodeskolen går elevene inn på salaby.no og logger inn med brukernavn og passord. Brukeren kan gå inn på hvilket som helst trinn, trykke på temasider og finne kodeskolen der. Du finner også kodeskolen i matematikkfaget hos Byparken (siden for 3. og 4. trinn) og Kanal S (siden for 5.-7. trinn). Der møter elevene sirkusdirektør Salami som har fått problemer med at det nesten er tomt for bier i Salaby. Han bygger en robot, men vet ikke hvordan han skal styre den. Elevene må altså hjelpe Salami med å programmere roboten til å fange bier.

Når en går inn på kodeskolen er det en instruksjonsvideo elevene kan velge å se, den gir oppgavene en historie og hver verden har en egen video der vi blir introdusert til nye problemer Salami og roboten møter på og hvordan de kan håndteres med nye funksjoner. Det er i alt seks verdener med seks brett. Altså er det i alt 36 brett, der det gradvis blir lagt inn nye funksjoner. For hver nye verden får vi se en ny video om hvordan det går med sirkusdirektøren og roboten. Det er i alt seks videoer.

Kodeskolen har en lærerveiledning. Den viser løsningen på noen av de «vanskeligste brettene». I beskrivelsen skriver de at lærerveiledningen skal inneholde hint og løsninger. Her kommer det frem at noen av brettene kan ha flere løsninger, og noen av brettene viser de halve løsningen på. Det virker som at det er disse som blir referert til som hint på løsningen. Lærerveiledningen består altså av skjermbilder av løsninger på noen av brettene. Her finner en også informasjon om hvordan læreren kan gå frem dersom de skulle oppleve tekniske problemer. For eksempel hvis en elev opplever at all fremgangen hens har forsvunnet så kan læreren ta kontakt med Salaby som kan hjelpe med å gi tilbake fremgangen. Lærerveiledningen er ikke inkludert i den vertikale analysen da det ikke er lærerveiledning til hvert brett, men heller en fasit.

4.1.2 Vertikal analyse, Fabrikken – brett 1

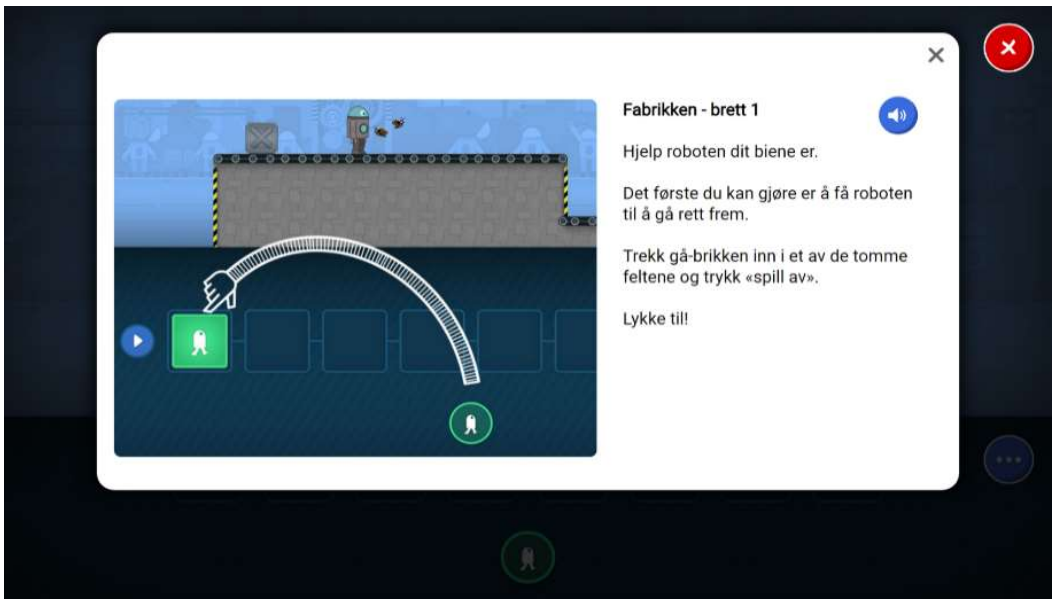
I dette kapittelet skal jeg ta for meg undervisningsmateriellet Fabrikken – brett 1. Dette er det første brettet fra den første verdenen. Jeg tar da for meg den første videoen og det første

brettet tilhørende kodeskolen. Det er disse to jeg videre referer til som undervisningsmateriell for denne Fabrikken – Brett 1. Jeg går inn i hvordan og hvem undervisningsmateriellet informerer til, og hvordan algoritmisk tenkning kommer frem gjennom analysen. Rammeverket, basert på Brennan og Resnick (2012) og Benton et al. (2016), brukes for å analysere hvordan undervisningsmateriellet legger opp til anvendelse av algoritmisk tenkning.

4.1.2.1 Informasjon

Figur 5

Figur 6



Figur 7

Som vist i figur 5, 6 og 7 får elevene beskrevet at målet med oppgaven er å redde biene, både i videoen og i oppgaveteksten. Oppgaveteksten kommer i tillegg med instrukser til hvordan løse brettet og kommuniserer at målet er «Hjelp roboten dit biene er». En løser brettet med å kode roboten til å bevege seg til ruten biene befinner seg i.

Kodeskolen definerer ikke noen rammer for brettet. Om elevene skal arbeide individuelt eller i samarbeid blir ikke spesifisert. I videoen blir eleven/-e referert til som «du», jeg velger

derfor å tro at brettene er ment til individuelt arbeid. Eleven har mulighet til å løse brettet igjen dersom hen ikke er fornøyd med antall bier, eller bare har lyst. Det er altså mulig å gjøre brettet igjen, men undervisningsmateriellet legger ikke opp til noen form for etterarbeid da dette er valgfritt. Hvor mange elever som faktisk vil gjøre brettet om igjen er ikke noe jeg kan konkludere med. Videoen kan beregnes som førarbeid, men denne er valgfri å se.

Dette undervisningsmateriellet beskriver gjennom illustrasjon og tekst i oppgaveteksten hvordan eleven skal løse brettet. Teksten kan spilles av, da den er lest inn. Undervisningsmateriellet er derfor tilpasset slik at elever som ikke kan lese, eller har utfordringer med å hente ut informasjon fra tekst, fremdeles kan få med seg hvordan brettet skal løses. Undervisningsmateriellet avklarer ikke for eleven eller læreren om elevgruppen trenger noen form for forforståelse eller kompetanse innen programmering, matematikk eller algoritmisk tenkning for å kunne løse dette brettet.

Figur 8

Figur 8 er et skjermbilde av brett 1, Fabrikk. Her kan en se hva elevene arbeider med, og hvilke alternativ de har tilgjengelig. Elevene kan trykke på de blå sirklene på høyre side. Der kan de velge om rutenettet skal være synlig eller ikke, med å trykke på den blå sirkelen med en «#» i. Rutenettet gjør det tydeligere hvor langt roboten går med bruk av en gå-funksjon, noe som gjør det mulig for elevene å telle hvor mange gå-funksjoner roboten trenger for å komme til biene. Dette er en form for tilpasning elevene kan styre selv. Den øverste sirkelen, i'en, gjør at oppgaveteksten dukker opp igjen og den nederste med en pil som går i sirkel gjør at de starter brettet på nytt.

Elevene kan bruke så mye tid de ønsker på brettet. De blir ikke tatt tiden på og de blir ikke avbrutt på noe vis dersom de bruker lang tid.

Figur 9

Figur 10

Figur 9 og 10 er to mulige tilbakemeldinger elevene kan få på dette brettet. Etter å ha fullført det første brettet får de tilbakemeldingen «Gratulerer. Helt perfekt! Du fanget alle biene! Du brukte 1 funksjoner» eller «Gratulerer. Helt perfekt! Du fanget alle biene! Du brukte 8 funksjoner».

Jeg valgte å løse brettet både «feil» og «riktig» for å se om tilbakemeldingen forandret seg. Det eneste som ble forandret var antallet funksjoner som var beskrevet i tilbakemeldingen. Løsningen vurderes etter rett/feil løsninger av verktøyet og i dette brettet er løsningen uansett rett. De blir vurdert ut fra antall funksjoner, dette har de ikke fått beskrevet før de tar fatt på første brett.

4.1.2.2 Algoritmisk tenkning

Figur 11

Figur 11 viser brettet i fabrikk – brett 1, det er dette elevene har fremfor seg når de starter på oppgaven.

Den grønne firkanten, i figur 11, er en gå-funksjon jeg har lagt inn i kodefeltet. Nedenfor er det en grønn sirkel, dette er den ene funksjonen som er tilgjengelig i dette brettet. Oppgaveteksten referer til funksjonen som en gå-brikke. Hver brikke i kodeskolen tilsvarer en funksjon. Når jeg analyserer dette undervisningsmateriellet i lys av Brennan og Resnick (2012) kommer gå-brikken i kodefeltet frem som en *sekvens*. Koden kjøres når brukeren trykker på spill av knappen (den blå sirkelen til venstre for kodefeltet, som vist i figur 11), dette er en *hendelse*. I dette undervisningsmateriellet har vi da de *algoritmiske konseptene: hendelse* og *sekvens*.

Elevene får en beskrivelse av hva de skal gjøre, gjør det og får uansett (gitt at de klarer å flytte brikken til kode-feltet) tilbakemelding om at alt er riktig. Dette gjør at elevene ikke arbeider med de *algoritmiske praksisene* presentert av Brennan og Resnick (2012).

Videre analyserer jeg undervisningsmateriellet i lys av Benton et al. (2016) rammeverk og ser da etter utforsking, forklare, å se for seg, utveksle og koble.

Før elevene tar fatt på brettet får de forklart gjennom tekst og illustrasjon hva som skal gjøres. For å løse brettet skal de ta en brikke inn i kode-feltet. De får forklart at brikken er en gå-brikke og at «det første du må gjøre er å få roboten til å gå rett frem». Dette undervisningsmateriellet legger ikke opp til *utforsking*, slik som det blir presentert i Benton et al. (2016), da løsningen og brikken blir forklart både gjennom tekst og illustrasjon, før elevene tar fatt på oppgaven. Siden elevene får rett på denne oppgaven uavhengig av hvor mange funksjoner de legger inn i kode-feltet blir det heller ikke en situasjon hvor de trenger å debugg».

Undervisningsmateriellet legger ikke opp til at elevene skal *forklare* det de har lært, eller hvordan de har løst det, da de bare skal løse brettet og gå videre til neste.

Det er mulig for elevene *å se for seg* utfallet siden koden ligger i kode-feltet helt til elevene trykker på start-knappen. Elevene trenger ikke å planlegge en rute eller hva de skal gjøre da de får riktig svar uansett hvor mange ganger de ber roboten om å gå frem. Undervisningsmateriellet legger ikke opp til at elevene skal se for seg løsninger eller reflektere over utfallet, men det er mulig at elevene gjør det metoden.

Oppgaven oppfordrer ikke elevene til å *utveksle* og bygge på andres ideer, da den er laget for individuelt arbeid.

Undervisningsmateriellet *kobler* algoritmisk tenkning til programmering, som er en del av matematikkfaget. Jeg tar for meg alle kompetansemålene for 1. – 7. trinn og fant et som elevene vil arbeide med gjennom dette undervisningsmateriellet, da de følger regler i spillet for å løse brettet:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill

Nedenfor er en tabell som oppsummerer funnene av algoritmisk tenkning. Brennan og Resnicks (2012) kategorier blir listet opp for å tydeliggjøre hva som ble funnet, Benton et al. (2016) konsept får et tall mellom 1 og 3. 1 viser til at konseptet ikke ble funnet i analysen, 2 viser til at konseptet kommer frem til en viss grad og 3 viser til at konseptet kom tydelig frem gjennom analysen.

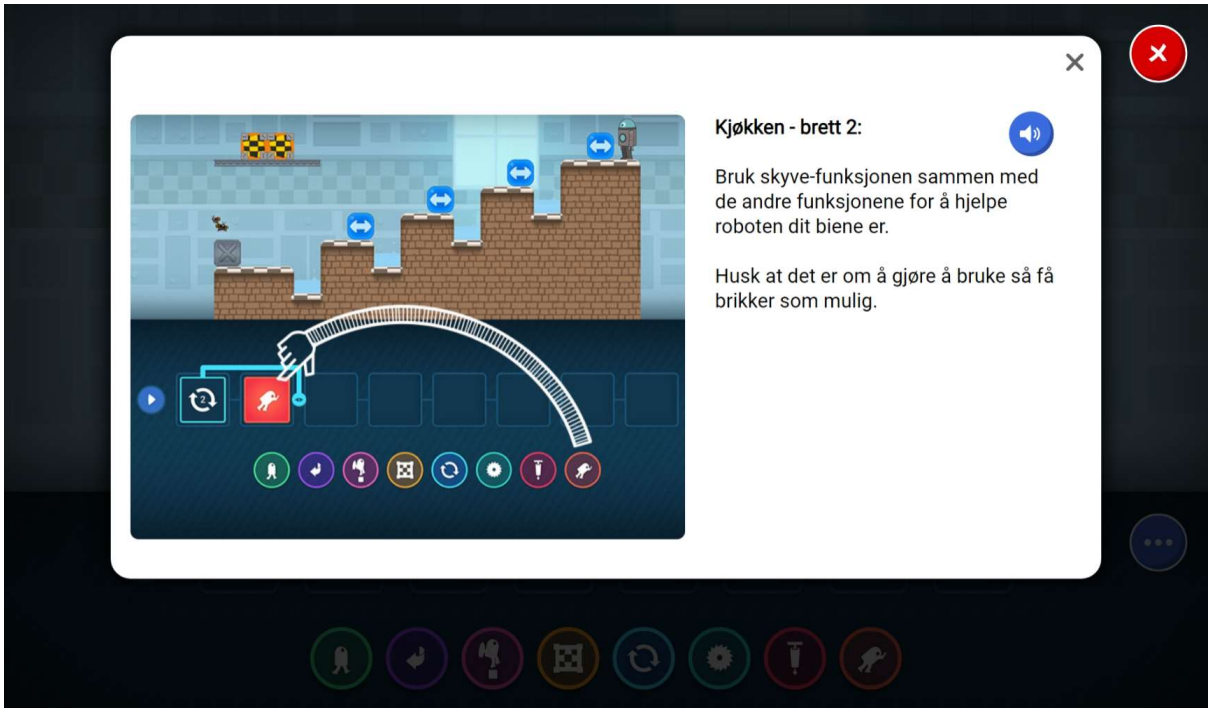
Tabell 4

Sammenfatning av Fabrikk – brett 1					
Brennan og Resnick (2012)	Algoritmiske konsept			Algoritmisk praksis	
	-Hendelse -Sekvens			O	
Benton et al. (2016)	Utforske	Forklare	Se for seg	Utveksle	Koble
	1	1	2	1	2

4.1.3 Vertikal analyse, Kjøkken – brett 2

I dette kapittelet skal jeg analysere undervisningsmateriellet tilhørende Kjøkken – brett 2. Dette gjør jeg ved hjelp av analyseverktøyet som er basert på Benton et al. (2016) samt Brennan og Resnick (2012). Kjøkken er den fjerde verdenen i kodeskolen. Brett 2 blir tilgjengelig etter de har fullført brett 1 i Kjøkken, Kjøkken blir tilgjengelig etter de har fullført oppgavene i Fabrikk, Industri og Hage, elevene har allerede løst 18 brett når de tar fatt på dette brettet.

4.1.3.1 Informasjon



Figur 12

Målet med brettet er å få roboten til biene, med bruk av færrest mulig funksjoner. Dette kommer frem gjennom oppgaveteksten, som vist i figur 12 «… hjelpe roboten dit biene er» og «Husk at det er om å gjøre å bruke så få brikker som mulig».

Verken elever eller lærer får spesifisert rammer av oppgaveteksten eller videoen. I videoen til fabrikk ble det referert til eleven/-e som «du», siden det ikke har kommet frem mer informasjon om dette går jeg ut fra at dette brettet også er ment å løses individuelt.

Figur 13

Figur 14

Før dette brettet kan videoen for Kjøkken anses som en form for førarbeid, i den blir de introdusert for den nye skyve-funksjonen, som vist i figur 13 Det er ikke nødvendig å ha sett videoen for å kunne løse brettet, så hvorvidt alle elevene velger å se den eller ikke er ikke mulig for meg å si. Etter oppgaven får eleven tilbakemelding på hvordan det har gått og muligheten til å starte oppgaven på nytt, altså er det ikke noe etterarbeid, men en mulighet for å prøve igjen dersom en ikke er fornøyd med tilbakemeldingen.

På dette brettet, slik som i fabrikk – brett 1, er det programmet som vurderer arbeidet. Elevene blir vurdert ut fra hvor mange bier de klarer å fange, de kan fange mellom en og tre bier. Antall bier minsker dersom de bruker flere funksjoner enn det som er anerkjent som helt riktig.

Figur 15

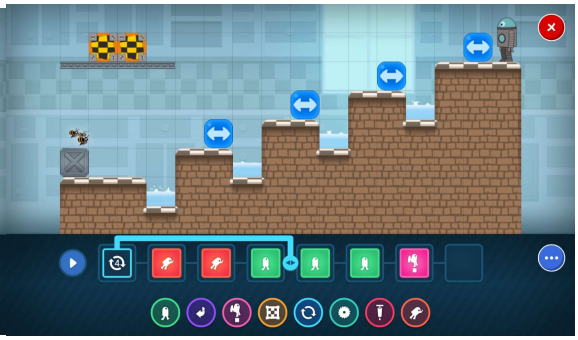
Figur 16

Figur 17

Figur 18

Tilbakemeldingen elevene kan få for dette brettet er vist i figurene over. Dersom de løser brettet med færrest mulig funksjoner får de tilbakemeldingen: «Gratulerer. Helt perfekt! Du fanget alle biene! Du brukte 4 funksjoner» (Figur 15).
Dersom eleven løser den med litt flere funksjoner, får hen tilbakemeldingen: «Supert! Du klarte å fange 2 bier. Du brukte 6 funksjoner» (Figur 16).
Dersom eleven brukte for mange funksjoner, kan hen få tilbakemeldingen: «Gratulerer. Veldig bra! Du klarte å fange 1 bie! Du brukte 15 funksjoner. Tips: prøv å bruke færre funksjoner neste gang» (Figur 17).
Dersom eleven får roboten til å falle utfor brettet eller i vannet får hen beskjeden: «prøv igjen!» (Figur 18), eleven har da muligheten til å løse brettet igjen.

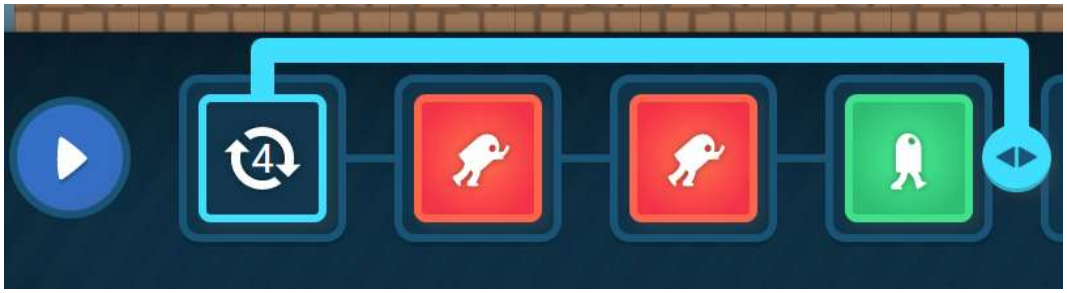
Figur 19



Figur 20

I figurene 19 og 20 har jeg «bygget» to ulike koder for å løse brettet. Her kommer det tydelig frem at det er antall funksjoner elevene blir vurdert etter og ikke nødvendigvis den mest effektive løsningen. Den rosa funksjonen/brikken gjør at roboten flyr opp en rute, en rute fremover og så ned, altså forflytter roboten seg fra utgangspunktet og gjennom tre ruter. Denne funksjonen er i en løkkefunksjon (den blå brikken med tallet 4) som da fører til at roboten gjentar dette fire ganger og den forflytter seg da gjennom åtte ruter før den ender opp hos biene. I figur 20 kan en se at jeg har valgt å få roboten til å gå to ganger, etter løkken, før roboten hopper opp til ruten med biene i. Den grønne funksjonen forflytter roboten bare en rute fremover og i stedet for åtte ruter forflytter roboten seg gjennom fire ruter. Denne koden er altså mer effektiv, i form av at roboten forflytter seg gjennom færre ruter, men gir «dårligere» vurdering da det krever bruk av flere funksjoner.

4.1.3.2 Algoritmisk tenkning



Figur 21

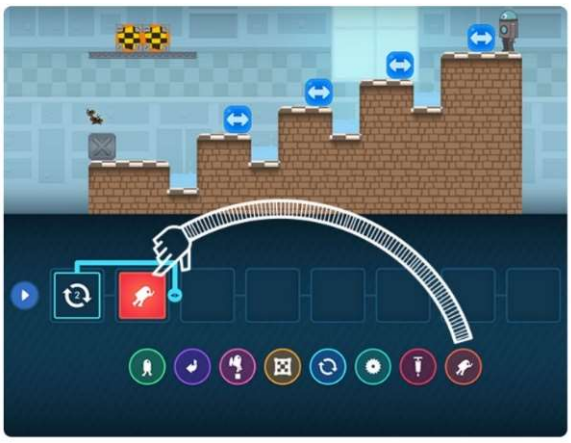
Figur 21 over er et utklipp av en *sekvens* i kodefeltet i Kodeskolen. I figuren kan en se de tre *algoritmiske konseptene*, som beskrevet av Brennan og Resnick (2012), *sekvens*, *hendelse* og *løkker*. De fire brikkene i kodefeltet er funksjoner som til sammen former en *sekvens*. *Løkken* beskriver hvilken del av *sekvensen* som skal gjentas, altså beskriver brikken med tallet 4 at de tre neste brikkene skal gjentas fire ganger. Når elevene trykker på start-knappen er dette en *hendelse* som fører til at koden kjøres.

Dersom en ikke klarer dette brettet på første forsøk får en mulighet til å prøve igjen, eleven kan altså teste ulike løsninger. Altså kan en finne den *algoritmiske praksisen: teste og debugge*. Etter eleven har kjørt koden forsvinner den, det gjør at selve debuggingen kan bli noe utfordrende. En kan altså debugge dersom en husker sekvensen en kjørte. At alle elevene kommer til å arbeide med dette er ikke garantert. Det er mulig noen av dem kan dette godt, eller er fornøyd uten å fange alle tre biene, og løser brettet bare en gang. Da vil ikke den algoritmiske praksisen: teste og debugge ta sted.

Det er en begrensing på antall funksjoner i kode-feltet, som en kan se i figur 21 Dette brettet kan løses med å skrive inn i kode-feltet bare en gang. Det er ikke garantert at alle gjør det på første forsøk. Dette åpner for å skrive koden i mindre parti, i stedet for bare en lang kode. Altså kan en på et vis arbeide med *abstrahering og modularisering*. Dette blir ikke kommunisert gjennom undervisningsmateriellet, men er mulig på grunn av begrensingen i kode-feltet.

Figur 22

Figur 22 viser et skjermbilde av videoen tilhørende kjøkken. Her blir vi introdusert til robotens nye evne, å skyve blå kasser. Nytten av denne egenskapen illustreres, nå kan roboten skyve disse blokkene ut i vannet slik at den kan komme seg tørrskodd over vannet. I oppgaveteksten, som vist i figur 12, står det: «Bruk skyve-funksjonen sammen med de andre funksjonene for å hjelpe roboten dit biene er». Dette begrenser elevenes mulighet til å *utforske* den nye funksjonen.



