Flø

Programmering i LK20

Da stod vi her, midt i oppstart av nye læreplaner, i tillegg til en pandemi. Det har ikke vært den enkleste høsten å være lærer, men jeg håper jeg kan bidra positivt med denne artikkelen. Jeg vil dele konkrete eksempler på undervisningsopplegg, spesielt knyttet opp mot matematikk og programmering. Flere digitale ressurser vil også gjøres tilgjengelig gratis via et nytt nettsted.

Ifølge de nye læreplanene (LK20) skal elevene lære å programmere i matematikk, naturfag, musikk og kunst og håndverk, der matematikk har hovedansvar for programmeringsopplæringen. Programmering i matematikkfaget handler om at elevene skal lære å bruke algoritmisk tenkning som en problemløsningsstrategi og vurdere når det er formålstjenlig å bruke ulike digitale hjelpemidler, inkludert det å programmere. Gjennomsnittlig ligger det ett kompetansemål innen programmering i matematikk hvert skoleår i grunnskolen, og gjennom de nye planene uttrykkes det et ønske om at elevene skal få opplæring i programmering for å få faglig utbytte innen for eksempel matematikk.

Ellen Egeland Flø
Universitetet i Oslo
elleneg@student.uv.uio.no

Ellen Egeland Flø fikk hederlig omtale i forbindelse med Holmboeprisen i 2020.

Det er formulert beskrivelser i overordnet del, i kjerneelementene og i kompetansemålene, om at elevene skal utforske og oppleve skaperglede i matematikken. Samlet sett skal LK20 bidra til å øke elevenes dybdelæring. Det er ikke akkurat et enkelt oppdrag lærerne har fått! Hvordan kan man som lærer operasjonalisere disse kravene?

Jeg foreslår å bruke tverrfaglige undervisningsopplegg som knytter ulike matematiske områder opp mot programmeringsoppgaver og utforskende arbeidsmåter, der elevene utvikler og lager en fysisk gjenstand for å løse oppgaven. Elevene jobber i samsvar med en modell som er basert på en kombinasjon av naturvitenskapelige metoder og designtenking. Den består av at elevene arbeider seg gjennom syv faser:

1. Undersøke
2. Planlegge
3. Gjennomføre
4. Teste
5. Evaluere
6. Forbedre
7. Dokumentere

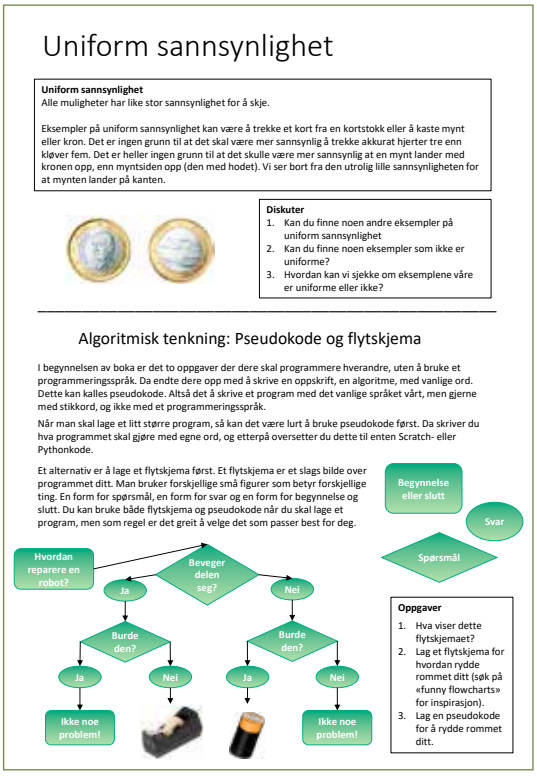
Disse fasene inngår i figur 1, som viser eksempel på et undervisningsopplegg basert på denne modellen. Hovedpoenget er ikke at elevene slavisk trenger å følge denne modellen, den skal heller ikke oppfattes kronologisk. Man hopper gjerne mellom de ulike fasene og er

Flø, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

innom forskjellige faser mange ganger. Dette er sentralt, siden et viktig poeng med denne måten å jobbe på er å «lære av sine feil». For å legge til rette for dette bør elevene bli oppmuntret til å evaluere hvordan arbeidet går, og å korrigere/justere kursen videre. Fasene kan særlig være knagger for lærerne å ha når de utvikler slike undervisningsopplegg. Jeg leder for tiden forskningsprosjektet ProSkap-SL, som undersøker undervisning basert på denne modellen, se gjerne mer på prosjektets nettsted (https://www.uv.uio.no/iped/forskning/prosjekter/morch-prokraft/).

