Masteroppgave

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

Eirun Flaten Sandvin

Blokkprogrammering som verktøy i arbeid med multiplikasjon på 4. trinn

En kvalitativ studie av en gruppe elever sin bruk av blokkprogrammering som verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon

Masteroppgave i Grunnskolelærerutdanning 1-7
Veileder: Yvonne Grimeland, Benedikte Grimeland, Oda Tingstad Burheim og Torunn Klemp
Mai 2023

NTNU
Kunnskap for en bedre verden

Eirun Flaten Sandvin

Blokkprogrammering som verktøy i arbeid med multiplikasjon på 4. trinn

En kvalitativ studie av en gruppe elever sin bruk av blokkprogrammering som verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon

Masteroppgave i Grunnskolelærerutdanning 1-7
Veileder: Yvonne Grimeland, Benedikte Grimeland, Oda Tingstad Burheim og Torunn Klemp
Mai 2023

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Denne studien har som hensikt å undersøke hvordan blokkprogrammering fungerer som et verktøy, i arbeid med sammenhengen mellom multiplikasjon og divisjon, for en gruppe elever på 4. trinn. Jeg har valgt å bruke programmeringsverktøyet *micro:bit*, med det tilhørende blokkbaserte programmeringsspråket *MakeCode*. Studien er gjennomført som en kvalitativ studie med induktiv tilnærming. Funnene fra analysen blir diskutert opp mot ulike teorier som er sentrale innenfor matematikkdidaktikk. Mest sentralt er Vygotskij (1978) sitt læringssyn med fokus på medierende redskaper gjennom Rabardel (1995; 2001) sin teori om skapelsen av et instrument. Funnene blir også sett opp mot teori om multiplikasjon, tidlig algebra og Utdanningsdirektoratet sin beskrivelse av programmering som arbeidsmetode.

Studien er gjennomført for alle elevene på trinnet, men studien fokuserer i hovedsak på arbeidet til seks fokuselever. Undervisningsopplegget er todelt, og gjennomført i to undervisningsøkter i løpet av en uke i praksis. Undervisningsopplegget er utformet med tanke på å skape situasjoner der divisjon kan brukes i arbeid med multiplikasjonsstykker. Dette forsøker jeg å legge til rette for ved å gi elevene multiplikasjonsstykker med ukjent multiplikator. Elevene jobbet sammen i par, og løste noen få oppgaver på ark og resten ved hjelp av blokkprogrammering på en Chromebook. Datamaterialet består hovedsakelig av skjermopptak fra fokuselevenes Chromebooker og mine egne observasjonsnotater. Jeg og en medstudent var deltakende observatører under gjennomføringen av undervisningsopplegget med fokuselevene. Transkriberte skjermopptak har blitt analysert gjennom sekvensinndeling og åpen koding. Observasjonsnotatene inneholdt ideer til mulige funn, og har fungert som innspill til analyseprosessen.

Funn fra fokuselevenes arbeid viser at blokkprogrammering fungerte best som et verktøy for å jobbe med egne matematikkunnskaper for dem, som i dette tilfellet var multiplikasjonsstykker med ukjent produkt. Elevene løste oppgavene ved hjelp av blokkprogrammering og så det de programmerte i sammenheng med andre registre og automatiserte tallfakta. Oppgaver med ukjent multiplikator, som elevene stort sett løste ved gjentatt addisjon i arbeid med oppgaver på ark, virket vanskeligere for dem. Kun ett fokuspar tok i bruk divisjon. Det var generelt få forsøk på å prøve ut strategier for å finne den ukjente multiplikatoren. Fokuselevene utforsket og ble kjent med funksjonene i programmeringsspråket undervegs i arbeidet. Det virket som at terskelen for å flytte på blokker og endre på programmet var høyere i det fokuselevene skulle «utforske» matematisk, altså lage en kode for å finne en ukjent multiplikator. Jeg kommenterer også hvordan mitt syn på programmeringsvariabler, ved at de automatisk kan bidra til generalisering, ikke virker å være likt hos fokuselevene.

Denne studien er gjennomført som en del av forskningsprosjektet LAB-Ted. Det medfører at jeg har vært en del av samme praksisgruppe, med samme veiledere/oppfølgingslærere og hatt praksis på samme skole siden starten av tredje studieår. Vi har hatt praksis i samme klasse flere ganger, og studentene, veiledere og elever kjenner hverandre derfor bedre enn det som er vanlig i et masterprosjekt. Deltaking i LAB-Ted medfører også at forskningsfokuset mitt er valgt i samarbeid med både NTNU og praksisskole.

Abstract

The purpose of this study is to investigate how block programming works as a tool when working with the connection between multiplication and division for a group of pupils in 4th grade. I have chosen to use the programming tool micro:bit, with the associated block-based programming language MakeCode. The study has been carried out as a qualitative study with an inductive approach. The findings from the analysis are discussed against various theories that are central to mathematics didactics. Most central is Vygotskij's (1978) view of learning with a focus on mediating tools, and Rabardel's (1995; 2001) theory about the creation of an instrument. The findings are also compared to theory of multiplication, early algebra and the Norwegian Directorate of Education's description of programming as a working method.

The study has been carried out for all students in grade 4 at the school although it mainly focuses on the work of six of them. The lesson plan is divided into two parts which are carried out in two teaching sessions during one week, and contains tasks where division can be used in work with multiplication problems. I try to facilitate this by giving the students multiplication problems with an unknown multiplier. The students worked together in pairs, solving a few problems on paper and the rest using block programming on a Chromebook. The data material mainly consists of screen recordings from the six students' Chromebooks and my own observation notes. We were two participating observers during the six students work with the tasks. Transcribed screen recordings have been analyzed through sequencing and open coding. The observation notes contained ideas for possible findings, and have served as input to the analysis process.

Findings from the students' work show that block programming worked best as a tool for working with mathematics already known to them, which in this case was multiplication problems with unknown product. The students solved the tasks using block programming and saw what they programmed in connection with other registers and automated number facts. Problems with an unknown multiplier, which the students mostly solved by repeated addition when working with problems on paper, seemed more difficult for them. Only one student pair used division. There were generally few attempts to try out strategies to find the unknown multiplier. The six students explored and became familiar with the functions of the programming language during the work. It seemed that the threshold for moving blocks and changing the program was higher when the students had to explore mathematically, i.e. create a code to find an unknown multiplier. I also comment on how my view of programming variables, in that they can automatically contribute to generalization, does not seem to be shared by the students.

This study has been carried out as part of a research project, LAB-Ted. This means that I have been part of the same practice group, with the same supervisors and with practice at the same school since the start of the third year of study. We have had practice in the same class several times, and the students, supervisors and students therefore know each other better than what is usual in a master's project. Participating in LAB-Ted also means that my research focus has been chosen in collaboration with both NTNU and the school.

Forord

Fem fine år er ved vegs ende. Takk til alle som har bidratt til å gjøre studieårene gode. Takk til venner som har gjort dagene, både på skolen og etter skoletid, innholdsrike og morsomme. Takk til familie som har støttet med gode råd og korrekturlesning, og for barnepass og trilleturer det siste året. Tusen takk til samboer og barn, som fyller hver eneste dag med mye kos og glede.

Det har vært et spennende og lærerikt siste år av utdanningen. Jeg har gjennomført et masterprosjekt om å ta i bruk programmeringsverktøyet micro:bit i matematikkundervisningen. Dette er et tema jeg har vært interessert i å lære mer om, siden programmering har blitt en del av læreplanen og siden jeg har begrensede kunnskaper om programmering selv. Takk til alle som har vært en del av masterprosjektet mitt. En stor takk til mine fire veiledere, Yvonne Grimeland, Benedikte Grimeland, Oda Tingstad Burheim og Torunn Klemp, for et godt samarbeid og for alle gode innspill og svar på spørsmål. Takk til resten av gjengen i forskningsprosjektet LAB-Ted, og spesielt til mine tre medstudenter for kjekke praksisperioder og støttende fellesskap i oppgaveskriving. Takk til praksislærer og rektor ved praksisskolen, som har vist interesse for prosjektet mitt og veiledet i praksisperioder. Tusen takk til alle elevene jeg har vært så heldig å få bli kjent med gjennom prosjektet. En ekstra takk til mine seks fokuselever, for at de ville være med i fokusgruppen min og for et godt samarbeid.

Nå ser jeg fram til å begynne som nyutdannet lærer til høsten. Jeg gleder meg til å ta i bruk det jeg har lært, prøve ut programmering i matematikkundervisningen og utvikle meg som lærer. Takk for fem fine år, NTNU!

Trondheim, mai 2023
Eirun Flaten Sandvin

Innhold

Figurer

Tabeller

1 Innledning

1.1 Bakgrunn

På hver eneste barneskole i Norge finnes det et klassesett med *micro:bit.* Dette er små datamaskinbrikker som kan programmeres. En micro:bit har en liten skjerm, to knapper, diverse inngangsporter og sensorer. Ved å koble micro:bit-en til en datamaskin, har man mulighet til å programmere den. I 2014 ble den første micro:bit-en utviklet. Det skjedde i forbindelse med *BBC* sin *Make It Digital*-satsning, med formål om å gi unge erfaring med programmering (Micro:bit Educational Foundation, u.å.c). I Storbritannia har det blitt delt ut over en million micro:bit-er til skoleelever. I Norge har Sparebankstiftelsen DNB finansiert utdelingen av et klassesett med micro:bit til alle skoler (super:bit, u.å.). Programmering for elever har vært en synlig trend i mange land de siste årene (Bråting & Kilhamn, 2021). Noen land har implementert programmering som et tverrfaglig tema i læreplanen, mens andre land har innført programmering som et eget fag i skolen (Bråting & Kilhamn, 2021, s. 170). I Norge er programmering vektlagt i kompetansemål og kjerneelementer i enkelte fag, deriblant matematikk (Kunnskapsdepartementet, 2019). I tillegg til å innføre programmering i læreplanen, har Utdanningsdirektoratet satt i gang flere tiltak for å aktualisere programmering for elever. *NRK Super*, den frivillige organisasjonen *Lær Kidsa Koding* og vitensentrene har gått sammen, på oppdrag fra Utdanningsdirektoratet, om å lage en programmeringskonkurranse for micro:bit (super:bit, u.å.). Årlig blir det arrangert super:bit-kamp, der alle skoler og kodeklubber i landet har mulighet til å delta. Dette tiltaket skal hjelpe skoler i gang med programmering, ved å øke elever og læreres forståelse.

Programmering blir ofte omtalt som viktige ferdigheter for det 21. århundre. I de kommende årene vil samfunnet bli stadig mer digitalisert, som vil føre til at det blir et stort behov for yrker der programmering står sentralt (Senter for IKT i utdanningen, 2016, s. 11). Programmeringsferdigheter vil være nødvendig for å utvikle teknologi videre. I tillegg til en økende andel yrker for teknologiutvikling som krever kvalifikasjoner for programmering, vil resten av samfunnet også ha behov for kunnskaper innen programmering. Det kreves forståelse for å kunne programmere og skape ny teknologi, men det kreves også en viss grad av forståelse for programmering for å kunne ta i bruk teknologi (Senter for IKT i utdanningen, 2016, s. 11). Behovet for programmeringsferdigheter har også vært diskutert i Norge. Senter for IKT i utdanningen (2016, s. 6) viser til to NOU-er og en rapport fra en ekspertgruppe, for å synliggjøre utviklingen her til lands. I 2013 kom *Hindre for digital verdiskaping* (NOU 2013:2) som viste manglende kompetanse for programmering blant nordmenn. To år etter kom *Fremtidens skole - Fornyelse av fag og kompetanser* (NOU 2015:8) som redegjorde for hvordan kompetanseutviklingen i skolen måtte framtidsrettes. Utdanningsdirektoratet oppnevnte en ekspertgruppe, som i 2016 anbefalte et eget fag for teknologi i norsk skole (Senter for IKT i utdanningen, 2016, s. 6).

Enn så lenge er programmering og teknologi kun innført som et eget valgfag på ungdomsskoler (Senter for IKT i utdanningen, 2016, s. 8). Programmering og algoritmisk tenkning ble imidlertid innført som et tverrfaglig tema i den norske læreplanen ved

skiftet til *Læreplanverket for Kunnskapsløftet 2020* (Kunnskapsdepartementet, 2019). Algoritmisk tenkning blir beskrevet som måten vi jobber på når vi programmerer, og er et begrep som rommer mye mer enn å bare tenke som en datamaskin (Wing, 2008). Utdanningsdirektoratet (2019) beskriver algoritmisk tenkning som en problemløsningsmetode, og framhever noen arbeidsmåter og verdier de mener er viktige for algoritmisk tenkning. De vektlegger spesielt systematikk og en utforskende tilnærming til oppgaveløsning. Elevene skal være systematiske i arbeidet sitt, blant annet ved å dele opp store problemer i mindre delproblemer og å ha en steg-for-steg-tilnærming til prosessen for å løse en oppgave. I arbeid med algoritmisk tenkning skal elevene i tillegg være nysgjerrige, prøve ut ulike løsninger og ha en eksperimenterende tilnærming til oppgavene. Elevene skal kunne feilsøke kodene sine og ha strategier for å rette opp feilene. I læreplanen for matematikkfaget, blir algoritmisk tenkning beskrevet som en viktig arbeidsmåte under kjerneelementet for *utforsking og problemløysing* (Kunnskapsdepartementet, 2019, s. 2). Også på et internasjonalt plan blir programmering løftet fram som en arbeidsmetode med viktige verdier. EU omtaler programmering som en arbeidsmetode med gode muligheter for samarbeid og kommunikasjon, og kan dermed bidra til at mennesker fra hele verden kan jobbe sammen i et programmeringsspråk som går på tvers av andre språk og landegrenser (Senter for IKT i utdanningen, 2016, s. 6). På bakgrunn av blant annet dette, oppfordrer EU alle sine land å ta i bruk programmering i skolen.

Det ligger mange muligheter i micro:bit, men det trengs mer forskning om, og støtte til lærere for, hvordan micro:bit kan brukes i undervisningen. Forsström og Kaufmann gjorde i 2018 en litteraturstudie innen temaet programmering i matematikkundervisning. De konkluderer blant annet med at det trengs mer forskning om hvordan programmering kan brukes i sammenheng med matematikk, slik at ikke programmeringen kun blir et eget isolert emne, men heller noe som henger sammen med resten av matematikkundervisningen. Det finnes allerede en del forskning der programmering blir brukt i geometriundervisning, men det er svært lite forskning som ser hvordan programmering kan bli brukt i arbeid med andre matematiske tema (Forsström & Kaufmann, 2018, s. 30).

1.2 Deltakelse i forskningsprosjektet LAB-Ted

Denne studien er gjennomført som en del av forskningsprosjektet *Learning, Assessment and Boundary crossing in Teacher education (LAB-Ted)*. Prosjektet fokuserer på å ha et trepartssamarbeid for oppgaveskriving på lærerutdanningen (Norges teknisk-naturvitenskapelige universitet, u.å.). Fokus for studentenes masterprosjekter er valgt i samarbeid med veiledere fra universitetet og praksislærer og rektor i praksisskolen. Slik er målet at masteroppgavene skal være relevante for lærerutdanningen på universitetet, i skolene, og for studentene selv.

Som deltaker i LAB-Ted har jeg vært del av samme praksisgruppe de siste tre årene av studiet mitt. Hver praksisperiode har vi vært på samme skole, med de samme oppfølgingslærerne. Både skolen, representert ved rektor og praksislærer, og oppfølgingslærerne fra NTNU er med i forskningsprosjektet. I perioden for praksis og innsamling av datamateriale, blir vi blitt veiledet av praksislærer som er oppdatert på oppgavene våre. I selve skrivearbeidet, har vi gruppeveiledning med alle fire matematikkstudenter og alle fire veiledere. Alle fire masteroppgavene har felles overordnet tema innen matematikkdidaktikk.

1.3 Min studie

I masterprosjektet mitt vil jeg undersøke hvordan programmering fungerer som arbeidsmetode i sammenheng med andre matematiske tema, slik Forsström og Kaufmann (2018) foreslår. Jeg har forsket på et 4. trinn, og så derfor til kompetansemålene for 4. trinn i læreplanen for matematikk (Kunnskapsdepartementet, 2019) for å finne hvilken matematikk som var mest relevant for prosjektet mitt. Flere av kompetansemålene for dette trinnet handler om divisjon, som for eksempel «utforske, bruke og beskrive ulike divisjonsstrategiar». Et annet kompetansemål tar for seg sammenhengen mellom regneoperasjoner, «utforske og forklare samanhengar mellom dei fire rekneartane og bruke samanhengane formålstenleg i utrekningar». Sammenhengen mellom multiplikasjon og divisjon er derfor noe elever på 4.trinn skal jobbe med. I en tidligere praksisperiode på trinnet fikk jeg innblikk i at denne sammenhengen var ukjent for elevene. De hadde jobbet en del med multiplikasjon og litt med divisjon, men svært få så hvordan den ene operasjonen kunne brukes i arbeid med den andre. Jeg tenkte derfor at det ville være et relevant fokus i studien min.

Læreplanen inneholder også spesifikke kompetansemål for algoritmisk tenkning. Et av kompetansemålene for matematikk på 4.trinn er «lage algoritmar og uttrykkje dei ved bruk av variablar, vilkår og lykkjer». Elever på 4.trinn skal blant annet jobbe med programmeringsvariabler. I undervisningsopplegget jeg har utformet for denne studien, har jeg tatt i bruk programmeringsvariabler i sammenheng med blokkprogrammering. Blokkbasert programmering er utformet for at yngre elever skal få tilgang til programmering, ved at blokkene fungerer som et stillas som muliggjør programmering uten å kunne syntaks (Sentance et al., 2019, s. 144). I studien min har jeg brukt micro:bit med det blokkbaserte programmeringsspråket *MakeCode* i matematikkundervisningen for 4. trinn. Det matematiske fokuset har vært på å se at multiplikasjon og divisjon er inverse operasjoner, og at dette kan være et nyttig forhold å kjenne til i arbeid med operasjonene. Forskningsfokuset mitt er: *Hvordan fungerer blokkprogrammering som et verktøy for å arbeide med sammenhengen mellom multiplikasjon og divisjon for en gruppe elever på 4. trinn?*

Jeg har utformet et undervisningsopplegg som skal jobbes med i MakeCode, med multiplikasjonsoppgaver som legger opp til at divisjon kan brukes. Funnene mine blir hovedsakelig sett i lys av Rabardels teori om *instrumentell skapelse* (*instrumental genesis*) (oversettelse hentet fra Moe, 2019), men også teori om tidlig algebra og multiplikasjon. Dette, samt en beskrivelse av micro:bit og MakeCode, blir presentert i oppgavens kapittel to. I kapittel tre gjør jeg rede for min bruk av skjermopptak for datainnsamling og den induktive analyseprosessen ved åpen koding. Analysen gav fem kategorier for funn av elevenes bruk av blokkprogrammering som verktøy. Disse er vist og forklart i kapittel fire. Hvorvidt blokkprogrammering fungerte som et verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon blir i kapittel fem diskutert opp mot nevnt teori. Oppgaven avrundes i kapittel seks, med konklusjon og implikasjoner for videre forskning og undervisning med programmering som verktøy.

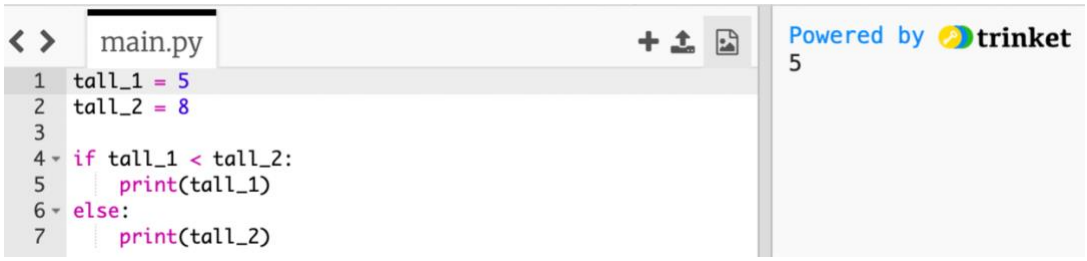
2 Teori

2.1 Programmering

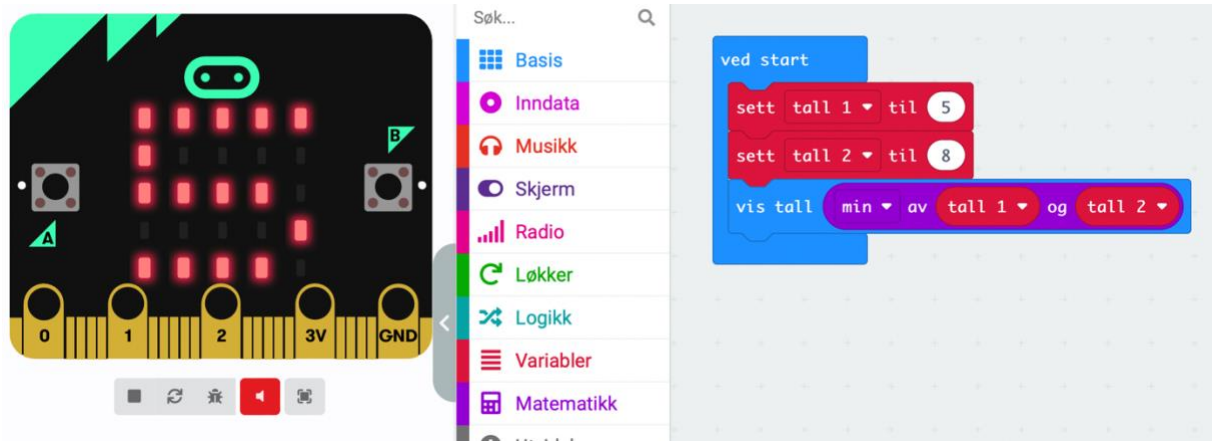
Programmering er å lage instruksjoner til en datamaskin som ber den om å utføre en oppgave (Senter for IKT i utdanningen, 2016, s. 9). I programmering møter vi begrepet *kode*. Dette er vidt, men jeg bruker det som et begrep for én enkelt instruksjon. Når flere instruksjoner blir satt sammen, får vi et *program* (Haraldsrud et al., 2020, s. 17). En elev som programmerer har et mål for hva datamaskinen skal utføre. For å få datamaskinen til å gjøre ønsket handling, skriver eleven koder som sammen utgjør et program. Når datamaskinen kjører programmet, leser den kode for kode for å forstå hva den skal gjøre.

Dersom eleven har formulert koder som gir mening for datamaskinen, vil den gjennomføre ønsket handling. Det finnes ulike *språk* man kan bruke for å programmere. De blir delt i hovedkategoriene *tekstbaserte programmeringsspråk* og *blokkbaserte programmeringsspråk* (Franklin et al., 2016). I tekstbasert programmering finnes språk som *Python* og *JavaScript*. Disse baserer seg på at elever må skrive inn riktig formulerte instruksjoner, tegn for tegn. Her er *syntaks* svært viktig. Elevene må skrive kodene helt likt slik de skal være, eller så forstår ikke datamaskinen hva de ber den om å gjøre (Haraldsrud et al., 2020, s. 18). Dersom en elev glemmer et kolon eller staver et ord feil, har ikke datamaskinen mulighet til å forstå hva eleven prøver å instruere den til. Tekstbaserte programmeringsspråk inneholder kun bokstaver, tall og tegnsettingssymbol (Franklin et al., 2016). I figur 2.1.1 er et eksempel på et program for å avgjøre hvilket av to tall som er minst, i det tekstbaserte språket Python.

Figur 2.1.1: Program i det tekstbaserte språket Python



Figur 2.1.2: Program i det blokkbaserte språket MakeCode



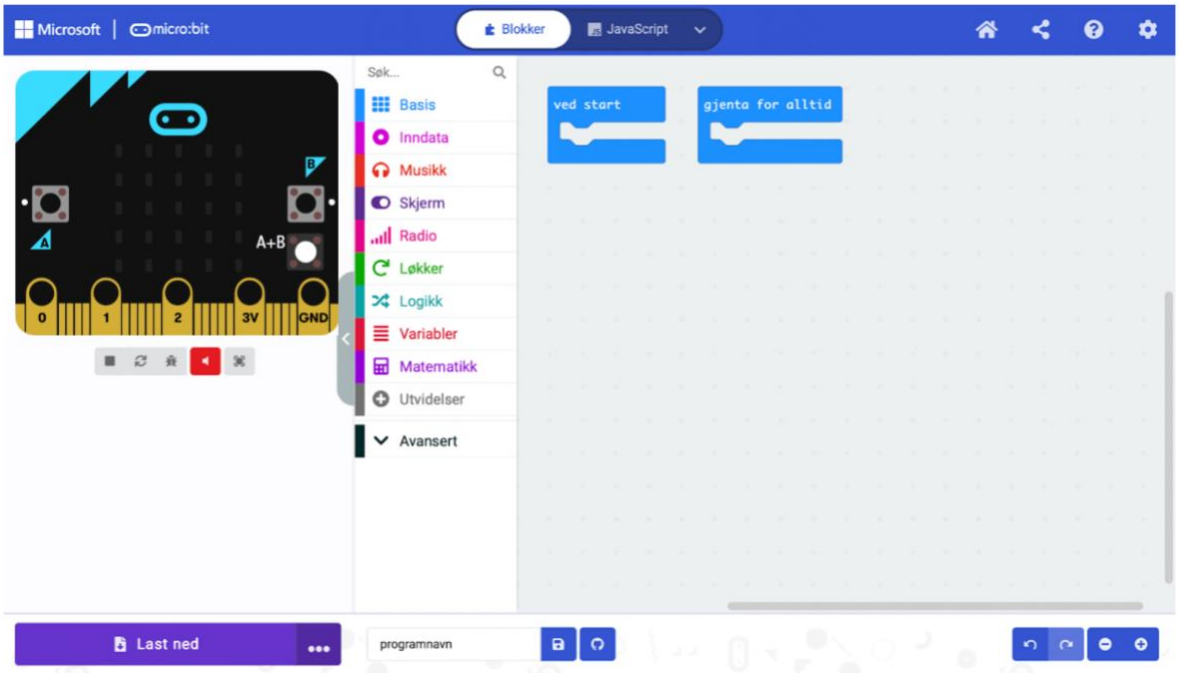
Eksempler på blokkbasert programmeringsspråk er *MakeCode* og *Scratch*. Dette er språk som har visuelle blokker med innbakte koder. Dette gir mulighet for at yngre elever som ikke har forutsetninger for å huske og formulere syntaks riktig, kan være med å programmere (Franklin et al., 2016, s. 217). Elevene trenger ikke å skrive inn koder tegn for tegn, men kan heller lete etter ønskede blokker i en meny. I tillegg til at blokkene støtter elevene gjennom å inneholde ferdigformulerte koder, tilbyr de støtte for elevene gjennom form og farge. Blokkene er delt i ulike farger etter hvilken kategori de hører til, og har ulik form som gjør at de kun kan settes sammen på måter som gir mening for datamaskinen. Blokkene med ferdige koder kan minne om lego-brikker, som skal kobles sammen til programmer. Dette kan sees som en form for stillasbygging, som støtter elever i arbeid med oppgaver som ikke hadde vært tilgjengelige for dem ved tekstbasert programmering (Sentance et al., 2019, s. 144). I figur 2.1.2 er tilsvarende program som i figur 2.1, med i det blokkbaserte språket MakeCode. Blokkprogrammering har blokker med innbakte koder, og er slik mer visuelt og intuitivt enn det tekstbaserte programmet i figur 2.1.1.