Figur 23

Figur 23 viser illustrasjonen elevene får opp sammen med oppgaveteksten før de tar fatt på brettet. Den illustrerer elevene må bruke både løkkefunksjonen og skyvefunksjonen for å fange tre bier. Noen av de andre funksjonene fører til at roboten løfter på skuldrene eller at den går utfor på høyre side av brettet. De andre gjør at elevene ikke kan løse brettet med færrest mulig funksjoner. Undervisningsmateriellet legger i liten grad opp til *utforskende* arbeid, slik jeg tolker Benton et al. (2016). Det er mulig elevene *utforsker* ulike løsninger, debugger og tester de ulike funksjonene, da de kan løse brettet så mange ganger de ønsker. Dette gjør det mulig å *utforske* noe, men det er ikke noe undervisningsmateriellet kommuniserer at de skal gjøre.

Undervisningsmateriellet legger ikke opp til at elevene skal *forklare* resultat eller løsningsmetode.

Undervisningsmateriellet kommuniserer ikke at elevene skal *se for seg* utfallet eller reflektere over det etterpå, men det er mulig at elevene gjør nettopp det mens de skriver koden og planlegger hvor roboten skal gå.

Undervisningsmateriellet legger ikke opp til at elevene skal *utveksle* spørsmål og ideer med medelevene, da de arbeider individuelt og ikke får instrukser om noe annet.

Oppgaven *kobler* matematikk i form av begrep og konsept som er typisk for programmering, som etter fagfornyelsen ble en del av matematikkfaget. Elevene følger regler i spillet som en kan finne igjen i et kompetansemål for 2. trinn. I oppgaveteksten anvendes også begrepet funksjon og elevene må bruke løkker for å fange både to og tre bier, disse finner vi igjen i to av kompetansemålene under:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker

Nedenfor er en tabell som oppsummerer funnene av algoritmiske tenkning i undervisningsmateriellet.

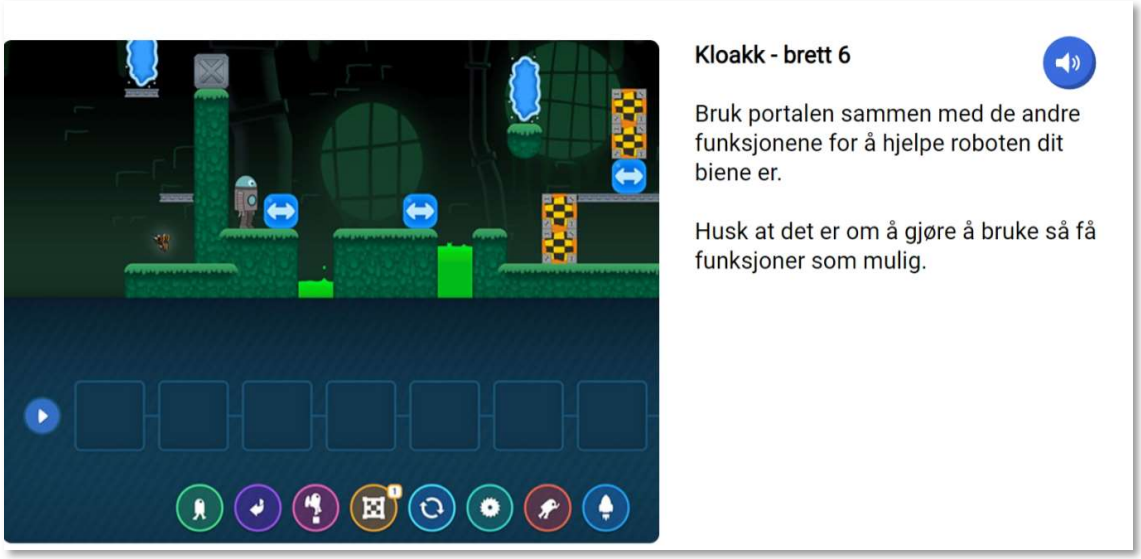
Tabell 5

<table>
<tr><td colspan="6">Sammenfatning av Kjøkken – brett 2</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="2">Algoritmiske konsept</td><td colspan="3">Algoritmisk praksis</td></tr>
<tr><td colspan="2">-Hendelse
-Sekvens
-Løkke</td><td colspan="3">-Teste og debugge
-Abstrahering og modularisering</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>1</td><td>2</td></tr>
</table>

4.1.4 Vertikal analyse, Kloakk – brett 6

Kloakk er den siste verdenen i kodeskolen, og brett seks er det siste brettet i denne verden. Altså skal jeg nå ta for meg det siste brettet i kodeskolen og den tilhørende videoen. Når elevene arbeider med dette, har de allerede gjennomført 35 brett.

4.1.4.1 Informasjon



Figur 24

Figur 24 viser oppgaveteksten og en illustrasjon elevene får opp før de tar fatt på brettet. Oppgaveteksten kommuniserer at målet med brettet er, slik som i de tidligere oppgavene, å få roboten til ruten med biene i med bruk av færrest mulig funksjoner: «Husk at det er om å gjøre å bruke så få funksjoner som mulig.».

Undervisningsmateriellet tar utgangspunkt i at elevene har kompetanse fra de tidligere brettene, da disse må løses for å komme til dette brettet, men oppgaveteksten minner elevene allikevel om å bruke portalene som er på brettet, som en kan se i figur 24: «Bruk portalen for å hjelpe roboten dit biene er». Undervisningsmateriellet kommuniserer ikke noen rammer for hvordan elevene skal arbeide med brettet. Siden det ikke har blitt kommunisert noe om elevene skal arbeide individuelt eller i samarbeid tar jeg utgangspunkt i at også dette brettet skal løses individuelt. Det blir heller ikke kommunisert hvor mye tid en kan forvente å bruke på brettet.

Figur 25

Figur 26

Figur 27

Figur 28

I de fire figurene over viser de fire ulike tilbakemeldingene elevene kan få på dette brettet. Verktøyet vurderer og gir tilbakemelding på arbeidet til elevene, slik som de tidligere brettene. Dersom elevene klarer brettet med færrest mulig antall funksjoner får de tilbakemeldingen «Gratulerer. Helt perfekt! Du fanget alle biene! Du brukte 16 funksjoner» (Figur 25), hvis de bruker for mange funksjoner får de enten tilbakemeldingen: «Gratulerer. Supert! Du klarte å fange 2 bier. Du brukte 21 funksjoner» (Figur 26), eller «Gratulerer. Veldig bra! Du klarte å fange 1 bie! Du brukte 31 funksjoner. Tips: prøv å bruke færre funksjoner neste gang» (Figur 27). På dette brettet er det også mulig å få roboten til å gå i feltene med grønn veske, da får eleven tilbakemeldingen: «Prøv igjen!» (Figur 28).

4.1.4.2 Algoritmisk tenkning

På dette brettet må en bruke *løkker* for å bruke færrest mulig funksjoner, og fange flest mulig bier. Funksjoner som står alene, har ikke blitt definert av Brennan og Resnick (2012). Brennan og Resnick (2012) beskriver *sekvenser* som en liste med flere instruksjoner. En instruksjon tilsvarer en funksjon i Kodeskolen. Å trykke på start er en *hendelse*, som fører til at koden kjøres. Derfor kan vi finne de *algoritmiske konseptene sekvenser, løkker og hendelse*.

Dersom eleven ikke er fornøyd med resultatet etter å ha forsøkt å løse brettet kan hen forsøke igjen. Derfor kan en finne den *algoritmiske praksisen: teste og «debugge»*. Koden forsvinner etter den har blitt kjørt. Det kan være krevende å huske en så lang kode og finne feil uten å ha den fremfor seg noe som gjør at debugging-prosessen kan bli utfordrende. Dersom en løser oppgaven riktig må en bygge/skrive kode i kode-feltet minimum tre ganger. Jeg skriver minimum da elevene fritt kan legge inn en sekvens med, for eksempel, tre funksjoner, kjøre denne og så arbeide videre og dermed skrive flere sekvenser. At elevene blir nødt til å dele brettet i flere små sekvenser og da også problem fører til at de arbeider med den *algoritmiske praksisen abstrahering og modularisering*.

Elevene arbeider individuelt med hver sin datamaskin. Undervisningsmateriellet legger ikke opp til at elevene skal arbeide med å uttrykke seg, eller samhandle, på andre måter heller. Elevene arbeider ikke gjennom å *uttrykke* slik det blir beskrevet av Benton et al. (2016).

Figur 29

Figur 30

Figur 31

I videoen for Kloakk blir vi introdusert til portalene i denne verden. De illustrerer og forklarer hvordan disse fungerer. De ulike funksjonene har ikke noen forklaringer på dette brettet, men

en kan teste flere av funksjonene. Dersom eleven tester gå-funksjonen eller sage-funksjonen vil roboten løfte på skuldrene, de andre kan testes fra robotens utgangspunkt i brettet. Derfor er det mulig å *utforske* hvordan disse fungerer. Det er ikke gitt at elevene vil gjøre dette, da noen trolig ikke har behov for å teste dem. Elevene har mulighet til å *utforske* ulike løsninger da de kan prøve brettet så mange ganger de ønsker. Undervisningsmateriellet kommuniserer ikke at elevene skal drive med *utforskende* arbeid, men det er mulig noen elever vil gjøre det.

Undervisningsmateriellet åpner ikke for å la elevene *forklare* seg da de verken stiller spørsmål til eleven eller legger opp til diskusjon i klasserommet.

Elevene har muligheten til å *se for seg* utfallet av sekvensen, før de kjører den. De lager en sekvens i kode-feltet, hvor den blir frem til de velger å kjøre den. Eleven har muligheten til å planlegge hvor stor del av brettet hen ønsker å løse om gangen og da planlegge koden litt og litt, se for seg utfallet, kjøre koden og reflektere over utfallet hen fikk. Dette er dog ikke omtalt av undervisningsmateriellet.

Undervisningsmateriellet legger ikke opp til at elevene skal *utveksle* oppdagelsene sine med medelever, da det bygger på individuelt arbeid.

Brettet anvender begrep og konsept som er typisk for programmering og *kobler* dermed algoritmisk tenkning til matematikkfaget. I oppgaveteksten finner jeg begrepet funksjon, i tillegg på elevene bruke løkker for å få mer enn en bie i vurderingen. Løkker finner vi igjen i kompetansemål for 4. og 5. trinn. Eleven vil følge reglene for spillet for å løse brettet, dermed kan en også koble undervisningsmateriellet med kompetansemål for 2. trinn:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker.
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker.

Nedenfor er en tabell 6 oppsummerer funnene av algoritmisk tenkning i undervisningsmateriellet.

Tabell 6

Sammenfatning av Kloakk – brett 6					
Brennan og Resnick (2012)	Algoritmiske konsept			Algoritmisk praksis	
	-Hendelse -Sekvens -Løkke			-Teste og debugge -Abstrahering og modularisering	
Benton et al. (2016)	Utforske	Forklare	Se for seg	Utveksle	Koble
	2	1	2	1	2

4.1.5 Oppsummering kodeskolen

I dette kapittelet ønsker jeg å oppsummere funnene fra analysen av kodeskolen. Målet til kodeskolen er at elevene skal «få en grunnleggende forståelse av koding og algoritmisk tenkning». Gjennom analysen kommer det frem at de oppnår noen aspekt av algoritmisk tenkning.

4.1.5.1 Informasjon

Målet på hvert brett er det samme. Elevene skal få roboten til ruten med biene, med bruk av færrest mulig funksjoner. Dette blir kommunisert i oppgavetekster og gjennom videoene tilhørende de ulike verdenene. Roboten og Salami kommer stadig over nye utfordringer og vi følger historien til Salami og roboten har reddet alle biene. Dette gjør at kodene blir stadig mer komplekse.

Rammene rundt verktøyet blir ikke kommunisert til elevene eller lærerne. I den første videoen blir brukeren av verktøyet referert til som «du», derfor konkluderer jeg med at verktøyet er ment å brukes i individuelt arbeid. Altså at elevene arbeider på hver sin datamaskin. Oppgaveteksten er lest inn slik at de som ikke har så gode lese-egenskaper fremdeles kan delta. Tidsbruk, før og etterarbeid eller andre rammer blir ikke kommunisert.

Alle brettene blir vurdert på samme måte, som baserer seg på antall funksjoner. Uavhengig om det betyr at roboten må forflytte seg lengre i antall ruter. Får en færrest mulig funksjoner får en tre bier, så to eller en om det brukes for mange funksjoner.

4.1.5.2 Algoritmisk tenkning

De anvender de algoritmiske konseptene *sekvens, løkke* og *hendelse*. De bruker ikke disse begrepene, men jeg har funnet bruk av gjenta-funksjon og loop-funksjon. Gjenta-funksjon og loop-funksjon henviser begge til løkke-funksjonen. De bruker begrepet funksjon jamnlig gjennom oppgavetekstene og noen av videoene. Begrepene blir ikke definert, men de blir brukt i ulike sammenhenger. Elevene får kjennskap til disse *algoritmiske konseptene* gjennom kodeskolen.

Kodeskolen kommer innom noen former for *algoritmisk praksis*. I analysen har jeg funnet *testing og debugging, iterativ og inkrementell* og *abstrahering og modularisering*. Elevene kan teste ulike løsninger, men debugging er noe krevende da koden forsvinner etter den er kjørt. Det kan være elevene husker hele koden og klarer det, men undervisningsmateriellet har ikke lagt til rette for at elevene kan arbeide med dette.

Gjennom verktøyet er det noe begrenset hvor mye en kan teste og debugge. Elevene har muligheten til å gjøre brettene om igjen så mange ganger de vil. Derfor er muligheten der for å teste ulike løsninger, for så å debugge kodene og prøve igjen. Kodene forsvinner etter de er kjørt og brettene krever gradvis flere funksjoner og sekvenser. Å huske en sekvens bestående av tre funksjoner er enklere å huske enn tre fulle kode-felt. Dette gjør at debugging blir gradvis mer krevende og hvor mye elevene arbeider med denne algoritmiske praksisen kan diskuteres.

Brettene krever gradvis at en anvender flere av funksjonene, men fremdeles så få som mulig for å komme i mål. Det første brettet har få funksjoner og er trolig ment for å introdusere elevene til verktøyet før de gradvis gjør brettene mer avansert. Hvert brett er i seg selv ikke iterativ og inkrementell, men gjennom å arbeide med flere brett vil elevene arbeide med den *algoritmiske praksisen iterativ og inkrementell*.

Abstrahering og modularisering innebærer å bygge noe stort av å sette sammen flere små deler. Muligheten til å kjøre en funksjon om gangen om elevene ønsker det eller at noen brett krever at elevene fyller kode-feltet flere ganger gjør at de arbeider med noe stort bestående av flere små deler.

De nye mulighetene i undervisningsmateriellene blir presentert gjennom videoene, slik som vi så i Kjøkken – brett 2. Undervisningsmateriellene kommuniserer ikke at elevene skal drive med *utforskende* arbeid, slik jeg tolker Benton et al. (2016) rammeverk, men dersom elevene glemmer hva noen av brikkene er så er det mulig å teste hva de gjør så lenge roboten er plassert et sted der funksjonen kan kjøres. Elevene kan løse brettene så mange ganger de ønsker og kan derfor teste ulike løsninger. Undervisningsmateriellet kommuniserer ikke at de skal anvende denne metoden.

Elevene skal arbeide individuelt, noe som gjør at det ikke blir mulighet for dem å *utveksle* kunnskap. Undervisningsmateriellene gir ingen spørsmål elevene må svare på, så de blir ikke nødt til å forklare løsninger eller tanker.

Elevene kan *se for seg* utfallet av sekvensene og reflektere over det etterpå, men dette er ikke kommunisert gjennom undervisningsmateriellene.

Disse undervisningsmateriellene *kobler* algoritmisk tenkning og programmering sammen. Jeg fant til sammen tre kompetansemål i utvalget fra kodeskolen. I analysen ble det ikke funnet at undervisningsmateriellet legger opp til arbeid med andre matematiske fagområder. Alle tre hadde dette kompetansemålet:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill
I Kjøkken - brett 2 og Kloakk - brett 6 fant jeg også disse to kompetansemålene:
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker.
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker.

Tabell 7 oppsummerer funnene av analysen av de tre brettene, tabellen tar bare for seg rammeverkene av Brennan og Resnick (2012) og Benton et al. (2016). Brennan og Resnicks (2012) kategorier blir listet opp for å tydeliggjøre hva som ble funnet, Benton et al. (2016) konsept får et tall mellom 1 og 3. 1 viser til at konseptet ikke ble funnet i analysen, 2 viser til at konseptet kommer frem til en viss grad og 3 viser til at konseptet kom tydelig frem gjennom analysen.

Tabell 7

<table>
<tr><td colspan="6">Sammenfatning av Kodeskolen</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Hendelse
-Sekvens
-Løkke</td><td colspan="2">-Teste og debugge
-Abstrahering og modularisering
-Iterativ og inkrementell</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>1</td><td>2</td></tr>
</table>

4.2 KUBO

I dette kapittelet skal jeg ta for meg undervisningsmateriellet til KUBO. Jeg tar først for meg den horisontale analysen hvor formålet er å gi en oversikt over hva verktøyet består av. Den horisontale analysen består av bakgrunnsinformasjon og gjennomgående struktur, slik det er beskrevet i analyseverktøyet i metodekapittelet. Videre går jeg inn i den vertikale analysen hvor jeg analyserer tre undervisningsmateriell slik det er beskrevet i rammeverket jeg har satt sammen av Benton et al. (2016) og Brennan og Resnick (2012).

4.2.1 Horisontal analyse

Formålet med dette kapittelet er å få frem KUBO som helhet. Jeg tar først for meg bakgrunnsinformasjon før jeg går inn i den gjennomgående strukturen.

4.2.1.1 Bakgrunnsinformasjon

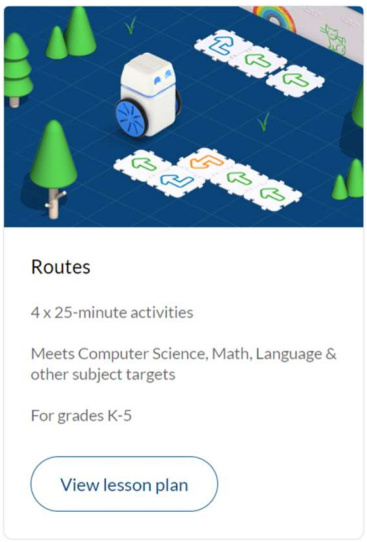
KUBO Robotics er utgiveren av KUBO og ble grunnlagt i 2016 i Danmark. KUBO er en robot som er designet for å lære om programmering og algoritmisk tenkning til alle i alderen 4-10+.

Jeg sendte KUBO e-post for å avklare hvor mange brukere de har i Norge. De kunne dele at de har hatt et stort prosjekt i Bergen hvor de har solgt 500 roboter. Til sammen i Norge har de solgt omtrent 1000 roboter som engasjerer 5000 til 12 000 elever rundt om i Norge (Vedlegg 2).

4.2.1.2 Gjennomgående struktur

KUBO er en robot som en kan programmere med blokkprogrammering, brikkene i KUBO kalles Tag-Tiles. Grunnpakken er en grunnleggende pakke med det mest nødvendige, blant annet roboten, men med utvidelsene får en flere muligheter og alternativ i form av flere Tag-Tiles.

UBO har 12 læreplaner og 34 tverrfaglige læreplaner. De 12 læreplanene er sortert etter hvilke pakker fra KUBO som trengs for å kunne løse oppgavene. Under KUBO Coding er det fire læreplaner, KUBO Coding + har tre læreplaner, KUBO Coding ++ har to og KUBO Coding Math har tre læreplaner.



Figur 32

I figur 32 kan en se et eksempel på en oppgave en kan hente ut fra KUBO. En slik oppgave inneholder lærerguide, læreplan, to arbeidsark, diplom, TagTiles som kan skrives ut på A4 ark, og kompetansemål fra USA og England. Det er bare lærerguiden og læreplanen jeg ser på i den vertikale analysen og det er disse to jeg videre referer til som undervisningsmateriellet.

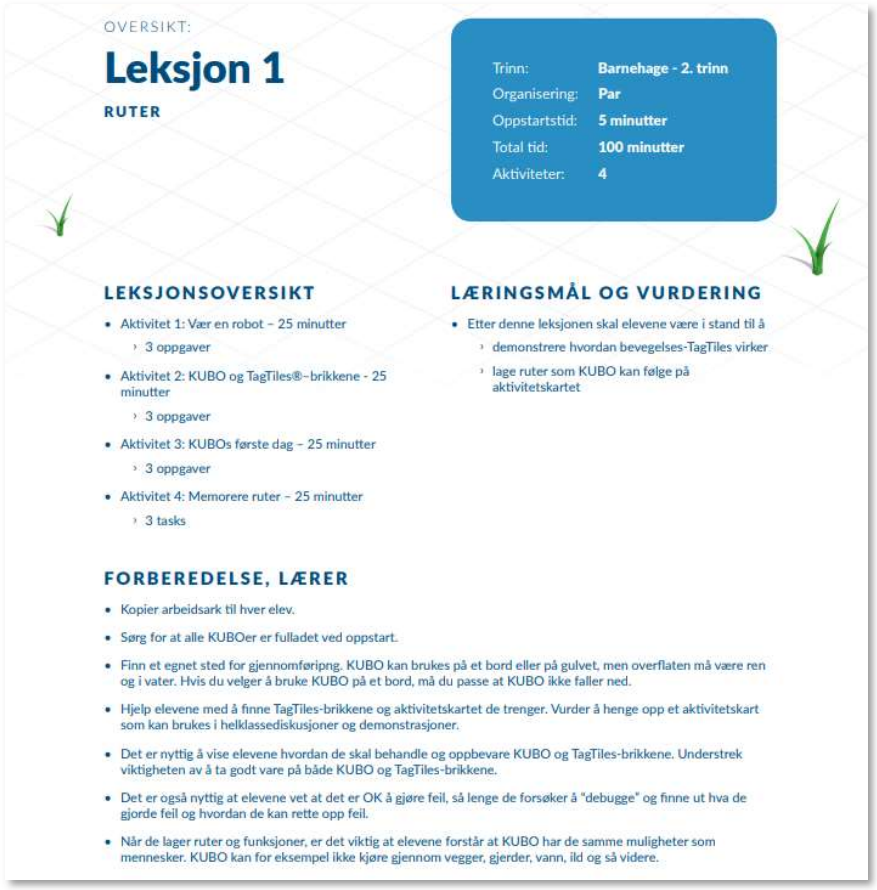
I tillegg til dette har KUBO fem lærerguider, tre magasin de har publisert med informasjon om tanken bak KUBO, hvordan KUBO kan brukes og teoretisk begrunnelse for hvordan og hvorfor de har tatt de valgene de har gjort.

4.2.2 Vertikal analyse – KUBO Coding: Routes

I dette kapittelet skal jeg gjennomføre den vertikale analysen av undervisningsmateriellet KUBO Coding: Routes. Jeg tar for meg læreplanen og lærerguiden. Lærerguiden er ment for å kommunisere hvordan timen skal gå for seg til læreren. Læreplanen skal være tilgjengelig for elevene da dette er aktivitetene de skal arbeide med. I analysen går jeg først inn i hvordan undervisningsmateriellet formidler informasjon, videre ser jeg på hvordan de ulike aspektene av algoritmisk tenkning kommer frem. Dette gjør jeg med bruk av analyseverktøyet basert på Brennan og Resnick (2012) og Benton et al. (2016). Til slutt har jeg lagt funnene av algoritmisk tenkning i en tabell for å oppsummere funnene.

4.2.2.1 Informasjon

Jeg skal først ta for meg punktene i rammeverket under «informasjon». Dette er for å gi et innblikk i hvordan undervisningsmateriellet informerer lærer og elev om mål, rammer og vurderinger.