Som lærer i nesten ti år har jeg gjort meg noen personlige erfaringer med å jobbe på denne måten. Blant annet har jeg brukt et undervisningsopplegg der elevene bygger solfangere (se figur 5), da solfangeres virkemåte har vært pensum for Vg1 fram til høsten 2020. Mine foreløpige erfaringer tilsier at elevene har veldig ulike tilnærminger, noen veldig systematiske/matematiske, mens andre bare prøver seg fram, men alle har endt opp med noe bra til slutt. En elev som var faglig svak i matematikk og naturfag, nærmest strålte da hun fikk til å lage sitt første program for temperaturmåling. Og en diskusjon jeg hadde med en elevgruppe om gyldighetsområde for deres modell (en tredjegradsfunksjon) av temperaturen som funksjon av tid for solfangeren deres, kommer jeg ikke til å glemme. De fleste på gruppa hadde ikke vært borti regresjon (tre av fire elever), men alle snakket i munnen på hverandre da de prøvde å forklare hvorfor modellen ikke gjaldt for alle verdier av x. Mitt håp er at denne typen undervisningsopplegg kan bidra positivt til tilpasset opplæring og mestring for et større mangfold av elever.

Jeg har utformet denne typen undervisningopplegg på en slik måte at venstresiden er et sammendrag av tilhørende fagstoff, som elevene bør ha jobbet med før de skal i gang med det praktiske arbeidet. I figur 1 er sannsynlighet temaet, samt en kort introduksjon til pseudokoding og flytskjema, som er begreper hentet



Figur 1: Undervisningsopplegg om sannsynlighet der elevene skal lage terning eller lykkehjul. Last ned PDF av alle undervisiningsoppleggene fra www.caspar.no/tangenten/2021/flo.pdf

fra programmeringsopplæring. Det jobbes med kompetansemålet på 9. trinn: «simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering». Høyresiden er delt opp i de syv fasene med veiledende tips og spørsmål til elevene som leder dem i prosessen med å gjennomføre oppgaven, men uten noen form for ferdig oppskrift eller fremgangsmåte. Elevene må støttes i å tenke ut selv. Lærerens rolle blir da mer som en rollemodell som demonstrerer hvordan elevene kan gå fram for å finne ut av det de lurer på, heller enn å komme med de «riktige» svarene.

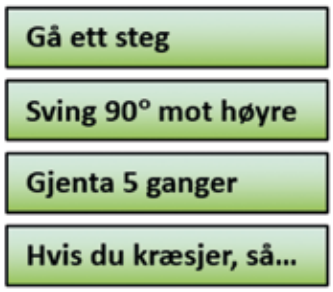
I undervisningopplegget i figur 1 skal elevene lage et program som simulerer et mulig utfall og beregner sannsynligheten for dette utfallet, slik det står i kompetansemålet som det er henvist til over. Angående programmeringsbegreper,

Fløe, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

slik som løkker, variabler, funksjoner, vilkår o.l., sier Grover et al. (2015) og Meerbaum-Salant et al. (2013) at det er krevende for elevene å lære disse, og at det er viktig at læreren tar opp disse begrepene eksplisitt i sin undervisning. De fremhever at elevene ikke lærer programmering og programmeringsbegrepene automatisk ved å lage egne programmer (Grover et al., 2015; Meerbaum-Salant et al., 2013). En populær tilnærming til begrepslæring innen programmering kalles analog programmering (unplugged programming), der man programmerer uten datamaskin, nettbrett eller annen digital teknologi.