2.1.1 Beskrivelse av micro:bit og MakeCode

Figur 2.1.3: Framside (til venstre) og bakside av en fysisk micro:bit

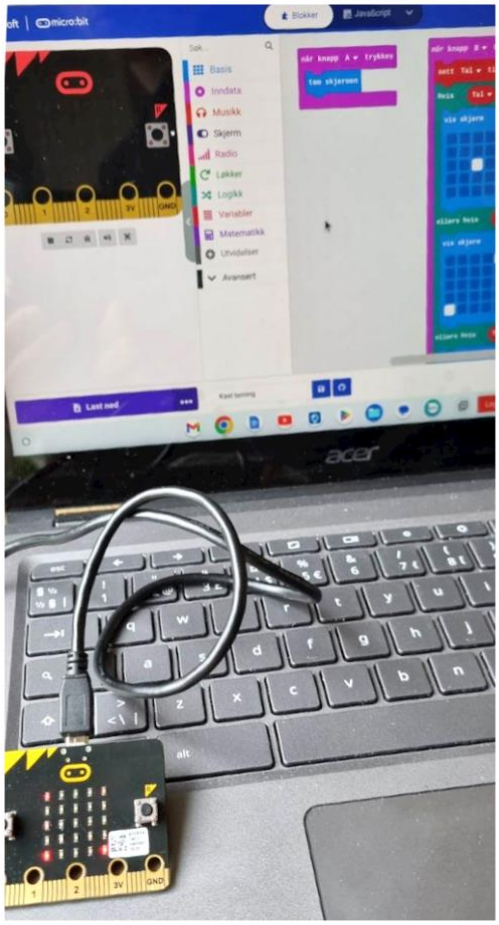
En *micro:bit* er en liten brikke som inneholder en datamaskin. Denne kan programmeres, og kan få koblet til sensorer og annet ekstrautstyr (Micro:bit Educational Foundation, u.å.a). I figur 2.1.3 ser vi en fysisk micro:bit. Den har to knapper (merket A og B), innebygde sensorer, høyttaler og bluetooth-funksjon (merket med navn på baksiden). Mellom knappene på framsiden er det en skjerm som består av 5x5 piksler. Hver piksel har en liten lysdiode som kan slåes på og av. Det betyr at det til sammen er 25 små dioder som kan tennes. Dette gir muligheter for å vise bokstaver, tall eller ikoner på skjermen. I figur 2.1 viser micro:bit-skjermen et smilefjes-ikon. Diodene i skjermen setter også begrensninger for hvor mye som kan vises om gangen. Skjermen har kun mulighet til å vise én bokstav, ett tall eller ett ikon. Dersom micro:bit-en blir programmert til å vise et flersifret tall eller et helt ord eller setning, "ruller" symbolene over skjermen. For eksempel vil tallet 152 først vises ved sifferet 1, som beveger seg mot venstre. Når det nesten er ute av skjermen, blir sifferet 5 gradvis synlig fra høyre. Likt blir 2 gradvis synlig etter som 5 er på veg ut av skjermen. Elever må følge godt med når flersifrede tall blir vist, siden det kun vises ett siffer om gangen, og hvert siffer kun er synlig i omtrent ett sekund mens det beveger seg over skjermen.

Figur 2.1.4: Den offisielle editoren til micro:bit



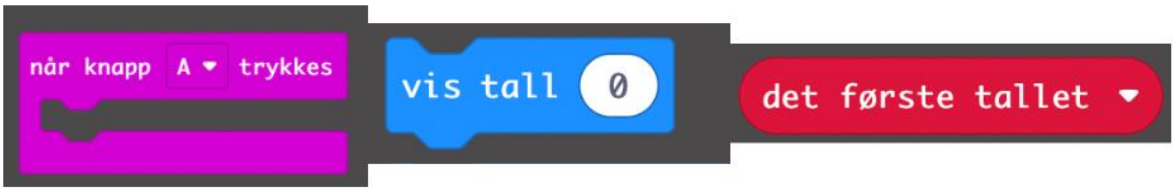
Både blokkbasert og tekstbasert programmeringsspråk kan brukes for å programmere en micro:bit. For å programmere den, trengs en *editor*, et verktøy for lage programmer til micro:bit-en. Den offisielle editoren for micro:bit finnes på makecode.microbit.org (Make:code Educational Foundation, u.å.b). Denne har blokkbaserte MakeCode som hovedprogrammeringsspråk. Det er mulig å bytte språk til de tekstbaserte Python og JavaScript. MakeCode inneholder blokker som har farge etter hvilken kategori de hører til. Blokkene har også ulike former, for å vise hvilke som kan kobles sammen. Blokkene inneholder ferdige koder man kan sette sammen, som lego-klosser, til programmer. Editoren inneholder (fra venstre på figur 2.1.4) en animasjon av en micro:bit, en meny med kategorier for de ulike blokkene og et stort felt for selve programmeringen. Animasjonen av micro:bit-en er koblet sammen med programmeringsfeltet, og fungerer som en simulator. Her kan elever teste programmet sitt, før de laster det over på en fysisk micro:bit. Muligheten for å simulere, gjør også at man ikke nødvendigvis trenger å ha en fysisk micro:bit. Det er mulig å lage og kjøre programmer i editoren. Animasjonen har noen forskjeller fra den fysiske micro:bit-en, som er tilpasninger for datamaskinformatet, som blant annet en egen knapp for å trykke begge knappene samtidig, kalt knapp A+B (se animasjonen på figur 2.1.4), og funksjoner for å justere verdier for sensorer. Editoren i seg selv kan være et redskap for programmering, men kan også kobles sammen med en fysisk micro:bit. Ved å koble micro:bit-en inn i datamaskinen med en kabel, kan elever overføre programmene de lager i editoren til en fysisk micro:bit. På denne måten kan elever programmere micro:bit. I figur 2.1.5 blir et program overført fra datamaskin til micro:bit, som gjør at micro:bit-en fungerer som terning.

Figur 2.1.5: Overføring av program til micro:bit



Videre vil jeg forklare hvordan et utvalg av blokkene i MakeCode fungerer. Som tidligere nevnt har blokkene ulik form og farge. Jeg vil forklare tre ulike geometriske former for blokker og fire ulike farger.

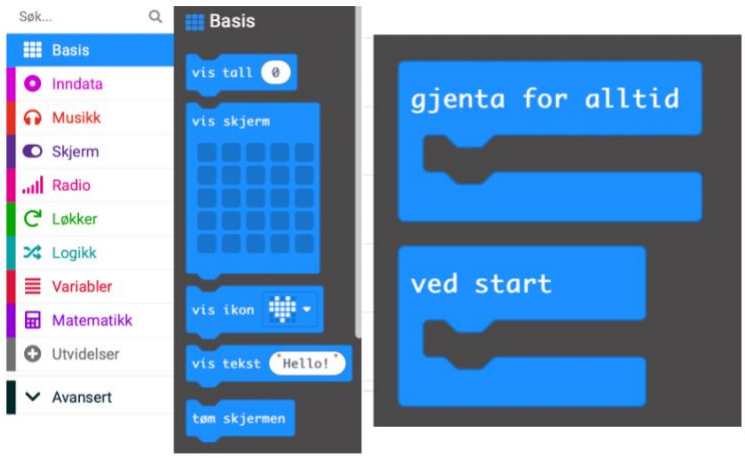
Figur 2.1.6: Startblokk, handlingsblokk og verdiblokk



Jeg velger å kalle de tre ulike formene for startblokk, handlingsblokk og verdiblokk. I figur 2.1.6 ser vi en rosa *startblokk*. Disse har mulighet til å få andre blokker plasserte inn i seg. De rette kantene over og under blokka viser til at de ikke kan kobles inn i andre. Slike blokker kalles startblokker, og må settes i starten av et program. På innsiden av blokka er det kanter med spor i, som en legokloss eller nøkkel, som viser til at andre blokker med samme spor kan settes inn der. Det finnes startblokker som blir aktiverte ved start og startblokker som blir aktiverte når noe skjer, som at knapper blir

trykte. I figur 2.1.6 ser vi også et eksempel på en blå *handlingsblokk*. Disse sier hva som skal skje, som for eksempel at micro:bit-en skal vise et tall. Det finnes også handlingsblokker for å vise tekst eller ikoner, eller å endre eller sette verdier til variabler. Handlingsblokker skal kobles inn i en startblokk, og det er mulig å koble en rekke handlingsblokker sammen. Alle handlingsblokkene har spor i både kanten over og under, og kan slik settes sammen inni startblokker eller til andre handlingsblokker. En del av handlingsblokkene har et hvitt ovalt felt i seg, slik som handlingsblokka i figur 2.1.6. I disse feltene skal elevene føre inn en verdi, som kan være både tall og bokstaver. Dette er mulig ved å enten trykke inn i feltet og skrive verdien eleven vil at handlingsblokka skal ha, eller så kan eleven dra en *verdiblokk* inn i feltet. Verdiblokker har oval form, slik som feltet i handlingsblokka. Verdiblokkene kan være variabler som har blitt tilordnet en verdi, eller være koblet til verdier fra sensorer. For eksempel finnes det verdiblokker for lysnivå eller lydnivå, som micro:bit-en henter informasjon om gjennom sensorer.

Figur 2.1.7: Et utvalg basisblokker

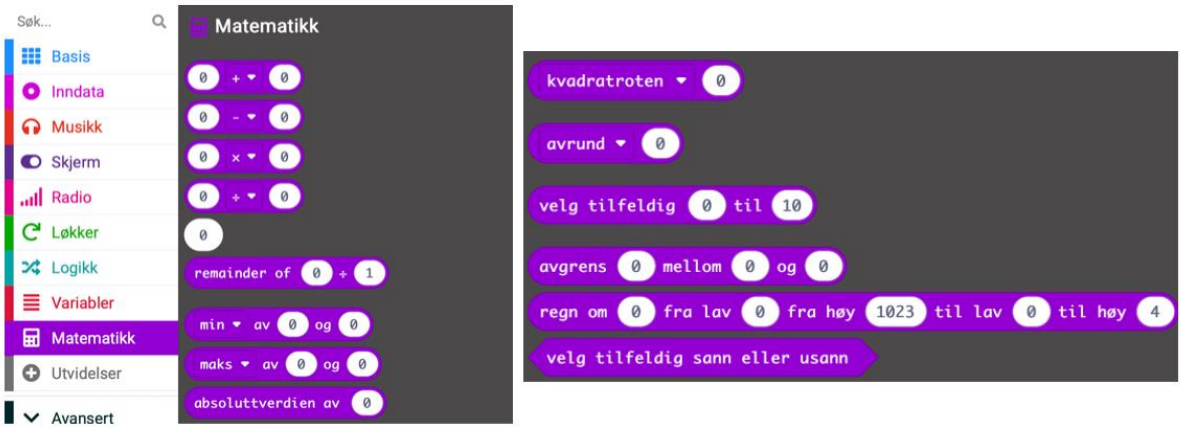


Blokkene har også farge etter hvilken kategori de hører til. Blå blokker er *basisblokker*. Det er stort sett handlingsblokker som får noe til å vise på skjermen til micro:bit-en. Til venstre på figur 2.1.7 ser vi handlingsblokker for å vise tall, tekst eller ikon. Til høyre er startblokker som kjøres automatisk ved start eller kontinuerlig gjennom hele programmet.

Figur 2.1.8: Et utvalg inndatablokker

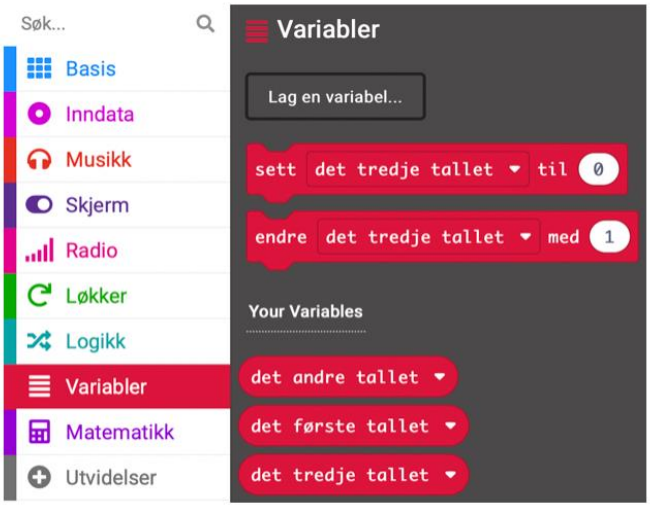
Inndatablokkene er rosa og handler om at micro:bit-en får informasjon fra knapper og sensorer. Startblokkene i inndatakategorien blir brukt for å lage programmer som skal bli aktiverte ved for eksempel knappetrykk. Det finnes tre startblokker for knappetrykk, "når knapp A trykkes", "når knapp B trykkes" og "når knapp A+B trykkes". Den siste viser til når elevene trykker inn knapp A og B samtidig, og denne handlingen kan altså aktivere et eget program. I denne kategorien finner vi også verdiblokker for sensorer, vist til høyre i figur 2.1.8.

Figur 2.1.9: Matematikkblokkene



De lilla blokkene er *matematikkblokker*. Her er det kun verdiblokker. Det finnes blokker for hver av de fire regneartene, blokker som settes til et tilfeldig tall, blokker for avrunding og for å ta kvadratrot blant annet. Verdiblokkene for matematikk har egne felt som elever kan skrive inn verdier eller sette inn andre verdiblokker i. For eksempel kan en elev skrive inn tallene to og tre i multiplikasjonsblokka, for å få verdien av produktet av tallene. Det er også mulig å sette inn verdiblokker fra sensorer eller variabler, eller å sette flere verdiblokker fra matematikk inni hverandre. Dersom en elev vil multiplisere tre tall sammen, kan hen sette en multiplikasjonsblokk inn i en annen, ved å dra blokka inn i et av de hvite feltene til den andre blokka.

Figur 2.1.10: Variabelblokkene

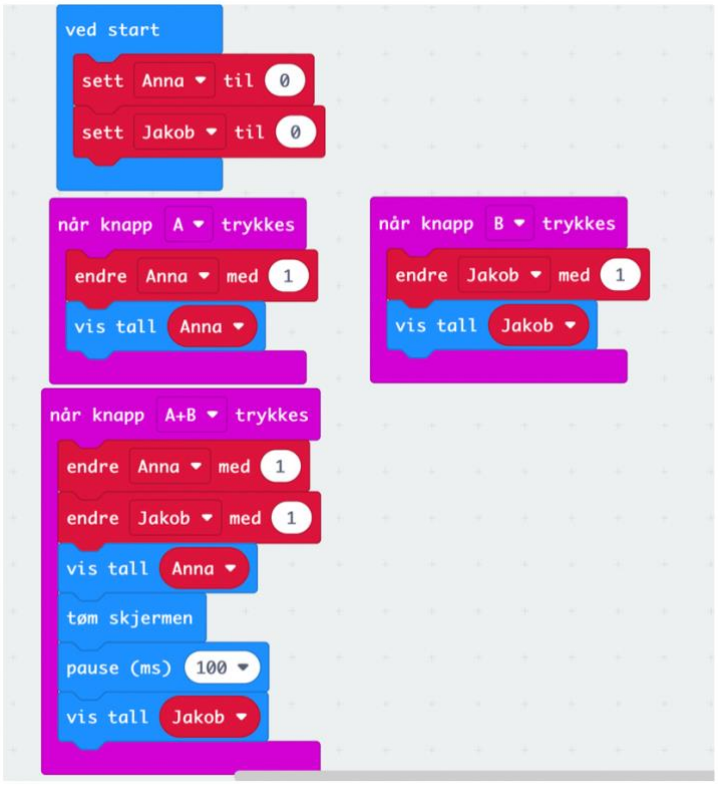


Variabelblokkene er røde, og finnes som handlingsblokker og verdiblokker. Handlingsblokkene kan *sette* eller *endre* variabelverdier. I menyen vises handlingsblokkene kun for variabelen som sist ble laget. Ved å trykke på den hvite pilen ved siden av variabelnavnet, kan elevene velge hvilken som helst variabel. Verdiblokkene for variabler kan settes inn i handlingsblokker for å bruke variabelverdiene. For eksempel kan elever sette verdiblokker for variabler inn i matematikkblokker for å regne med dem. Dette er ikke *algebraiske variabler*, men *programmeringsvariabler* som skiller seg litt fra de algebraiske. For å tydeliggjøre dette, vil jeg fra nå av referere til variablene brukt i programmet som *programmeringsvariabler*.

Programmeringsvariabler fungerer som et minne (Usiskin, 1988, s. 16). Alle involverte programmeringsvariabler i et program må defineres (Kilhamn et al, 2022). Det vil si at de må få tildelt en verdi for å eksistere i programmet. Det gjøres ved hjelp av en sett-blokk i MakeCode. I løpet av gjennomkjøringen av et program kan programmeringsvariabelen endre verdi. Når programmet startes og datamaskinen leser koder med handlingsblokker for programmeringsvariabler, sett- eller endreblokker, vil verdien til programmeringsvariabelen endres.

Navnet til programmeringsvariabler kan inneholde ord og mellomrom, og kan slik være beskrivende for hva de representerer (Kilhamn et al., 2022). I tidlig algebra er et naturlig språk viktig for å utvikle elevenes algebraiske tenkning (Kieran et al., 2016, s. 1). Ved å kunne gi programmeringsvariabler navn som refererer til elevenes naturlige språk, kan programmering gi elevene mulighet til å utvikle sin algebraiske tenkning.

Figur 2.1.11: Program med programmeringsvariabler



I figur 2.1.11 ser vi et eksempel på et program med programmingsvariabler. Programmet er laget for å holde tellingen på poeng i et spill med to spillere, *Anna* og *Jakob*. Her er det to programmeringsvariabler, som har fått navn etter spillerne. Når knapp A trykkes, blir *Anna* sine poeng endret med ett mer, og den totale summen hennes, verdien til *Anna*, vises på micro:bit-skjermen. Når knapp B trykkes, får *Jakob* ett poeng mer og den samlede verdien hans vises på skjermen. Når knappene trykkes inn på likt (knapp A+B), får begge ett poeng hver. Skjermen på micro:bit-en viser først *Anna* sine poeng og deretter *Jakob* sine.

2.2 Tidlig algebra

Tidlig algebra handler om at de yngre elevene også kan ta del i *algebraisk tenkning*, ved å jobbe med oppgaver der algebra kan bli brukt, men som også er meningsfulle å delta i uten algebra (Kieran, 2004, s. 149). Det handler om å fokusere på egenskaper ved tall og operasjoner, for å tilrettelegge for algebraisk tenkning (Carpenter et al., 2003, s. 3-4). Når elever tenker algebraisk, resonnerer de med uspesifiserte mengder, de ser likhetstegnet som et uttrykk for forhold, de generaliserer og de jobber med å oppdage og uttrykke matematiske strukturer (Blanton, 2010). Ved å introdusere elever for tidlig algebra får de tidligere mulighet til å se sammenhenger og til å generalisere matematikk, som kan bidra i deres matematiske utvikling og forståelse (Kaput & Blanton, 2001, s. 344).

Variabler blir omtalt som en del av kjernen i algebra (Kilhamn et al., 2022). Begrepet er ikke entydig, siden variabler kan ha ulike roller (Schou et al., 2014, s. 196). For eksempel vil alle bokstavene i $A = l \cdot b$, formelen for areal av et rektangel, bli oppfattet som variabler. Vi kan tenke oss at vi kjenner verdiene til *l og b*, og at de er henholdsvis tre og fire. Lengden og bredden, l og b, er da *uavhengige variabler*. Vi tilordner

variablene de kjente verdiene for lengden og bredden til rektangelet. Variablene kan stå for hvilke som helst positive reelle tall (Schou et al., 2014, s. 196), men blir i denne situasjonen satt til verdiene tre og fire. Siden $A = l \cdot b$, vil *A* bestemmes ut ifra verdiene til de uavhengige variablene *l og b*. Slik blir *A* en *avhengig variabel*, en variabel som også kan ta verdien til alle positive reelle tall, men som blir bestemt ut ifra verdiene til andre variabler (Schou et al., 2014, s. 197). Variabler kan på denne måten også være *kjente* eller *ukjente*. De kjente variablene kan vi tilordne en verdi ved start, som for *l og b* i eksempelet over. Verdien holder seg konstant gjennom hele utregningen, og bokstavene *l og b* er plassholdere for kjente verdier (Schou et al., 2014, s. 197). I eksempelet er variabelen *A* ukjent. Bokstaven for variabelen *A* blir en plassholder for en ukjent verdi helt fram til slutten av utregningen. Her vil A sin verdi bli kjent gjennom å multiplisere de kjente variablene 3 og 4, som blir 12. Variabelen *A* har da verdi lik 12. Variabler kan være avhengige og ukjente som *A*, eller uavhengige og kjente som *l og b.*

Det finnes også situasjoner der vi kan møte uavhengige variabler som også er ukjente. Variabler kan fungere som uspesifiserte mengder i utregninger (Kilhamn et al., 2022). Ved å behandle regnestykket over $A = l \cdot b$ uten å tilordne noen av variablene verdi, kan elever jobbe med den generelle sammenhengen at man alltid kan finne arealet av et rektangel ved å multiplisere lengde og bredde. I denne situasjonen representerer variablene alle positive reelle tall og viser til at dette er noe som gjelder «alle» tall og ikke bare for et spesifikt regnestykke.

Algebraiske variabler skiller seg fra programmeringsvariabler ved at algebraiske variabler har samme verdi gjennom hele utregningen (Kilhamn et al., 2022). Over har jeg vist til forskjellige variabler, men felles for alle er at de ikke endrer verdi i løpet av utregningen. Enten er variabelen kjent gjennom hele utregningen, eller ukjent fram til slutten av utregningen eller udefinert og står for "alle" tall for å representere generelle sammenhenger. I eksempelet om *Anna* og *Jakob* i figur 2.1.11, så vi hvordan programmeringsvariabler kan bli satt til en verdi eller få endret verdi flere ganger undervegs i gjennomkjøringen av et program. Programmeringsvariabler skiller seg også fra algebraiske variabler ved navngiving. Der variabler i algebra har lite beskrivende navn, ofte med kun én enkelt bokstav, kan programmeringsvariabler ha beskrivende navn med ord og mellomrom (Kilhamn et al., 2022).

2.2.1 Multiplikasjon

Å jobbe med tidlig algebra innebærer å fokusere på egenskaper ved operasjoner, i stedet for å bare se på selve utregningsoperasjonen (Carpenter et al., 2003, s. 3-4). Egenskapene skaper muligheter for å trekke inn algebraisk tenkning. Multiplikasjon som operasjon har et potensial for å brukes i arbeid med tidlig algebra, ved for eksempel å fokusere på egenskapene for assosiativitet og distributivitet (Enge & Valenta, 2011, s. 31). Den assosiative egenskapen handler om at dersom tre tall skal multipliseres, for eksempel $3 \cdot 4 \cdot 5$, er det det samme hvilke to tall som blir multiplisert sammen først (Van de Walle et al, 2020, s. 205). Den distributive egenskapen handler om at det er mulig å dele opp et tall, så lenge alle tallene blir med videre i utregningen. For eksempel kan man i arbeid med $5 \cdot 6$ dele opp sekstallet og slik få $5 \cdot 2 \cdot 3$, og fortsatt få samme svar (Van de Walle et al., 2020, s. 206). En annen nyttig egenskap for multiplikasjon er det inverse forholdet operasjonen har for divisjonsoperasjonen (Carpenter et al, 2003, s. 4). Dette kan gi elevene innsikt i en viktig matematisk sammenheng. De nevnte egenskapene gjør at multiplikasjonsoppgaver kan åpne for tidlig algebra ved at elever

kan oppdage, uttrykke og generalisere sammenhenger, men det er samtidig mulig å engasjere seg i oppgavene uten å ta i bruk algebraisk tenkning.

"For at en situasjon skal være etablert som multiplikativ, er det nødvendig å koordinere minst to sammensatte enheter, på en slik måte at en av de sammensatte enhetene er fordelt over elementene til den andre sammensatte enheten" (min oversettelse av Steffe, 1994, s. 19, sitert i Rønning & Burheim, 2020, s. 126). Videre kan multiplikasjon sees som en *del-hel-struktur* (Van de Walle et al., 2020, s. 197; Fosnot & Dolk, 2001, s. 36). En enhet viser til *antall deler*, og en annen enhet er *størrelse per del*, altså antall elementer i hver del. Det totale antallet elementer i delene utgjør *det hele*. Både antall deler, størrelsen på delene eller det hele kan være ukjent. Van de Walle et al. (2020, s. 197) ser på multiplikasjonsstykker som $antall\ grupper \cdot gruppestørrelse = produkt$. I studien min møter elevene kun multiplikasjonsstykker i en programmeringskontekst, og å bruke gruppebegrepet blir slik for løsrevet fra konteksten. Jeg velger derfor å heller bruke begrepene $multiplikand \cdot multiplikator = produkt$. Dette er kjente navn på de ulike enhetene av et multiplikasjonsstykke (Aubert et al., 2022), som ikke viser til grupper eller deler, og dermed passer en programmeringskontekst bedre. Jeg vil likevel referere til del-hel i noen eksempler der multiplikasjon som operasjon blir forklart.

Hvilken del av multiplikasjonsstykket som er ukjent, har mye å si for hvordan regnestykket kan løses. Multiplikasjonsstykker med ukjent produkt skaper et behov for multiplikasjon, mens der multiplikator eller multiplikand er ukjent, gir et behov for divisjon som operasjon (Van de Walle et al., 2020, s. 197). For eksempel kan vi tenke at dersom vi skal finne det hele, og har fem deler som hver inneholder seks elementer, må vi ta størrelsen, seks, like mange ganger som antall deler, fem. Det blir da behov for multiplikasjon. Dersom vi har et ukjent antall deler eller at størrelsen per del er ukjent, blir det et behov for divisjon. Dersom vi vet at det hele er 30 og at det er delt i fem deler, legges det opp til divisjon som operasjon, ved at 30 elementer skal fordeles på fem deler, for å finne ut hvor stor hver del er. Ved ukjent antall deler, skal det kjente hele fordeles på deler med kjent størrelse. Dette legger også opp til divisjon. De tre ulike oppgavestrukturene, er vist i tabell 2.1.

Tabell 2.1: Oppgavestrukturer for multiplikasjonsstykker

Regnestykke	Oppgavestruktur	Operasjon
$5 \cdot 6 = \_$	Ukjent produkt	Multiplikasjon
$5 \cdot \_ = 30$	Ukjent multiplikator	Divisjon
$\_ \cdot 6 = 30$	Ukjent multiplikand	Divisjon

Det finnes ulike modeller for multiplikasjon, som *like grupper*, *areal*, *multiplikativ sammenligning*, *multiplikativ endring*, *delingsdivisjon* og *målingsdivisjon* (Rønning & Burheim, 2020, s. 125). Det er viktig at elevene får kjennskap til ulike modeller for multiplikasjon. Slik får de mulighet til å kjenne igjen flere situasjoner der disse operasjonene kan være nyttige å bruke (Rønning & Burheim, 2020, s. 131). Elevene i studien min bruker stort sett *gjentatt addisjon*, en strategi innenfor like grupper-modellen. Dette blir kategorisert som en primitiv modell av Fischbein et al. (1985, s. 6).

De primitive modellene er ofte ineffektive og kan ha et begrenset bruksområde, som igjen åpner for en begrenset forståelse for multiplikasjon (Rønning og Burheim, 2020, s. 126, s. 131).

Måling- og delingsdivisjon er to andre modeller som kan brukes for multiplikasjon. Divisjon og multiplikasjon er inverse operasjoner av hverandre, og slik tett knyttet sammen (Rønning & Burheim, 2020, s. 130). Å være bevisst det inverse forholdet mellom multiplikasjon og divisjon bidrar også til en bedre forståelse av regneartene hver for seg (Carpenter et al., 2003, s. 4; Fosnot & Dolk, 2001, s. 70). Forholdet kan hjelpe elever både i arbeid med multiplikasjon og i arbeid med divisjon. Både multiplikasjon og divisjon baserer seg på *multiplikativ tenkning* (Schou et al., 2014, s. 31). Å tenke multiplikativt handler om å forstå at grupper med objekter også kan sees som enheter i seg selv (Van de Walle et al., 2020, s. 200).

2.2.2 Multiplikative tallfakta

Multiplikasjon, eller addisjon, av to tall som begge er mindre enn ti, kalles *tallfakta* (Van de Walle et al., 2020, s. 216). Det finnes også tallfakta for divisjon og subtraksjon. Det er de tilsvarende stykkene til tallfaktaene for multiplikasjon og divisjon (Van de Walle et al., 2020, s. 216). Altså kan vi si at $3 \cdot 5 = 15$ er tallfakta, siden det er multiplikasjon av to tall under ti. Likt blir $15 \div 5 = 3$ sett som tallfakta for divisjon, siden det er tilsvarende et multiplikasjonsstykke for to tall under ti.

Når elever automatiserer kunnskap om tallfakta, "vet" de svaret på regnestykkene uten å måtte regne (Van de Walle et al., 2020, s. 217). Det er tre faser i utviklingen av automatisert tallfaktakunnskap (Baroody, 2006, s. 22).

1. Teller: Elevene teller seg fram til svaret. For eksempel vil elever i arbeid med $5 \cdot 3$ telle to femmere opp fra fem, enten ved hjelp av fingre, klosser eller verbalt.
2. Resonnerer: Elevene bruker kjent kunnskap for å finne svaret. Her kan elever i arbeid med $5 \cdot 3$ bruke at de vet at $5 \cdot 2 = 10$, og så legge til den siste femmeren ved å telle seg fem opp fra ti.
3. Behersker tallfakta: Elevene "vet" svaret. I arbeid med $5 \cdot 3$, svarer elevene kjapt at de vet at svaret er 15, uten at de trenger å regne for å komme fram til det.