OVERSIKT:

Leksjon 1

RUTER

Trinn:	**Barnehage - 2. trinn**
Organisering:	**Par**
Oppstartstid:	**5 minutter**
Total tid:	**100 minutter**
Aktiviteter:	**4**

LEKSJONSOVERSIKT

- Aktivitet 1: Vær en robot – 25 minutter
  - 3 oppgaver
- Aktivitet 2: KUBO og TagTiles®–brikkene - 25 minutter
  - 3 oppgaver
- Aktivitet 3: KUBOs første dag – 25 minutter
  - 3 oppgaver
- Aktivitet 4: Memorere ruter – 25 minutter
  - 3 tasks

LÆRINGSMÅL OG VURDERING

- Etter denne leksjonen skal elevene være i stand til å
  - demonstrere hvordan bevegelses-TagTiles virker
  - lage ruter som KUBO kan følge på aktivitetskartet

FORBEREDELSE, LÆRER

- Kopier arbeidsark til hver elev.
- Sørg for at alle KUBOer er fulladet ved oppstart.
- Finn et egnet sted for gjennomføripng. KUBO kan brukes på et bord eller på gulvet, men overflaten må være ren og i vater. Hvis du velger å bruke KUBO på et bord, må du passe at KUBO ikke faller ned.
- Hjelp elevene med å finne TagTiles-brikkene og aktivitetskartet de trenger. Vurder å henge opp et aktivitetskart som kan brukes i helklassediskusjoner og demonstrasjoner.
- Det er nyttig å vise elevene hvordan de skal behandle og oppbevare KUBO og TagTiles-brikkene. Understrek viktigheten av å ta godt vare på både KUBO og TagTiles-brikkene.
- Det er også nyttig at elevene vet at det er OK å gjøre feil, så lenge de forsøker å "debugge" og finne ut hva de gjorde feil og hvordan de kan rette opp feil.
- Når de lager ruter og funksjoner, er det viktig at elevene forstår at KUBO har de samme muligheter som mennesker. KUBO kan for eksempel ikke kjøre gjennom vegger, gjerder, vann, ild og så videre.

Figur 33

Figur 33 er et utklipp av første side av lærerguiden i KUBO Coding: Routes.

Dette undervisningsmateriellet lister opp mål for aktivitetene elevene skal jobbe med/mot, som vist under læringsmål og vurderinger i figur 33. Målet med aktivitetene å arbeide med ulike tilnærminger til programmering og algoritmisk tenkning. Hver aktivitet har individuelle læringsmål. Disse blir presentert for læreren i lærerguiden og for elevene før de starter på ny aktivitet i læreplanen. Undervisningsmateriellet har også to overordnede læringsmål, disse blir beskrevet i lærerguiden og læreplanen, som vist i utklippet av lærerguiden over.

I den blå ruten i figur 33 blir læreren informert om at elevene skal arbeide med disse aktivitetene i par. Tidsrammen er på totalt 100 minutter. Aktivitetene er ment for barn fra barnehagealder til andre trinn. Undervisningsmateriellet tydeliggjør altså rammer for aktiviteten gjennom lærerguiden.

Figur 34

Figur 34 viser listen over utstyr elevene trenger for å gjennomføre denne aktiviteten. Denne finner en i læreplanen, læreren har også en liste i lærerguiden. Dette anser jeg som førarbeid da det må gjøres klart før aktiviteten kan starter. Læreren har i tillegg en liste med forberedelser i lærerguiden, som vist i figur 35.

FORBEREDELSE, LÆRER

- Kopier arbeidsark til hver elev.
- Sørg for at alle KUBOer er fulladet ved oppstart.
- Finn et egnet sted for gjennomføripng. KUBO kan brukes på et bord eller på gulvet, men overflaten må være ren og i vater. Hvis du velger å bruke KUBO på et bord, må du passe at KUBO ikke faller ned.
- Hjelp elevene med å finne TagTiles-brikkene og aktivitetskartet de trenger. Vurder å henge opp et aktivitetskart som kan brukes i helklassediskusjoner og demonstrasjoner.
- Det er nyttig å vise elevene hvordan de skal behandle og oppbevare KUBO og TagTiles-brikkene. Understrek viktigheten av å ta godt vare på både KUBO og TagTiles-brikkene.
- Det er også nyttig at elevene vet at det er OK å gjøre feil, så lenge de forsøker å "debugge" og finne ut hva de gjorde feil og hvordan de kan rette opp feil.
- Når de lager ruter og funksjoner, er det viktig at elevene forstår at KUBO har de samme muligheter som mennesker. KUBO kan for eksempel ikke kjøre gjennom vegger, gjerder, vann, ild og så videre.

Figur 35

Etter de har gjennomført aktiviteten får elevene og læreren presentert refleksjonsspørsmål og diskusjonsspørsmål, som et etterarbeid for aktiviteten. Undervisningsmateriellet kommuniserer hva som kreves av førarbeid og etterarbeid av både læreren og elevene.

Undervisningsmateriellet kommer med råd om tilpasninger underveis, i lærerguiden.
Aktivitet 1: Når elevene tegner rutene sine på arbeidsarket, kan det være nyttig for dem å se eller bruke TagTiles-brikkene.
Aktivitet 2: Noen elever kan ha vansker med å skrive svarene sine på arbeidsarket. For disse elevene kan det være nyttig å tegne svarene, eller at de får svaralternativer.

Figur 36

Figur 37

Figur 36 og 37 er hentet fra lærerguiden. Disse viser eksempler på løsninger for oppgave 1 og 2. KUBO viser til noen mulige løsninger i lærerguiden, som vist i de to figurene over. Under Info til lærer i lærerguiden står det også at: «Elevene kan lage og velge hvilken som helst rute som KUBO skal følge til lekeplassen. Det er OK at enkelte ruter er lengre enn andre».
Elevene blir ikke vurdert ut fra rett/feil løsning, da det er flere løsninger som treffer kriteriene til oppgaven.

Figur 38

Figur 38 er et utklipp av siste side av læreplanen. Elevene skal gjennomføre de to punktene i den rosa firkanten og det er opp til læreren å vurdere om elevene mestrer dette eller ikke. Hvordan elevene skal få tilbakemeldinger blir ikke beskrevet. Elevene får en form for tilbakemelding fra KUBO, KUBO kjører koden elevene har lagd og gir da elevene et utfall. De får da en form for tilbakemelding gjennom om de får forventet utfall eller ikke.

4.2.2.2 Algoritmisk tenkning

Videre analyserer jeg undervisningsmateriellet i lys av rammeverket jeg har satt sammen av Benton et al. (2012) og Brennan og Resnick (2016). Jeg tar først for meg Benton et al. (2012) aspekt av algoritmisk tenkning, før jeg går videre til de algoritmiske konseptene fra Brennan og Resnick (2016). Til slutt oppsummerer jeg funnene av algoritmisk tenkning i en tabell.

Elevene bygger *sekvenser,* som inngår i Brennan og Resnicks (2016) algoritmiske konsept, når de skal instruere KUBO med bruk av TagTiles. Når elevene setter KUBO på TagTiles er det en *hendelse* som får KUBO til å lese sekvensen. Det er flere TagTiles i grunnpakken, med flere *algoritmiske konsept*, men disse blir ikke anvendt gjennom dette undervisningsmateriellet. *Sekvenser* og *hendelse* er de eneste *algoritmiske konseptene* som anvendes gjennom dette undervisningsmateriellet.

Undervisningsmateriellet legger opp til anvendelse av den *algoritmiske praksisen testing og debugging* da det allerede på første side av lærerguiden, som vist i figur 33, står at: «Det er også nyttig at elevene vet at det er OK å gjøre feil, så lenge de forsøker å debugge og finne ut

hva de gjorde feil og hvordan de kan rette opp feil». Elevene får ikke forklart hvordan de ulike TagTiles fungerer, men får instrukser på hvordan de kan teste hva som skjer dersom de setter KUBO på de ulike TagTilesene. Elevene skal altså teste hvordan TagTiles fungerer og ulike løsninger, samt debugge dersom de ikke oppnår ønsket resultat.

Figur 39

Figur 39 er et utklipp fra aktivitet 1, oppgave 1, i undervisningsmateriellet KUBO Coding: Routes.

Elevene blir introdusert til TagTiles gjennom embodiment, som vist i figur 39. De bruker de ulike TagTilesene for å føre medeleven til et punkt i klasserommet, før de bruker TagTiles sammen med KUBO. Når de skal begynne å arbeide med KUBO får de instrukser om å utforske en og en TagTile for å undersøke hva de ulike kommandoene gjør. De får så instrukser om å lage spesifikke ruter med bruk av TagTiles, sette KUBO på disse rutene og undersøke hva som skjer. Oppgavene blir trinnvis mer utfordrende, altså er de *inkrementell*. I figur 39 ber de elevene om å gjenta samme prosess helt til de er tomme for TagTiles. Dette er *gjentagende* arbeid og er derfor, slik jeg forstår det, *iterativ*. Altså er her den *algoritmiske praksisen inkrementell og iterativ*.

Elevene blir oppfordret til å *utforske*, slik det er beskrevet av Brennan og Resnick (2012), gjennom å teste hva de ulike TagTilesene gjør. De får ikke forklart ting, men de blir fortalt hvordan de skal finne ut hva de ulike TagTilesene gjør. En del av konseptet *utforske* er å feilsøke og debugge. I info til læreren står det: «Det er også nyttig at elevene vet at det er OK å gjøre feil, så lenge de forsøker å debugge og finne ut hva de gjorde feil og hvordan de kan rette opp feil». Altså legger undervisningsmateriellet opp til at elevene skal *utforske* gjennom å undersøke hvordan TagTiles fungerer, men også gjennom å gjøre feil og løse disse feilene.

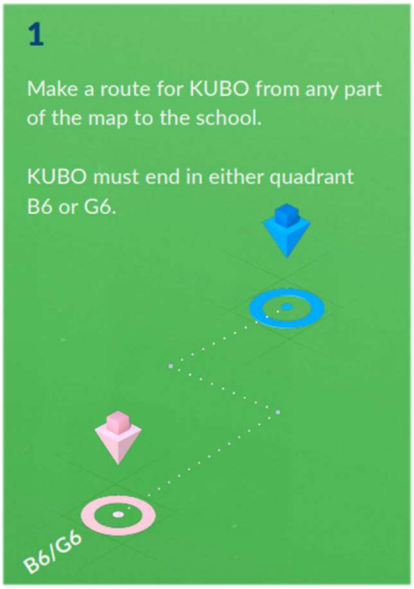
Læreren får en liste med diskusjonsspørsmål for klasseromsdiskusjoner. Dette, i tillegg til refleksjonsspørsmål som elevene får, viser at undervisningsmateriellet legger opp til at elevene skal *forklare* hva de arbeider med og lærer.

Figur 40

Figur 40 er et utklipp fra oppgave 2, i den første aktiviteten i dette undervisningsmateriellet. Her skal elevene lage en rute for å komme fra et sted i klasserommet til det andre. De skriver ned en kode og tester den. De bytter så kode med partneren og gjennomfører koden til partneren. Undervisningsmateriellet legger opp til å bruke det *algoritmiske konseptet å se for seg* gjennom denne oppgaven, som krever at de planlegger ruter mot et bestemt mål og tegner opp et tenkt utfall, som vist i figur 40. Oppgaven krever at elevene planlegger og tester utfall slik det blir beskrevet av Benton et al. (2016).

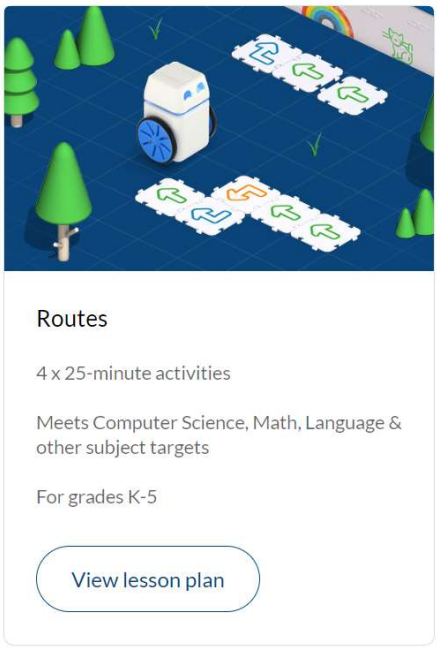
Undervisningsmateriellet kommuniserer at elevene skal arbeide i par, som vist i utklippet av lærerguiden, figur 33. Elevene skal samarbeide gjennom alle aktivitetene. Undervisningsmateriellet beskriver ikke at elevene må *utveksle* ideer, men i en samarbeidsoppgave vil det være naturlig at elevene arbeider sammen og *utveksler* ideer for å finne en god løsning på oppgaven.

Figur 41



Figur 42

KUBO aktivitetskartet består av et koordinatsystem med ulike illustrasjoner i, som vist i figur 41. Dette er et kart som en kan kjøpe med grunnpakken og illustrerer en skole med uteområde og trafikk. Figur 42 viser oppgaveteksten i aktivitet tre, oppgave en. Her *kobles* aktiviteten til det matematiske fagområdet koordinatsystem. KUBO *kobler* algoritmisk tenkning til programmering da TagTiles brukes for å programmere KUBO.



Figur 43

Figur 44

Figur 43 er et utklipp fra nettsiden til KUBO, dette er linken en må trykke på for å komme frem til lærerguiden og læreplanen. Figur 44 viser et utklipp fra lærerguiden.

I figur 43 står det at dette undervisningsmateriellet er ment for elever i «K-5». I lærerguiden, som vist i figur 44, står det at undervisningsmateriellet passer for «trinn: Barnehage – 2. trinn». Jeg har valgt å ta for meg kompetansemålene fra første til og med syvende trinn, da jeg ser på dette som mest hensiktsmessig. Dersom jeg begrenser søket etter kompetansemål kan jeg risikere at verdifulle poeng fra læreplanen sklir ut. Jeg tar ikke for meg rammeplanen for barnehage, da min oppgave omhandler barneskolen. Jeg tar bare for meg kompetansemål som omhandler programmering, da jeg ikke har teoretisk grunnlag til å spekulere i de andre kompetansemålene. Med disse kriteriene står jeg igjen med to kompetansemål som elevene arbeider med gjennom dette undervisningsmateriellet:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
3. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet.

Nedenfor er en tabell for å oppsummere funnene for algoritmisk tenkning, som beskrevet av Benton et al. (2016) og Brennan og Resnick (2012). Funnene av Brennan og Resnicks (2012) algoritmiske tenkning blir listet opp, mens Benton et al. (2016) får et tall mellom 1 og 3. Scoren 1 viser at konseptet ikke ble funnet i analysen, 2 viser at det ikke kom tydelig frem, men at konseptet kan ta sted og 3 viser at konseptet kom tydelig frem i analysen.

Tabell 8

<table>
<tr><td colspan="6">Sammenfatning av KUBO Coding: Routes</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Sekvens
-Hendelse</td><td colspan="2">-Teste og debugge
-Inkrementell og iterativ</td></tr>
<tr><td>Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td></td><td>3</td><td>3</td><td>3</td><td>2</td><td>3</td></tr>
</table>

4.2.3 Vertikal analyse – KUBO Coding ++: Super Coders

I dette kapittelet skal jeg gjennomføre den vertikale analysen av undervisningsmateriellet KUBO Coding ++: Super Coders. Dette undervisningsmateriellet innebærer læreplanen, lærerguiden. Læreplanen består av flere aktiviteter. Lærerguiden henviser seg til læreren og kan fungere som en veileder for læreren gjennom undervisningen. Aktivitetene og oppgavene kommer frem i læreplanen, denne skal være tilgjengelig for elevene gjennom økten. I den vertikale analysen tar jeg først for meg punktet for informasjon, slik det er presentert i metode-kapittelet, før jeg går inn i algoritmisk tenkning. Jeg analyserer for algoritmisk tenkning slik det er presentert av Brennan og Resnick (2012) og Benton et al. (2016). Jeg avslutter analysen med å oppsummere funnene av algoritmisk tenkning i en tabell, for å tydeliggjøre funnene mine.

4.2.3.1 Informasjon

I den vertikale analysen tar jeg først for meg punktet med «informasjon». Jeg skal se etter målet og læringsmålet for undervisningen, hvilke rammer som blir satt og hvordan elevene vurderes.

OVERSIKT:

Læreplan 1

SUPERPROGRAMMERERE

Klassetrinn:	3-5
Gruppestørrelse:	Par
Oppsettid:	5 minutter
Total tid:	180 minutter
Aktiviteter:	3

OVERSIKT OVER LÆREPLAN

- Aktivitet 1: Variabler - 45 minutter
  - 3 oppgaver
  - Introduserer variabler og bruk av disse i løkker.
- Aktivitet 2: Betingelser - 90 minutter
  - 3 oppgaver
  - Skape programmer med betingelser.
  - Bruk hendelser til å styre KUBO.
  - Gjøre KUBOs bevegelser tilfeldige.
- Aktivitet 3: KUBOs nye ferdigheter - 45 minutter
  - 3 oppgaver
  - Bruk TagTiles® angi hastighet og snu til å justere bevegelsene til KUBO.

RESULTATENE

- I slutten av denne delen skal elevene kunne:
  - Demonstrere og forstå hvordan KUBO Coding++ TagTiles® fungerer.
  - Bruke TagTiles med variabler og operatorer i funksjoner.
  - Skape programmer med betingelser.
  - Opprette hendelser ved hjelp av TagTiles i Coding++.

VURDERING

Elevene kan vise at de mestrer innholdet ved å:

- Demonstrere og vise en forståelse av hvordan brikkene i Coding++ fungerer.
- Bruke hendelser, variabler og betingelser til skape funksjoner med Coding++-brikker.
- Bruke tilfeldige hendelser til å variere bevegelsene til KUBO.
- Variere hastigheten og retningen til KUBO ved hjelp av brikkene øk hastighet og snu.

NØDVENDIG FORHÅNDSKUNNSKAP

- KUBO Coding++ Læring 1 eller kodelisens læring 1-4
  - Elevene bør ha kunnskap om programmering med KUBO før de nye brikkene tas i bruk. Settet Coding++ er beregnet som en utvidelse av det originale Coding-settet.
  - Hvis det har vært en stund siden elevene har jobbet med KUBO så kan det være smart å gå gjennom ordene som benyttes og oppgavene fra Coding++ læreplan 1.

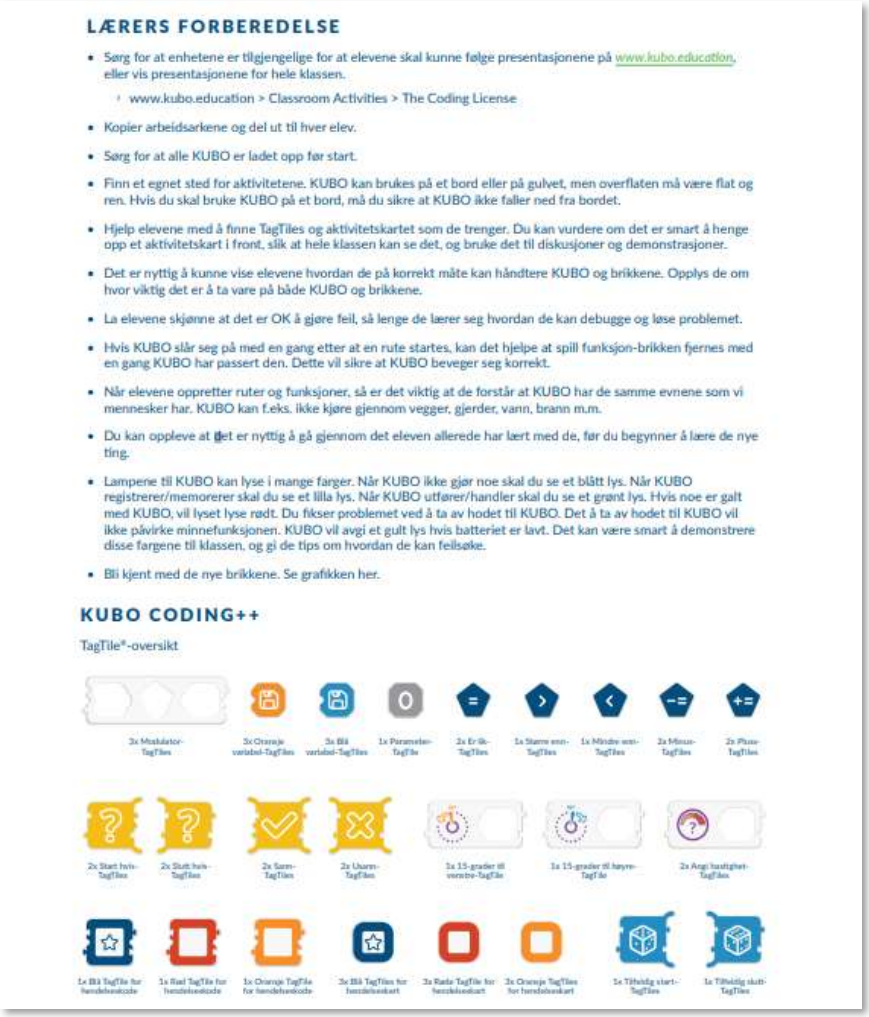
Figur 45

Figur 45 er et utklipp fra lærerguiden. Her er oversikten over læringsmål under «resultatene» og «vurdering». De tilsvarende målene finner en i læreplanen. I tillegg til disse har hver aktivitet egne læringsmål og mål for oppgavene. Et eksempel på mål for en aktivitet er: «Opprette en funksjon med en betingelse».

Figur 46

Figur 46 er et skjermbilde fra læreplanen, dette utklippet viser til hvilket utstyr elevene må ha tilgjengelig for å kunne gjennomføre aktiviteten. I lærerguiden står det at elevene skal ha læreplanen tilgjengelig gjennom aktivitetene, enten på egne enheter eller på en skjerm fremme i klasserommet.

I lærerguiden (figur 45), under nødvendige forkunnskaper, kommer det frem at: «Elevene bør ha kunnskap om programmering med KUBO før de nye brikkene tas i bruk. Settet Coding++ er beregnet som en utvidelse av det originale Coding-settet» og «Hvis det har vært en stund siden elevene har jobbet med KUBO så kan det være smart å gå gjennom ordene som benyttes og oppgavene fra Coding++ læreplan 1». Noen aktiviteter har «Nye ord» og andre har «Gjennomgang av ord som er benyttet tidligere».



LÆRERS FORBEREDELSE

- Sørg for at enhetene er tilgjengelige for at elevene skal kunne følge presentasjonene på www.kubo.education, eller vis presentasjonene for hele klassen.
  - www.kubo.education > Classroom Activities > The Coding License
- Kopier arbeidsarkene og del ut til hver elev.
- Sørg for at alle KUBO er ladet opp før start.
- Finn et egnet sted for aktivitetene. KUBO kan brukes på et bord eller på gulvet, men overflaten må være flat og ren. Hvis du skal bruke KUBO på et bord, må du sikre at KUBO ikke faller ned fra bordet.
- Hjelp elevene med å finne TagTiles og aktivitetskartet som de trenger. Du kan vurdere om det er smart å henge opp et aktivitetskart i front, slik at hele klassen kan se det, og bruke det til diskusjoner og demonstrasjoner.
- Det er nyttig å kunne vise elevene hvordan de på korrekt måte kan håndtere KUBO og brikkene. Opplys de om hvor viktig det er å ta vare på både KUBO og brikkene.
- La elevene skjønne at det er OK å gjøre feil, så lenge de lærer seg hvordan de kan debugge og løse problemet.
- Hvis KUBO slår seg på med en gang etter at en rute startes, kan det hjelpe at spill funksjon-brikken fjernes med en gang KUBO har passert den. Dette vil sikre at KUBO beveger seg korrekt.
- Når elevene oppretter ruter og funksjoner, så er det viktig at de forstår at KUBO har de samme evnene som vi mennesker har. KUBO kan f.eks. ikke kjøre gjennom vegger, gjerder, vann, brann m.m.
- Du kan oppleve at det er nyttig å gå gjennom det eleven allerede har lært med de, før du begynner å lære de nye ting.
- Lampene til KUBO kan lyse i mange farger. Når KUBO ikke gjør noe skal du se et blått lys. Når KUBO registrerer/memorerer skal du se et lilla lys. Når KUBO utfører/handler skal du se et grønt lys. Hvis noe er galt med KUBO, vil lyset lyse rødt. Du fikser problemet ved å ta av hodet til KUBO. Det å ta av hodet til KUBO vil ikke påvirke minnefunksjonen. KUBO vil avgi et gult lys hvis batteriet er lavt. Det kan være smart å demonstrere disse fargene til klassen, og gi de tips om hvordan de kan feilsøke.
- Bli kjent med de nye brikkene. Se grafikken her.