Et eksempel på en analog programmeringsaktivitet kan være å dele elevgruppa inn i par, der hver gruppe får tildelt en egen destinasjon. Elevene skal lage en skriftlig oppskrift på hvordan den andre eleven i paret skal bevege seg for å nå denne destinasjonen. Denne aktiviteten fungerer også for uteskole i koronatider! I andre aktiviteter kan en elev i et elevpar lage en forklaring på hvordan en enkel tegning kan tegnes uten at den andre eleven ser på. Googler man «unplugged programming», finner man mange gode forslag til denne typen aktiviteter.

Ifølge Grover et al. (2015) vil det være viktig å bevisstgjøre elevene på sammenhengen mellom det konkrete innholdet i aktiviteten og de tilhørende programmeringsbegrepene. For destinasjonsaktiviteten kan det gjøres ved at elevene skal gjennomføre aktiviteten en gang til, men til en annen destinasjon som bare den ene eleven i elevparet vet om. Denne gangen kan elevene bare bruke ferdige lapper for å lage oppskriften sin. Lappene kan for eksempel se ut som de i figur 2 med ulike programmeringsbegreper på. Etter gjennomført aktivitet kan læreren ta opp hvilke programmeringsbegreper de nå har brukt, hva som var formålet med dem, og hensiktsmessige bruksområder, og drøfte dette med elevene. Ved senere programmeringsaktiviteter kan læreren hjelpe elevene med å vedlikeholde disse begrepene ved å repetere dem sammen med elevene, og gjerne koble dem mot bruk i andre sammenhenger. Dette vil kunne styrke elevenes forståelse av begrepene da det blir bevisstgjøring av begrepene samtidig som de abstrakte begrepene blir eksemplifisert i flere ulike sammenhenger. Slike analoge aktiviteter har jeg erfaring med at også fungerer på en fin måte for å få nye klasser til å bli kjent med hverandre. Det virker som en ganske ufarlig måte å samarbeide på, og det fungerer for elever fra skrivedyktig alder til voksne lærere.



Figur 2: Eksempler på ferdige lapper til destinasjonsaktiviteten.

Man kan vurdere om elevene i utgangspunktet skal bruke blokk- eller tekstbasert programmering. LK20 krever ikke tekstbasert programmering på ungdomstrinnet. Alle kompetansemålene kan nås ved hjelp av blokkbasert programmering. Men dette kan potensielt endres når man får tilgang til eksamensoppgaver for 10. trinn, og flertallet av forlagene har valgt å satse på det tekstbaserte programmeringsspråket Python. Meerbaum-Salant et al. (2013) viser til at blokkbaserte programmeringsspråk er fine å bruke for nybegynnere innen programmering, da man slipper å fokusere så mye på den detaljerte syntaksen, som semikolon, parenteser, innrykk og stor bokstav. Type programmeringsspråk kan inngå i lærerens differensieringsstrategi, da de fleste programmeringsoppgaver på ungdomstrinnet vil kunne gis for begge typer.

Det å tilpasse opplæringen innen programmering kan også gjøres ved å ta i bruk ulike oppgavetyper som innebærer større eller mindre grad av stillasbygging rundt elevens

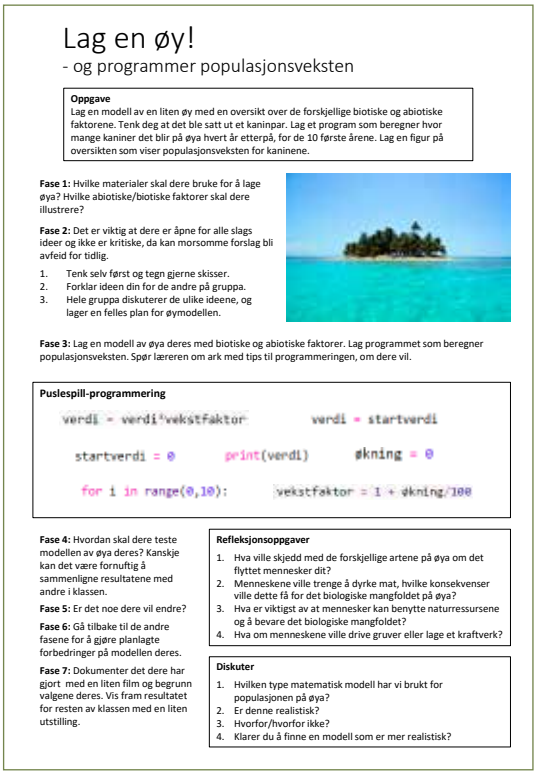
Flø, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