2.3 Bruk av redskaper i læringsprosessen

Å bruke redskaper for å utvikle forståelse for et matematisk objekt er sentralt i læringsteorien til Lev Vygotskij (Sentance et al., 2019). Han ser læring som en prosess som går fra et sosialt plan til et individuelt plan. Elever starter å resonnere i fellesskap, de tilegner seg måter å tenke eller handle på som finnes i kulturen, og gjør dem til sine egne (Dahl et al., 2020, s. 163). For å binde sammen det sosiale og det individuelle planet, trengs *mediering*. Vygotskij fokuserer på at læring skjer gjennom å bruke hjelpemidler, *medierende redskaper* (Sentance, 2019). Disse skal bidra til å dele kunnskap sosialt og til å utvikle kunnskap individuelt. Læringsteoretikeren ser språket som det viktigste redskapet for mediering (Vygotskij, 1978, i Dahl et al., 2020, s. 163). Andre medierende redskaper som bidrar til å dele og utvikle kunnskap kan være unifix-kuber, en tallinje eller blokkprogrammering.

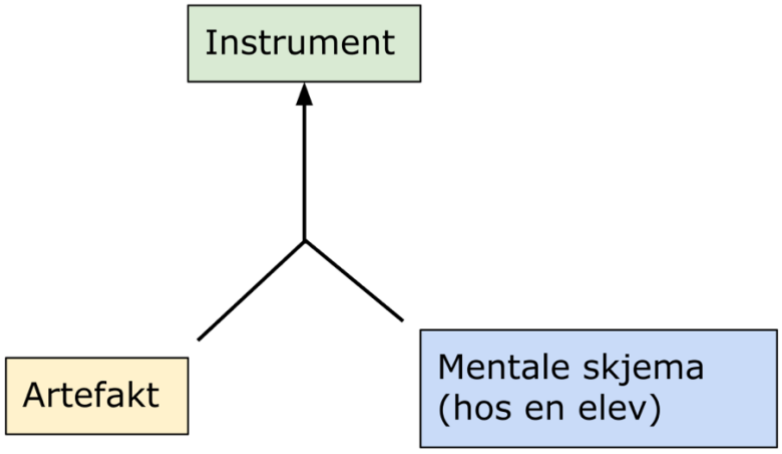
Teoriene til Vygotskij danner grunnlaget for kjente teorier som *sosiokulturelt læringssyn* og *aktivitetsteori*. Vygotskij sin teori om at aktivitet er mediert gjennom redskaper, er

også en del av det teoretiske grunnlaget brukt i teorien om *instrumentell skapelse* (*instrumental genesis)* (Rabardel & Samurcay, 2001; Verillon & Rabardel, 1995, s. 81).

2.3.1 Instrumentell skapelse

Innenfor forskningsfeltet for matematikkundervisning fokuserer teorien om instrumentell skapelse på utviklingen fra en artefakt til et instrument (Rabardel & Samurcay, 2001). En *artefakt* er en betegnelse for alle objekter i kulturen som mennesker har tilgang til, som kan bidra til utvikling og som dermed har et potensial for å fungere som et verktøy (Verillon & Rabardel, 1995, s. 81). Et *instrument* blir sett som en individuell mental konstruksjon av en artefakt, og er et begrep for når en elev ser artefakten som et nyttig verktøy. Artefakten blir til et instrument gjennom en prosess, kalt instrumentell skapelse, der en elev utvikler en artefakt til å bli et meningsfylt verktøy for seg selv (Rabardel & Samurcay, 2001). Teorien om instrumentell skapelse handler om å lære med redskaper, og ser spesifikt på hvordan et redskap blir et nyttig verktøy for en elev.

Figur 2.3.1: Todelt instrument



Et instrument består av en artefaktdel og en skjemadel, som vist i figur 2.3.1 (Rabardel & Samurcay, 2001). Skjemadelen viser til en oppfatning av at hver elev har *mentale skjema* som endres under instrumentell skapelse-prosessen. En elevs mentale skjema har en sosial dimensjon og en individuell dimensjon (Rabardel & Samurcay, 2001). Den sosiale dimensjonen av det mentale skjemaet viser til at skjemaet er avhengig av både eleven og utvikler av artefakten (Rabardel & Samurcay, 2001; Verillon & Rabardel, 1995, s. 87). For eksempel vil både eleven som bruker MakeCode og designeren som har utviklet programmeringsspråket, bidra til det mentale skjemaet eleven har i sin instrumentelle skapelse-prosess. Utvikleren bak MakeCode har lagt føringer for hvordan den kan brukes, og legger slik rammer for elevens utforsking med artefakten. I MakeCode er for eksempel kommandoer innebygd i blokker, noe som gjør at elevene ikke kan prøve ut egne ideer til koder. Eleven endrer skjemaet sitt gjennom å gjøre seg erfaringer med MakeCode og blir kjent med hvordan det best kan brukes. Den individuelle dimensjonen av det mentale skjemaet viser til at skjemaet er personlig og unikt for hver elev (Verillon & Rabardel, 1995, s. 87).

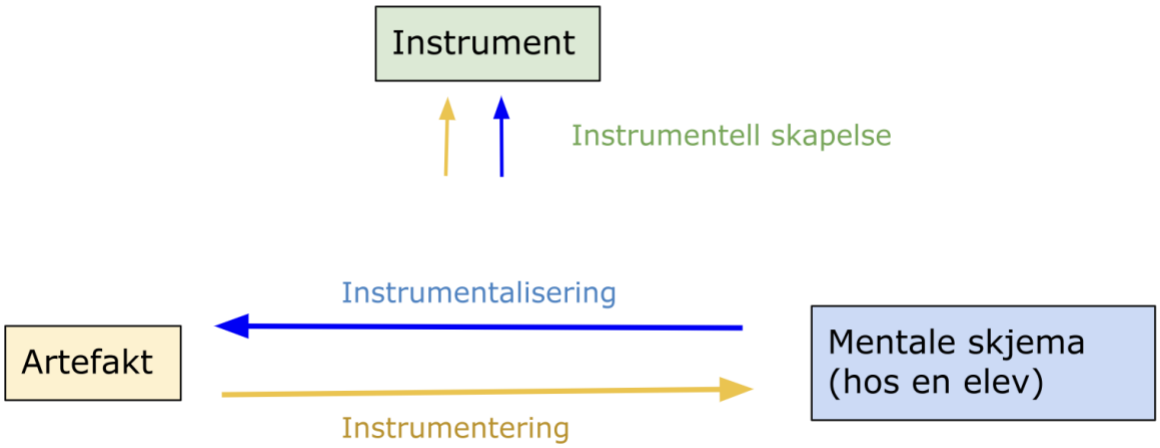
De mentale skjemaene kan deles i *bruksskjema (usage schemes)* og *instrumenterte handlingsskjema (instrumented action schemes)* (Trouche, 2003, s.789). Det førstnevnte

er et generelt skjema for artefaktens bruk, samt hvilke muligheter og begrensninger en elev ser i artefakten. Dette endres etter som eleven gjør seg erfaringer med artefakten. Et instrumentert handlingsskjema har et mer spesifikt mål, og handler om hvordan en spesiell type oppgave kan løses (Trouche, 2003, s. 789). Her kan elever koble sammen matematikkunnskapene sine og brukskjemaene de har for hvordan artefakten kan brukes. I undervisningsopplegget mitt skulle elevene lage et program i MakeCode for å løse multiplikasjonsstykker med ukjent multiplikator. Her spiller både elevenes kjennskap til den involverte matematikken, og kjennskap til programmeringsverkøyet micro:bit, inn. Elevene må bruke matematikkunnskapene sine inn i programmeringen, men kan begrenses av hvilke muligheter de kjenner til av blokker og sammensetninger. Her legger både kunnskaper om den involverte matematikken og artefakten rammene for hvilke muligheter elevene har for å kunne løse oppgaven, og på hvilken måte de velger å løse den. Det instrumenterte handlingsskjemaet blir utvidet ettersom eleven blir kjent med hvordan denne typen matematikk kan kobles sammen med MakeCode.

Det kan være vanskelig å observere endringer i mentale skjemaer. Ved å bruke Vergnaud (1996, sitert i Trouche, 2003, s. 789) sin definisjon av et mentalt skjema, kan vi likevel påstå at vi kan si noe om andres mentale skjema. Vergnaud omtaler mentale skjema som organisering av atferd for spesifikke situasjoner. Handlinger kan sees som en observerbar form for atferd, og vi kan slik få et innblikk i en persons mentale skjema. For eksempel kan en lærer studere en elevs prøving og feiling i arbeid med blokkprogrammering for å få innblikk i hvilke funksjoner eleven kjenner til, eller hva eleven forventer at skal være mulig eller ikke. På denne måten får læreren et inntrykk av elevens mentale skjema. Endring kan også observeres ved at læreren ser at elevene prøver å løse samme type oppgave senere eller bruker artefakten på en annen måte. Dersom en elev som tidligere brukte udefinerte programmeringsvariabler i programmene sine begynner å alltid gi alle variabler verdier, kan det indikere endring i mentale skjema. Kanskje har eleven oppdaget at variabler i blokkprogrammering må tilordnes en verdi for å kunne fungere i et program?

Prosessen for å utvikle et instrument, instrumentell skapelse-prosessen, er delt i instrumentalisering og instrumentering (Rabardel & Samurcay, 2001). Instrumentet er som nevnt todelt, i artefaktdel og skjemadel, og er like avhengig av å utvikle både artefakten og de mentale skjemaene til eleven for å bli et instrument med verdi for eleven.

Figur 2.3.2: Prosessen for å skape et instrument



Instrumentalisering handler om å utvikle artefaktsdelen av instrumentet. Her former eleven artefakten. Eleven kan utforske, gjøre avgrensninger og valg for å gjøre seg kjent med artefakten og for å gjøre artefakten mer personlig (Rabardel & Bourmaud, 2003, sitert i Trouche, 2003, s. 789). Elever som jobber med å programmere i MakeCode, gjør seg hovedsakelig kjent med artefakten gjennom å prøve ut hvordan blokker kan settes sammen og brukes, eller hvordan editoren fungerer. På denne måten lager elevene seg en oppfatning av hvilke muligheter og begrensninger blokkene gir dem. Kan elevene finne en måte å sette opp multiplikasjonsstykker med flere faktorer enn to? Lager elevene programmeringsvariabler, og hvilke navn gir de dem? Slike handlinger kan bidra til instrumentalisering og prosessen for å skape et instrument.

Instrumentering fokuserer på skjemadelen ved instrumentet. Elevenes arbeidsmåter blir formet av artefakten. I instrumenteringsprosessen tar elevene i bruk skjemaene sine for bruk av artefakten for å løse oppgaver (Rabardel & Bourmaud, 2003, sitert i Trouche, 2003, s. 789). Elevene skal tilegne seg den sosiale dimensjonen av skjemaet, og utvikle den individuelle dimensjonen av det mentale skjemaet (Rabardel & Samurcay, 2001). Når elevene utforsker blokkprogrammering i MakeCode, slik som i eksempelet over, vil de mentale skjemaene bli endret. Elevene utvider bruksskjemaet for hvordan artefakten fungerer, og kan legge til ny informasjon til et instrumentert handlingsskjema, dersom eleven utforsker hvordan programmet kan brukes til å løse en spesifikk oppgave. Den sosiale dimensjonen av skjemaet tilegnes etter som eleven blir kjent med hvilke muligheter og begrensninger utformingen av MakeCode for multiplikasjon gir. Instrumentering er synlig når eleven velger arbeidsmåter som er tilpasset artefakten eller oppnår ny kunnskap ved å bruke artefakten.

Instrumentell genesis er, gjennom instrumentaliserings- og instrumenteringsprosessene, en sammensmelting av mentale skjema og teknikker for å bruke artefakten (Drijvers et al., 2010, s. 214). Mulighetene og begrensningene som er innbakt i artefakten blir koblet opp mot elevens kunnskap og arbeidsmåter (Trouche, 2003, s. 798-799). Gjennom denne prosessen utvikler eleven artefakten til å bli et instrument, og blir sett på som et verktøy.

2.4 Tidligere forskning om programmering i matematikkundervisning

Programmering i skolen er relativt nytt, og det er derfor begrenset med forskning innen emnet (Forsström & Kaufmann, 2018). Sanna Erika Forsström og Odd Tore Kaufmann gjorde i 2018 en litteraturstudie for programmering i matematikkundervisning, med fokus på undervisningspotensialet programmering har innen matematikkfaget. De undersøkte 15 artikler og fant fire gjennomgående tema, som de tar med inn i diskusjonen av potensialet til programmering i matematikkundervisningen (Forsström & Kaufmann, 2018, s. 22). De fire temaene er: Elevenes motivasjon for matematikk, elevenes prestasjoner, samarbeidet mellom elevene og læreren som har fått endret rolle i møte med programmering. Studiene Forsström og Kaufmann har undersøkt inneholder elever i arbeid med robotprogrammering. Deltakerne i studiene er fra fem til 19 år og bruker forskjellige programmeringsverktøy. Ingen av studiene omhandler micro:bit.

I de 15 artiklene finner Forsström og Kaufmann (2018, s. 26) ingen generaliserbar sammenheng mellom programmering og motivasjon eller prestasjon i studiene. Flere av studiene konkluderer med at elevene ble mer motiverte eller prestere bedre, men Forsström og Kaufmann trekker fram ulike grunner til at studiene ikke er generaliserbare (Forsström & Kaufmann, 2018, s. 28). De ser videre at den nye lærerrollen som følge av programmering i undervisningen henger sammen med samarbeidet mellom elevene. I programmeringsaktiviteter er samarbeid mye brukt, og læreren skal veilede elevene i programmeringen deres og samtidig se til at samarbeidet fungerer (Forsström & Kaufmann, 2018, s. 27). Den nye lærerrollen fokuserer i større grad på veiledning enn at læreren skal stå foran elevene og undervise.

Forsström og Kaufmann (2018, s. 30) konkluderer med at det trengs mer forskning på hvordan matematikk og programmering skal knyttes sammen. Det blir trukket fram at programmering og algoritmisk tenkning bidrar til å tenke logisk og til å utvikle problemløsningsstrategier. Dette er viktige egenskaper i matematikk, og det trengs derfor forskning som kobler programmering til ulike deler av matematikkfaget. Det finnes forskning som kobler programmering til geometri, men lite om andre matematikktema (Forsström og Kaufmann, 2018, s. 30). Programmering er mer enn et verktøy i seg selv, siden det også kan sees som en måte å jobbe på innen andre tema (Forsström og Kaufmann, 2018, s. 28). Litteraturstudien viste lite eksisterende forskning om undervisningspotensialet i programmering som metode.

3 Metode

Jeg har gjennomført en studie for å undersøke hvordan blokkprogrammering fungerte som et verktøy for å arbeide med sammenhengen mellom multiplikasjon og divisjon. Studien er gjennomført for et helt 4. trinn, men jeg fokuserer på hvordan seks fokuselever brukte blokkprogrammering i arbeid med oppgavene. Undervisningsopplegget er todelt, og ble gjennomført over to økter. I dette kapittelet skal jeg gjøre rede for valgene jeg har tatt undervegs i prosjektet, hvilken type studie jeg har planlagt og hvordan den ble gjennomført, og samlet sett synliggjøre hele forskningsprosessen min.

3.1 Kontekst

Datamaterialet er samlet inn i løpet av min siste praksisuke på lærerutdanningen. Praksisskolen ligger i en storby, og har i underkant av 100 elever på hvert trinn. Elevene jeg har forsket på går i 4. klasse. Undervisningsopplegget er gjennomført for alle klassene på trinnet, men av ulike lærere. Halvparten av trinnet fikk gjennomført opplegget av en praksisstudent, og andre halvdel av en matematikklærer som jobber fast på trinnet. Seks av elevene var valgt ut som fokuselever og gjennomførte opplegget delvis adskilt fra resten av klassen, med meg og en medstudent til stede. Vi har vært i praksis på dette trinnet på både høst- og vårsemesteret dette studieåret, i tillegg til å ha vært i praksis på samme trinn et tidligere studieår også. Slik kjenner vi elevene, og de oss, litt bedre enn det som er vanlig i praksis.

Medstudenten min forsker på elevers engasjement i arbeid med programmering, og hennes studie satt rammene for at undervisningsopplegget skulle vare over to økter. Utenom dette er undervisningsopplegget utformet for mitt forskningsfokus. Undervisningsopplegget er altså todelt og gjennomført i to økter i løpet av én praksisuke. På mandag i praksisuka gjennomførte jeg første del av opplegget, og andre økta ble gjennomført på torsdag. Begge øktene var i utgangspunktet 90 minutter, men ble oppstykket av pauser med dansing etter som elevene fikk behov for å bevege seg. Undervisningsopplegget ble gjennomført for hele 4. trinn. Fokuset var likevel på kun seks av elevene, som var fokuselever for prosjektet. De gjennomførte på eget rom i første økt av opplegget. I andre økt var fokuselevene samlet på et gruppebord bakerst i klasserommet sammen med resten av klassen. Jeg og medstudenten min var til stede med fokuselevene i begge øktene for å svare på og stille spørsmål til elevenes arbeid.

3.1.1 Utvalg

Undervisningsopplegget i masterprosjektet mitt ble gjennomført for hele trinnet, men jeg fokuserer på hvordan seks av elevene jobbet med oppgavene. Fokuselevene er valgt ut av medstudenten min, i samråd med praksislærer. Hun forsker på samme opplegg som meg, ved å se på elevenes engasjement, men gjennomførte noen økter med de samme elevene før jeg startet datainnsamlingen til mitt prosjekt. Elevene var slik valgt ut av henne og praksislærer før datainnsamlingen i mitt prosjekt startet. Fokuselevene er valgt på bakgrunn av faktorer som at samtykkeskjema fra foreldre var levert i tide, at elevene selv hadde lyst til å delta og at elevene kunne sies å være på et relativt gjennomsnittlig faglig nivå for fjerdeklassinger. I øktene for datainnsamling til prosjektet mitt jobbet

fokuselevene med oppgavene adskilt, eller delvis adskilt, fra resten av klassen sin, slik at det var mulig å ta opp lyd av samtalene deres og for at vi kunne filme dem uten at andre elever kom med. Fokuselevene hadde alltid minst én voksen hos seg, og fikk slik noe tettere oppfølging enn resten av elevene på trinnet. Rammene ellers var like. Alle elever jobbet sammen i par, noen var tre sammen, og hvert par hadde tilgang på én Chromebook og ett sett med oppgaveark.

3.1.2 LAB-Ted

Masteroppgaven er en del av forskningsprosjektet *Learning, Assessment and Boundary crossing in Teacher education* (LAB-Ted). Prosjektet er et samarbeid mellom NTNU, Universitetet i Tromsø og King's College i London. Vi har vært to studentkull i prosjektet, som ved NTNU består av fire matematikkstudenter og fire kroppsøvingsstudenter per kull. Prosjektet har to målsettinger. Det første er å utvikle et samarbeid mellom praksisfelt, universitet og student for at forskning som skjer på lærerutdanningen skal være både praksis- og profesjonsorientert. Det andre er å oppdage hindringer og spenninger som kan oppstå i et slikt samarbeid (Norges teknisk-naturvitenskapelige universitet, u.å.).

Vi er fire matematikkstudenter på kull to i LAB-Ted, som siden tredje studieår har utgjort en praksisgruppe. Hver praksisperiode har vi vært på samme skole, som er med i prosjektet. Vi har fire oppfølgingslærere, som også er veiledere for masteroppgavene våre og var veiledere da vi skrev FoU-oppgaver. Tre av dem er veiledere fra matematikkseksjonen ved lærerutdanningen og én fra pedagogikk. Vi skriver individuelle masteroppgaver, men har felles overordnet tematikk for dem, og stort sett felles gruppeveiledning. Før sommerferien 2022 ble studentene, veiledere fra NTNU og praksislærer og rektor ved praksisskolen enige om at oppgavene våre skulle ha implementeringen av programmering i matematikkfaget som overordnet tema. Vi ble enige om temaet på grunn av at det er relativt nytt i skolen, og noe som alle parter ønsket å lære mer om. Videre bestemte vi oss for å bruke programmeringsverktøyet micro:bit og det tilhørende blokkbaserte programmeringsspråket MakeCode. Hovedargumentet vårt er at alle norske skoler har et klassesett med fysiske micro:bit-er, i tillegg til at nettsiden med editor og MakeCode-språk er gratis og fremstår enkel og intuitiv for oss.

3.1.3 Forberedelser

Siden vi allerede hadde bestemt overordnet tema for masteroppgavene våre da høstsemesteret i femte studieår begynte, benyttet vi anledningen til å gjennomføre noen introduksjonsøkter til micro:bit for fjerdeklassingene. Elevene hadde på dette tidspunktet ikke brukt verktøyet før, og var så godt som nybegynnere i programmering. Vi startet med to aktiviteter der vi øvde på å gi presise beskjeder i riktig rekkefølge, som er viktig for programmering. Disse aktivitetene foregikk uten datamaskiner, og kan slik kalles *unplugged programmering*. Dette blir av Busuttil og Formosa (2020, s. 570) referert til som en praktisk arbeidsmåte for å forstå prinsipper for programmering, uten å bruke datamaskiner. Første økta var uteskole, og elevene programmerte en medelev med bind for øynene gjennom en hinderløype i skogen. Neste økt jobbet elevene også i par, men skulle nå programmere hverandre til å tegne en figur. Den ene eleven fikk se en figur, som den andre skulle tegne etter instruksjoner fra medeleven.

Deretter gikk vi over til å programmere i editoren til micro:bit med MakeCode som språk. For økta som var elevenes første møte med MakeCode, hadde vi laget fem oppgaver som var utformet med tanke på at elevene skulle bli kjent med programmet. Her valgte flere elever bort oppgavene vi hadde gitt og utforsket programmet på egen hånd. Elevene viste god evne til å utforske, og prøvde og feilet med et stort utvalg blokker. Det kom fram mange interessante program, som eksempelvis elever som fikk micro:bit-en til å spille musikk, mens den løste enkle regnestykker og viste tekst eller ikoner på skjermen. I en annen økt ble elevene gitt bilder av programmer og koder i MakeCode som de skulle forklare skriftlig. Noen dager senere, som en stasjon under stasjonsarbeid, fikk elevene laste et enkelt program som simulerte en terning over på en fysisk micro:bit. Elevene viste stort engasjement for både micro:bit og unplugged programmering. I denne praksisperioden jobbet elevene med multiplikasjon og divisjon også. Elevene møtte på flere situasjoner som kunne blitt løst enkelt dersom de hadde visst at divisjon er den inverse operatoren til multiplikasjon, men som de ikke viste kjennskap til. En del av elevene benyttet seg av gjentatt addisjon, og brukte strategien i møte med både multiplikasjon og divisjon.

Etter praksisperioden på høsten, og frem til vi kom tilbake i praksis i januar, har elevene hatt noen økter i Scratch. Det er også et programmeringsspråk som er blokkbasert. Blokkene i Scratch ligner en del på blokkene elevene møter i MakeCode, men språkene har noen forskjeller. Elevene hadde ikke brukt variabler i arbeid med programmering i Scratch. To av studentene på praksisgruppa var derfor på besøk i en av klassene på trinnet uka før praksis, for å introdusere programmeringsvariabler for elevene. Programmeringsvariabler er en viktig del av både mitt og flere av mine medstudenters masterprosjekter.

3.2 Forskningsdesign

I studien min vil jeg finne ut hvordan blokkprogrammering kan fungere som et verktøy for arbeid med sammenhengen mellom multiplikasjon og divisjon for 4. trinn. Jeg har laget et matematikkopplegg som tar utgangspunkt i at elevene skal arbeide i MakeCode. Matematikkoppgavene er utformet med tanke på at de skal legge opp til at elevene kan bruke divisjon i arbeidet med multiplikasjon.

Studien jeg har gjennomført kan klassifiseres som en *intervensjonsstudie*. Dette er et vidt begrep, også kjent under andre navn, som for eksempel *designstudie* (Prediger et al., 2015). Intervensjonsstudie rommer forskjellige typer forskning, og har ikke like mange faste rammer som andre forskningsdesign. Studiene, som kan kalles intervensjonsstudier, har likevel noen fellestrekk. Det er spesielt sentralt å generere nye teorier for intervensjonsstudier (Prediger et al., 2015, s. 877). Denne typen forskning innen matematikkdidaktikk inneholder tre antakelser for læring, som kan sees i sammenheng med blant annet Vygotskij sitt læringssyn (Prediger et al., 2015, s. 881-882):

1. Det er viktig å få med elevers perspektiver i forskning. Elever har egne erfaringer og relevante synspunkter. De er ikke "uferdige voksne", men har en verdi her og nå.
2. Læringsprosesser utvikles over tid.
3. Tanke og handling hører sammen. Handlinger blir styrt av tanker, og handlinger kan utvikle tanker.

For å forstå hvordan blokkprogrammering er et verktøy for elever er det relevant å se hva de gjør og stille dem spørsmål. Undervegs i gjennomføringen av opplegget ba jeg elevene forklare koder eller bruksområder for program. Jeg ser elevenes oppfatning av verktøyet som verdifull og vil ha dette med i tolkningen min av elevenes bruk av verktøyet. Masterprosjektet mitt har begrensede rammer for tidsbruk, men jeg har valgt å undersøke hvordan elevene bruker blokkprogrammering som et verktøy i arbeid med matematikk de allerede har hatt som fokus på 4. trinn. Det tar tid å utvikle forståelse for noe, og jeg ville derfor velge et matematisk tema elevene allerede er i gang med å utvikle forståelse for. Jeg ser også elevenes handlinger som svært relevante for å kunne svare på forskningsfokuset mitt. Vergnaud (1996, sitert i Trouche, 2003, s. 789) ser handlinger som en observerbar form for atferd, og mentale skjema som organisering av atferd for forskjellige situasjoner. Likt tenker jeg at elevenes handlinger i arbeid med oppgaver med ukjent multiplikator, eller i oppgaver med blokkprogrammering, sier noe om hvordan skjemaene til elevene er og om hva de tenker at er mulig i de gitte situasjonene.

Intervensjonsstudie er et bredt forskningsdesign, og har ingen faste kriterier. Cobb et al. (2003, sitert i Prediger et al., 2015) har likevel funnet fem kjennetegn ved intervensjonsstudier:
1. Utforme og studere nye undervisningsformer. Dette er hensikten med intervensjonsstudier.
2. Generere teori. Målet med intervensjonsstudier er å produsere teorier om læringsprosessen og læringsmidler.
3. Teori og studie virker på hverandre. Teori påvirker studien, og etter gjennomføring virker studien tilbake på teorien. Teorien er med å styre hvordan vi velger å gjennomføre studien. Etter gjennomført studie blir teorien påvirket gjennom refleksjon rundt forskjell i ventet og faktiske funn.
4. Gjentakende sykluser for invensjon og revisjon. Intervensjonsstudier inneholder justeringer i form av å tilpasse aktiviteter eller teori, enten undervegs i en studie eller mellom studier.
5. Gyldighet til praksisfeltet. Forskningen må beskrive hvilke forhold den er gjort i. Klasserom er komplekse kontekster og gir umulige forhold for å kopiere studier. Det er viktig å redegjøre for hvilke faktorer som kan ha påvirket studien, og på hvilken måte.

Disse kjennetegnene finnes også i min studie. Hensikten min er å studere det nye redskapet i undervisning, blokkprogrammering. Jeg vil, gjennom mine funn, bidra til å generere teori og utforme blokkprogrammering i undervisningen videre. Jeg vil finne ut mer om hvordan elever bruker blokkprogrammering som et verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon. På bakgrunn av dette har jeg laget oppgaver til elevene som jeg mener gir mulighet til å se at divisjon kan bidra nyttig i arbeid med multiplikasjon. Jeg analyserer elevenes arbeid, ved å se på det de sier og gjør, med fokus på hvordan elevene bruker blokkprogrammering som et verktøy. Undervisningsopplegget ble gjennomført over to økter på to ulike dager. Dette gav meg mulighet til å gjøre justeringer mellom øktene. Også undervegs gjorde jeg små justeringer, som å åpne for å bruke MakeCode til å løse en oppgave som i utgangspunktet var tenkt uten. Gjennom å blant annet opplyse om faktorer som at jeg har større kjennskap til elevene gjennom LAB-Ted, forklare forholdene for fokuselevenes gjennomføring av oppgaven og å fortelle om elevenes møter med programmering før

studien, gir jeg et bilde av konteksten jeg forsket i. Dette skal bidra til at studien oppleves som gyldig.