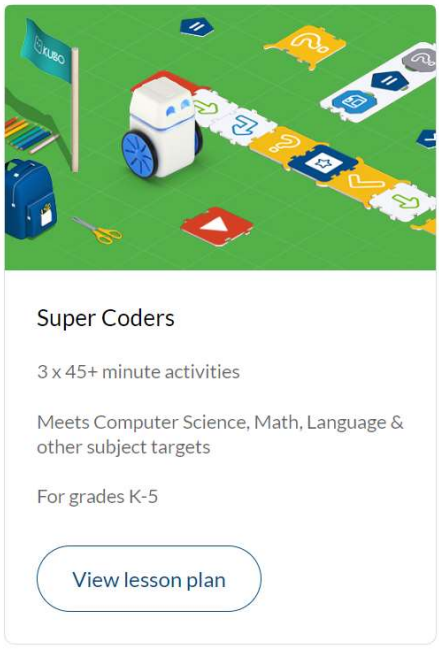
KUBO CODING++

TagTile®-oversikt

Figur 47

Figur 47 er et utklipp fra læreplanen og viser en liste over forberedelsene en lærer burde gjøre før hen tar fatt på undervisningen. Førarbeid for dette undervisningsmateriellet kommer tydelig frem her.

I lærerguiden er det diskusjonsspørsmål til hver aktivitet, det er opp til læreren hvordan hen ønsker å bruke. I tillegg har hver aktivitet to oppsummeringsoppgaver, i lærerguiden står det at læreren må bestemme selv om hen ønsker å få skriftlige eller muntlige svar på de ulike oppgavene. Jeg anser både diskusjonsspørsmål og oppsummeringsoppgavene som etterarbeid for aktiviteten. Undervisningsmateriellet kommuniserer altså både før- og etterarbeid med både lærer og elev.



Figur 48 *Figur 49*

Figur 48 viser linken en trykker på for å komme til undervisningsmateriellet. Figur 49 viser informasjon fra første side i lærerguiden. De to figurene gir ulik informasjon når det kommer til hvilken aldersgruppe undervisningsmaterialet passer for. For å ikke ekskludere noe vesentlig velger jeg å ta utgangspunkt i at materiellet er tilegnet barn fra barnehage til 6. trinn, da K-5 tilsvarer dette i Norge. Andre rammer blir beskrevet gjennom lærerguiden, under «lærers forberedelse» og «administrasjon». De informerer om at elevene skal/burde arbeide i par, som vist i figur 49. Elevene skal dele på et KUBO-sett per gruppe. KUBO anbefaler også å gi elevene roller slik at de bytter på å styre KUBO.

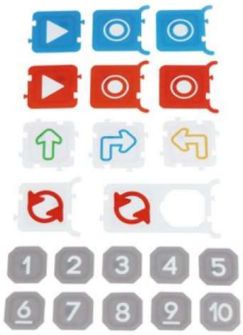
Jeg kan ikke finne at undervisningsmateriellet kommuniserer informasjon om spesifikke tiltak for tilpasning, men undervisningsmateriellet beskriver at læreren må gå gjennom ulike ting dersom noe er uklart for elevene og hjelpe dem dersom det er noe de ikke forstår. Blant annet under aktivitet 1, oppgave 1: «Hvis elevene ikke forstår hva variabler er, så vil det være smart å gå gjennom og lære de det før du fortsetter». Det er anbefalt at elevene spør hverandre først: «spør tre, så meg» -regelen blir referert til. Dette innebærer å spør tre medelever før eleven spør læreren om hjelp.

I dette undervisningsmateriellet får læreren en liste over egenskaper eller kompetanse elevene skal ha tilegnet seg gjennom de tilhørende oppgavene, dette kan du se under «vurdering» i figur 45. Svarene til elevene blir ikke vurdert, men heller om elevene har kompetansen til å kunne demonstrere punktene under «vurdering». Undervisningsmateriellet beskriver ikke

hvordan læreren skal gi tilbakemeldinger. Roboten gir en form for tilbakemelding da den utfører koden den har fått beskjed om å kjøre. Dersom utfallet er som forventet/uventet er dette og en form for tilbakemelding.

4.2.3.2 Algoritmisk tenkning

Videre setter jeg søkelyset mot algoritmisk tenkning. Jeg analyserer først etter algoritmisk tenkning, slik det er presentert av Benton et al. (2012) før jeg tar for meg de algoritmiske konseptene presentert av Brennan og Resnick (2016). Avslutningsvis setter jeg inn funnene i en oppsummerende tabell for å tydeliggjøre hva som kom frem gjennom analysen.



Figur 50

Figur 51

Figur 50 og 51 er skjermbilder fra lærerguiden, her illustreres TagTilesene som anvendes i dette undervisningsmateriellet. Figur 50 viser mulighetene som er i grunnpakken. De røde og blå TagTilesene øverst er brikker som brukes for å få KUBO til å lese koden, play-TagTilesen er den som får KUBO til å kjøre koden. Under er det tre ulike piler som gjør at KUBO går frem, til høyre eller til venstre. Disse kan settes sammen i løkker (TagTilesene under pilene) med tall som er nederst i figuren, tallene vil fungere som variabler.

Et av målene er å arbeide med de *algoritmiske konseptene variabler, operatører, hendelser* og *betingelser*. I tillegg har jeg funnet de *algoritmiske konseptene sekvens, løkke og data.* Elevene bruker TagTiles fra KUBO Coding og KUBO Coding++. Grunnpakken har TagTiles som kan danne *sekvenser* og *løkker*. Start hvis-TagTiles og Slutt hvis-TagTiles, Sann- og Usann TagTiles åpner for å lage *betingelser*. KUBO++ inkluderer *operatører* som =, >, <, -= og +=, og *data* i form av Oransje variabel- og Blå variabel-TagTiles. Disse to variablene lagrer *data* og gjør at en på et mer effektivt vis kan bytte ut verdier i en *sekvens* uten å måtte skrive om hele *sekvensen*. *Hendelse* tar sted hver gang elevene, for eksempel, setter roboten på en TagTile.

Figur 52

Figur 52 er et utklipp fra Aktivitet 1, oppgave 5. Oppgaven figuren viser til går ut på å debugge koden til medeleven. Det står i lærerguiden at det er viktig at elevene vet at de kan gjøre feil så lenge de debugger. Undervisningsmateriellet kommuniserer at elevene skal arbeide med den *algoritmiske praksisen testing og debugging*.

I lærerguiden under lærernotater for aktivitet 3, oppgave 2, står det at elevene kan arbeide med hver sin del av funksjonen før de setter den sammen, da de to funksjonene de skal lage i denne oppgaven er uavhengige av hverandre og uansett skal settes sammen til en samlet funksjon. Dette beskriver den *algoritmiske praksisen abstrahere og modularisere*.

Figur 53

Figur 54

Figur 53 og 54 er to deloppgaver i aktivitet 1. Her får de en kode de skal justere for å få KUBO til å bevege seg i bane, først en gang så to ganger. Dette gjør de med å forandre variabelen. Å *remikse* koden på denne måten går, slik jeg forstår det, under den *algoritmiske praksisen gjenbruk og remiks*. Elevene arbeider i par gjennom hele undervisningsmateriellet, så det kan oppstå flere situasjoner hvor denne praksisen kan ta sted.

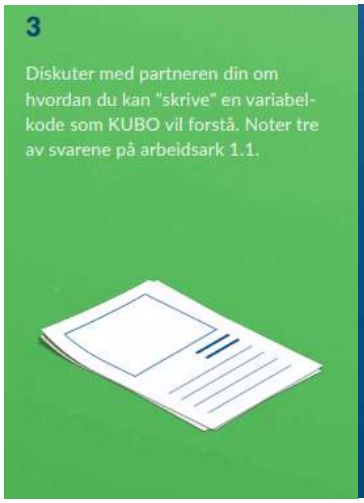
Figur 55

Figur 56

Figur 55 er et utklipp fra aktivitet 2, oppgave 1, og figur 56 er fra aktivitet 3, oppgave 1. Begge oppgavetekstene referer til å teste ut ulike ting. I den første skal eleven teste om funksjonene fungerer. I den andre skal elevene utforske tilfeldig-brikkene. Dette er første oppgaven elevene arbeide med disse TagTilesene, så her skal de *utforske* hvordan de fungerer. Undervisningsmateriellet legger også opp til *utforsking*, slik det blir presentert av Benton et al. (2016), gjennom å lage egne mål på kartet, finne egne løsninger og debugge der det blir behov. KUBO starter i rute A1, elevene får selv velge hvor på kartet de to målene skal være.

Figur 57

Figur 57 viser en oppsummeringsoppgave fra aktivitet 1. Her blir elevene oppfordret til å finne et felles svar på spørsmålet «Hva er en variabel», de skal være forberedt på å dele *forklaringene* sine med resten av klassen. Undervisningsmateriellet legger altså opp til å anvende det Benton et al. (2016) beskriver som å *forklare*.



Figur 58

Figur 59

I figur 58 er del tre i oppgave 1. Her ber KUBO elevene om å planlegge hvordan de kan skrive variabler. Figur 59 viser deloppgave 4 i oppgave 1. Her spør undervisningsmateriellet om KUBO forstod koden elevene hadde lagd. Dette er formulert som et ja/nei spørsmål, men det kan fungere som et spørsmål som får elevene til å reflektere rundt hvordan det fungerer. Jeg mener derfor at dette undervisningsmateriellet legger opp til anvendelse av det *algoritmiske konseptet å se for seg*.

I figur 58 finner vi det Benton et al. (2016) beskriver som å utveksle. Elevene arbeider i par og blir oppfordret til å *utveksle* tanker og ideer gjennom å finne en felles mening for begrepet «variabel», undervisningsmateriellet legger altså opp til at elevene skal *utveksle* kunnskap og ideer.

Figur 60 *Figur 61*

Figur 60 og 61 er utklipp fra aktivitet 3, oppgave 3.

Undervisningsmateriellet *kobler* algoritmisk tenkning sammen med de matematiske kunnskapsområdene koordinatsystem, tid, fart og grader. Dette kan en se iblant annet i figurene 60 og 61, disse to trinnene finner en i aktivitet 3, oppgave 3. Tidligere beskrev jeg aldersgruppen undervisningsmateriellet er tiltenkt. Når jeg ser etter kompetansemål som kan kobles til undervisningsmateriellet tar jeg ikke for meg rammeplanen for barnehagen da dette ikke er relevant for min oppgave. Jeg ser etter kompetansemål for elever i 1. til 7. trinn og finner at undervisningsmateriellet legger opp til arbeid med disse:

2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill

3. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet

4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker

5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker

Nedenfor er en tabell som oppsummerer funnene av algoritmisk tenkning.

Tabell 9

Sammenfatning av KUBO Coding++: Super Coders					
Brennan og Resnick (2012)	Algoritmiske konsept		Algoritmisk praksis		
	-Sekvens -Løkke -Variabel -Hendelse -Betingelse -Operatør -Data		-Teste og debugge -Abstrahere og modularisere -Gjenbruke og remikse		
Benton et al. (2016)	Utforske	Forklare	Se for seg	Utveksle	Koble
	3	3	3	3	3

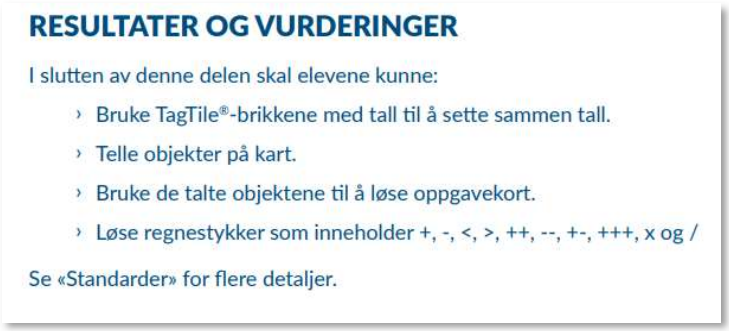
4.2.4 Vertikal analyse – Læreplan 3 – KUBO Coding Math: Advancing

Dette undervisningsmateriellet består av «Task Cards», hurtigstart guide og lærerguiden. Jeg tar først for meg hvordan undervisningsmateriellet informerer lærer og elev. Videre analyserer jeg undervisningsmateriellet i lys av analyseverktøyet som er basert på Brennan og Resnick (2012) og Benton et al. (2016). I denne oppgaven blir kartet i figur 62 anvendt, sammen med noen nye TagTiles som kan kombineres med de andre utvidelsene.

Figur 62

4.2.4.1 Informasjon

Læringsmålene blir presentert i lærerguiden, som vist i 62.



Figur 63

Elevene skal altså kunne å bruke de nye TagTile brikkene med tall, telle objekter på kartet, bruke de talte objektene på kartet for å løse oppgavene, løse regnestykker bestående av ulike regnearter. Videre refererer lærerguiden til «Standarder». De tre siste sidene i lærerguiden består av læringsmål. Disse er kompetansemål fra et læreverk i USA, disse kommer jeg ikke til å ta med videre i analysen, da jeg ikke anser kompetansemål fra USA som relevant for min oppgave.

I dette undervisningsmateriellet har elevene muligheten til å bruke en ny TagTile som gjør at KUBO beveger seg i en av tre forhåndsbestemte ruter i kartet. Dette er de tre fargede linjene i kartet, som vist i figur 62. Da vil ikke elevene lenger trenge å kode, men bare fokusere på å regne matematikk. I dette undervisningsmateriellet er fokuset å regne matematikk.

Figur 64

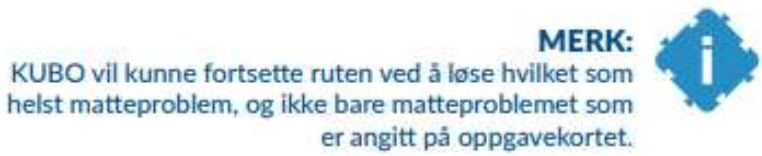
KUBO kommuniserer rammer for aktiviteten gjennom figur 64, som er et utklipp fra lærerguiden. Elevene skal arbeide i par. En aktivitet krever minimum 5 oppgavekort og hver aktivitet tar 20 minutter. Det er til sammen 180 oppgavekort og det er da mulig å bruke totalt

720 minutter med dette undervisningsmateriellet. De skal bruke butikkartet, som du kan se i figur 62.

I oversikten over materiell er det en «Quick Start Guide KUBO Coding Math». Det er 15 sider med informasjon om de nye TagTiles og hvordan disse kan brukes i dette undervisningsmateriellet. Denne blir lærerne bedt om å lese før de kommer i gang med undervisningen. Læreren får informasjon om annet førarbeid under «Lærers forberedelse» i lærerguiden.

I «Quick Start Guide KUBO Coding Math» er det en oversikt over de ulike utvidelsene og sammenhengene mellom dem. Før elevene tar fatt på KUBO Coding Matte er det anbefalt å ha arbeidet med KUBO Coding.

I dette undervisningsmateriellet vurderer KUBO om elevene regner rett eller ikke. KUBO går over TagTilesene fra venstre til høyre og gir tilbakemelding om regnestykket er løst riktig, eller ikke. Har elevene regnet riktig vil KUBO snurre og lyse-grønt, dersom oppgaven er løst feil vil KUBO lyse-rødt og snu seg fra side til side. KUBO kan ikke gi tilbakemelding om hva som stemmer eller ikke, men gi tilbakemeldingen at det er regnet rett eller feil. KUBO vet ikke hvilket regnestykke elevene arbeider med, så om elevene har telt feil antall objekter, men regner riktig vil de få tilbakemeldingen at de har gjort riktig. Dette står også i «Quick Start Guide KUBO Coding Math», som vist i figur 66.



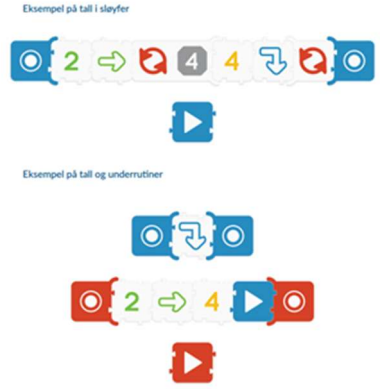
Figur 65

Elevene får utdelt oppgaver, kartet, TagTiles og KUBO. De får ikke utdelt en oppgavetekst eller beskrivelse for hva de skal gjøre, undervisningsmateriellet kommuniserer til læreren gjennom «Quick Start Guide KUBO Coding Math».

Figur 66

Oppgavekortene har oppgaver som er markert med en eller to stjerner, som en kan se til høyre i øvre hjørne av kortet på figur 66. Disse markerer to ulike vanskelighetsgrader. Læreren har mulighet til å tilpasse oppgavene med å gi mer utfordrende regnestykker til de som har behov for det. Læreren får også beskrevet at hen kan gi mer krevende oppgaver med å inkludere flere operatorer. Under «utvidelse» i «Quick Start Guide KUBO Coding Math», som vist i figur 68, står det at de kan eksperimentere med TagTiles fra grunnpakken.

4.2.4.2 Algoritmisk tenkning



Figur 67 *Figur 68*

Dersom de velger å integrere grunnpakken og kode KUBO til å gå ruter i kartet i tillegg kan vi finne flere *algoritmiske konsept*. Da kan elevene bygge *sekvenser, løkke, data* og *hendelse*. Anvendelse av grunnpakken finner vi i «Quick Start Guide KUBO Coding Math», under overskriften «utvidelse», som vist i figur 68, De har valgt å kode tallene som en form for *løkke*. Som vist i figur 67 så kan de legge 2-tallet fremfor en bevegelses-TagTile og 4-tallet fremfor en Spill av-TagTile. Da vil KUBO gå frem to ganger og kjøre Spill av-TagTile fire ganger. Spill av TagTilesen er blå og trigger da den blå funksjonen over. De to TagTilesene med sirkler i, som en ser i figur 67, er for å markere start og slutt på en funksjon. Disse

funksjonene blir lagret som *data*. At elevene setter KUBO på spill av TagTilesen er en *hendelse*.

I dette undervisningsmateriellet finner en i den algoritmiske praksisen testing og debugging i lærerguiden. Der blir læreren bedt om å informere elevene om at det går helt fint å gjøre feil så lenge de prøver å finne hva som gikk feil gjennom debugging. Dersom de bruker KUBO Coding pakken vil de bygge sekvenser og har da mulighet til å teste ulike løsninger og debugge dersom de ikke oppnår ønsket resultat.

Elevene arbeider i par, så de har muligheten til å diskutere og remikse hverandres koder. Altså kan en arbeide gjennom den *algoritmiske praksisen å gjenbruke og remikse*, men at de har muligheten til å arbeide med denne praksisen betyr ikke at alle kommer til å gjøre det.

En del av å *utforske* er å debugge og finne løsninger der de gjør feil. Gjennom dette undervisningsmateriellet kan læreren velge om elevene skal bruke TagTiles til programmering eller ikke, dersom de bruker TagTiles vil det oppstå situasjoner hvor de kan *teste og debugge*. Undervisningsmateriellet kommuniserer at elevene kan debugge dersom de gjør feil og at dette er ok. Det er mulig de arbeider med *utforsking*, men det kommer ikke tydelig frem gjennom undervisningsmateriellet.

I lærerguiden, under «trinn for trinn» står det at det kan være nyttig å henge opp kartet for bruk i diskusjon og demonstrasjon. Dette er det eneste undervisningsmateriellet kommuniserer om faglige samtaler. Elevene skal arbeide i par og har også der muligheten til å *forklare* løsninger og kunnskap. Undervisningsmateriellet legger til en viss grad opp til å anvende det algoritmiske konseptet *forklare*.

Elevene kan *se for seg* mulige løsninger, dersom de bruker KUBO grunnpakke i tillegg til matematikk utvidelsen, da dette vil kreve at de planlegger ruten til KUBO. I KUBO Coding Math kan elevene arbeide med bare regning og da er behovet for det *algoritmiske konseptet å se for seg* mindre. Det blir ikke kommunisert gjennom undervisningsmateriellet at elevene skal arbeide med dette *algoritmiske konseptet*.

Elevene har muligheten til å *utveksle* ideer når de arbeider med dette undervisningsmateriellet da de arbeider i par, dette står i utklippet fra lærerplanen, figur 64. Elevene arbeider sammen og oppgaveløsningen er ikke særlig lærerstyrt. Det er ikke en garanti at faglige samtaler vil ta sted, men det er mulig.

KUBO Coding Math implementerer arbeid med de ulike regneartene og koordinatsystem. Her finner en altså flere matematiske fagområder. I hvor stor grad elevene arbeider med algoritmisk tenkning kommer, i dette undervisningsmateriellet, an på om læreren velger å anvende det sammen med KUBO Coding settet. Når jeg skal se på hvordan undervisningsmateriellet *kobler* algoritmisk tenkning til matematikk tar jeg utgangspunkt i at både KUBO Coding Math og KUBO Coding blir anvendt. Da kan en arbeide med dette kompetansemålet:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
3. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet.

Nedenfor er en tabell med funnene av algoritmisk tenkning.

Tabell 10

<table>
<tr><td colspan="6">Sammenfatning av KUBO Coding Math: Advancing</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Sekvenser
-Løkke
-Data
-Hendelse</td><td colspan="2">-Teste og debugge
-Å gjenbruke og remiks</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>2</td><td>2</td><td>2</td><td>2</td><td>3</td></tr>
</table>

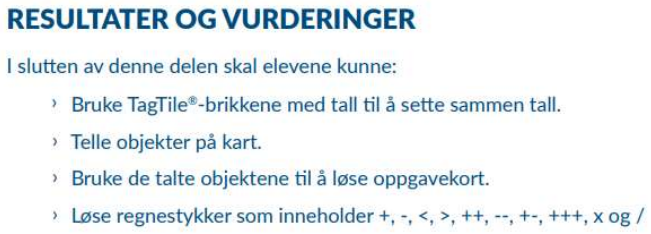
4.2.5 KUBO oppsummering

KUBO's mål er å lære barn algoritmisk tenkning gjennom koding. Gjennom analysen har jeg fokusert på algoritmisk tenkning og hvor godt de ulike aspektene av algoritmisk tenkning kommer frem gjennom undervisningsmateriellet. Dette kapittelet er en oppsummering av den vertikale analysen av KUBO, jeg tar først for meg funnene under informasjon og avslutter med algoritmisk tenkning.

4.2.5.1 Informasjon

KUBO Coding og KUBO Coding ++ lister opp «læringsmål og vurdering» i lærerguiden. Her står det for eksempel: «demonstrere hvordan bevegelses-TagTiles virker», «Opprette hendelser ved hjelp av TagTiles i Coding++» eller «telle objekter på kart». Målene for arbeid med KUBO Coding og KUBO Coding ++ er altså å oppnå disse læringsmålene.