forståelse. Harms et al. (2016) oppsummerer at forskningen på feltet gjerne tar utgangspunkt i å redusere elevenes opplevde vanskegrad gjennom at elevene får andre typer oppgaver enn å skrive egen kode fra bunnen. Noen eksempler på slike oppgavetyper:

- Et ferdig program som elevene skal forklare hva gjør
- Et ferdig program som elevene skal beskrive «output» til
- Begynnelsen av et program som de skal skrive ferdig
- Kode med feil som elevene skal rette opp
- Kode der elevene skal bytte ut en del (substituering)
- Puslespilloppgaver der elevene får koden hulter til bulter («Parsons problems»)
- Sammensatte oppgaver basert på PRIMM-modellen

Ericson et al. (2017) viser at å skrive kode fra bunn er krevende for elevene, mens det å endre, justere eller tilpasse ferdig kode oppleves som enklere. Det å hjelpe elever med å kategorisere kodesnutter virker positivt på deres læring i følge Ericson et al., og henger trolig sammen med at elevene på den måten får hjelp til å abstrahere begrepene og gjøre dem mer generelle. Da vil det kunne være enklere for elevene å bruke begrepene i nye situasjoner. To av oppgavetypene det er forsket mest på, er «Parsons problems» og PRIMM-modellen for programmeringsoppgaver. I figur 3 og figur 4 er det eksempler på begge oppgavetypene bygget inn i større undervisningsopplegg for matematikk (og naturfag).

«Parsons problems» kan gis med eller uten informasjon om innrykk (som er sentralt i Python da det definerer hva som hører til for eksempel en løkke) og med eller uten ekstra linjer med kode (eller kodeblokker) som ikke skal benyttes i det ferdige programmet. Dette kalles gjerne distraktorer, og alt etter hvordan de utformes, kan de brukes til målrettet trening


Lag en øy!

- og programmer populasjonsveksten

Oppgave
Lag en modell av en liten øy med en oversikt over de forskjellige biotiske og abiotiske faktorene. Tenk deg at det ble satt ut et kaninpar. Lag et program som beregner hvor mange kaniner det blir på øya hvert år etterpå, for de 10 første årene. Lag en figur på oversikten som viser populasjonsveksten for kaninene.

Fase 1: Hvilke materialer skal dere bruke for å lage øya? Hvilke abiotiske/biotiske faktorer skal dere illustrere?

Fase 2: Det er viktig at dere er åpne for alle slags ideer og ikke er kritiske, da kan morsomme forslag bli avfeid for tidlig.

1. Tenk selv først og tegn gjerne skisser.
2. Forklar ideen din for de andre på gruppa.
3. Hele gruppa diskuterer de ulike ideene, og lager en felles plan for øymodellen.

Fase 3: Lag en modell av øya deres med biotiske og abiotiske faktorer. Lag programmet som beregner populasjonsveksten. Spør læreren om ark med tips til programmeringen, om dere vil.

Puslespill-programmering

```
verdi = verdi*vekstfaktor        verdi = startverdi
startverdi = 0        print(verdi)        økning = 0
for i in range(0,10):        vekstfaktor = 1 + økning/100
```

Fase 4: Hvordan skal dere teste modellen av øya deres? Kanskje kan det være fornuftig å sammenligne resultatene med andre i klassen.

Fase 5: Er det noe dere vil endre?

Fase 6: Gå tilbake til de andre fasene for å gjøre planlagte forbedringer på modellen deres.

Fase 7: Dokumenter det dere har gjort med en liten film og begrunn valgene deres. Vis fram resultatet for resten av klassen med en liten utstilling.