3.2.1 Kritikk

Forskningsdesignet intervensjonsstudie får hovedsakelig kritikk for å være et for bredt og lite definert design for forskning (Prediger et al., 2015). Det er stor variasjon innenfor kategorien. For et forskningsdesign med lite variasjon og med en tydelig standardisert metode for datainnsamling og analyse, vil det være lett å følge stegene som er gjort og å tro på det studien konkluderer med. Det gir troverdighet å gjennomføre forskning på en etablert måte. Ifølge Clark et al. (2021) er et forskningsdesign et rammeverk for innsamling og analyse av datamateriale. Intervensjonsstudie har ikke standardiserte metoder for dette. Slik kan det være mindre tydelig om valg av metoder i en intervensjonsstudie gir funn av god kvalitet.

3.3 Utforming av undervisningsopplegget

3.3.1 Inspirasjon for utforming

Før jeg utformet oppgaver til elevene, leste jeg om PRIMM - en modell for hvordan lage undervisningsopplegg for programmering i undervisning (Sentance et al., 2019). Jeg synes modellen inneholdt flere viktige poeng, som jeg hadde i bakhodet mens jeg lagde oppgaver til undervisningsopplegget i studien min. PRIMM-modellen har vært til inspirasjon for meg, men ikke noe jeg har fulgt som en oppskrift.

PRIMM sier noe om hvilken rekkefølge ulike oppgavetyper skal bli introdusert for elever (Sentance et al., 2019, s. 148-151):

- P - *predict*: Det er viktig å kunne *lese* kode før man begir seg ut på å *skrive* kode. Her skal elevene forutsi hva som vil skje når et program kjøres.
- R - *run*: Etter at elevene har forutsett hva de tror kommer til å skje i det programmet kjøres, må de teste hva som faktisk skjer ved å lage og kjøre programmet.
- I - *investigate*: Når programmet har blitt testet, skal elevene få oppgaver som legger opp til at de skal utforske strukturen i koden. Elevene kan forklare deler av koden, eller feilsøke den, dersom koden ikke fungerer som ønsket.
- M - *modify*: Videre skal elevene få oppgaver der de må endre på koden. Kanskje oppdaget de gjennom undersøkelsene i forrige steg at koden ikke fungerer optimalt, eller at den har et begrenset bruksområde som kan utvides.
- M - *make*: Til slutt får elevene oppgaver der de skal lage programmer på egen hånd, som løser et nytt problem ved hjelp av strukturene elevene har jobbet med gjennom de foregående stegene i PRIMM.

3.3.2 Beskrivelse av oppgavene

Undervisningsopplegget mitt inneholder syv oppgaver. I første økt ble oppgave 1-4 gjennomført, mens oppgave 5-7 ble gjennomført i andre økt. Fokuset for studien min, å bruke blokkprogrammering i arbeid med sammenhengen mellom multiplikasjon og divisjon, er mest tydelig i andre økt. Oppgavene mine kan deles i to typer. Den første typen, som jeg kaller *erfaringsoppgaver*, er utformet slik at elevene skal bli kjent med den aktuelle matematikken og relevante MakeCode-funksjoner for å arbeide med den neste oppgavetypen. De neste oppgavene kaller jeg *divisjonsoppgaver*. For å skape

situasjoner der elevene kunne bruke divisjon i arbeid med multiplikasjon, tok jeg utgangspunkt i at elevene skulle gå fra oppgaver med ukjent produkt til oppgaver der multiplikator er ukjent. Ukjent produkt skaper, ifølge Van de Walle (2020, s. 197), et behov for multiplikasjon, mens ukjent multiplikator gir et behov for divisjon.

Erfaringsoppgavene er oppgave 1, 2, 3 og 5. Divisjonsoppgavene er oppgave 4, 6 og 7. Begge øktene av undervisningsopplegget starter med erfaringsoppgaver, for å gi elevene relevant erfaring med aktuell matematikk og funksjoner i MakeCode. I erfaringsoppgavene møter elevene multiplikasjonsstykker av hver av de tre oppgavestrukturene på ark, men kun regnestykker med ukjent produkt i sammenheng med blokkprogrammering. I divisjonsoppgavene møter elevene multiplikasjonsstykker med ukjent multiplikator i MakeCode.

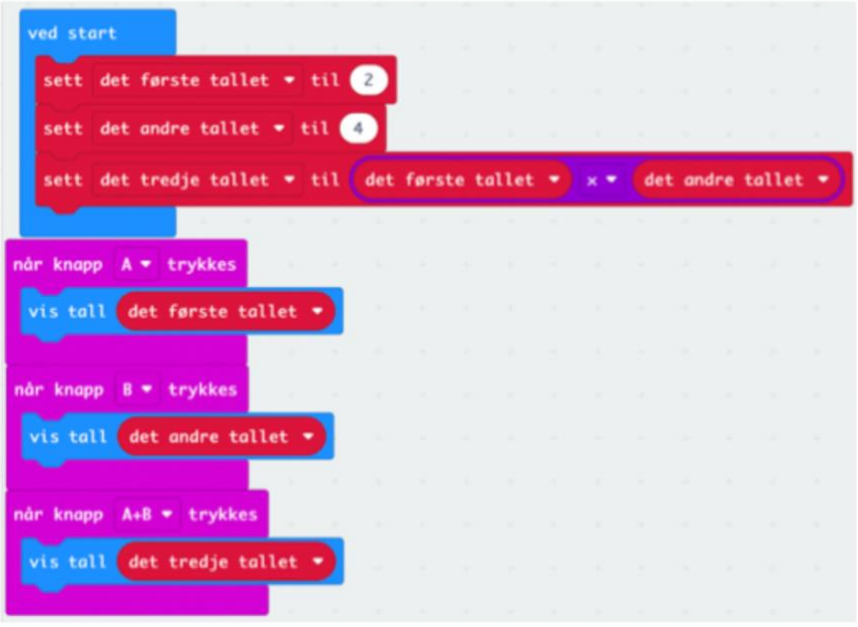
Oppgave 1

Denne oppgaven skal løses på ark, og programmering skal *ikke* brukes. Oppgaven inneholder en liste med ni multiplikasjonsstykker, tre av hver oppgavestruktur. Under de ni regnestykkene får elevene et tiende multiplikasjonsstykke, med ukjent multiplikator, som de skal forklare løsningen på. Her ville jeg undersøke hvilke strategier som var i bruk blant elevene.

Denne oppgaven fungerer som en kartleggingsoppgave. Det var rundt tre måneder siden sist praksisperiode. Den gang merket jeg meg at elevene strevde med oppgavestruktur av annen type enn med ukjent produkt. Elevene hadde få strategier for hvordan disse multiplikasjonsstykkene skulle løses, og prøvde seg mest fram med gjentatt addisjon. Jeg ville undersøke om elevene kunne løse oppgaver av alle strukturer og hvilke strategier de tok i bruk for å finne ukjent multiplikator.

Oppgave 2

Figur 3.3.1: Programmet i oppgave 2



I denne oppgaven får elevene oppgitt bildet over, som er av et program i MakeCode, med tre spørsmål under og en instruksjon til slutt. Spørsmålene handler om hva som

skjer når de ulike knappene på micro:bit-en trykkes. Ved å se hva elevene svarer her vil jeg få et innblikk i hvordan elevene forstår hva som skjer i programmet. Dersom de ikke forstår programmet, vil det være vanskelig å bruke eller modifisere det senere i opplegget.

Programmet inneholder tre programmeringsvariabler. Når programmet startes, blir *det første tallet* satt til to og *det andre tallet* til fire. Den siste variabelen, *det tredje tallet,* blir satt til *det første tallet* multiplisert med *det andre tallet*. Når knapp A trykkes, viser micro:bit-en verdien til *det første tallet*, to. Knapp B viser *det andre tallet* sin verdi, fire. Når knappene trykkes samtidig, knapp A+B, vises verdien til *det tredje tallet*, altså åtte.

Etter at elevene har svart på spørsmålene til kodeanalysen, får de instruksjon om at de skal lage programmet på bildet i MakeCode og teste det. De blir slik kjent med hvor de finner relevante blokker i menyen, og de må lage programmeringsvariabler. De får så testet om det de svarte på spørsmålene stemmer. I løpet av oppgave 2 jobber elevene med de to første stegene i PRIMM-modellen, *predict* og *run*.

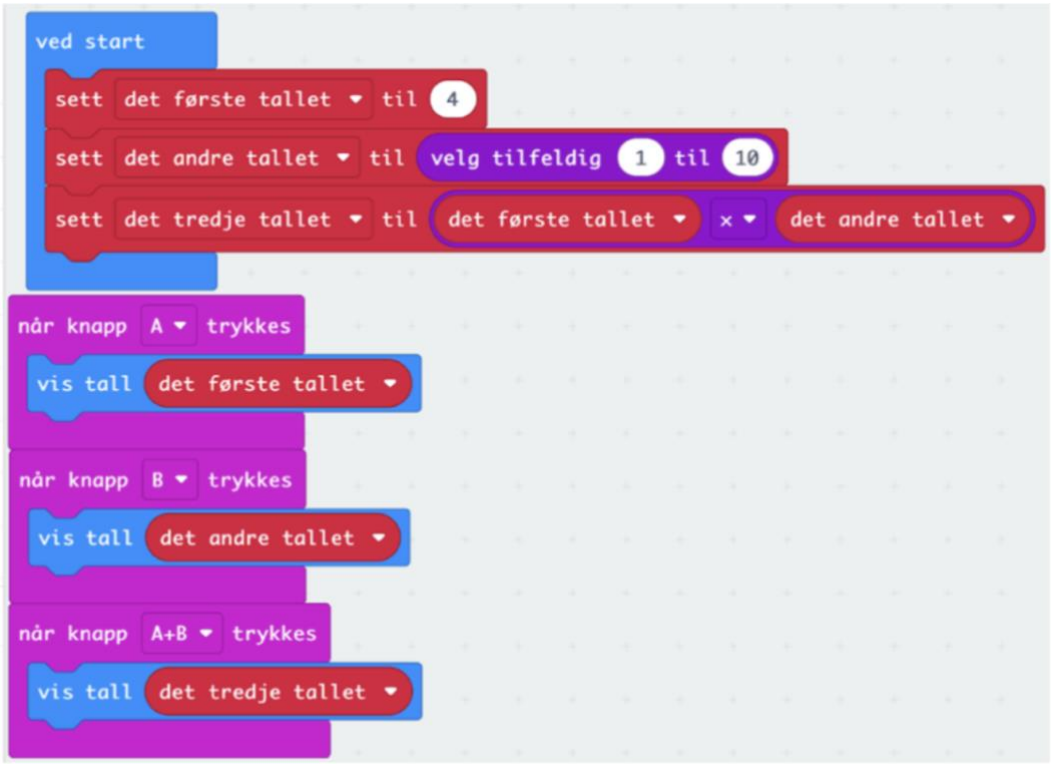
Oppgave 3
Her skal elevene bruke programmet fra oppgave 2 til å løse tre multiplikasjonsstykker. Elevene må sette verdiene til programmeringsvariablene *det første tallet* og *det andre tallet*, til verdiene som blir oppgitt i regnestykket. For eksempel kan $7 \cdot 6$ løses ved å sette *det første tallet* til syv og *det andre tallet* til seks. Når elevene trykker knapp A+B, vil svaret vise seg på skjermen til micro:bit-en. Denne oppgaven er utformet med tanke på å undersøke elevenes forståelse av programmet ytterligere ved at elevene må endre og bruke programmet. For å løse denne oppgaven må elevene ikke bare lese kode, men også kunne bruke den. Denne oppgaven kan bli sett i sammenheng med I - *investigate* og M – *modify* i PRIMM-modellen, ved at elevene blir bedre kjent med strukturene i programmet ved å ta det i bruk.

Oppgave 4
Dette er elevenes første møte med en divisjonsoppgave. Elevene skal i denne oppgaven løse multiplikasjonsstykkene fra oppgave 1 ved hjelp av programmet de har i editoren. Elevene har til nå bare løst multiplikasjonsstykker med ukjent produkt i MakeCode. Nå skal de løse de samme stykkene som de gjorde i oppgave 1, der det var tre multiplikasjonsstykker av hver oppgavestruktur, og de vil dermed få utfordringer med hvordan de skal bruke programmet for å finne ukjent multiplikator eller multiplikand.

Oppgaven er lagt opp for at elevene skal få et behov for å endre på koden, slik at den kan løse stykker der multiplikator eller multiplikand er ukjent. Denne oppgaven kan bli sett i sammenheng med den første M-en i PRIMM, *modify*, ved at elevene skal endre på koden. Siden dette var siste oppgave i første økt, var tanken at elevene kun skulle få tid til å gruble og teste ut ideer. Divisjonsoppgavene er i større fokus i andre økt, og oppgave 4 er ment til å sette i gang elevenes undring til denne økta.

Oppgave 5

Figur 3.3.2: Programmet i oppgave 5



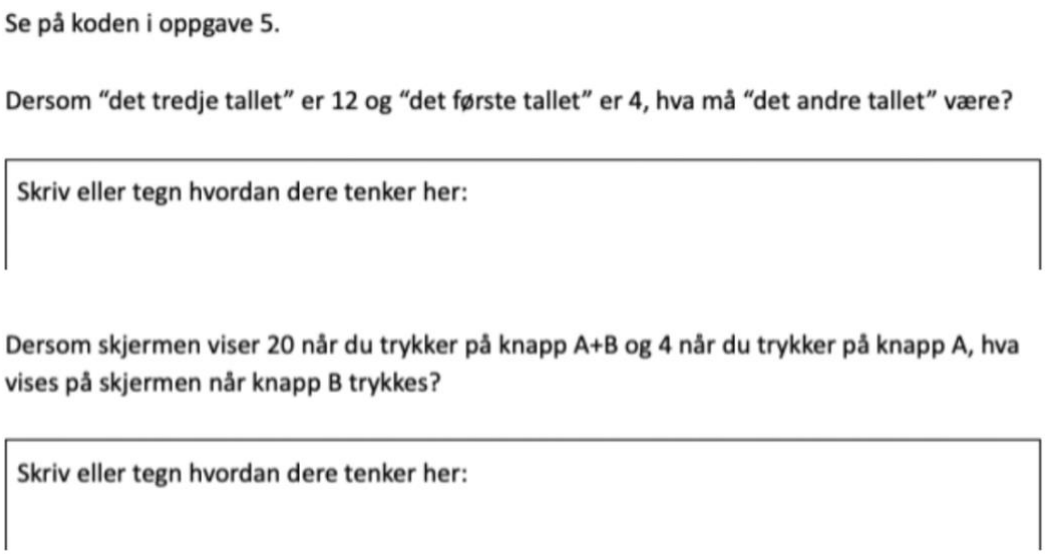
Andre økt starter med en erfaringsoppgave, med en analyse av et relativt likt program som i oppgave 2. Forskjellen er at i det nye programmet blir programmeringsvariabelen *det andre tallet* satt til et tilfeldig tall mellom én og ti.

Poenget med at *det andre tallet* er satt til et tilfeldig tall er å gjøre den til en ukjent. Alle programmeringsvariabler må være definerte for å kunne brukes, altså være satt til en verdi med en sett-blokk. Her er *det andre tallet* definert og satt til et tall mellom én og ti. Det er ikke synlig i koden hvilket spesifikt tall *det andre tallet* er satt til, men ved å trykke på knapp B, vil micro:bit-en vise hvilken verdi programmeringsvariabelen har fått tildelt. Hver gang programmet kjøres på nytt, vil koden velge et tilfeldig tall mellom én og ti for *det andre tallet*. Blokka for tilfeldig tall bidrar til at programmet blir mest mulig likt det elevene ble kjent med i første økt. Programmeringsvariabelen *det tredje tallet* er fortsatt satt til å være produktet av de to andre variablene. På denne måten overfører forhåpentligvis elevene forståelsen sin fra forrige økt om at programmet fungerer for å løse multiplikasjonsstykker. Operasjonen er den samme, og er synlig ved multiplikasjonstegnet i sett-blokka til *det tredje tallet*, men oppgavestrukturen gir nå en ukjent multiplikator.

Oppgaveteksten inneholder et bilde av programmet, tre spørsmål om hvordan programmet fungerer og en instruksjon om å lage programmet til slutt. Elevene skal svare på spørsmål om hva micro:bit-en viser når de ulike knappene blir trykket, for å vise at de forstår koden. Deretter skal de lage programmet i MakeCode og teste hva koden faktisk gjør. Også denne økta starter altså med PRIMM-modellens *predict* og *run*.

Oppgave 6

Figur 3.3.3: Oppgaveteksten til oppgave 6



Dette er en divisjonsoppgave, der elevene møter to tekstoppgaver, som har formuleringer som viser til MakeCode og micro:bit. Oppgaveteksten informerer elevene om å se oppgaven i sammenheng med programmet i oppgave 5. Den første tekstoppgaven er et multiplikasjonsstykke som bruker navnene til programmeringsvariablene fra det kjente programmet i opplegget. Den andre oppgaven bruker formuleringer som viser til knappene til micro:bit-en. Begge tekstoppgavene representerer multiplikasjonsstykker med ukjent multiplikator. Her har elevene mulighet til å dividere produkt på multiplikand.

Hver av tekstoppgavene har et felt under for å forklare hvordan elevene tenkte for å løse oppgavene. Det står ikke spesifisert at oppgavene skal løses i MakeCode eller at de skal løses uten MakeCode. Jeg hadde i utgangspunktet tenkt her at elevene ikke skulle bruke MakeCode, for å se hvordan de tolket formuleringene og om de kunne løse oppgavene på ark. Beskjeden om å løse oppgavene uten MakeCode ble glemt gitt til fokuselevene, og fokuset mitt for oppgaven blir dermed noe annerledes. Siden elevene her stod fritt i valg av strategi, er det interessant å se hvor mange av dem som valgte å bruke MakeCode.

Oppgave 7

Her møter elevene enda en divisjonsoppgave. Denne oppgaven skiller seg fra de andre divisjonsoppgavene, ved at det her blir eksplisitt opplyst om at elevene skal lage en kode for å finne den ukjente multiplikatoren.

I oppgaveteksten blir det opplyst om at elevene skal få micro:bit-en til å vise verdien til *det andre tallet*, men uten å bruke blokka som heter *det andre tallet*. Programmeringsvariabelen endrer verdi hver gang programmet kjøres, siden den er satt til et tilfeldig tall mellom én og ti, som gjør at elevene må lage en generell kode i stedet for å skrive inn et spesifikt tall. Fram til nå har elevene kunne se hvilken verdi *det andre tallet* har, ved å sette inn en verdiblokk for *det andre tallet* i en handlingsblokk som viser tall. Ved at denne muligheten blir tatt bort, må elevene finne en annen måte å finne verdien til *det andre tallet*. Elevene kan komme fram til koden *det tredje tallet* dividert

med *det første tallet,* som i figur 3.3.4. Denne koden vil alltid finne riktig verdi for *det andre tallet*. Programmeringsvariabelen *det andre tallet* får ny verdi hver gang programmet kjøres, men det gjør også *det tredje tallet*. Denne programmeringsvariabelen er definert som *det første tallet* multiplisert med *det andre tallet*. For at koden for *det andre tallet* alltid skal finne riktig verdi, kan *det tredje tallet* med fordel brukes i koden. Programmeringsvariabelen *det tredje tallet* kan sees som en avhengig algebraisk variabel, som avhenger av verdiene til de uavhengige variablene *det første tallet* og *det andre tallet*.

Figur 3.3.4: Ønsket kode

Jeg har en hypotese om at elevene her vil synes det er tungvint å skulle bruke det Rønning og Burheim (2020) refererer til som primitive modeller for multiplikasjon. Et eksempel på en strategi innen primitive modeller er gjentatt addisjon. Elevene kunne prøvd å bruke gjentatt addisjon i MakeCode også, men måtte da ha brukt mange blokker. En addisjonsblokk i MakeCode har kun to verdifelt, og elevene måtte derfor ha koblet sammen mange addisjonsblokker for å addere flere tall. Dersom elevene skulle løst $5 \cdot \_ = 30$ ved hjelp av gjentatt addisjon, måtte elevene vært gjennom flere steg. Her hadde den ukjente vært antall ganger fem blir addert for å nå 30, og det finnes ingen blokk i MakeCode som teller antall verdifelt. Koden i figur 3.3.5 hadde altså vist tallet 30, men elevene hadde trengt en blokk som viste at de hadde lagt sammen seks femmere.

Figur 3.3.5: Gjentatt addisjon i MakeCode

Denne oppgaven er den som minner mest om M - *make* i PRIMM-modellen. Elevene skal ikke lage et helt program på egen hånd, men de skal lage en ny kode i et kjent program. Denne nye koden bidrar til at multiplikasjonsprogrammet også kan brukes ved ukjent multiplikator, og utvider slik bruksområdet til programmet. Oppgaven kan slik sees som en oppgave for M - *modify* også. I denne oppgaven er det et utforskende element, som Utdanningsdirektoratet (2019) løfter fram som passende for programmering. Siden jeg hadde et matematisk mål som skulle nås i oppgaven, å bruke divisjon for å finne multiplikatoren, er ikke oppgaven helt utforskende likevel. For å veilede utforskingen til elevene hadde jeg med tipslapper som elevene kunne be om eller få, der lærerne tenkte det var aktuelt, undervegs. Tipsene gikk ut på hvilke blokker som kunne brukes og at elevene kunne forsøke å bruke noen av de andre matematiske operasjonene i matematikkblokkene.

3.4 Innsamling av datamateriale

Undervisningsopplegget mitt er delt i to og ble gjennomført over to økter. I den første økta var seks fokuselever sammen med meg og en medstudent på et eget rom. Elevene

ble filmet med videokamera, det ble brukt lydopptaker for å ta opp samtalene deres og det ble tatt skjermopptak av Chromebookene elevene brukte. I tillegg hadde jeg et observasjonsskjema jeg skrev notater i undervegs. Etter gjennomført økt satt jeg meg ned for å skrive flere notater fra det jeg hadde observert. Disse notatene inneholdt også tilbakemeldinger fra andre lærere som hadde gjennomført samme opplegg for resten av trinnet.

I andre økt av undervisningsopplegget var innsamlingen av datamateriale relativt lik. Fokuselevene måtte denne gang sitte i klasserommet med resten av klassen, på grunn av mangel på ledige rom. De satt imidlertid helt bakerst på et gruppebord, slik at vi fortsatt kunne filme og ta opp lyd uten at andre elever kom med. Også denne gang tok vi skjermopptak av fokuselevenes Chromebooker. Jeg noterte observasjoner inn i et skjema og noterte videre etter gjennomført økt. Her fikk jeg notert ned tilbakemeldinger med observasjoner gjort av andre lærere. Det ble samlet inn besvarte oppgaveark fra hele trinnet i begge øktene. Av disse tok jeg vare på fokuselevene sine, og skrev ned en felles oversikt over hvilke strategier som var i bruk og hvilke oppgaver som virket utfordrende for resten av elevene på trinnet.

3.4.1 Oversikt over innsamlet datamateriale

Tabell 3.1: Oversikt over innsamlet datamateriale

	ØKT 1	**ØKT 2**

Fokuselever	• Besvarte oppgaveark
	• Skjermopptak
	  ○ Fokuspar VA: 14 min og 21 sek
	  ○ Fokuspar KJ: 17 min og 26 sek
	  ○ Fokuspar LH: 20 min og 01 sek
	• Videoopptak
	• Lydopptak
		• Besvarte oppgaveark
		• Skjermopptak
		  ○ Fokuspar VA: 22 min og 29 sek
		  ○ Fokuspar KJ: 46 min og 22 sek
		  ○ Fokuspar LH: 47 min og 08 sek
		• Videoopptak
		• Lydopptak

Annet	• Utfylt observasjonsskjema (meg selv) • Notater som inneholder kommentarer om observasjoner fra andre lærere	• Utfylt observasjonsskjema (meg selv) • Notater som inneholder kommentarer om observasjoner fra andre lærere

For å svare på hvordan elevene brukte blokkprogrammering som et verktøy, har jeg sett på både handlinger og uttalelser. Jeg er interessert i å både se hvordan elevene bruker blokkprogrammering i arbeid med multiplikasjonsoppgavene, og høre hva de sier mens de programmerer. Datamaterialet som ble samlet inn er vist i tabell 3.1. Den viser fordeling i økter og skiller mellom datamateriale hentet direkte fra fokuselevene og annet datamateriale i form av observasjonsnotater.

Skjermopptakene fra fokuselevene er den viktigste delen av datamaterialet mitt. Her får jeg se hva fokuselevene gjør i editoren, samtidig som jeg hører hva de sier. Varigheten på skjermopptakene varierer etter når i undervisningsopplegget opptakene ble startet og stoppet. Noen ble tatt av med en gang elevene følte seg ferdig med oppgavene og noen hadde opptak på i dansepauser og en stund etter at de var ferdige å snakke om oppgavene. Initialene VA viser til fokusparet Vilma og Andrine. KJ viser til fokusparet Knut og Johannes. LH viser til fokusparet Lily og Hedda. Alle navnene er fiktive. De besvarte oppgavearkene til fokuselevene blir også bruk for å se hvordan elevene svarte på de skriftlige oppgavene. Skjermopptakene var av god kvalitet, som gjorde at jeg ikke fikk behov for datamaterialet samlet med lydopptaker og videokamera. Disse er derfor ikke tatt med videre til analyse, og er merket i grått i tabellen. De ble samlet inn, men ble slettet i stedet for å bli brukt.

Datamaterialet i raden «annet» består av mine observasjoner for fokuselevenes arbeid, samt mine notater for andre læreres observasjoner av elevene på resten av trinnet sitt arbeid. Datamaterialet i denne raden er lite brukt i analysen. Jeg fokuserer hovedsakelig på fokuselevenes arbeid, men siden opplegget ble gjennomført for hele trinnet på omtrent hundre elever, gir det en god mulighet for å sammenligne mitt lille utvalg elever opp mot et større utvalg. Notatene mine med oversikten fra trinnet viser at det ikke var særlige forskjeller i strategivalg mellom fokuselevene og resten av trinnet. Observasjonsnotater ble også brukt for å framkalle hvilken oppfatning jeg hadde av elevenes verktøybruk i situasjonen, som inspirerte meg i analyseprosessen.

3.4.2 Observasjon

Jeg og en av mine medstudenter var til stede hos fokuselevene under gjennomføringen av begge delene av undervisningsopplegget. Vi var *aktive observatører* ved å svare elevene på spørsmål og å stille spørsmål til det de gjorde. Dette kan kalles *deltakende observasjon*. Clark et al. (2021, s. 392) definerer denne formen for observasjon som der forskeren er med en gruppe i felt for å undersøke hvordan deltakerne gir mening til noe. Forskeren har da mulighet til å stille spørsmål, høre samtaler mellom deltakerne og se hva de gjør. Vi fortalte fokuselevene at vi skulle se på hvordan de jobbet med programmering og bruke det anonymisert i masteroppgaver. Slik var forskerrollene våre *åpne*.

Før gjennomføringen av undervisningsopplegget lagde jeg observasjonsskjema til meg selv. I skjemaet skrev jeg ned spørsmål om ting jeg var interessert i å følge med på. Observasjonsfokusene var ikke styrt av teori, men egne tanker for hva som kunne si noe om hvilken forståelse elevene hadde for micro:bit og MakeCode. Jeg lagde eksempelvis en tabell, der jeg skrev opp statistikk for om elevene svarte riktig på spørsmålene til programmet i oppgave 2 og 5, med et felt til høyre for å skrive inn hva som var eventuelle utfordringer. Jeg hadde også skrevet opp spørsmål jeg ville stille elevene, og felt for å skrive inn det elevene svarte. Et eksempel på dette er «Hvilke tall fungerer denne kalkulatoren for?». Kalkulator var et navn jeg brukte for programmet i opplegget. Spørsmålet er tenkt for å gi meg innblikk i hvilken forståelse elevene har for programmet. Dette er en form for *strukturert observasjon*. Clark et al. (2021, s. 256) omtaler dette som observasjon der forskeren på forhånd har laget et skjema som styrer observasjonen. Skjemaet inneholder kategorier der handlinger blir plasserte inn, og oversikt over spørsmål forskeren vil stille. Observasjonsskjemaet gjorde at jeg husket å

stille spørsmål som kunne gi meg innblikk i elevenes forståelse. Selv om jeg hadde laget felt for å skrive ned svar fra elevene, brukte jeg dette minimalt. Det fungerte mest som en påminner om å følge med på hvordan elevene forstår MakeCode og programmet. I etterkant av øktene noterte jeg ned alt jeg merket meg som interessant i løpet av gjennomføringen. På denne måten kan jeg kategorisere observasjonene mine som *delvis strukturerte.* De var litt styrt av et skjema, men stort sett basert på hva jeg oppfattet som interessant i situasjonen.