I hurtigstartguiden til KUBO Coding Math står det at elevene skal telle objekter, men dette kommer an på hvilket kart læreren velger å bruke. Jeg har tatt utgangspunkt i at det er butikkartet de bruker, da det er det som blir kommunisert i lærerguiden tilhørende dette undervisningsmateriellet. Da er målet at elevene lager og regner ut ulike regnestykker basert på prisene, som vist i utklippet fra læreplanen i figur 69.



Figur 69

Et punkt som var med i alle undervisningsmateriellene er at det er helt greit at elevene gjør feil «så lenge de debugger for å se hva de gjorde feil, og finne ut hvordan de fikser det». Læringsmålene og delmål kommer tydelig frem gjennom undervisningsmateriellet. Både i lærerguiden og i læreplanen.

Undervisningsmateriellene kommuniserer anbefalte rammer for læreren i lærerguiden. Elevene skal arbeide i par, hvert undervisningsmateriell har informasjon om hvor lang tid en kan beregne å bruke på hvert opplegg. På første side er det en rute med litt informasjon om rammene, som vist i figur 70, i tillegg kommer mer beskrivende rammer frem under «info til lærer». I KUBO Coding Math kommer rammene frem under «lærers forberedelse» og «administrering». Undervisningsmateriellene kommuniserer også førarbeid for læreren. Både KUBO Coding: routes og KUBO Coding Math: Advancing kom det frem former for tilpasninger.

Figur 70

I KUBO Coding Math bruker elevene TagTiles til å skrive utregningen, setter KUBO på regnestykket og de får tilbakemelding om de regnet riktig eller ikke. KUBO vet ikke hvilket regnestykke elevene skal svare på, så her får elevene riktig så lenge utregningen er riktig. Hva elevene skal vurderes etter blir beskrevet gjennom mål og læringsmål i undervisningsmateriellene. Hvordan de skal vurderes og få tilbakemelding blir ikke beskrevet, men elevene får spørsmål som krever at de vurderer eget arbeid og de får tilbakemelding fra KUBO ved utfallet de får. Eleven kan så vurdere om utfallet var forventet eller ikke.

4.2.5.2 Algoritmisk tenkning

Gjennom disse undervisningsmateriellene får en introdusert:
De *algoritmiske konseptene sekvens, løkke, hendelse, betingelse, operatør* og *data*. Disse arbeides med da noen av elevenes mål er å anvende spesifikke algoritmiske konsept, som for eksempel i analysen av KUBO Coding ++: Super Coders, hvor et mål er å «Opprette en funksjon med betingelse».

Gjennom analyse av disse tre undervisningsmateriellene fant jeg fire *algoritmiske praksiser: testing og debugging, gjenbruke og remikse, inkrementell og iterativ samt abstrahere og modularisere.*

I analysen fant jeg *utforskende* arbeid, som beskrevet av Benton et al. (2016), gjennom at undervisningsmateriellet legger opp til å teste og debugge. Når elevene får introdusert nye brikker får de instrukser om å *utforske* hvordan disse fungerer, de får heller ingen fasit da poenget er at elevene tilegner seg kompetansen og ikke at alle får samme svar. Elevene får altså muligheten til å *utforske* brikkenes egenskaper og ulike løsninger. Dette blir ikke kommunisert i KUBO Coding Maths, men det kommer tydelig frem i de to andre undervisningsmateriellene jeg har analysert.

Alle undervisningsmateriellene legger opp til at elevene skal anvende det *algoritmiske konseptet å forklare*. Det kommer frem gjennom at undervisningsmateriellet kommuniserer til læreren at de skal ha klasseromsdiskusjoner og gjennom konkrete refleksjonsspørsmål til elevene.

Gjennom undervisningsmateriellene til KUBO Coding og KUBO Coding ++ vil det oppstå flere situasjoner hvor elevene må planlegge og reflektere rundt ulike løsninger. Da vil trolig flere av elevene *se for seg* ulike løsninger. Dette *algoritmiske konseptet* kommer ikke frem i KUBO Coding Math da det ikke krever planlegging på samme måte, fokuset i KUBO Coding Math er på å regne og ikke å programmere eller planlegge ruter.

Elevene arbeider i par og har mulighet til å *utveksle* ideer mens de arbeider med oppgaver. I KUBO Coding og KUBO Coding Math kom det frem i en viss grad gjennom samarbeid. Undervisningsmateriellet i KUBO Coding ++ legger tydeligere opp til diskusjon rundt ulike begrep hvor det da vil være naturlig at elevene *utveksler* sine tanker om disse begrepene.

Alle undervisningsmateriellene *kobler* algoritmisk tenkning sammen med kompetansemål tilhørende matematikkfaget. Alle tre undervisningsmateriellene kunne kobles til disse to kompetansemålene:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill
3. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet
I KUBO Coding ++: Super Coders fant jeg disse to kompetansemålene i tillegg:
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker

Under er en sammenfatning av funnene fra alle tre undervisningsmateriellene. Her har jeg lagt inn alle de algoritmiske konseptene og algoritmiske praksisene som ble funnet. For å finne scoren på nederste linje i tabellen har jeg lagt sammen scoren i alle de tre undervisningsmateriellene og delt de på tre, så har jeg rundet av til nærmeste tall.

Tabell 11

<table>
<tr><td colspan="6">Sammenfatning av KUBO</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Sekvenser
-Løkke
-Variabel
-Hendelse
-Betingelse
-Operatør
-Data</td><td colspan="2">-Teste og debugge
-Gjenbruke og remikse
-Abstrahere og modularisere
-Inkrementell og iterativ</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>2</td><td>3</td></tr>
</table>

4.3 Spheroball

I dette kapittelet skal jeg ta for meg et utvalg av Spheros undervisningsmateriell. Jeg går først inn i den horisontale analysen for å få et overblikk over verktøyet, denne delen består av bakgrunnsinformasjon samt den gjennomgående strukturen slik det er beskrevet i metodekapittelet. Videre tar jeg for meg den vertikale analysen. Her tar jeg for meg hvordan undervisningsmateriellet formidler informasjon og hvilke aspekter av algoritmisk tenkning som kommer frem.

4.3.1 Horisontal analyse av Spheroball

Formålet med dette kapittelet er å legge frem funnene fra den horisontale analysen av Sphero. Under bakgrunnsinformasjon ser jeg etter hva verktøyet er, hvem verktøyet er laget av og hvem som bruker det. Videre ser jeg på den gjennomgående strukturen av verktøyet, her tar jeg for meg oppsett, antall undervisningsmateriell og hjelpemidler.

4.3.1.1 Bakgrunnsinformasjon

Sphero er basert i Colorado, USA og ble grunnlagt 1. Februar 2010 (Crunch, 2022). Sphero er en robot som kan programmeres gjennom en app. Det er mulig å bruke ulike former for programmeringsspråk i Sphero.

For å finne ut hvor stor brukerbase Sphero har i Norge sendte jeg dem en e-post. De kunne meddele at de har solgt over 10 000 roboter til Norge (Se vedlegg 3). De fleste av dem til skoler, omtrent 85%. Sphero leverer flere ulike roboter, blant annet Sphero Bolt. Denne står for hele 68% av robotene solgt til skoler i Norge er av denne typen. Jeg tok utgangspunkt i Sphero da jeg lagde kriteriene for utvalget mitt, men denne roboten kan også anvendes i disse tre undervisningsmateriellene.

4.3.1.2 Gjennomgående struktur

Sphero har et stort antall aktiviteter da lærere kan dele sine aktiviteter i tillegg til de Sphero selv har publisert. Jeg fant 30 aktiviteter som var relevant for denne oppgaven med begrensingene: gitt ut av Sphero, for elever i fra 1. til 7. trinn og matematikk. Undervisningsmateriellene publisert av Sphero inneholder vanligvis læringsmål, en liste over nødvendig utstyr og instruksjoner. Noen undervisningsmateriell inkluderer tips til læreren og refleksjonsspørsmål, men dette varierer.

I tillegg til disse undervisningsmateriellene har Sphero inkludert muligheten for lærere å dele egne undervisningsmateriell med andre. De kan dele både oppgaver og løsninger med andre interesserte. Sphero inkluderer også flere fagfelt. Det er altså et stort utvalg undervisningsmateriell i dette verktøyet.

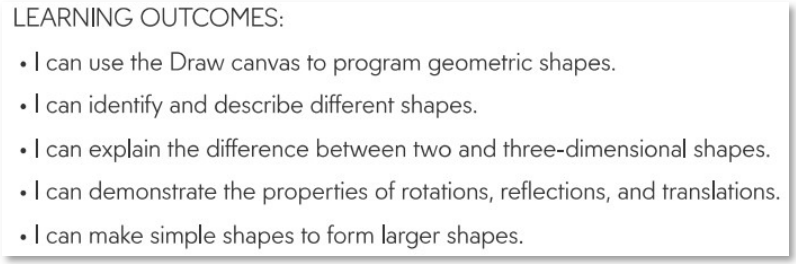
4.3.2 Vertikal analyse av Sphero – Geometric Transformations

I dette kapittelet skal jeg analysere undervisningsmateriellet Geometric Transformations, gitt ut av Sphero. Dette gjøres i lys av rammeverket som er sammensatt av rammeverket til Brennan og Resnick (2012) og Benton (2016). Jeg tar først for meg punktene under informasjon før jeg setter søkelyset mot algoritmisk tenkning i undervisningsmateriellet.

4.3.2.1 Informasjon

Undervisningsmateriellet henvender seg til den som skal løse oppgavene, altså eleven. Jeg antar derfor at målene og læringsmålene blir kommunisert til eleven. Både læringsmålene og målet for undervisningsmateriellet blir beskrevet på første side av dokumentet.

Målet med dette undervisningsmateriellet er å bruke tegne-lerretet i SpheroEdu-appen i arbeid med geometriske transformasjoner. De henviser til de geometriske transformasjonene: speiling, rotasjon og forskyvning. Læringsmålene for undervisningsmateriellet blir listet opp som vist i figur 71.


LEARNING OUTCOMES:

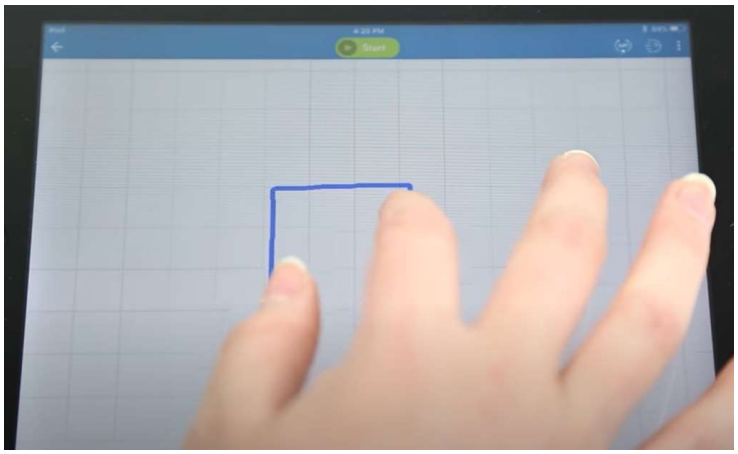
- I can use the Draw canvas to program geometric shapes.
- I can identify and describe different shapes.
- I can explain the difference between two and three-dimensional shapes.
- I can demonstrate the properties of rotations, reflections, and translations.
- I can make simple shapes to form larger shapes.

Figur 71

Dette undervisningsmateriellet presiserer rammer for individuelt arbeid og samarbeid. Det første steget i undervisningsmateriellet er individuelt arbeid der elevene så blir bedt om å gå sammen med en partner og diskutere funnene sine. I steg to blir det ikke spesifisert videre om de skal arbeide sammen eller ikke. I steg tre er det individuelt arbeid der de blir bedt om å gå sammen med en partner igjen for å dele arbeidet, komme med tilbakemeldinger og forbedre programmet sammen. I steg fire skal de arbeide med partneren gjennom hele steget. Der får de velge mellom tre ulike mål. Elevene skal enten tegne med en tusj, male med å dyppe

Sphero i maling eller å bruke foto med «long exposure» (Å ta bilde over tid slik at lysene til Sphero blir fanget opp som i et maleri).

Elevene får instruks om å se en video før de tar fatt på oppgaven: «Math planet's quick explanation of common types of transformation». Dette er en video, på et minutt, som forklarer begrepene: reflection, rotation og dilation. I steg to får elevene anbefalt å se en introduksjonsvideo for Draw-lerretet i Sphero edu-appen, dersom de ikke har vært borti det før. Denne videoen viser og forklarer hvordan elevene kan tegne på lerretet, fjerne tegningen, sikte inn Sphero og kjøre koden, dette kan du se i figur 72.



Figur 72

Begge videoene er før-arbeid for elevene, da de skal/burde ses før de tar fatt på oppgaven. For læreren er det listet opp utstyr som trengs for oppgavene, før-arbeidet til læreren er altså å klargjøre dette. Undervisningsmateriellet kommuniserer ikke spesifikke metoder for å tilpasse undervisningsopplegget til ulike elever i klassen.

Sphero opplyser ikke om hvordan arbeidet skal vurderes, men de oppfordrer til å ta bilder av arbeidet for å vise frem hva elevene har skapt.

4.3.2.2 Algoritmisk tenkning

Videre skal jeg ta for meg de algoritmiske konseptene og algoritmiske praksiser, slik det er beskrevet av Brennan og Resnick (2012), og de fem e'ene slik de blir presentert av Benton et al. (2017).

Elevene har muligheten til å lagre kodene sine, dette er det *algoritmiske konseptet data*. Når elevene trykker på start knappen kjøres koden, dette går under det *algoritmiske konseptet*

hendelse. Dette undervisningsmateriellet baserer seg på å tegne som en form for koding. Dette gjør at flere algoritmiske konsept, som løkker og sekvens, faller ut.

Elevene skal lage tre ulike tegninger som demonstrerer rotasjon, refleksjon og forskyvning. De skal gå sammen med partneren sin og dele arbeidet sitt. Elevene skal så samarbeide om å lage en tegning, eller kode. De har da muligheten til å teste ulike løsninger, og selv om de ikke har en kode de kan debugge, så har de en tegning de kan arbeide med for å finne bedre løsninger. De kan altså arbeide med den *algoritmiske praksisen testing og debugging* gjennom å teste ulike løsninger.

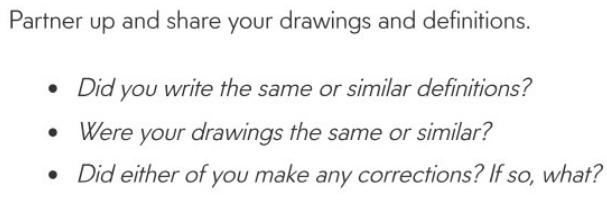
Brennan og Resnick (2012) beskriver den *algoritmiske praksisen gjenbruk og remiks* som en praksis hvor elevene tar nytte av hverandres koder og løsninger. Når elevene arbeider sammen for å lage en best mulig løsning og bygger på hverandres ideer, slik det er beskrevet i avsnittet over, så samsvarer dette med denne algoritmiske praksisen.

Undervisningsmateriellet legger opp til anvendelse av det *algoritmiske konseptet utforsking* gjennom å gi instruks om å *utforske* ulike muligheter i Sphero-appen. Elevene kan for eksempel *utforske* hvordan størrelsen på tegningen samsvarer med hvordan Sphero kjører. De kan også *utforske* farger og hurtigheten til Sphero.

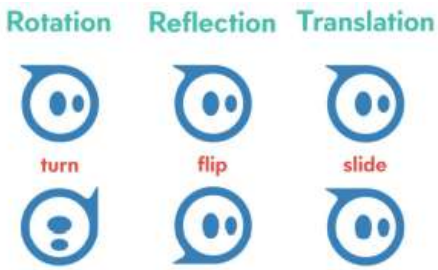
En av tingene elevene skal gjøre er å lage definisjoner til noen begrep, da skal de arbeide for seg selv. Dette krever at de må formulere seg. Videre blir de satt sammen med en partner som de skal diskutere definisjonene sammen med. De anvender altså *det algoritmiske konseptet å forklare* seg både skriftlig og muntlig.

Elevene skal skape noe konkret sammen, når de tegner kan de *se for seg* hvordan det de tegner blir oversatt til Sphero, for så å se hvordan dette vil bli oversatt til papiret de skal tegne, male eller ta bilde av. De kan så vurdere om de fikk ønsket utfall eller ikke. Dette blir ikke direkte kommunisert av undervisningsmateriellet, men det kan forekomme.

Undervisningsmateriellet legger opp til det *algoritmiske konseptet utveksle*, da det instruerer elevene til å gå sammen for å dele tegninger og definisjoner underveis i arbeidet. For eksempel i del 4 av første oppgave, som vist i figur 73.



Figur 73



Figur 74

Alle de *algoritmiske konseptene* blir arbeid med i geometrisk sammenheng, som vist i figur 74. Her *kobles* altså algoritmisk tenkning til geometri i matematikkfaget. Dette kommer tydelig frem gjennom målene for undervisningsmateriellet samt fagbegrepene som blir anvendt.

Denne oppgaven er for «Grades 6 to 8», dette tilsvarer 5. til 7. trinn i norsk skole. Dette undervisningsmateriellet baserer seg på en type programmering som ikke blir omtalt i disse trinnene. Jeg tror den faglige tyngden av geometrien er grunnen til at de har valgt så høye klassetrinn, men jeg ønsker å dra frem at undervisningsmateriellet legger opp til arbeid med kompetansemål innenfor programmering for lavere trinn.
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.

Funnene fra analysen har jeg sammenfattet i tabellen under. Brennan og Resnicks (2012) dimensjoner for algoritmisk tenkning blir listet opp, mens Benton et al. (2016) får et tall mellom 1 og 3. 1 beskriver at konseptet ikke ble funnet i analysen, 2 beskriver at det er funnet, men ikke kommer tydelig frem og 3 viser til at dette kommer tydelig frem i analysen.

Tabell 12

Sammenfatning av Sphero – Geometric Transformations					
Brennan og Resnick (2012)	Algoritmiske konsept			Algoritmisk praksis	
	-Data -Hendelse			-Å teste og debugge -Å gjenbruke og remikse -Abstrahering og modularisering	
Benton et al. (2016)	Utforske	Forklare	Se for seg	Utveksle	Koble
	3	3	2	3	3

4.3.3 Vertikal analyse av Maze Mayhem

I dette kapittelet skal jeg gå inn i den vertikale analysen av undervisningsmateriellet Maze Mayhem. Dette undervisningsmateriellet finnes på nettsiden, men det er noe mangelfullt i sammenligning med det samme undervisningsmateriellet i appen. I appen får læreren tips til hvordan gjennomføre undervisningsøkten, noe som er relevant for analysen. Jeg tar utgangspunkt i undervisningsmateriellet på nett da dette gir mer oversiktlige bilder til figurene i analysen, men jeg kommer til å hente tipsene til læreren fra appen. Dette blir analysert i henhold til analyseverktøyet i metodekapittelet. Jeg går først inn i hvordan undervisningsmateriellet formidler informasjon før jeg tar for meg algoritmisk tenkning, slik det er beskrevet av Benton et al. (2016) og Brennan og Resnick (2012).

4.3.3.1 Informasjon

Målene for undervisningsmateriellet blir presentert øverst i dokumentet. Målet for undervisningsmateriellet er at elvene skal programmere Sphero gjennom en labyrint. Her beskriver de også hvordan eleven skal nå målet, de skal nemlig samle data og finne den mest effektive veien gjennom labyrinten.

Læringsmålene blir presentert i en liste øverst i dokumentet og i appen som vist i figur 75.

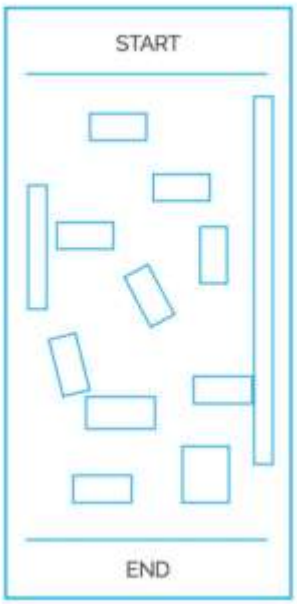
Figur 75

Figur 76

Figur 75 viser det første som kommer frem når du trykker på undervisningsmateriellet. Figur 76 er et skjermbilde av instruktørtips gitt i den første deloppgaven i dette undervisningsmateriellet.

Undervisningsmateriellets rammer for aktiviteten kommer frem i form av tips til læreren i SpheroEdu-appen, som vist i figur 76. Læreren får tips om å dele elevene i grupper bestående av 3-4 elever før de tar fatt på første steg. Undervisningsmateriellet foreslår også å gi elevene spesifikke roller når de tar fatt på steg tre og fem. Det står at aktiviteten tar 1-2 timer. Læreren får også informasjon om omtrent hvor lang tid hvert steg tar. Steg 1: 15min, steg 2: 15 min, steg 3: ingen informasjon, steg 4: 30 min, steg 5: 60-90 min. Videoen i figur 76 viser et eksempel på en labyrint elevene/læreren kan bygge.

En del av aktiviteten er at elevene skal bygge en labyrint som de skal bruke når de skal programmere Sphero. Sphero kommer med tips til læreren om at hen kan bygge labyrinten selv, så kan elevene ta fatt på resten av aktiviteten. Dersom elevene bygger labyrinten kan de gjøre det ut fra materialer som er tilgjengelig i klasserommet, eller med å teipe en labyrint på gulvet. Undervisningsmateriellet kommer med eksempel på labyrint som vist i figur 77 og 78.



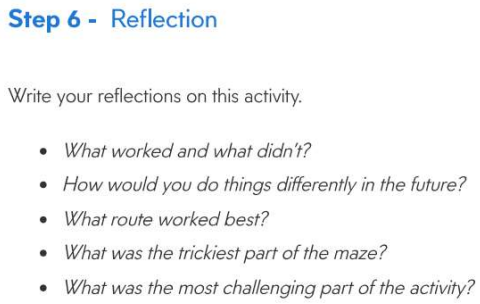
Figur 77 *Figur 78*

Undervisningsmateriellet lister opp ting som kan være nyttig når elevene skal bygge en labyrint, samt et par måleredskaper. Dette er ting som er lurt å ha klart, eller tilgjengelig, før elevene tar fatt på aktiviteten, jeg teller derfor dette som før-arbeid for læreren. Som

avslutning på aktiviteten får elevene noen refleksjonsspørsmål. Disse skal gjøres etter de er ferdig med programmeringen, spørsmålene kan fungere som etterarbeid.

Sphero presiserer ikke hvordan oppgaven kan tilpasses for ulike elever.

Dette undervisningsmateriellet viser ikke til noen metode for å vurdere arbeidet fra lærerens perspektiv, men det siste steget er å reflektere over eget arbeid, som vist i figur 79. Dette åpner for å la elevene selv vurdere hvordan arbeidet med aktiviteten har gått og reflektere over hvordan de kunne ha løst den nå som de sitter igjen med erfaringen fra denne aktiviteten.