Refleksjonsoppgaver

1. Hva ville skjedd med de forskjellige artene på øya om det flyttet mennesker dit?
2. Menneskene ville trenge å dyrke mat, hvilke konsekvenser ville dette få for det biologiske mangfoldet på øya?
3. Hva er viktigst av at mennesker kan benytte naturressursene og å bevare det biologiske mangfoldet?
4. Hva om menneskene ville drive gruver eller lage et kraftverk?

Diskuter

1. Hvilken type matematisk modell har vi brukt for populasjonen på øya?
2. Er denne realistisk?
3. Hvorfor/hvorfor ikke?
4. Klarer du å finne en modell som er mer realistisk?

Figur 3: Eksponentialfunksjoner kombinert med økologi. Et «Parsons problem» der elevene skal modellere populasjonsvekst. Oppgaven kan byttes ut med enklere eller mer krevende alternativ.

på forskjellige nivåer. For eksempel vil man kunne oppgi en kodelinje med to alternativer, der eneste forskjell mellom dem er en stor/liten bokstav i et variabelnavn. Da trener man elevene i syntaks, og man kan samtidig ta opp viktigheten av språklig presisjon innen programmering. Andre alternativer kan være å fokusere på det matematiske innholdet til et program, der man kan ha to alternativer til hvor stor vekstfaktoren blir ved 2,0 % årlig rente. Bruk av distraktorer gjør at slike oppgaver oppleves som vanskeligere for elevene enn om de får oppgaver uten distraktorer (Harms et al., 2016). Generelt gir det å jobbe med «Parsons problems» minst like stort læringsutbytte som å skrive egen kode, og det skjer også mer effektivt, altså på kortere tid (Ericson et al., 2017).

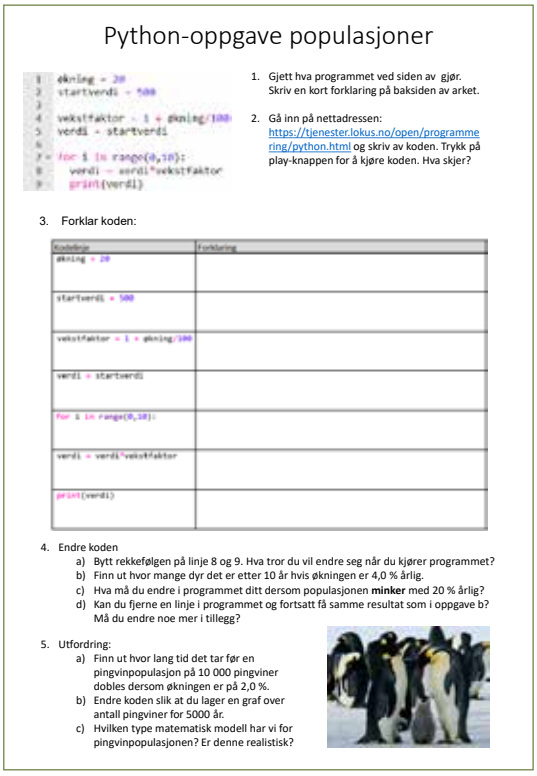
Flø, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

www.caspar.no/tangenten

PRIMM-modellen beskriver en sammensatt oppgavetype (eller måte å strukturere timene innen programmeringsopplæring på), der det legges stor vekt på samarbeid og diskusjon av hva som skjer i programmene, og på å gi elevene nok stillaser for å støtte elevenes læring av programmeringsbegreper. PRIMM er forkortelse for predict, run, investigate, modify og make, og beskriver gangen i modellen. Først får elevene utdelt en kode / et program som de skal forutsi hva gjør, deretter skal elevene kjøre koden og se hva som faktisk skjer, før de videre undersøker koden gjennom å jobbe med spørsmål, feilsøking e.l. Deretter skal elevene modifisere eller endre koden, og til slutt kan de skrive en helt egen kode, og i dette arbeidet fremheves det å la elevene jobbe sammen, gjerne i par. Sentance et al. (2019) fant at PRIMM-modellen er en lovende tilnærming til programmeringsopplæring, også for elevene som strever faglig. Se gjerne nettsidene til PRIMM-prosjektet ved King's College London – https://blogs.kcl.ac.uk/cser/research-projects/primm-project/.