Andre lærere var også involverte i gjennomføringen av undervisningsopplegget mitt. De var alle deltakende observatører, men uten skjema eller spesielle fokus. Jeg fikk muntlig, og én skriftlig, tilbakemelding fra hvordan de synes elevene fikk til å jobbe med oppgavene i MakeCode. Disse kommentarene har jeg skrevet ned som egne notater etter gjennomført økt.

For å hjelpe meg til å samle inn observasjoner av fokuselevene, hadde jeg flere hjelpemidler. Fokuselevene satt sammen i par rundt et gruppebord. Det lå en lydopptaker midt på bordet og et videokamera på stativ et par meter unna som filmet dem. Hvert elevpar hadde en Chromebook foran seg med skjermopptak, som også tar opp lyd. Skjermopptaket filmer det som skjer på skjermen, mens vi hører hva elevene sier. Skjermopptak kan sies å være en form for observasjon. Ved å ta skjermopptak, blir alle handlingene og utsagnene til elevene fanget opp, også det som skjer når læreren er et stykke unna. Slik kan vi kanskje få et enda bedre bilde av elevenes *naturlige* atferd. På skjermen kan vi se all deres prøving og feiling, og ikke bare et ferdig produkt som er klart til å bli vist til læreren. Vi får også høre alt de sier, alle ideer de har undervegs, men også alle avsporinger. Totalt sett får skjermopptak med mye. Utfordringene med skjermopptak, slik jeg ser det, er at det kan være vanskelig å høre hva elevene sier, og hvem som sier hva, dersom de sitter nært andre elever.

3.4.3 Transkripsjon

Det første jeg gjorde etter å ha samlet ferdig datamaterialet, var å begynne å *transkribere*. Ifølge Nilssen (2012, s. 46) handler å transkribere om å gjøre tale til tekst, og å skrive ned observasjoner og tanker. Som nevnt tidligere, valgte jeg bort lydopptakene og videoopptakene, siden skjermopptak fra elevenes Chromebooker gav meg det jeg trengte av observasjoner av handling og utsagn. Selv om det kun var skjermopptakene som skulle transkriberes, tok det lang tid. Jeg var interessert i både det elevene gjorde på skjermen og det de sa samtidig, noe som gjorde at jeg måtte se gjennom alt minst to ganger; Én gang for å skrive ned handlinger og én gang for å skrive ned utsagn. Jeg hadde til sammen seks skjermopptak, ett fra hvert fokuspar fra hver økt.

Siden en av mine medstudenter også skulle bruke samme datamateriale, fordelte vi transkriberingen oss imellom. Det er i utgangspunktet best å transkribere alt selv, siden ideer til koding kan komme og med hensyn til elevene som har sett for seg at det bare er læreren som skal høre gjennom (Nilssen, 2012). Siden elevene var kjent med at både medstudenten min og meg skulle forske på dem, og siden mengden materiale ble så stor, fant vi ut at det i vårt tilfelle var best med en fordeling. Ulemper med transkripsjon er ifølge Nilssen (2012) at man går glipp av kroppsspråk, ser ikke dersom elever for eksempel peker, kan blande sammen stemmer og høre feil. Alle disse utfordringene

møtte vi på. Det er steder i transkripsjonen der det er tydelig at elever peker, uten at vi kan være sikre på hvor. Stemmer blandet seg også ofte sammen, og noen ganger var det ikke mulig å høre hva elevene sa.

Utdrag fra transkripsjoner fra elevenes arbeid er presentert i resultatkapittelet. Her er elevenes handlinger forklart i innledningen før elevenes utsagn blir presentert. Handlinger som skjer mens elevene snakker er skrevet i hakeparenteser slik: [elevhandling]. Hakeparentes med tre punktum i, slik [...], markerer at elevutsagn og -handlinger er utelatt. Dette skjer i situasjoner der elevene snakker om eller gjør noe som ikke er relevant for fokuset, men der det som kommer før og etter er relevant. Alle utsagn er skrevet ned ordrett, med tenkepauser notert som tre punktum, slik "…". Jeg har tilpasset utsagnene fra elevenes dialekter til standardisert bokmål, for å lette leseligheten. Elevnavnene som blir presentert i transkripsjonen er fiktive. Fokusparene er kalt Vilma og Andrine, Knut og Johannes og Lily og Hedda. Lærernavnene «lærer E» og «lærer G» viser til henholdsvis meg og medstudenten min som deltok under fokuselevenes gjennomføring.

3.5 Analysemetode

Transkripsjonen av skjermopptakene gav meg et altfor stort datamateriale. Jeg delte hvert fokuspar sin transkripsjon inn i deler for de ulike oppgavene. Deretter satte jeg sammen delene som handlet om samme oppgave, fra hvert fokuspar, sammen i ett dokument. På denne måten håpet jeg å se om det var likheter eller mønster som viste seg i elevenes arbeid. Mengden datamateriale var likevel for stor. For at datamaterialet skulle være overkommelig å analysere, måtte jeg redusere mengden. Jeg valgte derfor å dele datamaterialet inn i sekvenser. Dette er inndelinger som avgrenser deler av transkripsjonen der én type ting skjer (Nilssen, 2012, s. 106). Prosessen for å dele inn i sekvenser krevde en grovanalyse av de transkriberte skjermopptakene. Jeg startet med å lese gjennom alt, for å få noen tanker om hva som ville være interessant å se etter for å svare på forskningsfokuset mitt. Jeg brukte også observasjonsnotatene mine her, for å få med meg hva jeg merket meg som spesielt interessant i gjennomføringen. Siden divisjonsoppgavene er de som er mest relevante for forskningsfokuset mitt, var det i utgangspunktet her jeg lette etter interessante sekvenser. Enkelte situasjoner fra elevenes arbeid med erfaringsoppgavene er likevel aktuelle for å diskutere fokuset mitt, ved at de kan vise til elevenes utvikling i bruk av blokkprogrammering som verktøy.

I inndelingsprosessen av sekvenser hadde jeg kriterier som jeg så etter, som tok utgangspunkt i hva jeg tenkte ville være interessant for forskningsfokuset mitt. Sekvenser kunne enten omhandle en situasjon der elevene diskuterte matematikk, gjorde endringer i programmet, brukte MakeCode til å løse en oppgave, eller diskuterte muligheter eller begrensninger i programmet eller editoren. Denne inndelingen gav meg 23 sekvenser. De fleste sekvensene inneholder flere av kriteriene nevnt over. 15 av sekvensene inneholdt diskusjon rundt funksjoner i programmet eller editoren, ni sekvenser inneholdt matematiske diskusjoner, fire sekvenser viste elever som løste en divisjonsoppgave ved hjelp av MakeCode og fire sekvenser handlet om at elevene endret på programmet. Kun én av sekvensene viser alle fire kriteriene. Sekvensene med relevans for forskningsfokuset ble med videre. Resten av datamaterialet valgte jeg bort, og er ikke med videre i analyseprosessen. På denne måten var det kun situasjoner som

handlet spesifikt om matematikk og bruk av blokkprogrammering som ble analysert nærmere.

3.5.1 Åpen koding

Etter å ha kortet ned mengden datamateriale forsøkte jeg å finne ut hva det fortalte meg. Jeg prøvde å finne en måte å gi mening til datamaterialet mitt. Dette kan beskrives som en *induktiv tilnærming*. Her er teori utfallet av studien, og ikke det som styrer analysen (Clark et al., 2021, s. 20). I induktive analyser skal jeg som forsker kode datamaterialet, og samle koder i kategorier gjennom å lete etter mønster i kodene (Nilssen, 2012). Kategoriene forteller hva som er funnene i studien. Med en induktiv tilnærming skal jeg gå inn i kodingsprosessen med et åpent sinn, men det er samtidig viktig å være klar over at dette aldri er fullstendig mulig (Nilssen, 2012, s. 26). Alle har erfaringer og verdier som farger en selv, og som ikke er mulig å bare «koble av».

Jeg forsøkte å finne svar på forskningsfokuset mitt i datamaterialet, som er å se på hvordan blokkprogrammering fungerer som et verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon. Ved å dele inn i og velge ut sekvenser, hadde jeg allerede gjort en grovanalyse. For å finne mer ut om hvordan blokkprogrammering fungerer som et verktøy, forsøkte jeg å identifisere tegn på at elevene oppfattet det som et verktøy eller ikke i sekvensene. Jeg lagde først koder for alt jeg synes kunne være relevant, og endte opp med alt for mange koder. Deretter lagde jeg koder for om elevenes strategier fører dem til mål, om de får galt svar og om de står fast eller gir opp. Dette fant jeg ut at ikke var relevant for fokuset mitt, og kategoriene ble dermed forkastet. Deretter så jeg på hvor elevene bruker blokkprogrammering i oppgaveløsningen og hvor de velger det bort, hvor de bruker matematikk for å begrunne programmeringen sin, hvor de ikke gjør det, og hvor elevene viser tillit til programmet og hvor de viser mistillit. Jeg har ikke vært styrt av noen teori eller ferdige kategorier i analysen min av datamaterialet. Kodingen min kan slik sies å være *åpen*. Nilssen (2012, s. 78) beskriver *åpen koding* som å møte datamaterialet med et åpent sinn. Forskeren forsøker å finne ut hva datamaterialet kan fortelle oss, og slik skape en ny teori ut fra det. Dette er en induktiv tilnærming, der man først samler datamateriale som deretter gir oss nye teorier (Clark et al., 2021, s. 20). I åpen koding prøver forskeren ut ulike koder, leter etter mønstre mellom koder og lager til slutt kategorier (Nilssen, 2012). Analysen min førte til slutt fram til kategoriene:

Tabell 3.2: Kategorier for funn i analysen

Kategorinavn	Navn på underkategorier	Beskrivelse
Bruker divisjon for å finne ukjent multiplikator	-	Elevene tar, som ønsket, i bruk divisjon i arbeid med multiplikasjon
Unngår å lage kode for å finne ukjent multiplikator	- Endrer oppgavestrukturen til multiplikativ operasjon - Endrer fra variabel til konstant verdi	Elevene går bort fra strukturen med en ukjent multiplikator, og har slik ikke behov for divisjon lenger
Bruker egne matematikkunnskaper for å gi mening til programmet	- Sjekker at programmet regner riktig - Bruker automatiserte tallfakta for å gi mening til programmet - Gjør et registerskifte for å gi mening til programmet	Elevene bruker matematikkunnskapene sine for å forklare det som skjer i programmet eller for å kontrollere at programmet gir riktig svar
Gjør seg kjent med funksjoner i MakeCode	- Lager egne oppgaver for å teste programmet - Oppdager snarveger - Oppdager at variabler må defineres	Elevene oppdager hvordan blokkene i MakeCode brukes eller tester hvilke tall programmene er gyldige for
Ser ikke muligheter i MakeCode	- Får ikke gjort det de vil i MakeCode - Løser oppgaven uten MakeCode	Elevene vurderer om en oppgave kan løses i MakeCode, men konkluderer med at det ikke er mulig, selv om de ikke har prøvd

Noen av kategoriene er spisset mot kun divisjonsoppgavene, og har derfor kun funn innenfor disse oppgavene. Dette gjelder *bruker divisjon for å finne ukjent multiplikator, unngår å lage kode for å finne ukjent multiplikator* og *ser ikke muligheter i MakeCode.* De siste to kategoriene er synlige i arbeid med alle oppgavene, både divisjons- og erfaringsoppgaver.

3.6 Etiske prinsipper

I denne studien har jeg forsket på elever på 4. trinn. I forskning er det alltid viktig å verne om deltakerne, spesielt i møte med barn (Nilssen, 2012, s. 150). Fokuselevene i denne studien har fått fiktive navn som kun jeg, og min medstudent som jeg deler datamateriale med, kjenner de ekte identitetene til. Gjennom anonymisering og at jeg ikke oppgir verken skolenavn eller viser til skriftlig elevarbeid, skal elevene ikke være mulige å kjenne igjen. Datamaterialet ble anonymisert rett etter innsamling gjennom koder som viste til elever. Elevene som ble med i fokusgruppa er valgt blant annet gjennom kriterier som at de selv hadde lyst til å være med og at de hadde levert samtykkeskjema fra foreldre som godkjente at barna deres kunne bli med. Undervegs i gjennomføringen kunne elevene når som helst trekke seg. De kunne også be om pauser, som elevene gjorde etter som de ble slitne og trengte å bevege på seg.

Vi sørget for at resten av trinnet jobbet med akkurat samme opplegg som oss, slik at det ikke skulle være noe problem dersom en fokuselev ikke ville være i fokusgruppa lenger. Dette bidro også til at det ikke skulle være noen ulempe for elevene å være med i forskningsprosjektet mitt. De gikk ikke glipp av noe, siden alle jobbet med det samme. Det er et viktig etisk prinsipp i samfunnsforskning at det ikke skal være noen ulempe å være deltaker i et forskningsprosjekt (Clark et al., 2021, s. 113). Et annet viktig etisk prinsipp er å be om skriftlig samtykke. Alle fire studentene på praksisgruppa sendte ut felles informasjons- og samtykkeskjema til elevenes foreldre i god tid før vi kom for å samle inn datamateriale. Siden de deltakende elevene kun er rundt ni år, måtte foreldrene godkjenne deltakelse. Prosjektet ble forklart muntlig for elevene, på en måte

som var tilpasset dem. I informasjonsskrivet til foresatte opplyste vi om prosjektene våre, om hvordan vi skulle samle og bruke datamateriale, og om anonymisering og oppbevaring av innsamlet datamateriale på krypterte eksterne enheter. Studien er meldt til *Norsk senter for forskningsdata* (NSD, nå SIKT).

I samfunnsforskning er det viktig å være bevisst sin egen *forskerrefleksivitet* (Nilssen, 2012). Som tidligere nevnt, kommer alle mennesker med verdier og erfaringer som farger alt vi opplever. Det er umulig å unngå dette, men det er heller noe forskere skal være bevisst (Nilssen, 2012). Det kan være spesielt utfordrende der man til vanlig er en del av konteksten som det forskes i. Selv om jeg ikke er lærer for elevene i studien min, kjenner jeg dem godt gjennom å ha vært praksisstudent i klassen deres i flere uker, tilsvarende to måneder sammenlagt.

En del av å vise refleksivitet, er å være *transparent* gjennom hele forskningsprosessen (Nilssen, 2012, s. 140). Det innebærer å synliggjøre alt som er gjort og tenkt undervegs. Jeg har i min studie redegjort for prosessen min ved blant annet å fortelle om min relasjon til elevene i studien, som kan ha påvirkning for hvordan elevene oppfattet situasjonen og løste oppgavene. Jeg har synliggjort min prøving og feiling i koding av datamaterialet, utfordringer i arbeid med transkribering og vist til endring av fokus som følge av at beskjeder ble glemt gitt til elever undervegs i gjennomføringen av undervisningsopplegget.

Det er også viktig at hele forskningsprosessen synliggjøres for at studien skal bli oppfattet som *troverdig* (Nilssen, 2012, s. 141). I samfunnsforskning og kvalitativ forskning finnes det flere virkeligheter og mange faktorer som ikke kan styres (Clark et al., 2021, s. 24; Nilssen, 2012, s. 25). Dette gjør at studien min ikke kan kopieres og få samme resultat. Elevene i studien min er unike, og selv om jeg hadde gjennomført på nytt med samme elever, ville aldri dagsformen deres, erfaringene og nivået deres vært det samme. Arbeidet til elevene i studien kan også ha blitt påvirket ved at det var studenter på trinnet, og at det ble gjennomført en studie som så på de samme elevenes engasjement samtidig som jeg gjennomførte min. Forhåpentligvis vil funnene mine likevel bli oppfattet som troverdige, innenfor min kontekst, gjennom en nøye gjennomgang av prosessen min fra å bestemme tema som en del av trepartssamarbeidet i LAB-Ted til innsamling av datamateriale og åpen koding med en induktiv tilnærming.

4 Resultat

Fokuset for studien min er hvordan blokkprogrammering fungerer som et verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon på 4. trinn. I dette kapittelet ser vi hvordan seks fokuselever jobbet med undervisningsopplegget utformet for forskningsfokuset. Elevene med fiktive navn var delt i tre par, Andrine og Vilma, Johannes og Knut og Lily og Hedda. Jeg har gjennom en åpen koding, som har fokusert på hvordan elevene knytter inn matematikk og bruker artefakten, kommet fram til fem kategorier for funn. De er:

- Bruker divisjon for å finne ukjent multiplikator
- Unngår å lage kode for å finne ukjent multiplikator
- Bruker egne matematikkunnskaper for å gi mening til programmet
- Gjør seg kjent med funksjoner i MakeCode
- Ser ikke muligheter i MakeCode

4.1 Bruker divisjon for å finne ukjent multiplikator

Divisjon ble nesten ikke brukt av elevene i arbeidet deres med oppgavene. Lily og Hedda er det eneste elevparet blant fokuselevene som prøvde ut å bruke divisjon. Under er et utdrag fra jentene sitt arbeid med oppgave fire. Jentene har akkurat løst alle multiplikasjonsstykkene ved å endre oppgavestrukturen, slik at de passer med det ferdige programmet i oppgaveteksten. Medstudenten min, lærer G, veileder dem til å heller løse multiplikasjonsstykkene i MakeCode med strukturen de har på oppgavearket.

Utdrag 1

Lærer G: Men hva hvis dere tenker.. at disse oppgavene, at dere ser dem på nytt, uten at dem er regnet på, at det er tomt der, der, der. Hvordan kunne dere brukt denne her for å finne de tallene da?
Lily: Vet ikke.
Lærer G. Har du noen ideer?
Hedda: Hva sa du?
Lærer G: Hvordan kan dere bruke..
Lily: Hvis man tar deling, kanskje?
Lærer G: Okei! Hvordan tenkte du da?
Lily: For eksempel på denne her.. [Viser til regnestykket $5 \cdot \_ = 45$]
Lærer G: Hva ville du delt?
Lily: Hm, fem og 45.
Lærer G: Du ville delt fem og 45?
Lily: Mhm, på hverandre.
Lærer G: Fem delt på 45 og 45 delt på fem? Eller hva tenkte du?
Lily: Hm, begge deler kanskje?
Lærer G: Begge deler? Hva tror du? Går det an å bruke deling for å finne det ut?
Hedda: Eh.. ja.. Vet ikke helt.

I utdraget ser vi at Lily foreslår divisjon som en mulig strategi for å finne verdien til den ukjente multiplikatoren. Hun er derimot usikker på hvilken måte divisjon skal brukes for

å gi de svaret de er ute etter. Lily skjønner at divisjonsoperasjonen skal inneholde 5 og 45, men vet ikke hvilket tall som er dividend og hvilket som er divisor. Min medstudent, lærer G, veileder jentene videre i arbeidet i utdraget under.

Figur 4.1.1: Lily og Hedda bruker divisjon

Utdrag 2

Jentene setter programmeringsvariabelene *det første tallet* til fem og *det andre tallet* til 45. De bytter regnearten i verdiblokka for matematikk i *det tredje tallet* til divisjon. Det står nå altså at *det første tallet* er satt til fem, *det andre tallet* til 45 og *det tredje tallet* til *det første tallet* dividert på *det andre tallet*. Jentene trykker på knapp A+B og får 0,11 over skjermen.

Lily: Det ble ikke riktig..
Hedda: Men hvis vi tar..
Lærer G: Nei, hvorfor ble ikke det riktig?
Hedda: Hvis vi tar 45 på det første tallet, så deler vi det på fem i stedet.
Lærer G: Hvorfor tror du det vil fungere bedre?
Hedda: Fordi at… eh.. hvis vi har fem poser da, så kan vi, eller 45 poser, så kan man ikke dele fem godteri på alle da.

Lily setter *det første tallet* til 45 og *det andre tallet* til fem, trykker på knapp A+B og får ni på skjermen.

Lily: Ja, da blir det.. Det blir riktig.. Ja, da blir det rett..

Dette utdraget følger rett etter det første. Altså begynte jentene å endre på verdier for programmeringsvariablene og endre operasjon fra multiplikasjon til divisjon av seg selv. De setter først inn fem som verdi for *det første tallet* og 45 for *det andre tallet*. Deretter trykker de på multiplikasjonstegnet i koden for *det tredje tallet*. Her får de opp en meny med flere ulike matematiske symbol. Jentene velger divisjonstegnet som ser slik ut:÷. Ved å trykke på knapp A+B kjører programmet og micro:bit-en regner ut hva fem dividert på 45 er. Regnestykket gir de 0,11 i svar, som de raskt sier at ikke er riktig. Videre begrunner de dette for læreren ved hjelp av en forklaring som tar utgangspunkt i en posemodell. Deretter bytter de om på verdiene til programmeringsvariablene. Programmet kjøres og micro:bit-en gir 9 til svar. Dette sier de selv at er riktig svar.

4.2 Unngår å lage kode for å finne ukjent multiplikator

I møte med oppgaver der multiplikasjonsstykkene inneholder en ukjent multiplikator, må elevene skrive ny kode og endre på programmet for å finne en ny ukjent. Før elevene

møter disse oppgavene, har de løst multiplikasjonsstykker med ukjent produkt i MakeCode. I stedet for å lage en ny kode som finner en ukjent multiplikator, velger elevene ved flere anledninger løsningsstrategier der de unngår å endre programmet. Dette skjedde i arbeid med alle divisjonsoppgavene, altså oppgave 4, 6 og 7. I denne kategorien presenterer jeg strategier elevene hadde for å unngå å lage kode for å finne en ukjent multiplikator. Jeg deler strategiene inn i *endrer oppgavestrukturen til multiplikativ situasjon* og *endrer fra variabel til konstant verdi*.

4.2.1 Endrer oppgavestrukturen til multiplikativ situasjon

Elevene gjør om på multiplikasjonsstykker som i utgangspunktet har ukjent multiplikator, slik at de heller får ukjent produkt. Elevene setter inn multiplikanden som verdi for programmeringsvariabelen *det første tallet*. Deretter setter elevene inn verdien de tror multiplikatoren skal ha for *det andre tallet*, som egentlig er den ukjente. Det kjente produktet blir brukt til å kontrollere om elevene satt inn korrekt verdi for multiplikatoren. Ved å trykke på knapp A+B, blir *det første tallet* og *det andre tallet* multiplisert sammen, og elevene sjekker om tallet micro:bit-en viser er det samme som produktet i regnestykket.

Dette gjorde to av de tre parene med fokuselever. Begge parene endret oppgavestrukturen i arbeid med oppgave 4. Det siste elevparet rakk ikke å gjøre oppgave 4. I oppgaven før denne, oppgave 3, løste elever multiplikasjonsstykker med ukjent produkt. Ved å endre strukturen på stykkene de møter i oppgave 4, kan elevene fortsatt bruke programmet fra forrige oppgave. Kombinasjonen med at elevene har et ferdig program for ukjent produkt, og at tallene i multiplikasjonsstykkene er lave nok til at elevene kan ta i bruk automatiserte tallfakta, gjør at elevene lett kan gjøre om oppgavestrukturen.

Figur 4.2.1: Program for ukjent produkt

Under ser vi et utdrag fra et av fokuselevparene, Vilma og Andrine, sitt arbeid:

Utdrag 3

Vilma og Andrine jobber med oppgave 4, der de skal bruke programmet i MakeCode til å løse regnestykkene fra oppgave 1. Jentene har kommet til regnestykkene med ukjent multiplikator. Her skal de løse $8 \cdot \_ = 16$. Jentene setter først programmeringsvariablene til 8 og 2, og får 16 over skjermen ved å trykke på A+B.

Vilma: Jeg vet at det er 16 [mens de skriver inn verdier for programmeringsvariabler].

Andrine: Ja, jeg også. Ja, det er..

Jentene går videre til neste neste regnestykke, som er $5 \cdot \_ = 45$.

Vilma: Fem ganger ni er lik 45.

Andrine: Er det?

Vilma: Det er jeg ganske sikker på faktisk!

Andrine: [Setter programmeringsvariablene til fem og ni, trykker A+B og får 45 over skjermen] 45!

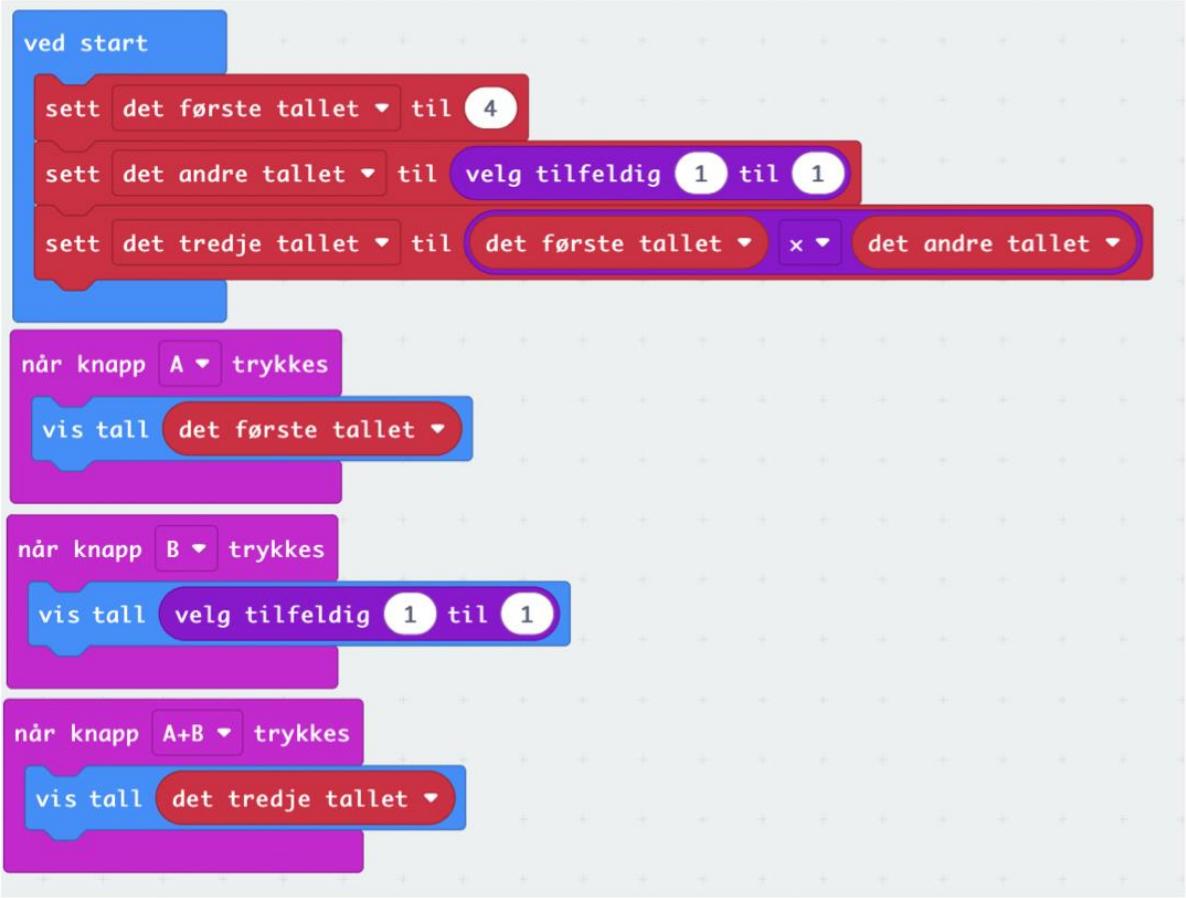
Det kan virke som at jentene bruker automatisert tallfaktakunnskap i arbeid med det første regnestykket. De «vet» at den ukjente skal være to og setter det inn som verdien til *det andre tallet*. Når de møter neste regnestykke, er de litt mer usikre på hvilket tall som den ukjente multiplikatoren er. De setter likevel inn det de ganske sikkert tror at er riktig, og sjekker ved å få programmet til å multiplisere sammen multiplikand og multiplikator. Her ser vi at jentene bruker programmet som om det var *det tredje tallet* som var ukjent. Ved å endre oppgavestrukturen slik, går multiplikasjonsstykket fra å ha et behov for divisjon til å få et behov for multiplikasjon.