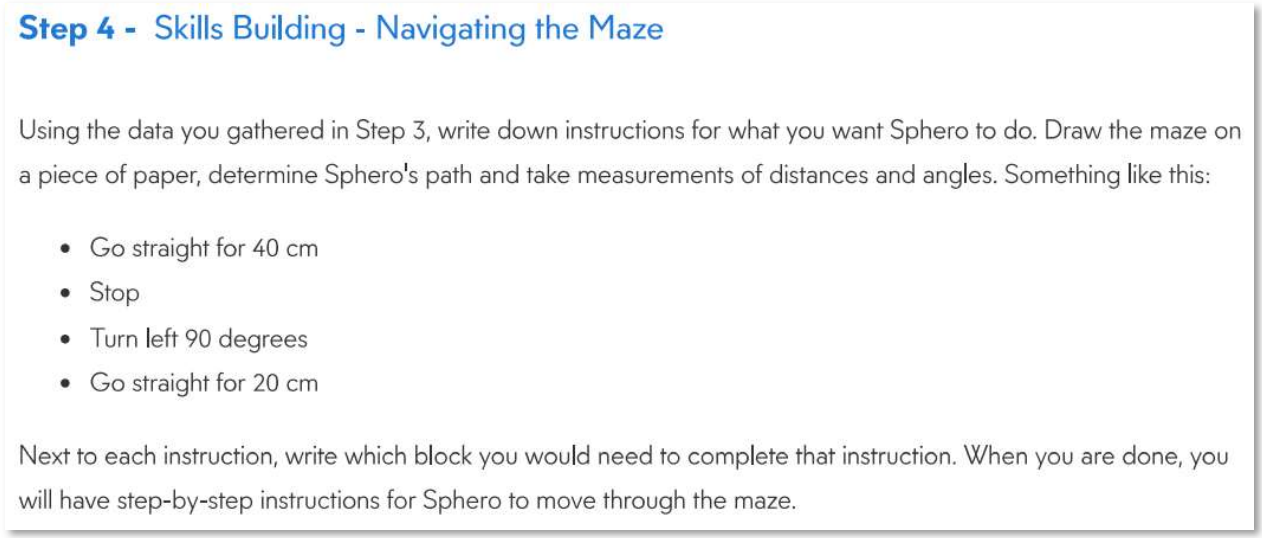


Figur 79

4.3.3.2 Algoritmisk tenkning

Dette undervisningsmateriellet begrenser ikke elevene til enkelte *algoritmiske konsept*. De programmerer gjennom blokkprogrammering, som gjør at elevene bare forholder seg til de blokkene som er tilgjengelig i appen. Disse innebærer muligheten til å bygge *sekvenser, løkker, paralleller, hendelser, betingelser, operatører* og *data*. Aktiviteten spesifiserer ikke hvilke av de *algoritmiske konseptene* som skal anvendes, men elevene står fritt til å arbeide med de ulike blokkene og utforske mulighetene. Dette fører til at noen elever vil anvende løkker og andre vil kanskje ikke det.

Undervisningsmateriellet legger opp til at elevene skal anvende den *algoritmiske praksisen teste og debugge*. De får instrukser på hvordan de kan samle inn data, skrive ned hva de ønsker at Sphero skal gjøre og sette sammen koden for å oppnå ønsket resultat. Når de skriver koden må de teste hvordan den fungerer og vurdere om det er mulig å forbedre den, altså debugge.



Step 4 - Skills Building - Navigating the Maze

Using the data you gathered in Step 3, write down instructions for what you want Sphero to do. Draw the maze on a piece of paper, determine Sphero's path and take measurements of distances and angles. Something like this:

- Go straight for 40 cm
- Stop
- Turn left 90 degrees
- Go straight for 20 cm

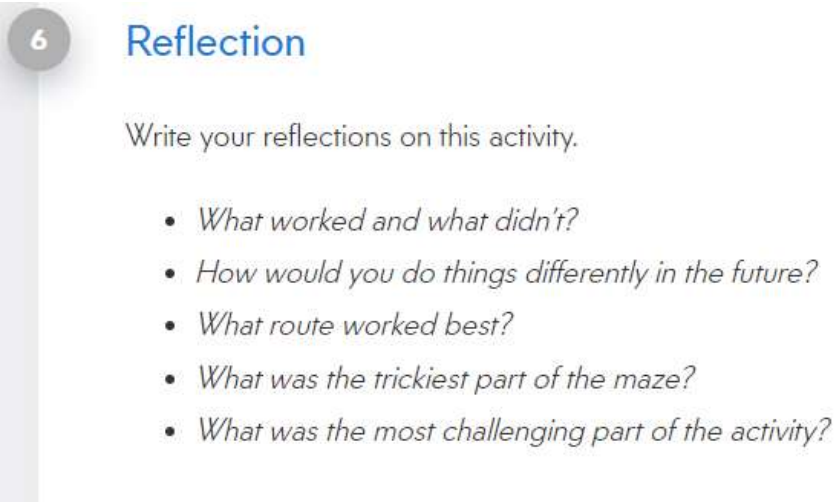
Next to each instruction, write which block you would need to complete that instruction. When you are done, you will have step-by-step instructions for Sphero to move through the maze.

Figur 80

Elevene får instrukser om å lage en liste over hva de ønsker at Sphero skal gjøre, som eksempelet en kan se i figur 80. Da vil elevene lage mange små koder for sine delmål, før de settes sammen til en lengre kode. Å sette sammen en kode av flere små deler tar bort unødvendig informasjon og en kan fokusere på mindre deler av programmeringen av gangen, dette går under den *algoritmiske praksisen å abstrahere og modularisere*

Elevene får muligheten til å *utforske* ulike løsninger, da de i steg fem skal programmere roboten og gjøre forandringer på koden underveis dersom de ønsker det. I dette steget får elevene hint om å *utforske* «the Delay Block» dersom de ikke får roboten til å svinge skarpt nok for labyrinten. Det er ingen fasit i undervisningsmateriellet, trolig fordi ulike klasser vil lage ulike labyrinter, dette åpner for at elevene kan *utforske* ulike løsninger for å finne den raskeste ruten.

Undervisningsmateriellet legger opp til at elevene skal *forklare* ulike aspekt av aktiviteten gjennom avsluttende refleksjonsspørsmål. Svarene på refleksjonsspørsmålene skal skrives ned, noe som gjør at elevene må sette ord på det de har gjort.



Figur 81

Før elevene koder i appen skal de skrive ned instruksjonene og blokkene som trengs for å få Sphero til å gjennomføre instruksjonene, dette kan en se i figur 81. Da kan elevene se for seg hvordan Sphero skal bevege seg, stegvis, gjennom labyrinten. Elevene får også refleksjonsspørsmål som vist i figur 81. Dette gjør at jeg mener at det *algoritmiske konseptet å se for seg* kommer tydelig frem i dette undervisningsmateriellet.

Undervisningsmateriellet legger opp til at elevene skal samarbeide. Dette åpner for at de kan arbeide gjennom den *algoritmiske praksisen utveksle* da de kan diskutere ideene sine og komme frem til en felles løsning. Undervisningsmateriellet kommuniserer ikke at elevene skal diskutere, men det er en mulighet. Å ha grupper tilsvarer ikke at det blir faglige samtaler. Jeg kan derfor ikke konkludere med at det vil ta sted, men at de kan det.

Undervisningsmateriellet *kobler* algoritmisk tenkning til de matematiske kontekstene programmering, fart, avstand og vinkler. Elevene må måle avstander, beregne farten og svingradiusen til Sphero og vurdere hvilke ruter som er mest effektive. Dette undervisningsmateriellet er tiltenkt trinnene «2 to 12», som tilsvarer 1. trinn til det siste året på ungdomsskolen. Jeg tar ikke for meg kompetansemålene for ungdomsskolen da det ikke er relevant for denne masteroppgaven, jeg tar for meg kompetansemålene for 1. til 7. trinn. Med disse kriteriene kom jeg frem til disse to kompetansemålene:
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker.
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker.

Under er en tabell som oppsummerer funnene av algoritmisk tenkning i dette undervisningsmateriellet.

Tabell 13

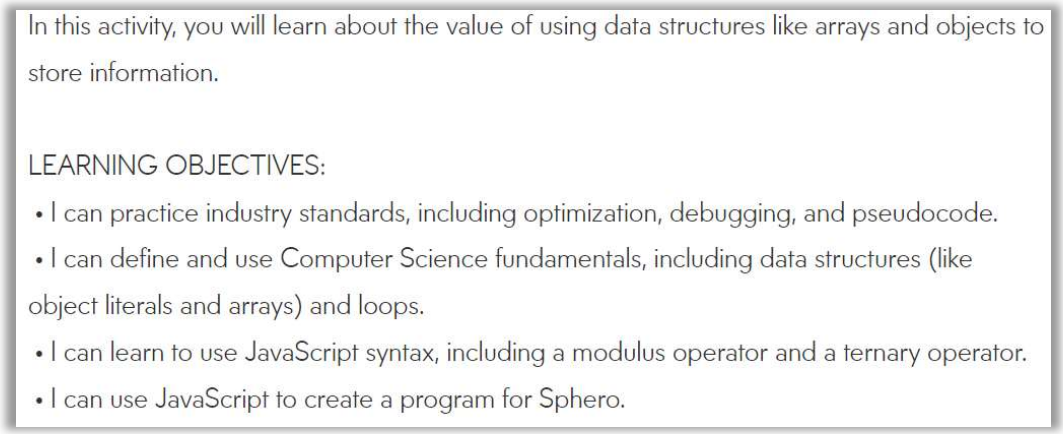
<table>
<tr><td colspan="6">Sammenfatning av Sphero – Maze Mayhem</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Data
-Hendelse</td><td colspan="2">-Testing og debugging
-Gjenbruk og remiksing
-Abstrahering og modularisering</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>2</td><td>3</td></tr>
</table>

4.3.4 Vertikal analyse av Morse Code & Data Structures

I dette kapittelet skal jeg ta for meg den vertikale analysen av undervisningsmateriellet Sphero, Morse Code. I dette undervisningsmateriellet skal elevene arbeide med tekst-programmering og anvende kodespråket JavaScript. Jeg tar først for meg punktet informasjon, her ser jeg etter målet til oppgaven, hvilke rammer som blir beskrevet og hvordan elevene skal vurderes. Jeg går så videre til algoritmisk tenkning slik jeg har beskrevet det av Benton et al. (2016) og Brennan og Resnick (2012).

4.3.4.1 Informasjon

Målet for denne aktiviteten er å lage og dele en melding med bruk av morsekode gjennom programmering av Sphero. Dette blir kommunisert øverst i undervisningsmateriellet før det går inn i læringsmålene, som vist i figur 82.



Figur 82

Undervisningsmateriellet kommuniserer noen rammer, men ikke om noen form for tilpasninger for ulike elever. Det kommer ikke tydelig frem i undervisningsmateriellet på nett om elevene skal arbeide individuelt eller i samarbeid, men i SpheroEdu-appen referer de til eleven/-e som «du». Jeg antar derfor at det er ment som individuelt arbeid. Undervisningsmateriellet legger inn en tidsrammer på 1-2 timer. Elevene får etterarbeid i form av refleksjonsspørsmål og spørsmål om definisjoner på ulike begrep. For dette undervisningsmateriellet vil det være fordelaktig å ha noe kompetanse eller forforståelse for programmeringsspråket JavaScript da de starter med å tyde kode i første del-oppgave.

Figur 83

Figur 83 er et skjermbilde tatt fra SpheroEdu-appen. Dette er tips til læreren/instruktøren om hvordan denne deloppgave 5 skal løses. Undervisningsmateriellet kommuniserer ikke hvordan elevene skal vurderes eller hvordan de skal få tilbakemelding, men elevene får i oppgave å evaluere hva de synes var vanskeligst og enklest, før de skal komme med beskrivelser på noen begrep og metoder. Hvordan roboten utfører koden er også en form for

tilbakemelding, da elevene kan se om de fikk forventet utfall eller ikke. Da har elevene mulighet til å selv reflektere og vurdere hvordan det har gått å arbeide med dette.

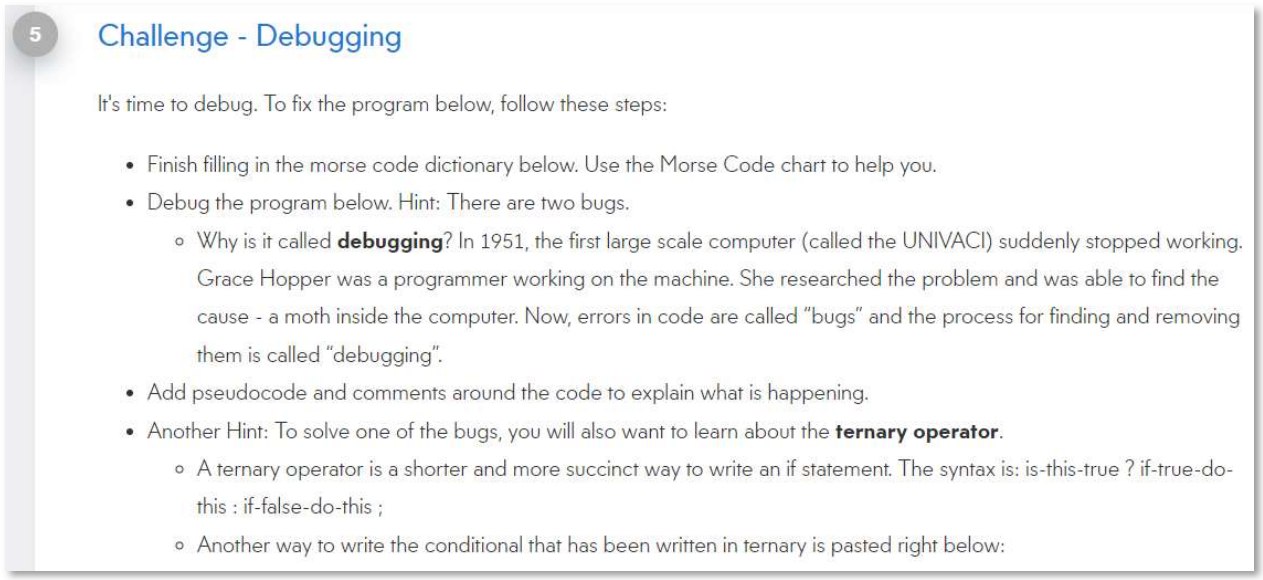
4.3.4.2 Algoritmisk tenkning

Dette undervisningsmateriellet får frem de *algoritmiske konseptene sekvens, hendelse, betingelse, operatører* og *data*. *Sekvens* er en liste over kommandoer som beskriver hva som skal skje når en *hendelse* tar sted. *Hendelsen* er i dette tilfelle å trykke på knappen for å kjøre koden. *Betingelse* kommer frem som hvis-setninger i undervisningsmateriellet. De bruker også *operatører* for eksempel når de skal tildele verdier til hver bokstav for å kunne enklere stave flere ord. Bokstaven A får tildelt [1,3] som gir et kort og et langt signal i morsekode. Denne verdien er en *operatør* og den blir lagret som *data*, slik det blir vist i figur 84.

```
const morseCode = {
 a: [1, 3],
 b: [3, 1, 1, 1],
 c: [3, 1, 3, 1]
}
```

Figur 84

Figur 84 viser eksempel på hvordan elevene kan skrive en liste over alfabetet. Dette viser til at en nå bruke bokstavene A, B og C i et ord på en mer effektiv måte. Når alle bokstavene har fått tildelt *data* på denne måten kan en skrive et ord i koden, koden vil da oversette ordet til morsekode.



5 Challenge - Debugging

It's time to debug. To fix the program below, follow these steps:

- Finish filling in the morse code dictionary below. Use the Morse Code chart to help you.
- Debug the program below. Hint: There are two bugs.
  - Why is it called **debugging**? In 1951, the first large scale computer (called the UNIVACI) suddenly stopped working. Grace Hopper was a programmer working on the machine. She researched the problem and was able to find the cause - a moth inside the computer. Now, errors in code are called "bugs" and the process for finding and removing them is called "debugging".
- Add pseudocode and comments around the code to explain what is happening.
- Another Hint: To solve one of the bugs, you will also want to learn about the **ternary operator**.
  - A ternary operator is a shorter and more succinct way to write an if statement. The syntax is: is-this-true ? if-true-do-this : if-false-do-this ;
  - Another way to write the conditional that has been written in ternary is pasted right below:

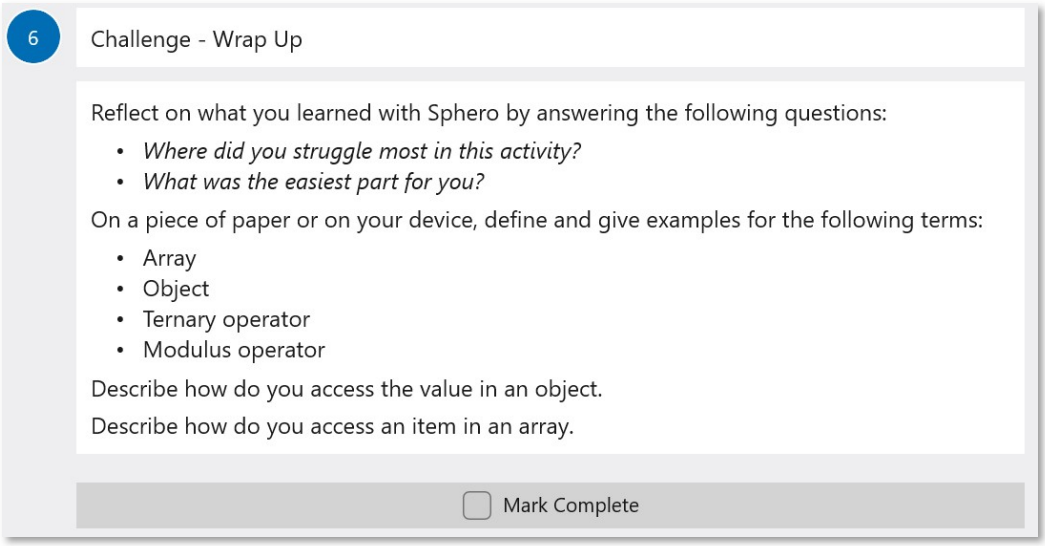
Figur 85

Debugging handler om feilsøking og å rette opp i feil. I undervisningsmateriellet finner jeg den *algoritmiske praksisen teste og debugge*, da undervisningsmateriellet instruerer elevene til å finne feilene i en kode for så å rette opp disse, som vist i punkt to i figur 85.

I oppgaven skal elevene ta for seg en kode som får roboten til å sende morsekode i form ut fra hvilket ord du vil koden skal sende. Elevene tar da utgangspunkt i koden som er der, samt listen over dataene for hver bokstav. De redigerer koden og bruker denne til å løse oppgaven. Her er det derfor snakk om den *algoritmiske praksisen gjenbruk og remiks*.

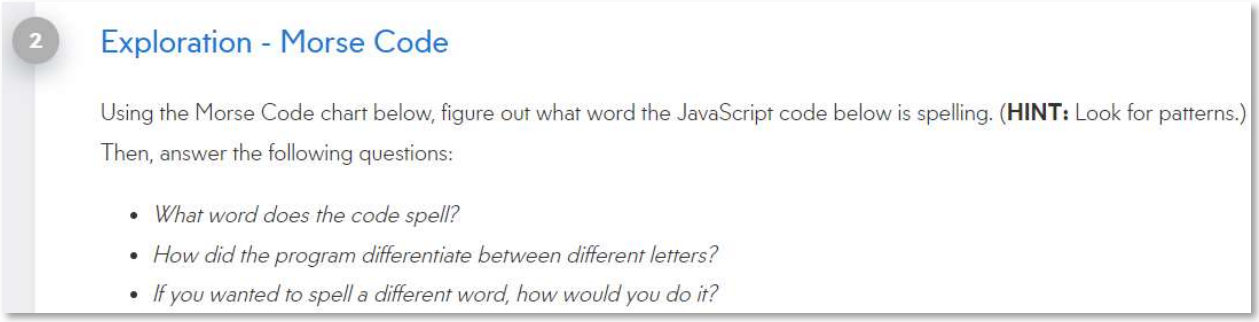
Elevene lager *sekvenser* som inneholder ulike komponenter, som *data* og *betingelser*. Dette går under den *algoritmiske praksisen abstrahere og modularisere*, da dette beskriver hvordan en kan bygge mange sekvenser som samarbeider i stedet for bare en kode. Dette undervisningsmateriellet viser nytten av å kunne lagre data slik at de kan skrive nye ord i morsekode på en mer effektiv måte.

Å *utforske* handler, slik jeg forstår Benton et al. (2016), om å la elevene få muligheten til å teste ulike ideer og debugge. Undervisningsmateriellet legger opp til *utforsking* slik det også legger opp til anvendelse av den algoritmiske praksisen testing og debugging.



Figur 86

I figur 86 får elevene refleksjonsspørsmål som ber dem om å *forklare* noen begrep og hva de synes var utfordrende og mindre utfordrende. At elevene må artikulere tankene og ideene de har tilegnet seg gjør at elevene utvikler en bedre forståelse for kunnskapen de nå har fått.



Figur 87

Figur 87 viser refleksjonsspørsmål elevene får i oppgave 2. Elevene får spørsmål om hva de ville gjort for å stave et annet ord dette kan få elevene til å *se for seg* ulike løsninger. Jeg mener derfor at undervisningsmateriellet legger opp til det *algoritmiske konseptet å se for seg*.

Undervisningsmateriellet legger ikke opp til å *utveksle* ideer og funn. Det blir ikke informert om elevene skal arbeide i grupper eller individuelt, her kommer det altså an på hvilke rammer læreren gir om de kommer til å ha denne muligheten til å arbeide gjennom *utveksling* eller ikke. Undervisningsmateriellet kommuniserer ikke andre muligheter for å *utveksle* kompetanse.

Dette undervisningsmateriellet *kobles* til matematikkfaget gjennom programmering. Det er ment for elever i «6th to 12th+ grade» som tilsvarer fra syvende trinn i grunnskolen i Norge. Jeg har valgt, slik som i de tidligere analysene, å ta for meg kompetansemål som omhandler programmering i alle trinn. Dette fordi jeg ikke ønsker å risikere å utelate gode poeng. Elevene lagrer data til hele alfabetet og arbeider med å bruke datasettet for å skrive ulike ord i morsekode:
7. trinn: bruke programmering til å utforske data i tabeller og datasett.

Nedenfor er en tabell hvor jeg har lagt inn funnene av algoritmiske konsept.

Tabell 14

<table>
<tr><td colspan="6">Sammenfatning av Sphero – Morse Codes & Data Structures</td></tr>
<tr><td rowspan="2">Brennan og Resnick (2012)</td><td colspan="3">Algoritmiske konsept</td><td colspan="2">Algoritmisk praksis</td></tr>
<tr><td colspan="3">-Sekvenser
-Hendelse
-Betingelse
-Operatør
-Data</td><td colspan="2">-Teste og debugge
-Abstrahere og modularisere</td></tr>
<tr><td rowspan="2">Benton et al. (2016)</td><td>Utforske</td><td>Forklare</td><td>Se for seg</td><td>Utveksle</td><td>Koble</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>1</td><td>2</td></tr>
</table>

4.3.5 Oppsummering Sphero

I dette kapittelet skal jeg kort oppsummere funnene fra de vertikale analysene for undervisningsmateriellene tilhørende Sphero. Jeg tar for meg punktene for informasjon først og går videre inn på algoritmisk tenkning slik det blir presentert i analyseverktøyet i metodekapittelet.

4.3.5.1 Informasjon

I alle tre undervisningsmateriellene kom målene for aktiviteten og læringsmålene tydelig frem, øverst i dokumentet. Målet for aktiviteten er det første som står i dokumentet og under denne beskrivelsen kom læringsmålene i en punktliste.

Undervisningsmateriellene hadde varierende mengde informasjon om rammene rundt aktiviteten. Aldersgruppen og hvor mye tid en kan regne med å bruke står øverst i alle tre, de har også en liste over hvilket utstyr som er nødvendig for å kunne gjennomføre aktiviteten, ting som må gjøres klart før aktiviteten kan starte går under før-arbeid. Hvorvidt elevene skulle samarbeide eller ikke kom tydelig frem i Geometric Transformations, men ble ikke nevnt i de gjenværende undervisningsmateriellene.

Verken lærer eller elev fikk informasjon om hvordan arbeidet skulle vurderes. Det som var felles for alle tre var at de inkluderte refleksjonsspørsmål, som oppfordret elevene til å vurdere arbeidet. Det var spørsmål om å definere begrep og å reflektere rundt hva som var krevende i aktiviteten. I Geometric Transformations kom disse spørsmålene underveis og sluttresultatet var et bilde/maleri. I de to andre kom disse etter de var ferdig med aktiviteten. Både i Maze Mayhem og Morse Code & Data Structures skulle de skrives ned. Det er altså muligheter for læreren for å vurdere og gi tilbakemelding på arbeidet til elevene, men det blir ikke kommunisert av Sphero at dette kreves.