Det har i flere tiår vært debattert hvordan lærere bør strukturere timene innen programmeringsopplæring. Debatten dreier seg i stor grad om hvor stor frihet elevene skal få, versus hvor styrt undervisningen bør være. Grover et al. (2015) oppsummerer at undervisningsopplegg der elevene får utforske fritt, ser ut til å gi liten kompetanse med hensyn til tidsbruk. På den andre siden vil hefter/bøker/nettsteder med forklaringer og gjennomgåtte eksempler heller ikke være spesielt effektive for læring (Harms et al., 2016). Da disse markerer to ytterpunkter for styring/frihet, vil det ikke være utenkelig at noe midt imellom vil fungere best, gjerne en form for moderat styrt utforskende tilnærming.

Men hvordan er sammenhengen mellom programmering og det å designe og utforme fysiske gjenstander? Det kalles gjerne for programmering av fysiske objekter, og Marshall (2007) fant ut at dette gir en positiv effekt på samarbeidslæring og aktiv læring. Videre fant Austin et al. (2020) og Sentance et al. (2017) at



Figur 4: Oppgave basert på PRIMM-modellen. Kan også brukes i Figur 3 som alternativ til elever som trenger mer støtte.

elevenes interesse for IKT-fag (computer science) stimuleres, og at elevenes forståelse for og holdninger til IKT-fag bedres. Om det bidrar direkte til at elevene lærer mer, er foreløpig ikke undersøkt.

Kombinasjonen av ønskene om å programmere fysiske objekter med det å jobbe skapende og utforskende (som kreves i LK20) for å lære matematikk (og gjerne andre fag) har munnet ut i undervisningsopplegg basert på en skaperverkstedmetode, med de syv fasene nevnt tidligere. Her er skaperverksted ikke et fysisk rom med mye spennende teknologi (som trolig er mindre gjennomførbart på alle skoler med presset økonomi). Det er heller en måte å jobbe på, der hovedfokuset er på det faglige innholdet, heller enn på digitale verktøy. Samtidig kan man gjerne bruke digitale verktøy som er tilgjengelig,

Flø, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

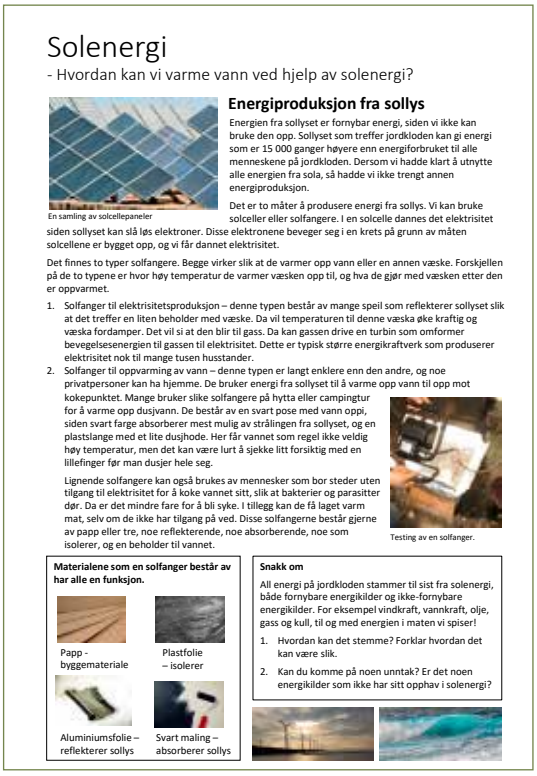
men det blir et verktøy for å nå kompetansemål, og er ikke et mål i seg selv. Gjennom det landsdekkende super:bit-prosjektet får alle barneskoler tildelt programmeringsutstyr basert på micro:bit (https://www.superbit.no/). Denne typen utstyr kan bidra til større muligheter for å koble sammen faginnhold med programmeringen. For å holde fokus på matematikken bør man primært ta utgangspunkt i fagets kjerneelementer og kompetansemål, og deretter se på teknologi som fremmer forståelse.