4.2.2 Endrer fra variabel til konstant verdi

I andre økt av undervisningsopplegget er programmeringsvariabelen *det andre tallet* satt til å være et tilfeldig tall, som skulle bidra til å gjøre den til en ukjent variabel i programmet. I og med at programmeringsvariabelen blir satt til et nytt tilfeldig tall hver gang programmet kjøres, bidrar det også til at elevene må lage en generell kode, som alltid finner ut hva *det andre tallet* er.

Et av elevparene, Johannes og Knut, utnyttet potensialet i verdiblokka for tilfeldig tall, som gjorde at de unngikk behovet for å lage en kode for å finne en ukjent multiplikator. Under ser vi et utdrag fra guttenes arbeid med oppgave 7, der de tilordnet en konstant verdi til programmeringsvariabelen *det andre tallet*.

Figur 4.2.2: Knut og Johannes sitt program for ukjent multiplikator



Utdrag 4

Guttene endrer på verdiblokka som velger tilfeldig tall. De bytter ti-tallet med et ett-tall, slik at programmeringsvariabelen *det andre tallet* nå er satt til verdien å velge et tilfeldig tall fra én til én.

Knut: Jeg er smart. Jeg har funnet ut noe.

Han drar ut en ny verdiblokk for å velge tilfeldig tall, og setter den inn i handlingsblokka for å vise tall i startblokka for knapp B. Også denne verdiblokka blir satt til å velge et tilfeldig tall fra én til én. Guttene har nå satt *det andre tallet* til et tilfeldig tall mellom én og én, og sagt at micro:bit-en skal vise et tilfeldig tall mellom én og én når knapp B trykkes.

Guttene manipulerer verdiblokka til å velge samme tall hver gang, slik at programmeringsvariabelen *det andre tallet* blir en konstant med en kjent verdi. De ber så programmet om å vise et tilfeldig tall fra én til én når knapp B trykkes. Dette er for så vidt samme verdi som de har gitt *det andre tallet*, men guttene har ikke brukt programmeringsvariabelen for å vise tallet på micro:bit-en. Koden for å vise tall og koden som definerer *det andre tallet* har slik ingen sammenheng. Guttene har ikke laget en ny kode for å finne den ukjente multiplikatoren, men har heller gitt den ukjente variabelen en kjent konstant verdi, og skrevet inn samme tall i handlingsblokka for å vise tall. Etter utdraget over kommer jeg bort til guttene og utfordrer dem til å finne en kode der verdiblokka for tilfeldig tall fortsatt er fra én til ti. Guttene er usikre og samtalen leder ikke fram til andre strategier.

4.3 Bruker egne matematikkunnskaper for å gi mening til programmet

Elevene brukte matematikkunnskaper for å forklare det de programmerte undervegs i arbeidet med alle oppgavene. Noen ganger fordi de virket å ha mistillit til programmet, andre ganger fordi de så programmet i sammenheng med kjente matematikkunnskaper og andre representasjoner for matematikk. Jeg har delt inn i underkategoriene *sjekker at programmet regner riktig*, *bruker automatiserte tallfakta for å gi mening til programmet* og *gjør et registerskifte for å gi mening til programmet.*

4.3.1 Sjekker at programmet regner riktig

Vilma og Andrine kom i en situasjon der de ikke var sikre på om programmet de hadde laget i MakeCode regnet riktig. Det var i første økta i arbeid med oppgave 3, der elevene skulle regne ut tre multiplikasjonsstykker ved hjelp av programmet i MakeCode. Jentene har regnet de to første stykkene uten problemer, men i det programmet gir jentene svaret på det tredje regnestykket, oppstår tvil.

Figur 4.3.1: Program som skaper usikkerhet for Vilma og Andrine

Utdrag 5

Vilma: Hva ble det?
Andrine: Det er.. 42. [mens hun trykker knapp A+B]
Vilma: Hæ, 42?
Andrine: Det er det som står her. [mens hun trykker knapp A+B igjen]
Lærer G: Stoler dere på den?
Vilma: Ja, vi stoler på den.
Andrine: Vent, jeg må regne.
Vilma: Det er jo riktig, fordi at..
Andrine: Ja, det er riktig..
Vilma: Det må jo være riktig..
Lærer G: Fordi?
Vilma: Fordi vi har gjort det sånn som man skal gjøre det.
Lærer E: Regner kalkulatoren riktig?
Vilma: Vet ikke.
Lærer E: Hva er det dere er usikre på da?

Vilma: Fordi at det høres litt mye ut.
Andrine: Ja, for de her, de ble jo også riktige.
Vilma: Ja, de ble det..
Lærer E: Kan dere gjøre syv ganger seks på ark da, for å sjekke om det er riktig?
Andrine: Vi kan gjøre det her.
Vilma: Okei, da blir det syv pluss syv, 14.
Lærer G: Kjenner dere til noen stykker som er ganske nært syv ganger seks som dere vet i hodet?
Vilma: Syv ganger fem.
Lærer G: Hva er det da?
Vilma: Det vet vi ikke.
Andrine: 5, 10, 15, 20, 25, 30, 35.
Vilma: 35. 35 + 7. 35 + 7.. Det blir… det var riktig!

Andrine og Vilma synes at 42 hørtes ut som at var litt for mye til å være svaret på $7 \cdot 6$. Etter forslag fra meg begynner elevene å kontrollregne på ark. De starter med gjentatt addisjon, men bytter til en mer effektiv strategi etter tips fra min medstudent. De tar utgangspunkt i $7 \cdot 5$, og ramser opp de syv første tallene av fem-gangen. Det blir 35, og jentene adderer deretter en siste syver. Svaret de fikk, 42, stemte overens med MakeCode sitt. Vilma og Andrine tar i bruk gjentatt addisjon og kunnskaper om gangetabellen for å kontrollere at programmet også regner riktig.

4.3.2 Bruker automatiserte tallfakta for å gi mening til programmet

Denne underkategorien skiller seg fra forrige ved at elevene her ikke virker å ha mistillit til svaret de får fra programmet. Elevene her gjør heller oppdagelser om at det de gjorde i MakeCode stemmer med tallfakta de kjenner til fra før. Dette skjedde i flere situasjoner i arbeid med divisjonsoppgavene, oppgave 4, 6 og 7, for begge jenteparene. Et eksempel på dette er Hedda og Lily sitt arbeid med oppgave 6.

Utdrag 6

Hedda: [Leser opp oppgaveteksten] Hvis skjermen viser 20 når du trykker på knapp A+B, og fire når du trykker på knapp A, hva viser skjermen når du trykker på knapp B.
Lily: Okei, da bare sjekker jeg det. Vi bare må få tak i 20. [Mens hun trykker flere ganger på knapp A+B.]
[...]
Lily: [Trykker knapp A+B igjen, og 20 vises på skjermen] Ja, vi fikk det til!
Lily: [Trykker på knapp B, og fem kommer på skjermen] Fem! Men det er ikke så rart da, for fire ganger fem er jo 20. Ja, bare skriv du at vi tenkte fire ganger fem er lik «twenty».

Her løser jentene oppgaven ved å kjøre programmet helt til de når verdien for trykk på knapp A+B de er ute etter, 20. Siden programmet setter programmeringsvariabelen *det andre tallet* til et nytt tilfeldig tall hver gang programmet kjøres, må jentene trykke flere ganger. Når de til slutt får 20, trykker de på knapp B for å se hvilken verdi *det andre tallet* har. Lily kommenterer at det ikke var uventet at fem var det ukjente tallet, siden fire multiplisert med fem er 20. Hun nevnte ikke dette undervegs i arbeidet med

oppgaven, men kommenterte til slutt at programmet kunne kobles opp mot automatiserte tallfakta hun hadde.

4.3.3 Gjør et registerskifte for å gi mening til programmet

Elevene får i flere av oppgavene multiplikasjonsstykker oppgitt i symbolform på et ark. Disse overfører elevene uten problem til programmet i MakeCode. I erfaringsoppgavene der elevene skulle analysere et ferdig program, fikk alle til å oversette programmet til regnestykker representert skriftlig ved symboler og muntlig i samtale med meg.

I arbeid med oppgave 4 trekker Lily og Hedda inn en form for posemodell, forklart med muntlig naturlig språk, for å forklare det de erfarer i MakeCode. I utdrag 2 så vi at Hedda bruker en posemodell for å forklare hvorfor svaret de fikk fra micro:bit-en ikke var det de var ute etter. Lily og Hedda ville prøve divisjon for å finne den ukjente multiplikatoren i regnestykket $5 \cdot \_ = 45$, men var usikre på hvilket tall som skulle stå som dividend og hvilket som var divisor. De prøvde å dividere fem på 45 og fikk 0,11, som de så var feil. For å forklare hvorfor det er feil, forklarer Hedda med muntlig naturlig språk at divisjonsoperasjonen kan bli representert ved å dele ut godteri i poser. Hun ser det de gjør i MakeCode i sammenheng med et annet kjent *register* for matematikk. Register er Duval (2006) sitt begrep for et representasjonssystem. Kilhamn et al. (2022) ser programmering som et eget register. Når Hedda konverterer divisjonssituasjonen fra blokkprogrammering til en muntlig form for posemodell, flytter hun det matematiske objektet over i et kjent register.

4.4 Gjør seg kjent med funksjoner i MakeCode

Elevene ble kjent med programmeringsspråket undervegs i arbeidet deres med oppgavene. MakeCode inneholder mange blokker og funksjoner, som elevene ble bedre kjent med bruksområdene til mens de løste oppgavene. Dette så jeg tegn på i elevenes arbeid med alle oppgavene. Jeg deler inn i underkategoriene *lager egne oppgaver for å teste programmet*, *oppdager snarveger* og *oppdager at programmeringsvariabler må defineres*.

4.4.1 Lager egne oppgaver for å teste programmet

Alle fokuselevene testet ut egne multiplikasjonsstykker i de ferdige programmene i undervisningsopplegget. Dette skjedde i begge øktene. Under ser vi et utdrag av Vilma og Andrine sin egenlagde utvidelse av oppgave 5. De testet hovedsakelig ut store tall og var systematiske i arbeidet sitt, ved at de skrev ned stykkene de prøvde ut og svarene de fikk på et ark.

Utdrag 7
Vilma og Andrine har laget koden for oppgave 5. I stedet for å lage en kode for å finne den ukjente multiplikatoren, velger jentene å teste programmet. De har i dette utdraget endret verdiblokka som velger et tilfeldig tall, som programmeringsvariabelen *det andre tallet* er satt til. *Det andre tallet* er nå et tilfeldig tall mellom 123456 og 234567. Videre diskuterer de hvilke andre endringer de skal gjøre før de kjører programmet.

Andrine: Skriv det der da.
Vilma: Okei, bare si tallene.

Andrine: Nei, *det første tallet* kan jo ikke være fire, det må være.. [Setter *det første tallet* til mange tilfeldige siffer etter hverandre.]

Figur 4.4.1: Vilma og Andrine sin egenlagde oppgave

Jentene har nå laget programmet slik som på bildet over. De trykker på knapp A+B, og *det første tallet* og *det andre tallet* blir multiplisert sammen. Begge leser opp tallet som ruller over skjermen. På grunn av størrelsen på tallet, blir det skrevet på standardform. Skjermen på micro:bit-en viser da et desimaltall, etterfulgt av bokstaven *e* og et tall med plusstegn eller minustegn foran. Multiplikasjonsstykket blir ifølge micro:bit 1,97105443e+24. Jentene har nå akkurat lest høyt svaret som rullet over skjermen. De leste *e* som «9, nei, 6!» og reagerer kun på plusstegnet.

Andrine og **Vilma**: *ler*
Vilma: Og så er det et plusstegn *ler* Hvordan? *ler*
Andrine: *ler* Jeg vet ikke, det var det som stod. Hvordan kom det?
Vilma: Se da! Se da! [Trykker på nytt og leser opp tallene i kor med Andrine.]
Andrine: Se om den er med da, pluss 24, se på slutten!
Vilma: Verdens vanskeligste regnestykke noensinne.

Jentene tester hva som skjer dersom de prøver å bruke programmet for store tall. Alle fokuselevene gjorde liknende testing av programmet som vist i utdraget. Først gjorde elevene oppgavene slik de stod på arket, deretter utfordret de programmet med å teste egenproduserte regnestykker. I noen tilfeller, slik som over, utforsket elevene programmet der de stod fast i oppgaveløsningen. Elevene lekte seg med å teste om programmet de hadde laget ville fungere for veldig store tall eller for negative tall.

4.4.2 Oppdager snarveger

Elevene koblet stort sett sammen blokker enkelt. Handlingsblokker kan settes sammen med hverandre eller inn i startblokker, ettersom de har samme spor. Verdiblokker kan settes inn i handlingsblokker som har verdifelt med samme ovale form som verdiblokkene. Handlingsblokkene for variabler vises kun for variabelen som sist ble laget. For å ta i bruk disse for andre variabler enn den som sist ble laget, må elevene trykke på en hvit pil på handlingsblokka som åpner en meny for alle variabler. Her kan elevene velge den variabelen som de skal sette eller endre verdi for. Denne *pilmenyen* finnes også i andre blokker, som gjør at det blir enklere å endre koden. I startblokkene for de ulike knappene kan man ved pilmenyen gjøre startblokka om til en startblokk for en annen knapp uten å slette og hente ny blokk. Det samme gjelder for matematikkblokkene for regneoperasjoner. Her kan elevene ved å bruke pilmenyen

endre fra for eksempel multiplikasjon til divisjon, uten å bytte ut hele blokka. Denne nyttige funksjonen utforsker Knut og Johannes i oppgave 2.

Utdrag 8

Her har Johannes allerede laget tre programmeringsvariabler, *det første tallet*, *andre tallet* og *det tredje tallet*. Nå styrer Knut Chromebooken og skal sette inn koder for å gi programmeringsvariablene en verdi ved start. Han trenger da handlingsblokker for å sette verdien til hver av variablene.

Knut: Og så skal vi ha enda mer variabler. Som heter *det andre*..

Johannes: Ja, vi har jo det!

Knut: Nei, vi kan ikke sett.. sette det eller endre det.

Johannes: Se!

Knut: Nei, se du kan ikke sette det eller endre det.

Johannes: Jo, se her. Du kan for eksempel gjøre sånn her! [Drar en verdiblokk av programmeringsvariabelen *andre tallet* og holder den over *det første tallet* i sett-blokka. Ingenting skjer.]

Figur 4.4.2: Johannes sitt forsøk på å sette sammen blokker

Knut lager etter dette en ny programmeringsvariabel, *det første talet*, for å få tilgang til handlingsblokker for denne variabelen.

Knut: Og så må du ta *det tredje*. Det er fire. Så må du sette *det tredje tallet* til *det første tallet* og *det andre tallet*.

Johannes: Variabler.. Sett.. [Trykker inn i menyen og drar ut en endre-blokk]

Knut: Nei..

Johannes: Du trenger ikke å skrive, se du kan bare gjøre sånn her! [Trykker på pilmenyen i endre-blokka og får opp lista over alle variablene de har laget]

Knut: Det er sant. Og så må du ta sett, ikke endre.

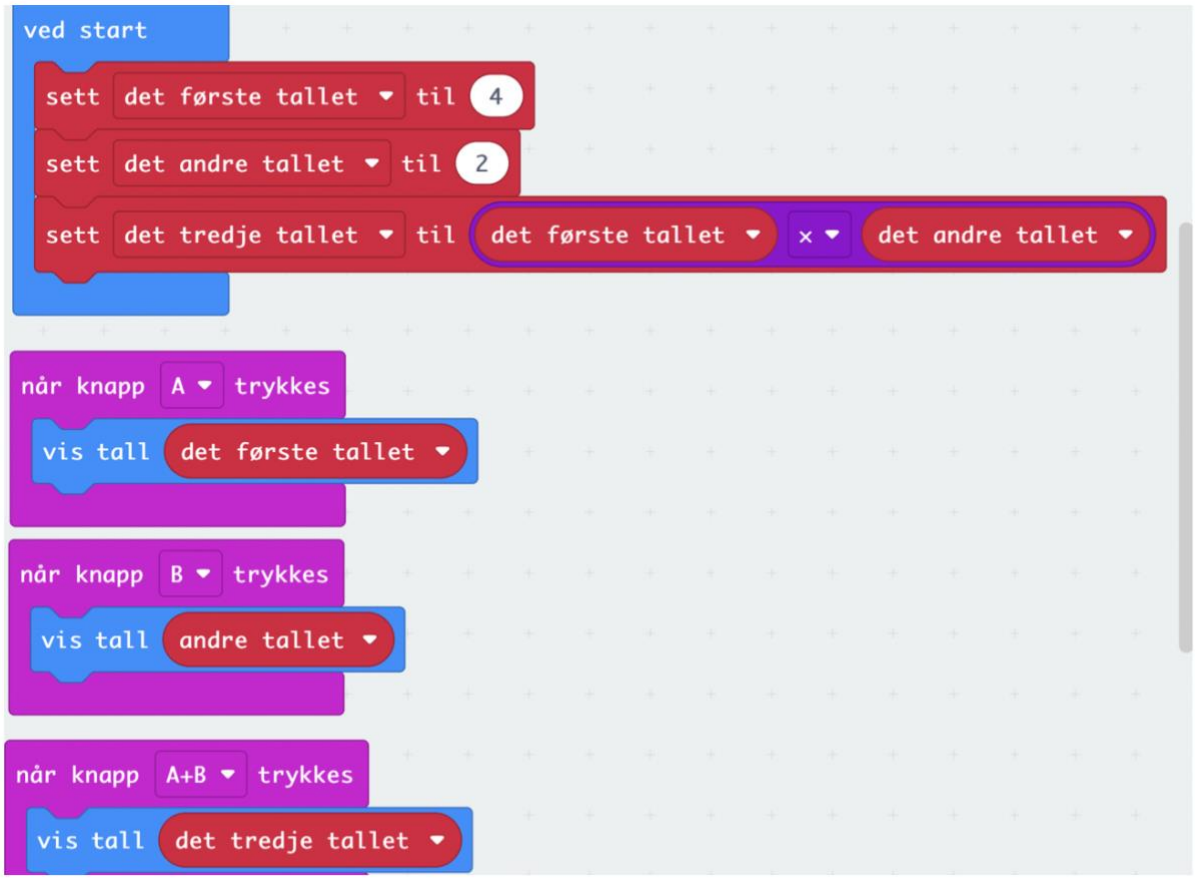
Figur 4.4.3: Pilmenyen for programmeringsvariabler

Pilmenyen finnes som nevnt for flere blokker. Dette kjente elevene til i varierende grad. Lily og Hedda brukte pilmenyen ved flere anledninger, noe som effektiviserte programmeringen deres. De slapp å finne nye blokker hver gang endringer skulle gjøres, og brukte i stedet pilmenyen til å gjøre endringer i startblokker og matematikkblokker. Guttene kjenner i starten av opplegget ikke til pilmenyen. Dette fører til at de, som allerede har laget alle tre variablene programmet krever, begynner å lage dem på nytt, slik at de får to versjoner med (nesten) samme navn for hver av de tre programmeringsvariablene. Johannes er inne på at de ikke trenger å lage variablene på nytt ved at han sier at de jo allerede har laget variablene. Han prøver å sette en verdiblokk av en programmeringsvariabel inn i handlingsblokka for en annen programmeringsvariabel. Som nevnt kan elever sette ovale verdiblokker inn i ovale verdifelt i handlingsblokker. Johannes prøver å sette en oval verdiblokk inn i et firkantet felt for variabelnavn. Blokka blir grå, som betyr at den er inaktiv, og at den må settes inn i programmet på en annen måte for å være gjeldende. Litt senere ser Johannes at de kan velge hvilken som helst variabel ved hjelp av pilmenyen, og viser Knut. Etter dette går bruken av variabelblokker lettere, men det doble settet med programmeringsvariabler gir dem utfordringer videre i arbeidet deres, som er vist i neste underkategori.

4.4.3 Oppdager at programmeringsvariabler må defineres

Programmeringsvariabler var relativt nytt for elevene da de gjennomførte opplegget i studien min. I noen situasjoner blir det tydelig at elevene er usikre på hvordan de fungerer. Da Johannes og Knut skulle lage programmet i den første økta, støtte de på utfordringer fordi de brukte udefinerte variabler. Det samme gjorde Vilma og Andrine i arbeid med oppgave 7. Elevenes arbeid er vist under.

Figur 4.4.4: Bruker udefinerte programmeringsvariabler



Utdrag 9

Guttene har laget ferdig programmet på bildet i oppgaveteksten. De skal nå teste det. Først trykker de på knapp A og får to på skjermen.

Johannes: To. Det er deilig.
[Trykker på knapp B og får null]
Johannes: Hæ? Hvorfor null?
Knut: Hm..
Johannes: Det står null.
Knut: Ja, det står null.. Hvorfor står det null?
Johannes: Jeg vet ikke..
...
Knut: Åja, det er fordi vi har tatt bare *andre tallet*. Her står det *det andre tallet*, her er bare *andre tallet*.
Johannes: Åjaa! [Sletter blokka for *andre tallet* og henter en *det andre tallet*-blokk, som de setter inn på samme plass]
Johannes: *Det* står det. *Det andre tallet*.

Guttene hadde etter misforståelsen vist i utdrag 8, laget flere programmeringsvariabler med nesten samme navn. Det endte med at de satte verdi til variabelen *det andre tallet*, men ba micro:bit-en vise verdien til *andre tallet*. Guttene definerte én programmeringsvariabel og ba om å vise verdien til en annen, som da er udefinert. Resultatet blir at micro:bit-en viser null, siden den viser verdien til en ikke-eksisterende programmeringsvariabel.

Vilma og Andrine fikk samme opplevelse da de prøvde å bruke udefinerte variabler i oppgave 7. Der skulle elevene lage en ny kode, slik at programmet kunne finne en ukjent multiplikator. Her prøvde jentene å endre koden i programmet ved å bruke programmeringsvariablene, men uten å begrunne endringene i matematikk.

Utdrag 10

Vilma: Kanskje vi kan gi *det andre tallet* et nytt navn?
Andrine: Åå?
Vilma: "Uten å bruke blokka som heter *det andre tallet*". Vi kan fortsatt bruke andre blokker som ikke heter *det andre tallet*.
Andrine: Det fjerde.. [Trykker seg inn i menyen og lager en ny variabel. Skriver inn navnet *det 4 tallet* for den nye variabelen.]
Vilma: Det fjerde tallet!
Vilma: Eh, fikk du.. tok du det samme tallet.. samme greia?
Andrine: [Setter en blokk for *det 4 tallet* inn i handlingsblokka for å vise tall under startblokka for når knapp B trykkes] Sånn! La oss teste den på nytt.
[...]
Johannes: Dere har gjort feil!
Vilma: Nei.. *leser oppgaveteksten*
Johannes: Den finnes ikke!

Her utforsker jentene om de kan gi programmeringsvariabelen et nytt navn eller om de kan lage en ny programmeringsvariabel. Det virker som at Vilma tenker de kan være lure ved å gi et nytt navn til variabelen, slik at de ikke trenger å lage en ny kode, men samtidig ikke bruker en variabel med navn *det andre tallet*. Hun foreslår å lage nytt navn og spør Andrine om hun tok «samme tallet.. samme greia». Andrine lager i stedet en ny variabel, *det 4 tallet*, men som Johannes kommenterer: «den finnes ikke». Jentene har ikke gitt den nye variabelen en verdi. Johannes har selv erfart behovet for at variabler må defineres, i utdrag 9, og ser her at jentene har møtt på samme utfordring.

4.5 Ser ikke muligheter i MakeCode

Noen ganger så elevene rett og slett ikke at en oppgave eller spesifikk handling var mulig å gjennomføre i MakeCode. Dette var tydelig i oppgave 6, der elevene kunne velge om de ville løse oppgaven ved å lage et program i MakeCode eller om de heller ville bruke en annen strategi. Her ville to av elevparene prøve ut noe i MakeCode, men som de ikke så at var mulig å gjennomføre. Situasjonene er noe ulike, og jeg deler i underkategoriene *får ikke gjort det de vil i MakeCode* og *løser oppgaven uten MakeCode*.

4.5.1 Får ikke gjort det de vil i MakeCode

Lily og Hedda valgte å løse oppgave 6 i MakeCode. De bruker programmet som de lagde i oppgave 5. Her er programmeringsvariabelen *det andre tallet* satt til et tilfeldig tall mellom én og ti. Hver gang programmet kjøres, blir et tilfeldig tall valgt som verdi for programmeringsvariabelen. For å bruke programmet fra oppgave 5 til denne oppgaven, kunne elevene med fordel gjort endringer for å effektivisere oppgaveløsningen. Dersom de bruker programmet slik det er, vil det fungere, men elevene må kjøre programmet på

nytt helt til de tilfeldigvis får de tallene de trenger. Her er et utdrag av hvordan Hedda og Lily jobbet:

Utdrag 11
Jentene jobber med andre deloppgave.

Hedda: [Leser oppgaveteksten] "Hvis skjermen viser 20 når du trykker på knapp A+B, og fire når du trykker på knapp A, hva viser skjermen når du trykker på knapp B?"

Lily: Okei, da bare sjekker jeg det. Vi bare må få tak i 20. [Trykker på knapp A+B flere ganger, slik at programmet kjøres på nytt og velger nye verdier for *det andre tallet*]

[...]

Hedda: Er det ikke enklere å bare skrive det der?

Lily: Nei, det går ikke. Det er akkurat det som ikke går. Fire, æsjebæsj. 36, nei.

Det virker som at Hedda tenker at de kan løse oppgaven mer effektivt ved å endre på koden. Hun sier «er det ikke enklere å bare skrive det der?». Lily svarer at det ikke går. Akkurat *hva* Hedda tenker å skrive *hvor*, kan jeg ikke være sikker på. Det kan tenkes at Hedda ville fylle inn tallet de lette etter, for å effektivisere. Likevel er dette noe som ikke kan konkluderes med. Jentene velger uansett å kjøre koden mange ganger, til de tilfeldigvis treffer tallet de leter etter, i stedet for å lage en kode der den oppgitte informasjonen kunne blitt satt inn. De prøver og feiler til de treffer riktig tall.

4.5.2 Løser oppgaven uten MakeCode

I arbeid med samme oppgave som jentene over, bruker Knut og Johannes matematiske løsninger uten MakeCode. De ville ta i bruk programmet, men så ikke hvordan det skulle gjøres. Her jobber guttene med første del av oppgave 6, som skal sees i sammenheng med programmet i oppgave 5. De har oppgitt at *det første tallet* er fire og *det tredje tallet* er 12. Oppgaven er å finne ut hva *det andre tallet* er.

Utdrag 12

Johannes: 12, fire?… Det gir ikke så mye mening, 12 og fire.. Går det ikke an å bare endre her da, til 12?

Knut: Nei, Johannes. Se her. *Det første tallet* er fire, og så skal det være.. *Det tredje tallet* skal være *det første tallet* pluss *det andre tallet*, og så skal vi gjøre det til 12.

Johannes vil først prøve å bruke MakeCode til å løse oppgaven, og sier «går det ikke an å bare endre her da, til 12?». Han legger det fra seg etter at Knut sier «nei». Etter dette snakker ikke guttene mer om å bruke programmering. De så det ikke som mulig å endre til 12 i programmet, slik Johannes foreslo. Videre bruker guttene gjentatt addisjon, forskjellige multiplikasjonsstykker og telling. Det er ikke så lett å henge med på resonnementene deres, og det virker som at guttene hopper mellom å forstå operasjonen i oppgaven som multiplikasjon og addisjon. Guttene virker usikre på formuleringen av oppgaven og omformulerer ikke informasjonen til et konkret multiplikasjonsstykke eller annet regnestykke muntlig. I feltet for oppgaveløsning på

arket skrev guttene $3 + 3 + 3 + 3 = 12$. De fant dermed et regnestykke som passet med oppgaveteksten, men guttene viser ikke tydelig hva de mener er svaret på oppgaven.

5 Diskusjon

Fokuset mitt for denne studien har vært å undersøke hvordan blokkprogrammering fungerer som et verktøy i arbeid med sammenhengen mellom multiplikasjon og divisjon. Jeg har utformet et undervisningsopplegg for dette, som er gjennomført for et 4. trinn. I studien min fokuserer jeg på hvordan seks fokuselever jobbet med oppgavene, og ser på hvordan de bruker blokkprogrammeringsverktøyet micro:bit med det blokkbaserte programmeringsspråket MakeCode i arbeid med oppgavene.