4.3.5.2 Algoritmisk tenkning

De ulike undervisningsmateriellene la opp til anvendelse av flere *algoritmiske konsept*. I Geometric Transformations fant jeg de algoritmiske konseptene *data* og *hendelse*. I Maze Mayhem ble det ikke inkludert eller ekskludert noen algoritmiske konsept. Her kunne elevene kode slik de selv ønsket, men målet var å finne den mest effektive ruten gjennom labyrinten. Dersom de ønsket å få Sphero til å lyse samtidig som den kjørte eller bruke løkker for å oppnå målet så var det ingenting som hindret disse mulighetene. I Morse Code & Data Structures ble de *algoritmiske konseptene sekvens, hendelse, betingelse, operatør* og *data* funnet.

Gjennom analysen av de tre undervisningsmateriellene ble det funnet tre former for algoritmisk praksis. Gjenbruk og remiksing, abstrahering og modularisering og testing og debugging.

Jeg fant Benton et al. (2016) beskrivelse av *utforsking* i alle tre undervisningsmateriellene. Elevene får instruks om å *utforske* ulike muligheter i undervisningsmateriellet Geometric Transformations. I Maze Mayhem skal elevene *utforske* gjennom å finne den mest effektive løsningen på oppgaven, og i Morse Code & Data Structures ble de instruert til å arbeide med debugging – som da går under utforskende arbeid.

I alle tre undervisningsmateriell ble det kommunisert at elevene skulle *forklare* ulike funn. I alle tre fikk de instrukser om å enten definere begrep og/eller reflektere rundt refleksjonsspørsmål og skrive ned svarene sine på dette.

Å se for seg kom frem gjennom undervisningsmateriellene, da det er mulig elevene vil se for seg ulike løsninger. Elevene får også refleksjonsspørsmål avslutningsvis.

Elevene fikk muligheten til å *utveksle* ideer og funn gjennom undervisningsmateriellet til Geometric Transformations. Da kommuniserer undervisningsmateriellet at elevene skal gå sammen for å sammenligne funnene sine og finne den beste løsningen. Denne måten å arbeide på kommer ikke frem gjennom de to resterende undervisningsmateriellene. Her blir det ikke kommunisert om elevene skal samarbeide eller ikke, så det kommer an på hvilke rammer læreren setter for aktiviteten.

De tre undervisningsmateriellene *kobler* algoritmisk tenkning til matematikk. Både Geometric Transformations og Maze Mayhem anvendte flere matematiske ferdigheter enn programmering, mens Morse Code & Data Structures holdt seg til programmering. Undervisningsmateriellet instruerer til å arbeide med disse fire kompetansemålene.
2. trinn: lage og følge regler og trinnvise instruksjoner i lek og spill.
4. trinn: lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker.
5. trinn: lage og programmere algoritmer med bruk av variabler, vilkår og løkker.
7. trinn: bruke programmering til å utforske data i tabeller og dataset.

Under er en sammenfatning av alle punktene for algoritmisk tenkning som ble funnet de vertikale analysene av undervisningsmateriellene tilhørende Sphero.

Tabell 15

Sammenfatning av Sphero					
Brennan og Resnick (2012)	Algoritmiske konsept			Algoritmisk praksis	
	-Sekvens -Data -Hendelse -Betingelse -Operatør			-Å gjenbruke og remikse -Abstrahering og modularisering -Testing og debugging	
Benton et al. (2016)	Utforske	Forklare	Se for seg	Utveksle	Koble
	3	3	3	2	3

5 Diskusjon

I dette kapittelet er hensikten å diskutere funnene fra analysen. Jeg skal ta for meg hvordan de ulike verktøyene legger opp til arbeid med de fem e'ene, de algoritmiske konseptene og de algoritmiske praksisene. Formålet med dette kapittelet er å få frem hvordan de ulike undervisningsmateriellene har lagt opp til arbeid med algoritmisk tenkning. Dette ønsker jeg å gjøre gjennom å belyse hvordan de ulike punktene kommer frem i verktøyene. Hensikten er ikke å henge ut noen av dem som mindre gode enn andre.

Hva kjennetegner så disse undervisningsmateriellene og programmeringsverktøyene? Gjennom analysen har jeg funnet stor variasjon mellom de ulike verktøyene og noe variasjon innad i verktøyene.

De tre verktøyene formidler informasjon i ulik grad. Kodeskolen formidler lite informasjon til læreren, men forklarer godt gjennom videoer og oppgavetekster hva det er elevene skal gjøre. KUBO har mye informasjon, dette er undervisningsmateriell som krever at en setter seg inn i de ulike delene (lærerguide etc.) før en starter undervisningen. I Sphero fant jeg varierende mengde informasjon. De var konsekvent i at målene kom tydelig frem, men rammene var varierte.

Det er med andre ord stor ulikhet i hva som kreves av læreren. Som nevnt i innledningen av oppgaven er det stor variasjon i hvor mye kompetanse lærerne har i programmering og algoritmisk tenkning, da dette er nye fagområder i læreplanen.

Fordi kodeskolen gir lite informasjon til læreren virker det ut som at kodeskolen legger opp til av elevene arbeider individuelt med lite innblanding fra lærere. Dette kan ses i sammenheng med at mange lærere har lite erfaring med programmering og de har dermed laget et lavterskel opplegg, som lærer med lite til ingen kunnskap kan bruke. Verktøyet legger ikke ansvar på at læreren skal tilegne seg kompetanse.

I kontrast til Kodeskolen gir KUBO store mengder informasjon. De inkluderer en liste som omhandler hva en må tenke på før timen, underveis og etter. KUBO beskriver at elevene skal samarbeide gjennom arbeidet med undervisningsmateriellene. Utenfor undervisningsmateriellene jeg har analysert har KUBO også videoer som er ment for å lære lærere hvordan KUBO fungerer. KUBO tilrettelegger for at lærerne kan tilegne seg kompetanse på feltet og gir lærerne et ansvar for å tilegne seg denne før de tar fatt på

undervisningen. Gitt at lærerne får tiden de trenger til å sette seg inn i all informasjonen KUBO gir så anser jeg dette som et verktøy de fleste lærere kan mestre i et klasserom.

Spheros undervisningsmateriell gir læreren varierende mengde informasjon. De avklarer ikke alltid hvorvidt elevene skal samarbeide eller ikke, da må man heller tolke teksten for å finne ut av det. Læreren har en rolle i undervisningen, slik som hen har i KUBO, men Sphero har ikke de ressursene som KUBO tilbyr for å lære lærere hvordan verktøyet deres fungerer. Anvendelse av Sphero i klasserommet «krever» at læreren har kompetanse innen algoritmisk tenkning og programmering.

5.1 Algoritmisk tenkning

Verktøyene la opp til anvendelse av flere ulike algoritmiske konsept. Kodeskolen la opp til anvendelse av *hendelse, sekvens, løkke* og *variabel. Sekvens* og *løkker* finner vi igjen i kompetansemålene i matematikk. I kompetansemålene nevnes også vilkår, men dette konseptet ble ikke funnet i Kodeskolen.

KUBO legger opp til bruk av *sekvens, løkke, variabel, hendelse, betingelse* (som da tilsvarer vilkår i læreplanen), *operatør* og *data*. I det første undervisningsmateriellet ble det bare funnet to *algoritmiske konsept*, dette kan skyldes at det er ment som en introduksjon til verktøyet, på lik linje som det første brettet i kodeskolen er en introduksjon til kodeskolen.

I Sphero fant jeg de *algoritmiske konseptene sekvens, data, hendelse, betingelse* og *operatør.* I undervisningsmateriellet til Sphero – Maze Mayhem ble det ikke kommunisert noen *algoritmiske konsept,* men siden aktiviteten baserer seg på å programmere Sphero til å kjøre gjennom en labyrint raskest mulig vil flere *algoritmiske konsept* tatt i bruk, da programmering bygger på disse konseptene. Hvilke *algoritmiske konsept* som vil bli brukt kan derfor ikke sies med noen form for sikkerhet.

Ved å sette verktøyene opp mot algoritmiske konsepter som finnes i læreplanen ser vi at KUBO treffer alle konseptene, mens Sphero og Kodeskolen kommer inn på noen av disse. Sphero har mange ulike typer undervisningsmateriell, at jeg ikke fant det i min analyse betyr bare at det ikke ble funnet. Sphero kan ha integrert flere *algoritmiske konsept* i andre undervisningsmateriell. Kodeskolen har lagt opp til anvendelse av færrest *algoritmiske konsept.* De har, til ulikhet med Sphero, gjentagende brett. Jeg kan ikke konkludere med at det ikke finnes flere *algoritmsike konsept* i Kodeskolen, men sannsynligheten for å finne flere

i deres undervisningsmateriell anser jeg som lavere enn at en kan finne flere *algoritmsike konsept* i andre undervisningsmateriell tilhørende Sphero.

Kodeskolens undervisningsmateriell legger ifølge analysen opp til anvendelse av tre *algoritmiske praksiser. Teste og debugge, abstrahere og modularisere* og *inkrementell og iterativ.* Kodeskolen er det eneste verktøyet som har et undervisningsmateriell hvor jeg ikke fant noen algoritmiske praksiser. Dette er trolig fordi det er det første brettet, som jeg antar er der for å introdusere elevene til verktøyet på en enkel måte.

I KUBO fant jeg fire *algoritmiske praksiser, teste og debugge, gjenbruke og remikse, inkrementell og iterativ* og *abstrahere og modularisere*.

I undervisningsmateriellet til Sphero fant jeg også tre *algoritmiske praksiser*. Der kom det frem at *gjenbruk og remiks, abstrahere og modularisere* og *testing og debugging* blir kommunisert.

De *algoritmiske praksisene* som går igjen i disse programmeringsverktøyene er å *teste og debugge* og *abstrahere og modularisere.* Det disse verktøyene har til felles innenfor *algoritmiske praksiser* er altså at de legger opp til at elevene skal, eller i alle fall har mulighet til, å teste ulike løsninger og debugge dersom de ikke får ønsket utfall ved første forsøk og at de legger opp til at elevene skal ta for seg mindre deler av et problem.

Selv om de har disse praksisene til felles så er de og veldig ulike. At en i kodeskolen kan løse et brett så mange ganger en vil gjør at en kan argumentere for at elevene kan bruke praksisen *teste og debugge.* Jeg vil dog påpeke at det er forskjell på denne typen testing opp mot å arbeide med debugging, slik som en får muligheten til i KUBO og Sphero. I både KUBO og Sphero har elevene mulighet til å gå inn i kodene sine for å finne ut hvorfor de fikk, eller ikke fikk, ønsket utfall. Kodeskolen kunne forbedret seg innenfor denne *algoritmiske praksisen* ved å ikke slette koden til elevene i det den kjøres.

Det ble også funnet variasjon i analysen av Benton et al. (2016) algoritmiske konsept. Undervisningsmateriellene fikk en poengsum i hver av kategoriene: Utforske, forklare, se for seg, utveksle og koble.

Kodeskolen fikk 2 i samlet vurdering i Benton et al. (2016) konsept å utforske. I denne kategorien fikk det første undervisningsmateriellet til kodeskolen den eneste eneren. Dette brettet er ment som en introduksjon og er lagt opp for å ikke kunne gjøre feil, så noen form

for utforsking her ble utelukket. Med unntak av det var det mulighet å utforske ulike løsninger, på samme premissene som at det er mulig å teste ulike løsninger – som jeg diskuterte under algoritmiske praksiser. KUBOs samlede vurdering i utforsking ble 3. To av undervisningsmateriellene kommuniserte tydelig at elevene skulle arbeide gjennom utforsking. Dette ble mindre tydelig i KUBO Coding Math, da fokuset der heller var på å regne mattestykker. Det er dog mulig å utforske så det ble en sum på 2. Sphero er det eneste verktøyet hvor alle undervisningsmateriellene fikk 3 i det algoritmiske konseptet utforsking, da de

Benton et al. (2016) beskriver konseptet å forklare som et konsept for å fremme diskusjon og forklaring, for at elevene skal artikulere tankene og ideene sine. Både KUBO og Sphero fikk 3. Undervisningsmateriellet KUBO Coding Math fikk 2, men med unntak av dette la undervisningsmateriellene opp til utforsking i stor grad. Alle de tre undervisningsmateriellene til Sphero la opp til at elevene skulle forklare seg. Kodeskolen legger opp til individuelt arbeid og gir ikke elevene spørsmål de må svare på, eller som oppfordrer dem til refleksjon, kodeskolen fikk derfor ikke like god uttelling i dette konseptet som de andre to verktøyene. Dette kan også ses opp mot at verktøyet ikke legger noen form for press på lærerens kompetanse.

Når det kommer til *det algoritmiske konseptet å se for seg* var det også noe variasjon. I kodeskolen er dette en metode som godt mulig kan bli brukt, men det blir ikke kommunisert at elevene må bruke tid på å planlegge og reflektere rundt løsningene til de ulike brettene. Dette gjorde at alle brettene fikk scoren 2. I KUBO ble det funnet oppgaver som oppfordret elevene til å planlegge ruter eller lignende, i tillegg har KUBO spørsmål underveis som oppfordrer til refleksjon rundt utfallene. KUBO Coding Math skilte seg ut fra de andre undervisningsoppleggene også når det gjelder det *algoritmiske konseptet å se for seg*. KUBO Coding Math skilte seg ut fra de andre to undervisningsmateriellene også på dette punktet. Det er mulig å arbeide gjennom *å se for seg,* men det ble ikke kommunisert at elevene skulle bruke disse metodene. I to av de tre undervisningsmateriellene tilhørende Sphero kom aspektet *å se for seg* tydelig frem.

Kodeskolen har lagt opp til individuelt arbeid. Dette fører til at elevene ikke har mulighet til å *utveksle* arbeid eller tanker med medelever og verktøyet fikk derfor 1 i score. KUBO Coding Routes og KUBO Coding Math har fått 2, dette er fordi de legger opp til situasjoner hvor det er mulig at elevene *utveksler* ideer og kunnskap. KUBO Coding ++ - Super Coders har fått

vurderingen 3, da undervisningsmateriellet instruerer elevene til å diskutere tankene sine. I Sphero har de tre undervisningsmateriellene fått tre ulike vurderinger. Et av undervisningsmateriellene er, ifølge analysen, ment for individuelt arbeid. Det andre er gruppearbeid, men undervisningsmateriellet kommuniserer ikke at elevene skal *utveksle* noe, og det siste kommuniserte at elevene skal anvende den *algoritmiske praksisen utveksle.* Fremkomsten av dette algoritmiske konseptet varierer altså mellom verktøy og verktøyene som har fått uttelling for dette har lagt opp til utveksling i varierende grad.

Verktøyene viste seg å være ulike også når det kom til hvordan de *koblet* algoritmisk tenkning til matematikk. Alle undervisningsmateriellene i analysen *koblet* algoritmisk tenkning til programmering. Dette gjorde at her bare er et skille mellom de som *kobler* programmering til algoritmisk tenkning og de som *kobler* flere matematiske fagområder inn i undervisningsmateriellene.

I Kodeskolen kan en ha et rutenett over brettet. Det går an å diskutere om hvorvidt dette kan være en start på å lære om koordinatsystem, men dette kommer helt an på læreren da det ikke blir kommunisert av undervisningsmateriellene. I analysen fant jeg derfor at Kodeskolen bare kobler algoritmisk tenkning til matematikk gjennom programmering. KUBO anvender et kart hvor det er koordinater.

I KUBO ble det funnet at undervisningsmateriellene da *kobler* algoritmisk tenkning til både programmering og koordinatsystemer. I Sphero fant jeg størst variasjon under dette algoritmiske konseptet. I dette verktøyet varierer undervisningsmateriellene mellom ulike matematiske fagområder, samtidig som de integrerer programmering og algoritmisk tenkning.

I Sphero – Geometric Transformations fant jeg at de anvendte geometri og i Maze Mayhem anvendte de fart, tid og distanse. I Sphero var det bare undervisningsmateriellet Morse Codes and Data Structures som ikke integrerte flere matematiske fagfelt. At Sphero inkluderer flere matematiske fagfelt og andre fagfelt som kunst gjør at de får et større bruksområde. Sphero kan anvendes i flere fag på grunn av dette, noe som er et godt argument for en skole dersom den skal investere i et slikt verktøy.

5.2 Programmeringsspråk

Kodeskolen har vist seg som et verktøy med lite variasjon i. Verktøyet bruker et blokkprogrammeringsspråk som bare er anvendelig i deres verktøy. At de anvender blokkprogrammering gjør at verktøyet har en lav inngangsterskel, som er et av poengene Papert (1980) beskriver som hensiktsmessig når barn skal lære programmering. Dolonen et al. (2019) forklarer hvordan et slikt programmeringsspråk også gjør det enklere for elevene å lære programmering uten å måtte fokusere så mye på syntaks. Dette fører til at overgangen til et programmeringsspråk som Python blir enklere.

Kodeskolen krever lite av en lærer, læreren trenger ikke å kunne å programmere fra før da elevene blir ledet gjennom spillet med veiledning innpakket i en historie om å redde biene. Elevene får hele historien gjennom verktøyet og blir introdusert til en og en funksjon. De får muligheten til å prøve seg på brettene så mange ganger de ønsker og får en form for tilbakemelding fra verktøyet. Brettene blir gradvis mer krevende da de integrerer flere funksjoner og lengre koder underveis.

Kodeskolen bidrar altså med å være en enkel start på det å lære å programmere og å arbeide med noen aspekt av algoritmisk tenkning, slik jeg har tolket Benton et. al (2016) og Brennan og Resnick (2012). Verktøyet har dog sine begrensninger. Papert (1980) beskriver at et programmeringsspråk også burde kunne anvendes slik at en har mulighet til å skape og/eller løse mer komplekse prosjekt. En kan argumentere med at Kodeskolen gradvis blir mer krevende, men Kodeskolen har en begrensing på 36 oppgaver i tillegg til at programmeringsspråket bare blir anvendt i Kodeskolen. Dette programmeringsspråket er med på å introdusere elvene til algoritmiske konsept og programmering. Dette er kompetanse som kan være nyttig for elevene å ta med seg videre, men språket i seg selv kan ikke anvendes andre steder. Papert (1980) beskriver også hvordan et verktøy burde kunne støtte flere ulike prosjekt for å engasjere flest mulig personer, med ulike interesser. Brettene blir mer krevende, men når elevene har løst alle brettene så er det ikke noe mer å gjøre i dette verktøyet, eller med dette programmeringsspråket. Altså har Kodeskolen og programmeringsspråket de anvender en lav inngangsterskel, men mangler muligheter for videre arbeid.

KUBO tilbyr mye informasjon og rettleiing til læreren. Her trenger læreren ikke å ha mye forkunnskaper om programmering. Dersom hen velger å følge undervisningsmateriellet steg for steg vil dette, slik jeg forstår det, gi en innholdsrik undervisningsøkt. For elevene er dette

et verktøy med lav inngangsterskel da de anvender blokkprogrammering, noe som gjør overgangen til skriftlig koding litt mykere (Dolonen et al., 2019). Dette er en av de tre kvalifikasjoner et programmeringsspråk burde inneholde, slik jeg forstår Papert (1980). Elevene blir gradvis introdusert til nye funksjoner og algoritmiske konsept. KUBO er det verktøyet som har fått høyest uttelling i analysen, men en kan kritisere KUBO da de mangler muligheten for å la elevene løse egne kreative problemer, basert på egne interesser, da verktøyet er lite fleksibelt. Det er også nevneverdig at KUBO Math ikke gav en like god uttelling som de andre to undervisningsmateriellene, selv om jeg tok utgangspunkt i at KUBO Coding ble tatt i bruk sammen med denne utvidelsen.

Før elevene tar fatt på å bruke roboten KUBO skal de arbeide med programmering gjennom embodiment. Dette er ifølge forskning (Sung et al., 2017; Dolonen et al., 2019; Popat og Starkey, 2018; Bocconi et al., 2018; Torkildsen og Gjøvik, 2019) en metode en kan anvende for å få kjennskap til konseptet programmering. KUBO er det eneste verktøyet hvor jeg har funnet implementering av embodiment. Grover og Pea (2013) stilte seg noe kritisk til metoden da elevene kan gå glipp av verdifulle aspekt ved algoritmisk tenkning. De mener at elevene risikerer å gå glipp av verdifull teknologiske aspekt av algoritmisk tenkning, men siden dette bare er brukt i første oppgaven anser jeg det som en introduksjon til programmering, som kanskje gjør introduksjonen til KUBOs blokkprogrammering noe mykere.

KUBOs programmeringsspråk åpner for å løse ulike problem med bruk av KUBO-roboten. Det er mulig å bruke roboten utenfor kartet til andre prosjekt, som for eksempel et kappløp. Si KUBO får instruks om å velge tilfeldig mellom -2 og 5, så kan en stille opp flere på rekke og se hvem som kommer først i mål. Dette språket og verktøyet åpner for å arbeide med flere typer prosjekter, slik som Papert (1980) beskriver. Det er dog noen begrensninger i dette verktøyet også. En kan ikke få KUBO til å lage lyder eller blinke på kommando for eksempel. KUBO blinker for å gi tilbakemelding til brukeren om det oppstår en feil i koden etc. KUBO kan heller ikke løse regnestykker for deg, men kan gi deg tilbakemelding på om du har løst en utregning riktig.

KUBO har også gitt ut tre «whitepapers» som nevnt i den horisontale analysen av KUBO. Den ene av dem heter «From KUBO's TagTile System to Scratch». Her beskriver KUBO hvordan en kan gå fra konseptene i KUBO til konseptene i Scratch, som da anvender et annet programmeringsspråk. Her viser KUBO at de er bevisst i at programmeringsspråket og

verktøyet deres har begrensinger og peker brukeren videre i en retning med høyere takhøyde og breiere vegger. Scratch er et mindre begrensende verktøy enn KUBO, da dette verktøyet anvender et blokkprogrammeringsspråk som er basert på Paperts blokkprogrammeringsspråk med navn LOGO, som blir brukt i flere verktøy.

Sphero gir læreren noe rettleiing, de har mange ulike undervisningsmateriell som krever i mer eller mindre grad forkunnskaper om programmering. Undervisningsmateriellene jeg har analysert har vært ulike i form av ulike programmeringsspråk, men også faglig innhold. Sphero har en lav inngangsterskel for elevene, da de tilbyr tre ulike programmeringsspråk. Hvorvidt tegning kan defineres som et programmeringsspråk eller ikke har jeg ikke mulighet til å konkludere med, men ifølge Forström og Kaufmann (2018) er programmering prosessen når en gir instrukser til et program som gjør at pcen, i dette tilfelle roboten, kan gjennomføre spesifikke oppgaver, løse problemer og støtte menneskelig interaksjon. Så slik jeg forstår det er denne formen for tegning en instruksjon eleven gir til roboten som får den til å følge en spesifikk instruks og da også en form for programmering. Tegning som programmeringsspråk er ikke noe jeg har sett i andre verktøy eller forskning, så takhøyden for programmeringsspråket er lav og veggene er smale, om jeg skal ta videre denne sammenligningen fra Papert (1980). Jeg anser det altså som brukervennlig, men at det har begrensinger som fører til at det ikke kan anvendes for å løse mer avanserte problemstillinger og det støtter ikke mange ulike interesser eller prosjekt.