Elevene samarbeider her i små grupper, der de jobber både praktisk med å lage en fysisk gjenstand og teoretisk med ulike matematiske begreper og beregninger. De jobber med problemløsning, og får teste problemløsningsstrategier, mens de jobber og selv merker behov for denne typen strategier. Her kommer også algoritmisk tenkning inn som en mulig problemløsningsstrategi ved at elevene veiledes i å jobbe systematisk og bryte komplekse problemer opp i mindre og enklere oppgaver. Se gjerne UDIRs nettsider om algoritmisk tenkning for mer detaljer (UDIR, 2019).

Den fysiske gjenstanden som elevgruppene samarbeider om å lage, fungerer som en prototype. Det vil si at det er meningen at det alltid vil være mulig å forbedre den, og gjerne i flere omganger, slik at neste versjon virker litt bedre. Da vil det forhåpentligvis føles som mindre farlig å gjøre feil, siden det blir en naturlig del av prosessen. Den fysiske gjenstanden, og utstyret elevene bruker for å lage denne, kan da fungere som konkretiseringsmateriell for abstrakte begreper og sammenhenger. Dette skal jeg se nærmere på i mitt doktorgradsarbeid.

For å bidra konkret til innføringen av LK20 har jeg sammen med det som tidligere var Snöball Film fått støtte fra Udir til å utvikle en digital gratisressurs for denne typen undervisningsopplegg i hovedsak for ungdomstrinnet. Den vil stå ferdig omtrent til høsten 2021 på www.kunnskapsfilm.no.

Store deler av denne ressursen kan være av interesse for videregående skole, spesielt de


Solenergi

- Hvordan kan vi varme vann ved hjelp av solenergi?

En samling av solcellepaneler

Energiproduksjon fra sollys

Energien fra sollyset er fornybar energi, siden vi ikke kan bruke den opp. Sollyset som treffer jordkloden kan gi energi som er 15 000 ganger høyere enn energiforbruket til alle menneskene på jordkloden. Dersom vi hadde klart å utnytte alle energien fra sola, så hadde vi ikke trengt annen energiproduksjon.

Det er to måter å produsere energi fra sollys. Vi kan bruke solceller eller solfangere. I en solcelle dannes det elektrisitet siden sollyset kan slå løs elektroner. Disse elektronene beveger seg i en krets på grunn av måten solcellene er bygget opp, og vi får dannet elektrisitet.

Det finnes to typer solfangere. Begge virker slik at de varmer opp vann eller en annen væske. Forskjellen på de to typene er hvor høy temperatur de varmer væsken opp til, og hva de gjør med væsken etter den er oppvarmet.

1. Solfanger til elektrisitetsproduksjon – denne typen består av mange speil som reflekterer sollyset slik at det treffer en liten beholder med væske. Da vil temperaturen til denne væska øke kraftig og væska fordamper. Det vil si at den blir til gass. Da kan gassen drive en turbin som omformer bevegelsesenergien til gassen til elektrisitet. Dette er typisk større energikraftverk som produserer elektrisitet nok til mange tusen husstander.
2. Solfanger til oppvarming av vann – denne typen er langt enklere enn den andre, og noe privatpersoner kan ha hjemme. De bruker energi fra sollyset til å varme opp vann til opp mot kokepunktet. Mange bruker slike solfangere på hytta eller campingtur for å varme opp dusjvann. De består av en svart pose med vann oppi, siden svart farge absorberer mest mulig av strålingen fra sollyset, og en plastslange med et lite dusjhode. Her får vannet som regel ikke veldig høy temperatur, men det kan være lurt å sjekke litt forsiktig med en lillefinger før man dusjer hele seg.

Lignende solfangere kan også brukes av mennesker som bor steder uten tilgang til elektrisitet for å koke vannet sitt, slik at bakterier og parasitter dør. Da er det mindre fare for å bli syke. I tillegg kan de få laget varm mat, selv om de ikke har tilgang på ved. Disse solfangerne består gjerne av papp eller tre, noe reflekterende, noe absorberende, noe som isolerer, og en beholder til vannet.