Undervisningsopplegget inneholdt erfaringsoppgaver og divisjonsoppgaver. I erfaringsoppgavene jobbet elevene med multiplikasjonsstykker av hver oppgavestruktur på ark eller lagde program i MakeCode for å bli kjent med aktuelle blokker. Jeg fokuserer mest på divisjonsoppgavene, der elevene jobbet med multiplikasjonsstykker med ukjent multiplikator i MakeCode. Ifølge Van de Walle et al. (2020, s. 197) gir ukjent multiplikator et behov for divisjon. Oppgaver med denne strukturen kan likevel løses ved hjelp av andre strategier også. I oppgave 1, der elevene skulle løse multiplikasjonsstykker av ulik oppgavestruktur på ark, viste det seg at gjentatt addisjon er den ledende strategien blant elevene på trinnet for å finne ukjent multiplikator. Dette er ifølge Rønning og Burheim (2020) en strategi innen den primitive modellen kalt *like grupper*. Rønning og Burheim kommenterer at det er en fordel for elever å kjenne til flere modeller for multiplikasjon, for å lettere kjenne igjen situasjoner der regnearten kan brukes. De nevner også divisjon som en modell for multiplikasjon. Før gjennomføring av undervisningsopplegget hadde jeg en hypotese om at elevene ville gå bort fra gjentatt addisjon i arbeid med blokkprogrammering. Dette tenkte jeg fordi det ville krevd mange blokker for å addere flere tall, og fordi det ikke finnes noen enkel måte for micro:bit-en å telle hvor mange ganger et tall er lagt sammen. På denne måten var det en sjanse for at elevene ville se etter andre mulige løsninger enn sin vante løsningsstrategi, gjentatt addisjon, i arbeid med oppgaver med ukjent multiplikator i MakeCode.

Jeg vil se funnene for forskningsfokuset mitt opp mot Rabardels teori om instrumentell skapelse. Et instrument er et begrep for når en artefakt er utviklet til å ha en funksjoll verdi for en elev, og består av en artefaktsdel og en skjemadel (Rabardel & Samurcay, 2001). De to delene viser til at utvikling må skje både i artefakten og i elevens mentale skjema, for å skape et instrument. Disse to prosessene kalles instrumentalisering og instrumentering (Rabardel & Samurcay, 2001). Jeg vil se etter instrumentalisering, at elevene utforsker artefakten, og instrumentering, der elevene blir formet av artefakten i måten de jobber på.

5.1 Valg av løsningsstrategi

Instrumenteringsprosessen er synlig ved at arbeidsmåtene til elevene er tilpasset artefakten. Elevene blir formet av artefakten. Hver elev har mentale skjema for hvordan artefakten kan brukes og mentale skjema for hvordan spesifikke oppgaver kan løses. I instrumenteringsprosessen er disse skjemaene i bruk og blir samtidig utvidet ved ny erfaring. Målet for opplegget var at elevene skulle bruke divisjon, som jeg forsøkte å skape et behov for ved å gi elevene multiplikasjonsstykker med ukjent multiplikator. Da

elevene skulle lage en kode for å finne den ukjente multiplikatoren, ble det tydelig at elevene ikke har så mange strategier å ty til fra skjemaene sine.

Det var i løpet av de to øktene bare to forsøk på å lage en kode for å finne den ukjente multiplikatoren ved bruk av blokkprogrammering som også var matematisk begrunnet. Det første var Lily og Hedda sitt, som ble presentert i utdrag 1 i resultatkapittelet. Disse er de eneste som tok i bruk divisjon av fokuselevene. Jentene kom muntlig fram til at de kunne bruke divisjon for å finne multiplikatoren, men var usikre på hvilket av de oppgitte tallene som skulle deles på det andre. De brukte blokkprogrammering som et verktøy for å finne en ukjent multiplikator, og tok også i bruk divisjon slik jeg ønsket. Jentene bruker mest naturlig språk som medierende redskap i denne situasjonen, og blokkprogrammering som et redskap for å finne riktig rekkefølge til tallene i divisjonsstykket. Det kan virke som at Lily og Hedda visste om sammenhengen mellom regneartene, siden de raskt foreslo divisjon som strategi og trengte heller ikke å utforske seg fram til det. I oppgavene med ukjent multiplikator på ark brukte de gjentatt addisjon og automatiserte tallfakta, og ikke divisjon. Jentene kjenner kanskje til flere strategier som de veksler mellom etter hvilken de tenker passer best til regnestykket. Det kan være at de er i en overgangsfase mellom å ha en primitiv modell for multiplikasjon, og holder på å forstå at divisjon også kan brukes, og derfor bare bruker divisjon i enkelte tilfeller. Det kan også tenkes at de ikke så gjentatt addisjon som en mulig strategi, siden det ikke finnes blokker for å telle antall ganger et tall er addert. Dette er en del av den sosiale dimensjonen av mentale skjema, der utviklerne bak MakeCode legger begrensninger for elevene gjennom hvilke blokker som finnes.

Hedda gjør et registerskifte til en regnefortelling med posemodell, gjennom et naturlig muntlig språk, i utdrag 2. Hun forklarer at det ikke går å dele fem godteri til 45 poser, og mener derfor kanskje at 0,11 ikke kan være riktig siden det er et desimaltall. Registerskiftet tyder på at Hedda ikke ser det som skjedde i programmet som isolert til blokkprogrammering, men at hun kunne identifisere det matematiske objektet i det de gjorde. Dette er presentert i utdrag 2, og er et av flere eksempler på at elevene brukte kjent matematikk i sammenheng med det de gjorde i MakeCode. Begge jenteparene så ved flere anledninger regnestykker i MakeCode i sammenheng med automatiserte tallfakta de hadde. Noen ganger bare som en kommentar om at MakeCode har riktig, fordi det er det samme som et multiplikasjonsstykke de «vet» svaret på, som vist i utdrag 6. Andre ganger førte det til at de endret hele strukturen på oppgaven, slik at de unngikk behovet for å lage en kode for en ukjent multiplikator. I utdrag 3 under *4.2 Unngår å lage kode for ukjent multiplikator*, ser vi at Vilma og Andrine sin bruk av automatiserte tallfakta gjør at oppgavene heller får ukjent produkt enn ukjent multiplikator. De har slik ikke et behov for divisjon lenger.

Instrumentering handler om at artefakten styrer hvordan eleven velger å løse en oppgave. Knut og Johannes valgte i oppgavene på ark å finne ukjent multiplikator ved å bruke gjentatt addisjon. I arbeid med oppgaver med ukjent multiplikator i MakeCode, valgte de en helt annen strategi. Som vist i utdrag 4 under kategorien *4.2 Unngår å lage kode for å finne ukjent multiplikator*, gav guttene den ukjente variabelen en kjent konstant verdi, slik at den ikke var ukjent eller hadde variabel verdi lenger. Dette gjorde at de unngikk behovet for å finne den ukjente multiplikatoren ved hjelp av å lage en kode. Denne strategien er i stor grad styrt av artefakten, og slik et synlig tegn på instrumentering. I oppgaver der guttene skulle funnet en ukjent multiplikator på ark, ville

det ikke gitt mening å bare sette inn et tilfeldig konstant tall for den ukjente. Det må være et tall som multiplisert med den kjente multiplikanden, blir det kjente produktet. I programmet elevene brukte i MakeCode var produktet, programmeringsvariabelen *det tredje tallet*, satt til multiplikand multiplisert med multiplikator, *det første tallet* multiplisert med *det andre tallet*. Da Knut satt verdien til multiplikatoren til et fast tall, ble også produktet konstant. Slik vil verdien de gir multiplikatoren automatisk passe inn med multiplikand og produkt. Strategien til Knut og Johannes gir mening i MakeCode, men ville ikke vært mulig ved regning på ark.

Ved å se funnene mine i lys av instrumenteringsprosessen, ser jeg at elevene velger arbeidsmåter som er tilpasset artefakten, men som også er tilpasset det som er kjente matematikkunnskaper for dem. Elevene ser det de gjør i MakeCode i sammenheng med andre register, automatiserte tallfakta og kjente strategier. Ved flere anledninger vinklet elevene oppgavene slik at de heller kunne bruke strategier som var kjent for dem, enn å utforske nye strategier. Dette førte til at oppgavene ikke inneholdt et behov for divisjon lenger, og hensikten med oppgaven ble borte. Utenom Hedda og Lily, var det ingen av fokuselevene som prøvde å ta i bruk divisjon. De endret heller på oppgavestrukturen slik at den passet med kjente matematikkunnskaper, eller så stoppet de helt opp.

Elevene hadde lite matematisk utforsking i MakeCode, og stod heller fast der matematikken var ukjent for dem. Det kan tenkes at det var enklere for elevene å regne multiplikasjonsstykker med ukjent produkt siden programmet i oppgaven hadde dette oppsettet. Likevel, det var få forsøk på å endre på programmet og å lage koder som kunne finne en ukjent multiplikator. Ifølge (Trouche, 2003) har elever instrumenterte handlingsskjema for hvordan de skal løse spesifikke oppgaver, som for å finne en ukjent multiplikator. Dette skjemaet er sammensatt av elevens bruksskjema for artefakten og aktuelle matematikkferdigheter for oppgaven. Det kan altså være lite kjennskap til MakeCode eller lite kjennskap til matematikken i oppgaven som er grunnen til at det stoppet opp for elevene. Som oppgave 1 viste er gjentatt addisjon den klart mest brukte strategien for å finne ukjent multiplikator blant elevene på trinnet, også for fokuselevene. Det kan tyde på at elevene hadde få strategier for multiplikasjonsstykker med denne oppgavestrukturen, og dermed manglet matematikkferdigheter for å kunne løse en slik oppgave. Elevene forsøkte derimot ikke å ta i bruk gjentatt addisjon i MakeCode. Kanskje så de det som umulig å bruke gjentatt addisjon i MakeCode når det ikke er en egen blokk for det? Eller kanskje oppgaveformuleringen gjorde at elevene tenkte annerledes?

5.2 Utforskingens rolle

Utdanningsdirektoratet (2019) løfter fram at programmering gir mulighet for å jobbe utforskende. Som nevnt over, utforsket elevene generelt mindre enn forventet i arbeid med undervisningsopplegget mitt. Med dette mener jeg at elevene i liten grad forsøkte å lage koder som kunne finne en ukjent multiplikator. Det var lite *matematisk* utforsking. I introduksjonsøktene til micro:bit i høst viste elevene stor entusiasme og utforsket mange ulike funksjoner i MakeCode. Den gang var ikke utforskingen knyttet til matematikk, men var heller fokusert rundt å bli kjent med editoren og algoritmisk tenkning. Utforsking av artefakten kan knyttes til instrumentalisering. Denne prosessen handler om at elevene blir kjent med og gjør personlige tilpasninger av MakeCode. Elevene kan se muligheter eller begrensninger i MakeCode og bli kjent med hvordan artefakten kan brukes på en nyttig måte for seg.

Instrumentaliseringsprosessen er synlig i flere av funnene mine. Dette ser vi i utdrag 5 i kategorien *4.3 Bruker kjent matematikk for å gi mening til programmet* gjennom at Vilma og Andrine bygget tillit til programmet og ble kjent med mulighetene som ligger i det. De var først usikre på om programmet, som gikk ut på å multiplisere sammen verdiene til to programmeringsvariabler, gav dem riktig svar. Jentene stolte på programmet helt til de møtte et multiplikasjonsstykke med relativt store tall, og måtte kontrollregne uten blokkprogrammering for å sjekke at svaret programmet gav stemte. Det kan virke som at jentene stolte på programmet så lenge de kunne kontrollere at det stemte med automatiserte tallfakta. De kommenterte selv at programmet jo hadde regnet riktig fram til nå. I det de møtte regnestykket de ikke hadde automatisert svaret på, ble de brått usikre. Dette endret seg imidlertid etter at de fikk kontrollert at stykket de var usikre på, var riktig. I utdrag 7 under *4.4 Blir kjent med funksjoner i MakeCode* ser vi at Vilma og Andrine bruker programmet uten å vise mistillit til svarene de får. Jentene virker trygge på at programmet regner riktig og tester for svært store tall. De kan selvsagt ikke vite at de får riktig svar uten å kontrollere, men de virker ikke å være opptatt av det heller. Jentene leker seg med programmet og tester MakeCode. Elevene startet med å bruke blokkprogrammering som et verktøy for å representere kjente multiplikasjonsstykker. Etter å ha blitt bedre kjent med mulighetene i programmet og utviklet tillit til det, bruker de det som et verktøy for å regne multiplikasjonsstykker de ikke har automatisert svaret for, men som passer med oppsettet i programmet deres.

Knut og Johannes får også brukt blokkprogrammering som et mer effektivt verktøy ved utvikling av instrumentaliseringsprosessen. I kategorien *4.4 Blir kjent med funksjoner i MakeCode* er det flere eksempler på at guttene støter på hindringer, men som de finner en løsning på. Guttene oppdager pilmenyen i utdrag 8, som både gjør det mulig for dem å lage programmer med flere variabler, og som også kan effektivisere programmering med andre blokker. Pilmenyen er en del av oppbygningen av MakeCode, og kan i mange tilfeller brukes som en snarveg. I henholdsvis utdrag 9 og 10, erfarer både guttene og Vilma og Andrine at udefinerte programmeringsvariabler alltid gir micro:bit-en verdien null. Guttene møter på dette før jentene, og kan dermed hjelpe dem med å se hva som er problemet i koden deres. De instrumentaliserer, blir kjent med artefaktens muligheter og oppbygning, som er med på å utvikle artefakten til et instrument. Disse oppdagelsene bidrar til at bruksområdet til verktøyet utvides for elevene. Etter at guttene lærte seg pilmenyen og behovet for definering av programmeringsvariabler, fikk de muligheten til å lage og bruke program med programmeringsvariabler i.

Alle fokuselevene hadde synlig utvikling av instrumentaliseringsprosessen. Som vist til i utdrag 7 under *4.4 Blir kjent med funksjoner i MakeCode*, hadde alle elevparene én eller flere faser med å teste programmet for veldig store eller små tall. De var interesserte i å se om det gikk an å multiplisere sammen tall med mange siffer eller negative tall, og hvordan det så ut på micro:bit-en når den multipliserte disse tallene. Elevene undersøkte om de møtte på noen begrensninger i artefakten og oppdaget muligheter for hvilke tall MakeCode klarer å regne med. Vilma og Andrine skrev i tillegg ned alle regnestykkene de gav micro:bit-en og svarene de fikk i retur. Disse kan sies å bruke MakeCode som et verktøy for å løse multiplikasjonsstykker, mens de to andre elevparene heller var interesserte i å bli kjent med MakeCodes muligheter.

Elevene ble kjent med muligheter i MakeCode og programmet undervegs. Ved å fokusere på hvordan instrumentaliseringsprosessen var synlig i elevenes arbeid, ser vi at blokkprogrammering ble brukt som et verktøy på to forskjellige måter. Blokkprogrammering blir brukt for å representere kjent matematikk på en ny måte, og etter hvert som elevene har bygget tillit til programmet, blir blokkprogrammering brukt for å finne svaret på ukjente regnestykker også. Dette gjelder bare for multiplikasjonsstykker med samme struktur som det oppgitte programmet. Altså bruker elevene kun blokkprogrammering til å finne svaret på regnestykker som ikke krever endring av oppbygningen av programmet, men bare endring av verdien til programmeringsvariablene. Vi ser også at instrumentaliseringsprosessen gjør at elevene får et utvidet bruksområde for blokkprogrammering som verktøy. Det krever kjennskap til oppbygning og muligheter i MakeCode for å kunne bruke det som et verktøy i flere situasjoner.

Elevene utviklet kunnskapen sin om artefakten mens de jobbet, og utvidet samtidig bruksskjemaet sitt. Bruksskjemaet handler om hvordan artefakten brukes og hvilke muligheter som ligger i den (Trouche, 2003). Selv om elevene gjorde flere oppdagelser undervegs, hadde de ikke bruksskjema som gjorde at de alltid så hvilke muligheter de hadde. I utdrag 11 og 12 under *4.5 Ser ikke muligheter i MakeCode* ser vi at både Hedda og Johannes foreslår å bruke MakeCode til å finne en ukjent multiplikator, men henholdsvis Lily og Knut sier at det ikke er mulig å bruke artefakten slik makkeren foreslår. Slik sett er det kanskje ikke bare matematikkferdighetene i det instrumenterte handlingsskjemaet som kommer til kort, men også elevenes bruksskjema. Selv om elevene utvidet bruksskjemaet undervegs i undervisningsopplegget, hadde de manglende erfaring med hvordan de kunne løse matematikkoppgaver i MakeCode og hvilke muligheter som ligger i programmeringsvariabler.

5.3 Verktøy for generalisering?

Før gjennomføring av undervisningsopplegget så jeg et potensial for generalisering i programmeringsvariablene. Jeg tenkte at ved å lage en programmeringsvariabel for hver av enhetene i et multiplikasjonsstykke, multiplikand, multiplikator og produkt, ville elevene bli nødt til å generalisere. Dersom de skulle finne verdien til programmeringsvariabelen *det andre tallet,* og kun kjente verdiene til *det første tallet* og *det tredje tallet*, måtte de lage en kode som uttrykte det generelle forholdet mellom delene og det hele i multiplikasjonsstykket. Elevene kunne dividert *det tredje tallet* på *det første tallet*, og ikke en spesifisert verdi på en annen. Her så jeg muligheter for å jobbe med tidlig algebra. Kaput og Blanton (2001) mener at å generalisere og å se sammenhenger, som i dette tilfellet mellom multiplikasjon og divisjon, kan bidra til å gi elever tidlig tilgang til viktig matematisk forståelse.

Jeg så flere tegn på at elevene ikke hadde samme oppfatning som meg av programmeringsvariablene. I utdrag 1 så vi hvordan Lily og Hedda, som det eneste fokuselevparet, brukte divisjon for å finne den ukjente multiplikatoren. Jentene skrev inn de kjente tallene, multiplikand og produkt, som verdier for programmeringsvariablene *det første tallet* og *det andre tallet*. Slik endret rollene til programmeringsvariablene seg. I utformingen min av opplegget tenkte jeg at det var viktig at programmeringsvariablene alltid representerte den samme plassen i et multiplikasjonsstykke, for eksempel at *det andre tallet* alltid representerte multiplikatoren. Jeg tenkte at det ville hjelpe elevene å

overføre det som skjedde i MakeCode til et regnestykke representert symbolsk. Måten Lily og Hedda løste oppgaven på, kan tyde på at dette ikke var så synlig for elevene likevel. Vi har sett at elevene i stor grad identifiserte det matematiske objektet i MakeCode, og overførte det til andre registre. Likevel så nok ikke elevene at programmeringsvariablene representerte hver sin enhet i et multiplikasjonsstykke. Dette kan vi også se lignende eksempel på i Andrine og Vilma sitt arbeid. Dersom de hadde sett programmeringsvariablene som generelle uttrykk for de ulike enhetene i et multiplikasjonsstykke, hadde de nok ikke plutselig blitt usikre på om programmet regnet riktig, som vi så i utdrag 7. De endrer kun på verdiene til programmeringsvariablene, altså bytter de tallene i multiplikasjonsstykket. Da er operasjonen og oppsettet ellers det samme, og det gir ikke mening at en datamaskin plutselig regner feil kun ved innsetting av nye tall. Jeg kan selvsagt ikke være sikker på hvordan jentene oppfattet programmeringsvariablene, men det kan tyde på at de ikke så dem som enhetene i oppsettet for et multiplikasjonsstykke.

Dersom elevene ikke ser programmeringsvariablene slik som jeg, vil generaliseringsaspektet falle bort. Hvis man ikke ser at programmeringsvariablene representerer de ulike enhetene i et multiplikasjonsstykke, er sjansen liten for at elevene ser at å dividere *det tredje tallet* på *det første tallet* alltid vil gi *det andre tallet*, eller at man alltid kan finne multiplikator ved å dividere produkt på multiplikand. Dersom elevene ikke overførte programmeringsvariablene til enhetene i et multiplikasjonsstykke, kan det være vanskelig å bruke dem til å lage nye koder også. De vil da bare bli oppfattet som navn som representerer de spesifikke verdiene de har satt programmeringsvariablene til. Å lage koder med navnene i stedet for tallene, vil da kunne kjennes meningsløst. Hypotesen min om at programmeringsvariablene ville bidra til at blokkprogrammering automatisk ble brukt som et verktøy for å uttrykke generelle sammenhenger, stemte ikke for elevene i arbeid med dette undervisningsopplegget. Det hadde muligens vært tydeligere dersom undervisningsopplegget hadde vært utformet annerledes, dersom elevene hadde vært eldre eller dersom poenget med programmeringsvariablene hadde blitt forklart tydeligere ved oppstart av undervisningsopplegget.

5.4 Metodediskusjon

Jeg har i tiden etter gjennomført innsamling av datamateriale tenkt på flere ting jeg ville gjort annerledes. I fokuselevenes arbeid med undervisningsopplegget var det tydelig at MakeCode fungerte best som et verktøy for multiplikasjonsstykker som passet i de ferdige programmene, altså at elevene kun trengte å sette inn nye verdier for programmeringsvariablene eller kjøre programmet helt til de traff ønskede verdier. Det hadde vært interessant å se hvordan elevene hadde brukt blokkprogrammering som verktøy dersom de ikke fikk et ferdig program, men måtte produsere all kode selv. Jeg tenkte på forhånd at jeg ville gi elevene et ferdig program som utgangspunkt, for å få de til å ta i bruk programmeringsvariabler, som igjen kunne bidra til generalisering. Det kan likevel tenkes at det hadde gått helt fint å gi elevene multiplikasjonsstykker som de selv skulle lage program for å løse. Elevene viste under introduksjonsøktene god evne til å utforske og lage egne program. Det hadde også vært interessant å se hvordan elevene hadde brukt blokkprogrammering som et verktøy i arbeid med multiplikasjon av større tall. I opplegget mitt ble automatisert tallfaktakunnskap brukt flere steder, og verktøyet hadde nok blitt brukt annerledes dersom dette ikke hadde vært mulig. Det kan tenkes at elevene ikke hadde kunne endret strukturen på oppgaven dersom tallene hadde vært

større, men det kunne også ført til større usikkerhet og mistillit som følge av at elevene ikke hadde hatt mulighet til å kontrollere programmet med automatiserte tallfakta.

Det er også ting jeg ville endret i selve gjennomføringen av innsamling av datamaterialet dersom jeg kunne gjort det på nytt. Jeg vil trekke fram gjennomføringen av andre økt som et punkt med potensial for innvirkning på fokuselevenes arbeid. Her satt fokuselevene bakerst i klasserommet med resten av klassen, i stedet for på et eget rom som under første økt. Det ble mindre arbeidsro for fokuselevene og situasjonen ble spesiell ved at kun disse elevene satt på et gruppebord med kamera og lydopptaker rundt seg. Det ble mye fokus på fokuselevene og spørsmål om prosjektet fra de andre elevene. Fokuselevene hadde samme utstyr rundt seg under første økt også, og slik sett var denne situasjonen kunstig også. I denne økta var de alene med to studenter på et rom, og elevene virket roligere og mer fokuserte enn i andre økt. Selv om fokuselevene kjenner meg og medstudenten min relativt godt ved at vi har vært i praksis for samme gjeng et år tidligere også, er vi likevel ikke en del av de voksne elevene vanligvis omgås med på skolen. Det kan spille inn på elevenes oppfatning av situasjonen.

6 Konklusjon

Forsström og Kaufmann (2018) konkluderte i sin litteraturstudie om programmering i matematikkundervisning at det trengs mer forskning på hvordan programmering kan brukes som arbeidsmåte, og at det må sees i sammenheng med andre matematiske tema enn geometri. Jeg har undersøkt hvordan blokkprogrammering kan brukes i arbeid med multiplikasjon, med spesielt fokus på å se sammenhengen til divisjon. I studien virker det som at elevene ser blokkprogrammering som et verktøy for å jobbe med sine egne matematikkunnskaper. Elevene overførte multiplikasjonsstykker representert på symbolform til et ferdig program i MakeCode, og løste dem der. Etter som elevene fikk erfaring med editoren og programmeringsspråket, gikk de fra å kontrollere alle svarene de fikk i MakeCode til å bruke blokkprogrammering som et verktøy for å finne svaret på multiplikasjonsstykker med større tall også.

Felles for elevenes bruk av blokkprogrammering i arbeid med multiplikasjonsoppgavene er at de har høy terskel for å gjøre endringer i programmet og få forsøk på å prøve ut nye koder og program. Elevene brukte programmene de fikk i oppgaveteksten, og tilpasset alle multiplikasjonsstykkene de fikk slik at de passet inn. Ett elevpar tok i bruk divisjon, slik jeg hadde håpet, og det virket som at dette var en kjent strategi for dem. Slik kan dette også støtte at elevene brukte blokkprogrammering som et verktøy for kjent matematikkunnskap. Elevene utforsket ikke nye matematiske strategier i arbeid med blokkprogrammering. Der matematikken var ukjent gjorde elevene enten om på oppgavene, slik at de kunne bruke kjente strategier, eller så stoppet arbeidet deres opp. Funn i studien min tyder på at blokkprogrammering fungerer best som et verktøy for å mediere kjente matematikkunnskaper og ferdige programmer, og i liten grad som et verktøy for å utforske og oppdage nye multiplikasjonsstrategier.

Før blokkprogrammering blir tatt i bruk i multiplikasjonsundervisning er det en stor fordel om elevene har hatt introduksjonsøkter med utforsking av artefakten. Elevene i studien min lærte seg bruk av blokker og snarveger i editoren undervegs i arbeidet, enda de hadde hatt egne introduksjonsøkter på forhånd. Det er viktig at elevene får kjennskap til blokker og funksjoner for at de skal kunne se hvilke muligheter de har. I min studie oppstod det situasjoner der elevene ikke så at en oppgave var mulig å løse i MakeCode. Spesielt programmeringsvariabler, som elevene kun hadde hatt én introduksjonsøkt med, var vanskelige for dem å se potensialet til. Etter noen introduksjonsøkter kjenner elevene artefakten bedre, og vet i større grad hvilke blokker og muligheter som finnes. Det er også mulig at bedre kjennskap til artefakten hadde muliggjort matematisk utforsking for elevene i større grad. I arbeid med undervisningsopplegget mitt brukte elevene kjente multiplikasjonsstrategier, og gjorde egne tilpasninger av oppgaver slik at de kunne løses med kjente strategier som passet i de ferdige programmene. Det finnes mange muligheter i MakeCode, noe som gjør det vanskelig å lage et undervisningsopplegg der elevene skal utforske seg fram til én bestemt strategi. Jeg var inspirert av PRIMM-modellen i utformingen av opplegget mitt. Dette innebar at elevene fikk et ferdig program, noe som virket å låse dem. Elevene brukte programmene til å løse multiplikasjonsstykker og lagde i tillegg egne stykker. De hadde imidlertid høy terskel for å endre på blokker eller struktur av det ferdige programmet. Det kan dermed tenkes at det hadde vært større sjanse for at elevene utforsket ulike strategier uten et ferdig program som utgangspunkt.

Etter å ha gjennomført studien ser jeg muligheter i blokkprogrammering for å representere allerede kjent matematikk for elevene. Det er mulig at blokkprogrammering som register er spesielt bra for å få fram enkelte egenskaper ved matematiske objekt, men dette må undersøkes nærmere av andre studier. Det blir også opp til andre å undersøke videre om det er mulig å utforske matematisk i MakeCode. Ideen høres god ut, men i mitt opplegg var det lite utforsking av nye multiplikasjonsstrategier. Som nevnt ser jeg et potensial for generalisering i programmeringsvariabler, men fikk ikke synliggjort det for elevene i denne studien. Jeg ser at blokkprogrammering muligens kan være et verktøy for å jobbe med tidlig algebra, både med generaliseringspotensialet og multimodale variabler generelt. Det ligger uten tvil mange muligheter i blokkprogrammering, men det trengs videre forskning for å tydeliggjøre hvilken rolle blokkprogrammering kan ha i matematikkundervisning. Skal programmering og algoritmisk tenkning være et eget emne, eller kan det være en nyttig representasjon eller arbeidsmåte for andre matematiske emner?

Referanser

Aubert, K. E., Briseid, E. M. & Hofmann, A. (2022, 27. desember). Multiplikasjon. I *Store norske leksikon.* Hentet 27. april 2023 fra https://snl.no/multiplikasjon

Baroody, A. J. (2006). Why children have difficulties mastering the basic number combinations and how to help them. *Teaching children mathematics*, *13*(1), 22-31. https://doi.org/10.5951/TCM.13.1.0022

Blanton, M. (2010). Early algebra. I Z. Usiskin, K. Andersen & N. Zotto (Red.) *Future curricular trends in school algebra and geometry* (s. 45-61). Information Age Publishing.