Sphero kan brukes i mange like prosjekt, som de selv viser med utvalget undervisningsmateriell. Blokkprogrammeringen som blir anvendt i Sphero kan tas videre til andre verktøy der det kan brukes for å løse prosjekt Sphero ikke kan brukes for. Sphero har undervisningsmateriell som passer sammen med kunst, matematikk og andre fag, og det kan derfor også anvendes sammen med mange ulike elever med ulike interesser. Blokkprogrammeringsspråket Sphero anvender passer, slik jeg forstår det, godt inn i kriteriene Papert (1980) definerer. Altså har brukerne av dette verktøyet mulighet til å skape og løse mer komplekse prosjekt etter hvert som de kommer videre i læringen, både ved å gå over til tekstprogrammering i Sphero, men også utenfor verktøyet.

Sphero har også et tredje programmeringsspråk, JavaScript. Dette er et tekstbasert programmeringsspråk som brukes i flere sammenhenger da dette også kan anvendes sammen med andre plattformer. Dette er et programmeringsspråk med høyere inngangsterskel enn tegningen og blokkprogrammeringsspråket, men med Sphero kan en gå fra

blokkprogrammeringen og over til tekstprogrammering når en er klar for denne utfordringen. Dette kan være med på å senke inngangsterskelen, slik som Dolonen et al. (2019) beskriver hvordan blokkprogrammering kan være et verktøy som gjør programmering til en kompetanse en enklere kan tilnærme seg. JavaScript begrenser seg, som sagt, ikke til å bare fungere i Sphero. Altså kan dette programmeringsspråket anvendes videre, dersom elevene tar for seg prosjekt som ikke kan løses i Sphero. Det kan for eksempel anvendes for å lage en nettside, noe verktøyet Sphero ikke kan bidra med, og som nevnt i avsnittet over kan Sphero integrere flere ulike interesser i arbeidet med programmering og algoritmisk tenkning.

5.3 Sosiokulturell læringsteori

Når jeg setter søkelyset mot hvordan undervisningsmateriellene legger opp til arbeid i sosiokulturelt perspektiv finner jeg noe nevneverdig. Både KUBO og Sphero legger opp til aktivitet hvor elevene interagerer med både verktøyet og medelever. Kodeskolen derimot legger ikke opp til arbeid med medelevene, men Sentance et. al (2019) argumenterer for at den mer vitende andre kan være ulike verktøy, som for eksempel en bok. Kan Kodeskolen da, selv om interaksjonen bare skjer mellom elev og verktøy, basere seg på sosiokulturell læring?

Slik jeg forstår Sentance et. al (2019) er Kodeskolen et sosiokulturelt verktøy, selv om de ikke inkluderer arbeid med medelever eller lærer. Elevene blir en del av en historie, hvor videoer og oppgavetekster viser hvordan elevene skal arbeide for å komme i mål. De får tilbakemelding på hvor godt de har løst brettet. De fortsetter da videre til neste utfordring, og også videre i historien om sirkusdirigent Salami og roboten hans. Verktøyet kan veilede eleven og fungere som en mer vitende andre. Læreren har autoritet i klasserommet og kan velge å sette elevene sammen i grupper for å løse brettene. Dersom dette blir gjort kan også kodeskolen anvendes i en multidimensjonal sammenheng, men dette er dog ikke noe verktøyet oppfordrer til.

Det som skiller Kodeskolen fra KUBO og Sphero blir da at elevene kan arbeide sammen, i tillegg til interaksjonen med verktøyet. Ifølge Abtahi et al. (2017) kan sosiokulturell læring også være multidimensjonal. Både KUBO og Sphero har scoret godt på Benton et al. (2016) algoritmiske konsept å utveksle, med unntak av undervisningsmateriellet Sphero: Morse Codes. Dette undervisningsmateriellet kan være den mer vitende andre i læringssituasjonen, slik som hos Kodeskolen siden teksten, ut fra min tolkning av Abtahi et al. (2017), formidler informasjon slik som en mer vitende andre kan.

Å utveksle baserer seg på å dele og bygge på medelevers ideer. Når elevene arbeider med KUBO og noen av undervisningsmateriellene til Sphero, kan de utveksle tankerekker og ideer for å komme frem til en god løsning på problemet de står ovenfor. Dette vil i så fall være en situasjon hvor multidimensjonal sosiokulturell læring kan ta sted. Elevene får informasjon og tilbakemelding både fra verktøyet/undervisningsmateriellet, men også fra hverandre.

6 Konklusjon

I denne studien har jeg tatt for meg undervisningsmateriell tilhørende programmeringsverktøy som anvendes i matematikkundervisningen, i norsk skole. Jeg har benyttet et rammeverk satt sammen av Charalambous et al. (2010), Brennan og Resnick (2012) og Benton et al. (2016). Dette valgte jeg å gjøre da jeg ikke fant et passende rammeverk i mitt litteratursøk. Dette rammeverket har jeg anvendt for å belyse hvilke deler av algoritmisk tenkning de ulike undervisningsmateriellene legger opp til i et forsøk på å besvare oppgavens problemstilling.

Ifølge Bråting et al. (2020) er lærerne som underviser i programmering mattelærere med lite kompetanse innenfor programmering. Dette er nødvendig kompetanse for å kunne se potensialet og fallgruvene i emnet, og da også verktøyene som anvendes, noe som er nødvendig for å kunne skape passende aktiviteter med spesifikke læringsmål (Bråting et al., 2020). Min undersøkelse går ikke inn i hvordan lærerne anvender disse verktøyene, så jeg kan ikke konkludere med at alt som blir kommunisert gjennom undervisningsmateriellet kommer inn i klasserommet og undervisningssituasjonen. Med andre ord kan jeg ikke konkludere med hvilket læringsutbytte elevene får ved at disse undervisningsmateriellene blir anvendt, slik som også Bråting og Kilhamn (2021) konkluderer med i sin undersøkelse av lærebøker i Sverige, jeg kan bare si noe om hvilke læringsmuligheter disse undervisningsmateriellene kan tilby.

Forskningsspørsmålet mitt søker etter kjennetegn i undervisningsmateriellene jeg har tatt for meg. Funn i analysen foreslår at det er få kjennetegn mellom de ulike verktøyene. Verktøyene har ulik utforming, noe som kan ha påvirket dette funnet. Jeg har valgt et verktøy som bare eksisterer på en datamaskin, et som består av en robot som krever en datamaskin for å anvendes og et som er uavhengig av datamaskiner utenfor roboten som følger med.

Alle tre verktøyene er mulige å anvende for nybegynnere innenfor programmering og algoritmisk tenkning. Noe av det som skiller dem er hvor mange aspekt av algoritmisk tenkning de inkluderer, hvor mye informasjon de formidler og til hvem, hvor mange andre matematiske fagområder de implementerer, om programmeringsspråkene er mulige å anvende videre og/eller hvilke begrensninger og fleksibilitet programmeringsspråkene gir elevene.

Det som går igjen av *algoritmiske konsept* er *sekvens, løkke* og *hendelse.* Av algoritmiske praksiser er det i hovedsak tre som går igjen – testing og debugging, inkrementell og iterativ, og abstrahering og modularisering. Disse ble funnet i alle de tre verktøyene og formidler noe om hvilke arbeidsmetoder verktøyene implementerer. *Gjenbruk og remiks* ble også funnet i KUBO og Sphero.

Slik jeg forstår Dolonen et al. (2019) har blokkprogrammeringsspråk lavere inngangsterskel enn tekstbaserte programmeringsspråk, siden en ikke trenger å fokusere på syntaksen i like stor grad. Alle tre verktøy anvender blokkprogrammering og har derfor lav inngangsterskel. Det vil si at alle tre er verktøy som er mulig for nybegynnere å ta i bruk. Både KUBO og Kodeskolen anvender programmeringsspråk som bare kan brukes i deres verktøy, mens Sphero anvender programmeringsspråk som kan tas i bruk i andre plattformer og verktøy.

Et kjennetegn jeg kan ta ut fra disse tre verktøyene er at de baserer seg på sosiokulturell læring, slik jeg forstår Sentance et al. (2019) og Abtahi et al. (2017). Kodeskolen legger ikke opp til arbeid med medelever, men heller har undervisningsmateriellet som den mer vitende andre da Kodeskolen formidler det elevene trenger å vite for å kunne løse brettene. Sphero legger opp til at elevene skal arbeide sammen i noen tilfeller og legger med dette opp til situasjoner hvor multidimensjonal sosiokulturell læring kan ta sted. Undervisningsmateriellene til Sphero formidler informasjonen elevene trenger og opptrer dermed som den mer vitende andre. KUBOs undervisningsmateriell kommuniserer at elevene skal arbeide i par i de undervisningsmateriellene jeg har tatt for meg. Undervisningsmateriellene formidler nødvendig informasjon til elevene slik at de kan løse oppgavene og legger opp til situasjoner hvor en kan få sosiokulturell læring, både mellom elev og verktøy, men også i multidimensjonal form.

Alle tre undervisningsmateriell gir en form for tilbakemelding til elevene. Kodeskolen gir tilbakemelding i form av en vurdering mellom 1 og 3 bier. Alle tre verktøyene gir tilbakemelding i form av hvilket utfall elevene får av koden. KUBO kan også gi en

feilmelding dersom koden og/eller regnestykket ikke går opp. For å oppsummere kan en altså si at alle tre verktøyene består av undervisningsmateriell som legger opp til sosiokulturelle læringssituasjoner, men at KUBO og Sphero også legger opp til multidimensjonale læringssituasjoner.

Da jeg startet denne forskningen, valgte jeg å sette meg i skoene til en lærer uten kompetanse innenfor programmering og algoritmisk tenkning. Jeg tenkte på hvordan jeg skulle valgt et verktøy som jeg kunne brukt i klasserommet. Jeg leste forskning og fant mye informasjon om Scratch, men jeg vet jo at det finnes et hav av verktøy. Hva er det som ville passet meg og min klasse best? Det kan jeg fremdeles ikke svare på, da jeg bare har analysert et lite utvalg undervisningsmateriell utgitt av tre utvalgte verktøy, samt at jeg foreløpig ikke har en egen klasse. Lærere som skal vurdere verktøy må lene seg på deres egne pedagogiske kompetanse og kunnskap om klassen i tillegg til kunnskapen om algoritmisk tenkning. Forskning sier at lærere har mangel på kunnskap om algoritmisk tenkning og jeg håper nå at oppgaven min kan fungere som en støtte for lærere i vurderingsprosessen. Oppgaven oppsummerer noen vinklinger og krav en kan ha til verktøy i undervisningssammenheng, men den svarer ikke på alle spørsmål en kan ha om tema. Jeg har gjennom denne forskningen tilegnet meg kompetansen til å argumentere for og mot ulike verktøy til ulik bruk. Jeg håper nå at masteroppgaven min kan bli nyttig for andre lærere på samme måte.

6.1 Videre forskning

I mitt litteratursøk fant jeg ikke noen tilsvarende undersøkelser, det nærmeste jeg kom var undersøkelser av lærebøker, som for eksempel Kilhamn og Bråting (2021). Det er stadig flere som tar for seg algoritmisk tenkning og programmering, da det er nytt i læreplanen, noe som forhåpentligvis vil skape positiv utvikling i skolen og mulighet for flere elever å tilegne seg en kompetanse som det er stort behov for i en digital verden.

Kodeskolen holder seg til tema programmering og algoritmisk tenkning. KUBO integrerer koordinatsystem og de ulike regneartene. Sphero integrerer flere ulike tema som kunst og et utvalg matematiske fagfelt. Det kunne vært interessant å gå dypere inn i oppgavene for å undersøke hvor kognitivt krevende de er. Videre kunne det vært interessant å undersøke hvordan lærere med ulik kompetanse tar i bruk disse verktøyene og i hvor stor grad elevene tilegner seg kompetanse i algoritmisk tenkning og programmering når de samtidig skal, eller ikke skal, forholde seg til andre matematiske fagområder.

Avslutningsvis ønsker jeg å belyse begrensninger i denne oppgaven. Oppgaven er begrenset til bare det som er tilgjengelig på nettet, som utgiveren av verktøyet selv har publisert og som er tilgjengelig uten ekstra kostnad. Jeg har også begrenset meg til bare tre verktøy, som alle er gitt ut i vestlige land. Det kunne vært interessant å sett på andre kulturers tilnærminger til denne kompetansen.

Jeg gjorde også begrensinger i utvalget mitt som jeg i ettertid stiller meg noe kritisk til. Etter å ha analysert undervisningsmateriellene tilhørende Sphero oppdaget jeg at også de har undervisningsmateriell som er ment som en introduksjon til verktøyet. Dette falt ut da det ikke dukket opp når jeg la inn filtrene i søkemotoren til Sphero. Det jeg anser som det første undervisningsmateriellet har jeg inkludert i de to andre verktøyene, å ha det fra Sphero også hadde vært et interessant tilskudd i min analyse og det er kritikkverdig at det ikke ble inkludert.

Bibliografi

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *MIT media lab*, ss. 1-25.

Abtahi, Y. (2017, Desember 6). Pupils, tools and the zone of proximal development. *Research in Mathematics education*.

Abtahi, Y., Lerman, S., & Graven, M. (2017, November). Conceptualising the more knowledgeable other within a multi-directional ZPD. *Educational Studies in Mathematics*, ss. 275-287.

Asdal, K., & Reinertsen, H. (2020). *Hvordan gjøre dokumentanalyse: En praksisorientert metode.* Oslo: Cappelen Damm.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016, Februar). Building mathematical knowledge with programming: insights from the ScratchMaths project. *Research Gate*.

Bocconi, S., Chioccariello, A., & Earp, J. (2018, Januar). *The nordic approach to introducing computational thinking and programming in compulsory education.* NORDIC@BETT2018 STEERING GROUP.

Bråting, K., & Kilhamn, C. (2021). The Integration of Programming in Swedish School Mathematics: Investigating Elementary Mathematics Textbooks. *Scandinavian Journal of Educational Research*. doi:10.1080/00313831.2021.1897879

Bråting, K., Kilhamn, C., & Rolandsson, L. (2020). Integrating programming in Swedish school Mathematics: description of a research project. *Sustainable mathematics education in a digitalized World: Proceedings of MADIF*, ss. 101-110.

Charalambous, C. Y., Delaney, S., Mesa, V., & Hsu, H.-Y. (2010). A comparative analysis of the addition and subtraction of fractions in textbooks from three countries. *Routledge*, ss. 118-150.

Christensen, H., & Ulleberg, I. (2015). *Klasseledelse, fag og danning.* Oslo: Gyldendal.

Christoffersen, L., & Johannessen, A. (2018). *Forskningsmetode for lærerutdanningene.* Oslo: Abstrakt forlag.

Christoffersen, L., Tufte, P. A., & Johannessen, A. (2021). *Introduksjon til samfunnsvitenskapelig metode.* Oslo: Abstrakt forlag AS.

Crunchbase. (2022). *Crunchbase*. Hentet fra Crunchbase: https://www.crunchbase.com/organization/orbotix

Flø, E. E. (2021). Programmering i LK20. *Tangenten - tidsskrift for matematikkundervisning*, ss. 3-9.

Fokides, E. (2017). Students learning to program by developing games. Results of a year-long project in primary school settings. *Journal of Information Technology Education: Research, Volume 16*, ss. 475-505.

Forsström, S. E., & Kaufmann, O. T. (2018, Desember). A litterature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, ss. 18-32.

Grønmo, S. (2016). *Samfunnsvitenskapelige metoder.* Bergen: Fagbokforlaget.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A Review of the state of the field. *Educational researcher*, ss. 38-43. doi: 10.3102/0013189X12463051

Gyldendal. (2022). *Salaby*. Hentet fra Salaby: https://www.salaby.no/hva-er-salaby

Hjerm, M., & Lindgren, S. (2011). *Introduksjon til samfunnsvitenskapelig analyse.* Oslo: Gyldendal.

Jacobsen, D. I. (2015). *Hvordan gjennomføre undersøkelser? Innføring i vitenskapelig metode.* Oslo: Cappelen Damm.

Kölling, M. (2015). Lessons from the Design of Three educational programming environments: Blue, BlueJ and Greenfoot. *International Journal of people-oriented programming*, ss. 5-32.

Krogh, T. (2014). *Hermeneutikk: om å forstå og tolke.* Oslo: Gyldendal akademisk.

Krogtoft, M., & Sjøvoll, J. (2018). *Masteroppgaven i lærerutdanninga.* Oslo: Cappelen Damm.

Larsen, K. A. (2007). *En enklere metode: veiledning i samfunnsvitenskapelig forskningsmetode.* Bergen: Fagbokforlaget.

Leseth, A. B., & Tellmann, S. M. (2014). *Hvordan lese kvalitativ forskning.* Oslo: Cappelen Damm.

Mørch, A., Dolonen, J. A., Kluge, A., & Litherland, K. (2019, November 8.). Litteraturgjennomgang av programmering i skolen. *Universitetet i Oslo*.

Olsson, H., & Sörensen, S. (2001). *Forskningsproceccen. Kvalitativa och kvantitativa perspektiv.* Oslo: Gyldendal Akademisk.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books.

Papert, S. (1993). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: HarperCollins Publishers.

Popat, S., & Starkey, L. (2018). Learning to code or coding to learn? A systematic review. *Computers & Education*, ss. 365-376.

Postholm, M. B., & Jacobsen, D. I. (2018). *Forskningsmetode for masterstudenter i lærerutdanningen.* Oslo: Cappelen Damm.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009, November). Scratch: Programming for all. *Communications*, ss. 60-67. doi:10.1145/1592761.1592779

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Routledge*, ss. 136-176.

Stigberg, H., & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a swedish primary school. *Sage*, ss. 483-496.

Sung, W., Ahn, J., & Black, J. B. (2017, 7 28). Introducing computational thinking to young learners: Practicing Computational perspectives Through Embodiment in Matchematics Education. *Springer + Business Media*, ss. 443-463.

Thagaard, T. (2018). *Systematikk og innlevelse: en innføring i kvalitative metoder.* Bergen: Fagbokforlaget.

Torkildsen, H. A., & Gjøvik, Ø. (2019). Algoritmisk tenkning. *Tangenten: Tidsskrift for matematikkundervisning*, ss. 31-37.

Utdanningsdirektoratet. (2019, 03 27). *udir.* Hentet fra udir.no: https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019, 11 15). *Udir*. Hentet fra Udir: https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer?lang=nob

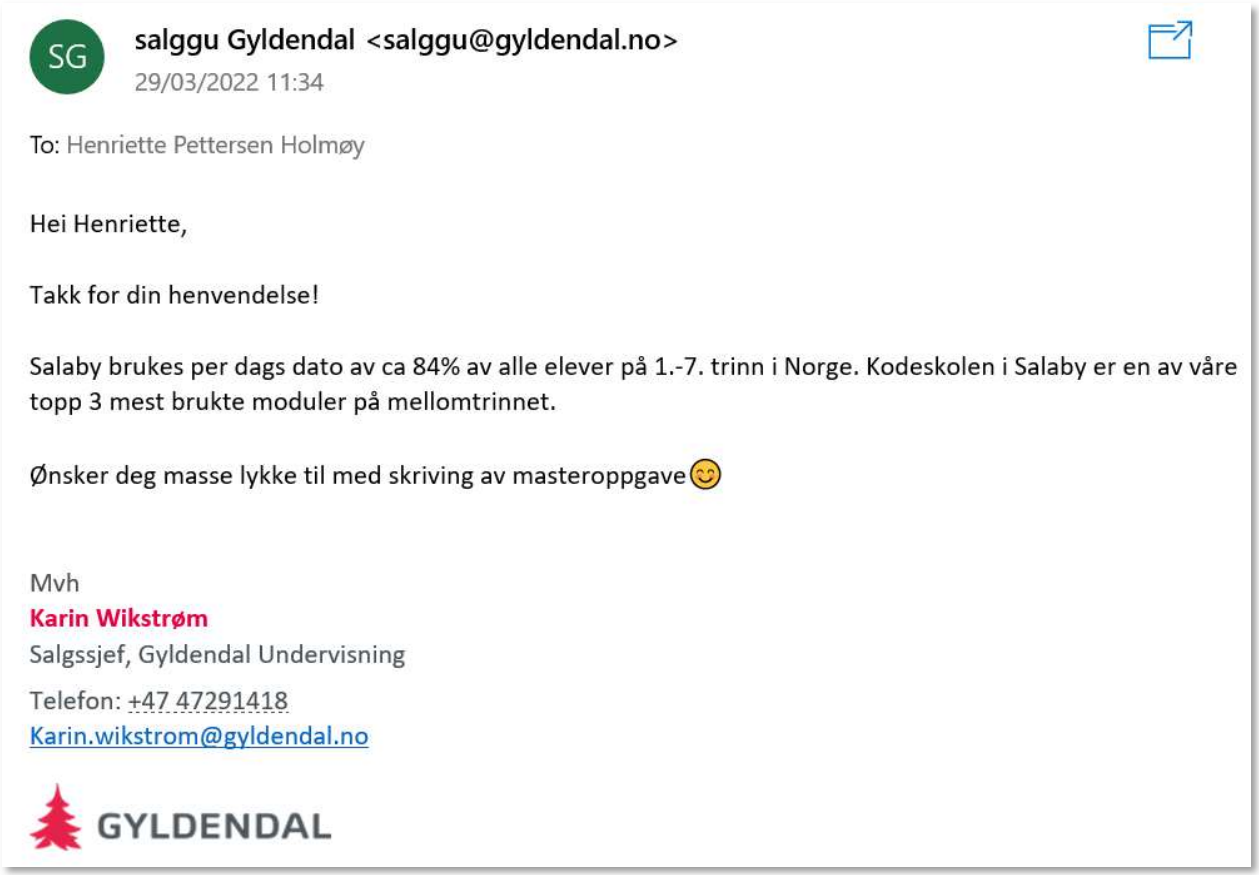
Utdanningsdirektoratet. (2020, 08 01). *Udir*. Hentet fra Udir: https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer?lang=nob&TilknyttedeKompetansemaal=true&anchorId=KE12#KE12

Utdanningsdirektoratet. (2020, 08 01). *Udir.no*. Hentet fra Udir: https://www.udir.no/lk20/mat01-05/om-faget/grunnleggende-ferdigheter?lang=nob

Wing, J. (2010, 11. 17). Computational thinking: What and Why? ss. 1-6.

Vedlegg

Vedlegg 1



SG

salggu Gyldendal <salggu@gyldendal.no>

29/03/2022 11:34

To: Henriette Pettersen Holmøy

Hei Henriette,

Takk for din henvendelse!

Salaby brukes per dags dato av ca 84% av alle elever på 1.-7. trinn i Norge. Kodeskolen i Salaby er en av våre topp 3 mest brukte moduler på mellomtrinnet.

Ønsker deg masse lykke til med skriving av masteroppgave😊

Mvh

Karin Wikstrøm

Salgssjef, Gyldendal Undervisning

Telefon: +47 47291418

Karin.wikstrom@gyldendal.no

GYLDENDAL

Vedlegg 2

PA

Preben Hansen from KUBO Robotics ApS <yourfriends@kubo-robot.com>

05/04/2022 13:26

To: Henriette Pettersen Holmøy

Hi Henriette
Thanks for reaching out. Norway has not been one of our core markets despite that Coding and Computational thinking has now been mandatory under the Math subject. We did have one big project in Bergen, where we did sell more than 500 sets of robots to them as well as some of our add on packs. So all in all I think there has been sold 1000 robots and that engage between 5000-12000 students across Noway.

BR
Preben

Vedlegg 3

From: Amanda Vaden
Sent: Wednesday, 4 May 2022 17:08
To: Henriette Pettersen Holmøy
Subject: Re: Sphero Press Teacher students masters degree - Norway

Hello Henriette,

Regarding Sphero's presence in Norway, what I can share with you is below:

Since 2017, we've sold over 10,000 robots in Norway, the vast majority (~85%) into schools there. Of that total, ~68% were Sphero BOLT.

Please let me know if you have any questions.

Thanks!
Amanda