Testing av en solfanger.

Materialene som en solfanger består av har alle en funksjon.

Papp - byggemateriale

Plastfolie – isolerer

Aluminiumsfolie – reflekterer sollys

Svart maling – absorberer sollys

Snakk om

All energi på jordkloden stammer til sist fra solenergi, både fornybare energikilder og ikke-fornybare energikilder. For eksempel vindkraft, vannkraft, olje, gass og kull, til og med energien i maten vi spiser!

1. Hvordan kan det stemme? Forklar hvordan det kan være slik.
2. Kan du komme på noen unntak? Er det noen energikilder som ikke har sitt opphav i solenergi?

Figur 5: Opplegg om modellering og solenergi det elevene bygger solfanger og måler temperaturøkningen med en micro:bit. Rettet mot ungdomstrinn, kan brukes på mellomtrinn med enklere fagstoff[1].

første årene etter innføringen av LK20. Ressurser spesielt for videregående utvikles også av dyktige lærere som deltar i forskningsprosjektet ProSkap-SL, og vil publiseres etter hvert. I prosjektet ønsker vi også å se på vanskegrad for ulike oppgavetyper innen programmering, og hvordan de kan bidra til tilpasset opplæring. Derfor har vi laget flere versjoner av en Pythonbasert programmeringstest ut fra kompetansemålene i programmering i grunnskolen, som vi ønsker at flest mulig gjennomfører. Både lærere, forskere og elever som har noe programmeringskunnskap, inviteres til å delta og til å være med i trekningen av et klassesett micro:bit (10 stykker) til sin skole. Testen er anonym, og finnes her: https://nettskjema.no/a/prog.

Lykke til med programmeringsundervisning i matematikken, folkens!

Fløe, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

Note

1 Opplegget er rettet mot ungdomstrinnet, kan legges til rette for mellomtrinnet og kan også tilpasses oppover ved å inkludere kalibrering og mer avansert fagstoff. Det er også mulig å måle temperaturen med en micro:bit og sende data til en annen micro:bit, som igjen kan logge data på en datamaskin, eller temperaturdata kan skrives til en fil på micro:biten som igjen kan behandles videre ved hjelp av Python. Alle disse forskjellige tilpasningene vil ligge tilgjengelig på den nye gratisressursen Kobling B på www.kunnskapsfilm.no.

Referanser

Austin, J., Baker, H., Ball, T., Devine, J., Finney, J., De Halleux, P., Hodges, S., Moskal, M. & Stockdale, G. (2020). The BBC micro:bit: From the UK to the world. *Communications of the ACM, 63*(3), 62–69.

Ericson, B. J., Margulieux, L. E. & Rick, J. (2017). Solving parsons problems versus fixing and writing code. I C. S. Montero & M. Joy (red.), *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (s. 20–29). ACM.

Grover, S., Pea, R. & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Harms, K. J., Chen, J. & Kelleher, C. L. (2016). Distractors in parsons problems decrease learning efficiency for young novice programmers. I J. Sheard, J. Tenenberg, D. Chinn & B. Dorn (red.), *Proceedings of the 2016 ACM Conference on International Computing Education Research* (s. 241–250). ACM.

Marshall, P. (2007). Do tangible interfaces enhance learning? I B. Ullmer & A. Schmidt (red.), *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (s. 163–170). ACM.

Meerbaum-Salant, O., Armoni, M. & Ben-Ari, M. M. (2013). Learning computer science concepts with scratch. *Computer Science Education, 23*(3), 239–264.

Sentance, S., Waite, J., Hodges, S., MacLeod, E. & Yeomans, L. (2017). «Creating cool stuff»: Pupils' experience of the BBC micro:bit. I M. E. Caspersen & S. H. Edwards (red.), *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (s. 531–536). ACM.

Sentance, S., Waite, J. & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education, 29*(2), 136–176.

Utdanningsdirektoratet (UDIR) (2019). *Algoritmisk tenkning*. www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Flø, E. E. (2021). Programmering i LK20 *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.