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*. *23*(2), 170-185. https://doi.org/10.1080/10986065.2020.1779012

Busuttil, L., & Formosa, M. (2020). Teaching Computing without Computers: Unplugged Computing as a Pedagogical Strategy. *Informatics in Education*, 19(4), 569-587. https://doi.org/10.15388/infedu.2020.25

Carpenter, T. P., Franke, M. L. & Levi, L. (2003). *Thinking mathematically: Integrating arithmetic and algebra in elementary school.* Heinemann.

Clark, T., Foster, L., Sloan, L. & Bryman, A. (2021). *Bryman's social research methods* (6. utg.). Oxford University Press.

Cobb, P., Confrey, J., Disessa, A., Lehrer, R. & Schauble, L. (2003). Design Experiments in Educational Research. *Educational Researcher*, 32, 9-13. https://doi.org/10.3102/0013189X032001009

Dahl, H., Klemp, T. & Nilssen, V. (2020. Språklige ressurser, en forutsetning for produktivt elevsamarbeid. I V. Nilssen & S.-M. Høynes (Red.), *Samtaleorientert matematikk: Et samspill mellom didaktiske og adidaktiske situasjoner* (s. 161-191). Fagbokforlaget.

Drijvers, P., Doorman, M., Boon, P., Reed, H. & Gravemeijer, K.. (2010). The teacher and the tool: Instrumental orchestrations in the technology-rich mathematics classroom. *Educational Studies in Mathematics*, 75, 213-234. https://doi.org/10.1007/s10649-010-9254-5

Duval, R. (2006). A cognitive analysis of problems of comprehension in a learning of mathematics. *Educational Studies in Mathematics*, 61, 103–131, https://doi.org/10.1007/s10649-006-0400-z

Enge, O., & Valenta, A. (2011). Argumentasjon og regnestrategier. *Tangenten – tidsskrift for matematikkundervisning*, *22*(4), 27-32. https://www.matematikksenteret.no/sites/default/files/attachments/MAM/Revisjon%2020-21/Enge_Valenta_Argumentasjon_og_regnestrategier.pdf

Fischbein, E., Deri, M., Nello, M. S., & Marino, M. S. (1985). The role of implicit models in solving verbal problems in multiplication and division. *Journal for Research in Mathematics Education, 16*(1), 3–17. https://doi.org/10.2307/748969

Forsström, S. A. & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research.* 17(12), 18-32. https://doi.org/10.26803/ijlter.17.12.2

Fosnot, C. T. & Dolk, M. (2001). *Young mathematicians at work: Constructing Multiplication and Division.* Heinemann.

Franklin, D., Hill, C., Dwyer, H. A., Hansen, A. K., Iveland, A., & Harlow, D. B. (2016, februar). Initialization in scratch: Seeking knowledge transfer. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 217-222. https://doi.org/10.1145/2839509.2844569

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

Kaput, J. J. & Blanton, M. (2001). Algebrafying the elementary mathematics experience, part I: Transforming task structures. I H. Chick, K. Stasey, J. L. Vincent & J. Vincent (Red.), *The future of the teaching and learning of algebra: Proceedings of the 12th ICMI study conference, Vol. I* (s. 344-351). The University of Melbourne. https://www.researchgate.net/profile/Kaye-Stacey/publication/330409545_The_future_of_teaching_and_learning_of_algebra

Kieran, C. (2004). Algebraic thinking in the early grades: What is it?. *The mathematics educator*, *8*(1), 139-151. https://gpc-maths.org/data/documents/kieran2004.pdf

Kieran, C., Pang, J., Schifter, D., & Ng, S. F. (2016). *Early algebra: Research into its nature, its learning, its teaching*. Springer Nature.

Kilhamn, C., Bråting, K., Helenius, O., & Mason, J. (2022). Variables in early algebra: exploring didactic potentials in programming activities. *ZDM–Mathematics Education*, *54*(6), 1273-1288. https://doi.org/10.1007/s11858-022-01384-0

Kunnskapsdepartementet. (2019). *Læreplan i matematikk (MAT01-05).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nob

Micro:bit Educational Foundation (u.å.a). *Make it: Code it*. Hentet 16. mai 2023 fra https://microbit.org/projects/make-it-code-it/

Micro:bit Educational Foundation (u.å.b). *Microsoft MakeCode*. Hentet 16. mai 2023 fra https://microbit.org/code/

Micro:bit Educational Foundation. (u.å.c). *Milestones for the BBC micro:bit.* Hentet 25. mars 2023 fra https://microbit.org/impact/case-studies/milestones-for-the-bbc-microbit/

Moe, T. I. (2019). *CAS – bruke for å lære, eller lære å bruke? En kvalitativ studie av instrumentell skapelse hvor elever i videregående skole benytter CAS i arbeid med matematikk* [materoppgave]. Norges teknisk-naturvitenskapelige universitet.

Nilssen, V. (2012). *Analyse i kvalitative studier.* Universitetsforlaget.

Norges teknisk- og naturvitenskapelige universitet. (u.å.). *LAB-Ted: Learning, Assessment and Boundary crossing in Teacher education*. Hentet 15. mai 2023 fra https://www.ntnu.edu/ilu/lab-ted

Rabardel, P. & Bourmaud, G. (2003). From computer to instrument system: A developmental perspective. *Interacting with Computers*, 15(5), 665-691, https://doi.org/10.1016/S0953-5438(03)00058-4

Rabardel, P. & Samurcay, R. (2001, 21.-23. mars). From Artifact to Instrument-Mediated Learning. *New Challenges to research on learning, International symposium organized by the Center for Activity Theory and Developmental Work Research, University of Helsinki.*

Rønning, F. & Burheim, O. T. (2020). Betydningen av en faglig og fagdidaktisk føranalyse i utviklingen av et undervisningsforløp. I V. Nilssen & S.-M. Høynes (Red.), *Samtaleorientert matematikk: Et samspill mellom didaktiske og adidaktiske situasjoner* (s. 121-157). Fagbokforlaget.

Schou, J., Jess, K., Hansen, H. C. & Skott, J. (2014). *Matematik for lærerstuderende: Tal, algebra og funktioner. 4.-10- klasse*. Samfundslitteratur.

Sentance, S., Waite, J. & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2-3), 136-176. https://doi.org/10.1080/08993408.2019.1608781

Senter for IKT i utdanningen. (2016, november). *Programmering i skolen* (notat, 1. reviderte utg.). https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Steffe, L. P. (1994). Schemes of action and operation involving composite units. *Learning and Individual Differences,* 4(3), 259-309.

Super:bit. (u. å.) *Hva er super:bit?* Hentet 25. mars 2023 fra https://www.superbit.no/hva-er-superbit/

Trouche, L. (2003). From artifact to instrument: Mathematics teaching mediated by symbolic calculators. *Interacting with Computers*, 15(6), 783–800, https://doi.org/10.1016/j.intcom.2003.09.004

Usiskin, Z. (1988). Conceptions of school algebra and uses of variables. *The ideas of algebra, K-12*, *8-19*. https://www.msri.org/attachments/workshops/454/Usiskin-Conceptions%20of%20School%20Algebra.pdf

Utdanningsdirektoratet. (2019, 27. mars). *Algoritmisk tenkning*. Hentet 03. februar 2023 https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Van de Walle, J. A., Karp, K. S. & Bay-Williams, J. M. (2020). *Elementary and Middle School Mathematics: Teaching Developmentally* (10. utg). Pearson Education Limited.

Vergnaud, G. (1996). The Theory of Conceptual Fields. I L. Stette, P. Nesher, P. Cobb, G. A. Goldin, & B. Greer (Red.), *Theories of Mathematical Learning* (s. 219-240). Lawrence Eribaum Associates.

Verillon, P., & Rabardel, P. (1995). Cognition and artifacts: A contribution to the study of thought in relation to instrumented activity. *European Journal of Psychology of Education, 10*(1), 77–101. https://doi.org/10.1007/BF03172796

Vygotskij, L. S. (1978). *Mind in Society. The development of higher psychological processes.* Harvard University Press.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Vedlegg

Vedlegg 1 - Oppgaveark

Navn: _____

Oppgave 1 - Multiplikasjon

Regn ut.

$3 \cdot 5 =$

$6 \cdot 4 =$

$3 \cdot 7 =$

$8 \cdot \quad = 16$

$3 \cdot \quad = 9$

$5 \cdot \quad = 45$

$\cdot 2 = 12$

$\cdot 3 = 18$

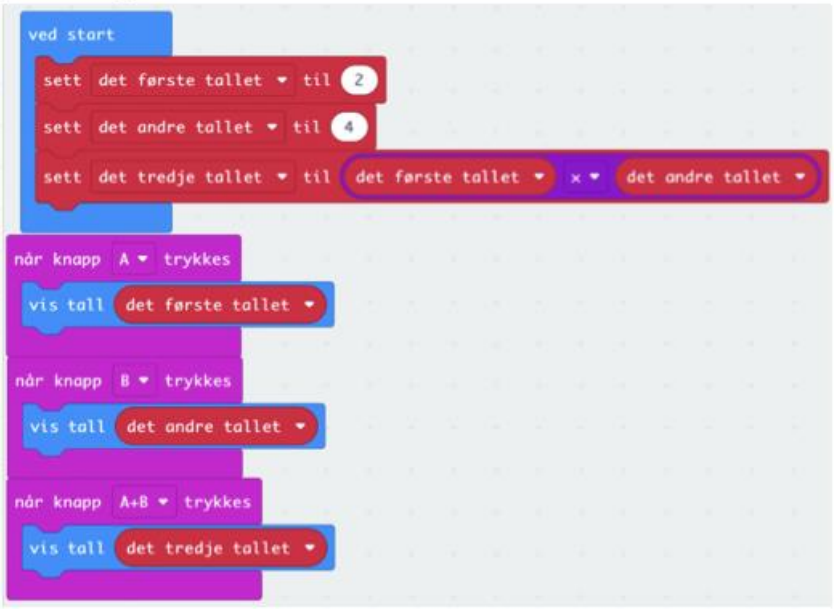
$\cdot 4 = 32$

Skriv eller tegn hvordan dere løste oppgaven:

$6 \cdot \quad = 24$

Navn: _____

Oppgave 2 - Lag kalkulator



Hvilket tall viser micro:bit´n når **knapp A** trykkes:_____

Hvilket tall viser micro:bit´n når **knapp B** trykkes: _____

Hvilket tall viser micro:bit´n når **knapp A+B** trykkes: _____

Hent chromebook. Lag programmet på bildet i MakeCode
(Gå inn på makecode.microbit.org og lag et nytt prosjekt med navn "multiplikasjon del 1")

Oppgave 3 - Regn med kalkulatoren

Løs oppgavene ved hjelp av kalkulatoren din i MakeCode og skriv svaret på arket:

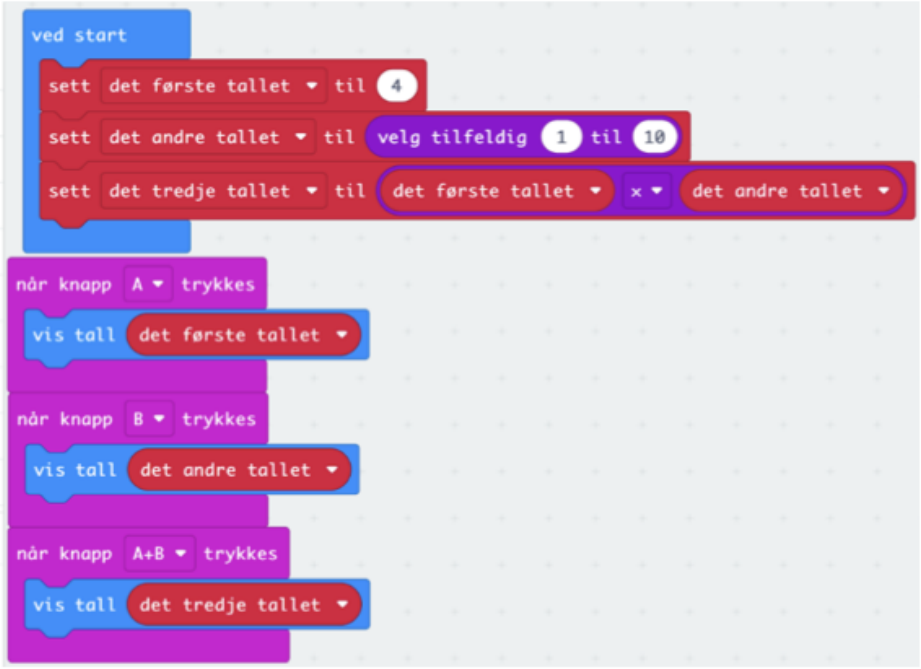
$3 \cdot 5 =$ ___ $9 \cdot 2 =$ ___ $7 \cdot 6 =$ ___

Oppgave 4 - Regn videre

Prøv å regne ut stykkene fra oppgave 1 i kalkulatoren dere lagde i MakeCode.
Dere kan endre på kalkulatoren dersom det trengs for å løse alle oppgavene.

Navn: _____

Oppgave 5 - Kodeanalyse



Hvilket tall viser micro:bit´n når **knapp A** trykkes: _____

Hvilket tall viser micro:bit´n når **knapp B** trykkes:

_____

Hvilket tall viser micro:bit´n når **knapp A+B** trykkes:

_____

Hent chromebook. Lag programmet på bildet i MakeCode
(Gå inn på makecode.microbit.org og lag et nytt prosjekt med navn "multiplikasjon del 2")

Navn: _____

Oppgave 6 - Regn ut

Se på koden i oppgave 5.

Dersom “det tredje tallet” er 12 og “det første tallet” er 4, hva må “det andre tallet” være?

Skriv eller tegn hvordan dere tenker her:

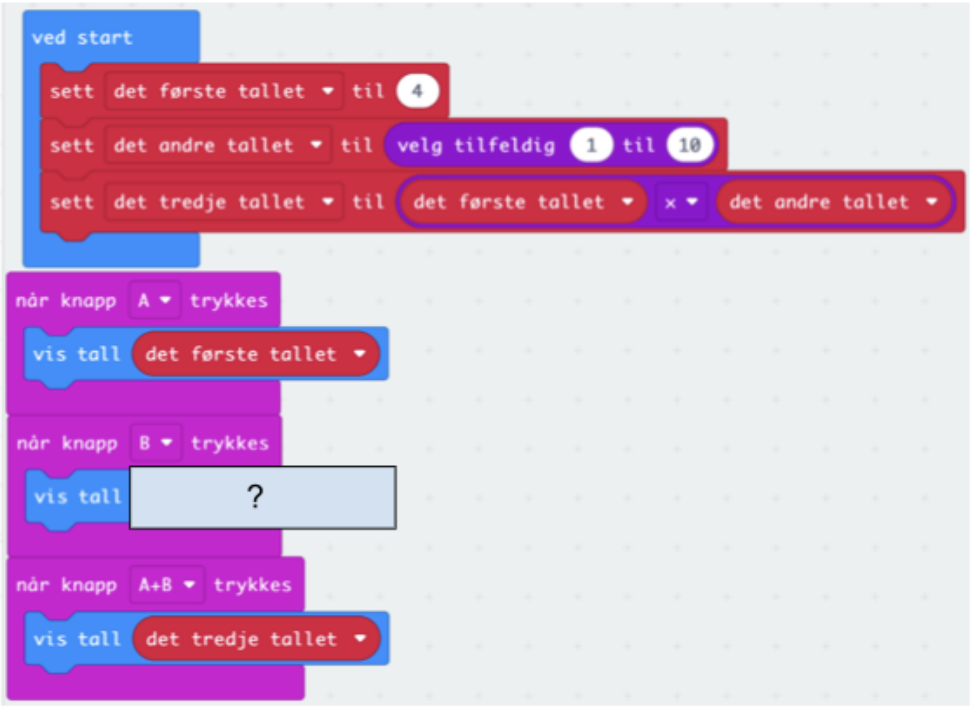
Dersom skjermen viser 20 når du trykker på knapp A+B og 4 når du trykker på knapp A, hva vises på skjermen når knapp B trykkes?

Skriv eller tegn hvordan dere tenker her:

Navn: _____

Oppgave 7 - Gjør om på kalkulatoren

Her skal du lage en ny kode for "det andre tallet".



Lag en ny kode for knapp B som gjør slik at kalkulatoren finner ut hva "det andre tallet" er, uten å bruke blokka som heter "det andre tallet".

EKSTRAOPPGAVE - IKKE BRUKT

Oppgave 8 - Regn ut med ny kalkulator

Bruk den nye kalkulatoren til å finne svaret på regnestykkene under. Skriv svaret på arket:

$4 \cdot \quad = 16$

$3 \cdot \quad = 21$

$5 \cdot \quad = 45$

$8 \cdot \quad = 72$

Navn: _____

EKSTRAOPPGAVE - IKKE BRUKT

Oppgave 9 - Regn ut

Bruk MakeCode til å løse oppgavene under.

$8 \cdot \quad = 48$

$9 \cdot \quad = 63$

$5 \cdot \quad = 45$

$250 \cdot \quad = 1500$

$\cdot 6 = 72$

$\cdot 3 = 18$

$\cdot 4 = 32$

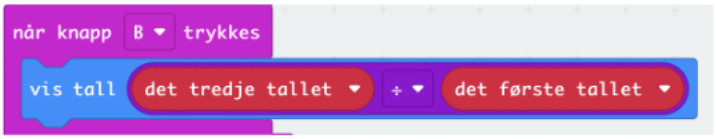
Vedlegg 2 – Tipslapper

Tips: Du kan bruke de andre variabelblokkene

Tips: Bruk en annen regneart

Tips: Bruk deletegnet i koden din

Tips: Slik kan koden se ut. Forstår du hvorfor den fungerer?



Vedlegg 3 – Observasjonsskjema

Tekst markert med " " = utsagn. Tekst uten markering er observasjoner.

Observasjon/intervju - Dag 1

Oppgave 1 - Multiplikasjon

Hvilke strategier bruker elevene?

Struktur 1	Struktur 2	Struktur 3

Oppgave 2/3/4 - Programmering

Svarer elevene korrekt på spørsmålene?

Ja	Nei	Utfordringer

"Hvilke tall fungerer denne kalkulatoren for? Hvordan/hvorfor?"

Tekst markert med " " = utsagn. Tekst uten markering er observasjoner.

Observasjon/intervju - Dag 2

Oppgave 5

Svarer elevene korrekt på spørsmålene?

Ja	Nei	Utfordringer

"Hvilke tall fungerer denne kalkulatoren for? Hvordan/hvorfor?"

Oppgave 6

Svarer elevene korrekt på spørsmålene? Utfordring knytt til formulering?

Ja	Nei	Utfordringer

Tekst markert med " " = utsagn. Tekst uten markering er observasjoner.

Oppgave 7

Forslag/strategier?

Generalisering?

Ja	Nei	Kommentar

Er det bruk for tips? Hvilke tips virker å være mest nyttige?

	Kommentar
1	
2	
3	
4	

Oppgave 8/9 - EKSTRA

Endrer elevene på kalkulatoren for oppgaver med samme struktur? Hvorfor/hvordan?

Endrer elevene på kalkulatoren for oppgaver med ulik struktur? Hvorfor/hvordan?

Vedlegg 4 – Informasjonsskriv til foresatte

Vil du delta i forskningsprosjektet

"Bruk av micro:bit i matematikkundervisning på barneskolen"?

Dette er et spørsmål til deg, som foresatt, om barnet ditt kan delta i et forskningsprosjekt hvor formålet er å undersøke ulike undervisningsopplegg i matematikk som bruker micro:bit. micro:bit er et programmeringsverktøy spesielt rettet mot barnetrinnet. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Vi er fire 5.-årsstudenter på Grunnskolelærerutdanningen på NTNU som skal skrive masteroppgaver våren 2023. I den anledning ønsker vi å samle inn datamateriale for å forske på bruk av programmering i matematikkfaget på barneskolen. Innsamlet datamateriale vil være grunnlag for masteroppgavene våre, og alle personopplysninger vil bli anonymisert. Denne forskningen innebærer observasjon, intervju, samt video- og lydopptak av elevene og læreren i klassen.

Vi ønsker å undersøke nærmere:
- Hvordan bruke micro:bit i matematikkundervisning?
- Hvordan kan matematikk inngå i programmering?
- Kan bruk av programmering i matematikkundervisning bidra til økt motivasjon for matematikkfaget generelt?

Masteroppgavene blir veiledet som en del av arbeid i forskningsprosjektet LAB-Ted. Det anonymiserte datamaterialet innsamlet til masteroppgavene kan inngå i annen forskning knyttet til LAB-Ted-prosjektet.

Hvem er ansvarlig for forskningsprosjektet?
Norges teknisk-naturvitenskapelige universitet er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Elever som har som matematikklærer ved 4. trinn ved barneskole får tilbud om å delta i prosjektet.

Hva innebærer det for deg å delta?
Eleven vil delta i den planlagte matematikkundervisningen. I enkelte økter, der micro:bit blir brukt, vil det bli tatt video- og lydopptak av elevenes arbeid. Dette bli lagret på en kryptert, ekstern harddisk, og vil slettes etter at materialet er transkribert og anonymisert.

I disse øktene vil vi snakke med elevene om deres tanker og arbeid med micro:bit. Dette kan enten foregå underveis i klasserommet, eller dersom det er praktisk umulig, vil vi be elevene om en samtale på grupperom etter endt økt. Potensielle spørsmål kan være:
- Hvordan synes du det fungerer å bruke micro:bit i undervisningen?
- Hvordan tenkte du i løsningen av denne oppgaven?
- Synes du programmering er en motiverende arbeidsform i matematikkfaget? - Hvordan brukte du kodebiten i løsningen din?
- Syns du det er noe som er vanskelig med bruk av micro:bit?

Hvis du som foresatt ønsker en fullstendig intervjuguide, ta direkte kontakt med oss.

Det er frivillig å delta

Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Elevene vil også bli spurt om de ønsker å delta i forskningsprosjektet, og elever som ikke ønsker dette vil ikke delta. Alle elevene vil få samme undervisningsopplegg, men det vil ikke bli samlet inn datamateriale av elever som ikke ønsker å delta.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger

Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Personer som vil ha tilgang til opplysningene som blir samlet inn er: studentgruppa, praksislærer og veiledere fra NTNU. Alle personopplysninger vil bli anonymisert, og datamaterialet vil bli lagret på en innelåst og kryptert ekstern harddisk.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes? Prosjektet vil etter planen avsluttes/oppgavene blir godkjent, noe som etter planen er juni 2023. Alle former for opptak som blir gjort under gjennomføring av prosjektet, vil anonymiseres fortløpende og selve opptakene vil slettes senest ved prosjektslutt.

Siden dette prosjektet er en del av et større prosjekt (LAB-Ted), vil de anonymiserte data fra prosjektet bli beholdt noe utover prosjektperioden, men vil slettes senest ved utgangen av juni 2025. Temaet for prosjektet er høyaktuelt for utviklingsarbeid og forskning i norsk skole, og derfor vil det anonymiserte materialet bli beholdt noe utover perioden masterprosjektene forgår. Formålet med den videre lagringen er at forskere ved NTNU, ILU, spesifikt Benedikte Grimeland, Yvonne Grimeland, Oda Tingstad Burheim og Torunn Klemp kan bearbeide materialet videre for å publisere forskningsarbeid slik at resultatene fra prosjektet når en større del av lærerutdanningsfeltet i Norge. Det er kun de nevnte forskere som vil ha tilgang til materialet. Materialet vil bli lagret på kryptert disk som ikke er koblet til nett, innelåst hos NTNU, og eid av NTNU.

Hva gir oss rett til å behandle personopplysninger om deg?

Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra NTNU har Personverntjenester vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter

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

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

- NTNU ved Yvonne Grimeland. Tlf. 48114352. Epost: Yvonne.grimeland@ntnu.no
- Vårt personvernombud: Thomas Helgesen. Tlf. 93079038. Epost: thomas.helgesen@ntnu.no

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:

- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Guro Kjeldstad (forsker) | Vilde Dahle (forsker) | Ellen-Marie Kristiansen (forsker) | Eirun Flaten Sandvin (forsker)

Yvonne Grimeland (veileder)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Bruk av micro:bit i matematikkundervisning på barneskolen», og har fått anledning til å stille spørsmål.

Jeg samtykker til at mitt barn har tillatelse til:

- ☐ å delta i intervju
- ☐ å delta i lyd-opptak
- ☐ å delta i video-opptak

Jeg samtykker til at opplysninger om mitt barn behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltakers foresatt, dato)

Vedlegg 5 – Oppgaver fra forberedelsesperioden

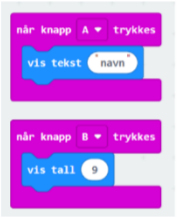
Utforsking av MakeCode

Oppgaver

1. Få "Hallo" til å vise på skjermen
2. Lag et hjerte som blinker 10 ganger
3. Få skjermen til å vise et tall når du trykker på en knapp
4. Få skjermen til å vise temperatur når du trykker på en knapp
5. Lag en terning som viser et tall når den ristes

Kodeanalyse

Kode	Beskriv tydelig og nøyaktig hva koden skal få micro:bit'n til å gjøre	Tegn hva som kommer til å vises på skjermen (micro:bit'n)
		
		

Kode	Beskriv tydelig og nøyaktig hva koden skal få micro:bit'n til å gjøre	Tegn hva som kommer til å vises på skjermen (micro:bit'n)
		
		Hva vises på skjermen hvis temperaturen er 5 (grader Celsius)? Hva vises på skjermen hvis temperaturen er 9 (grader Celsius)?
	Du ønsker at tallet 2 skal vises på skjermen når man trykker på knapp B. Etter en pause på 1 sekund skal tallet 4 vises på skjermen. Finn to feil med denne koden.	

Introduksjonsøkt til programmeringsvariabler

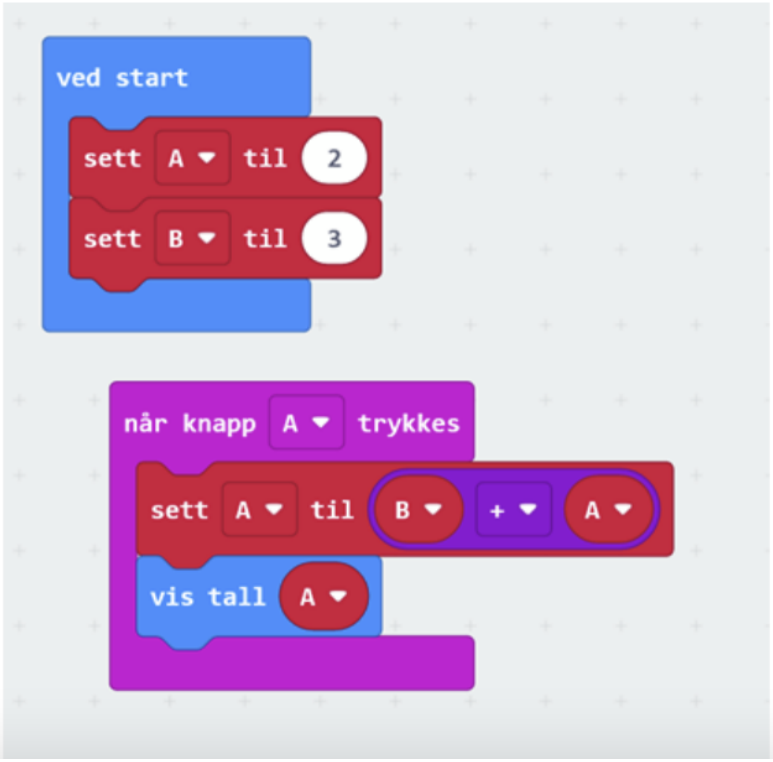
Oppgave 1: Bestem at micro:bit'n først skal vise tallet 1. For hver gang du trykker på knapp A skal micro:bit'n vise deg tallet som er én mer.

Oppgave 2: For hver gang du trykker på knapp A skal tallet som vises være én mer enn det forrige. Det første tallet skal være 5.

Oppgave 3: Lag et program med to variabler. Når knapp A trykkes, skal én av variablene øke med 5. Når knapp B trykkes, skal den andre variabelen øke med 2.

Oppgave 4: Lag et program med to variabler. Når knapp A trykkes, skal micro:bit`n legge sammen variablene og vise svaret på skjermen.

Oppgave 5: Beskriv hva som vil vises på micro:bit'n dersom du trykker flere ganger på knapp A.



NTNU

Kunnskap for en bedre verden