LEGO Mindstorms og MATLAB; anvendt matematikk/fysikk og programmering i skjønn forening

Tormod Drengstig
Institutt for data- og elektroteknikk
Universitetet i Stavanger
tormod.drengstig@uis.no

Sammendrag

Ved å *kombinere* LEGO Mindstorms og MATLAB kan studenter introduseres for programmering på en måte som i stor grad er prosjektbasert og som innebærer såkalt *studentaktiv læring.* Denne artikkel beskriver hvordan Universitetet i Stavanger benytter denne kombinasjonen for data- og elektrostudenter i første semester som en del av Ingeniørfaglig innføringsemne. Det blir også beskrevet hvordan studentene blir organisert i grupper og hvilke forutsetninger som må til for at *én* faglig ansvarlig skal kunne veilede 160 studenter og gi faglige tilbakemeldinger til samtlige. Videre beskrives hvordan det tilrettelegges for å oppnå individuell læring i gruppearbeidet. Programmeringen er knyttet opp mot praktiske anvendelser fra fysikken som gjør bruk av anvendt matematikk i form av blant annet numerisk integrasjon, filtrering og numerisk derivasjon. Dette åpner for mange motiverende prosjekter som studentene har stor glede av å gjennomføre.

1 Introduksjon

Bakgrunn

Emnet Ingeniørfaglig innføringsemne (ING100) ble fra høsten 2012 innført som et obligatorisk emne ved alle institusjoner i Norge som tilbyr bachelorstudier i ingeniørfag. Ved Universitetet i Stavanger (UiS) ble dette implementert som et todelt emne, med en felles del for alle ingeniørretningene og en fagspesifikk prosjektdel for hver av retningene *i*) maskin, *ii*) bygg, *iii*) kjemi, *iv*) petroleum, *v*) data og elektro. Temaet for fellesdelen er grunnleggende bruk av MATLAB, som både er et matematikk- og et programmeringsverktøy [1] (erfaringene med fellesdelen er beskrevet i [2]). Temaet for den fagspesifikke delen for retningen data og elektro er programmering av LEGO Mindstorms ved bruk av MATLAB. Siden LEGO-settene inneholder flere motorer og flere typer instrumentering (ultralydsensor, lyssensor, mikrofon, akselerometer, trykkbryter), er det store muligheter til å benytte dette innen temaet anvendt matematikk og fysikk.

Denne artikkelen ble presentert på konferansen NIK-2016; se http://www.nik.no/.

Motivasjon

Institutt for data- og elektroteknikk (IDE) utdanner data- og elektroingeniører på både bachelor- og masternivå, og felles for mange av instituttets egne fag er at de inneholder programmering, dog i varierende omfang. Av den grunn har elektrostudentene (til sin egen overraskelse) tradisjonelt sett opplevd at omfanget av programmering i utdanningen er større enn forventet. Ved innføringen av faget ING100 i 2012 benyttet IDE muligheten til å la den fagspesifikke delen bestå av små tematiske smakebiter fra noen av fagene som elektro-/datastudentene møter gjennom studiet, pakket inn i en ramme av programmering. I hovedsak er dette fag som matematikk, fysikk, instrumenteringsteknikk, datateknikk, signaler og systemer, samt reguleringsteknikk. Hovedutfordring var å finne et konsept som kunne integrere disse smakebitene og samtidig være på et tilstrekkelig overfladisk nivå til at studentene i første semester ville få utbytte av det.

Bakgrunnen for at ING100 ble todelt (først 6 uker forelesning og deretter 7 uker prosjekt), var de gode erfaringene med tilsvarende organisering i tidligere introduksjonsfag til programmering. Måten prosjektdelen gjennomføres på er derfor en videreutvikling av opplegget i disse tidligere fagene.

Ved innføringen av ING100 ble programmeringsspråket endret fra Java (brukt i det tidligere introduksjonsfaget) til MATLAB. Fordelene med MATLAB er at det lett å komme i gang, brukergrensesnittet er oversiktlig, det er interpreterende (trenger ikke å kompileres som Java), det er ikke objektorientert (som Java), velfungerende debug-modus for feilsøking, det gir god støtte til bruk i basisfag som Matematikk 1 og 2, og sist men ikke minst, MATLAB brukes i mange av fagene som studentene har senere i studiet.

Problemstilling

Tatt i betraktning at LEGO Mindstorms og FIRST LEGO League [3] har vært en suksess i grunnskolen (også i den videregående skole), var det naturlig å vurdere hvorvidt LEGO Mindstorms også kunne være aktuelt å anvende på universitetsnivå. Programmeringen i grunnskolen gjøres på PC ved hjelp av et grafisk programmeringsspråk hvor eksempelvis *for*- og *while*-løkker er grafiske byggeklosser, som deretter kompileres og overføres til selve hjernen i roboten, også kalt NXT-enheten (i siste versjon av LEGO Mindstorms heter enheten EV3 [4]). Siden denne form for programmering ikke representerer det som undervises på universiteter og høgskoler, undersøkte vi om MATLAB kunne brukes mot LEGO Mindstorms, og om dette kunne kjøres i sann tid.

Det viste seg at RWTH i Aachen [5] hadde utviklet en egen verktøykasse (*toolbox*) til MATLAB for nettopp dette formålet, og denne verktøykassen er nå en del av Mathworks sitt tilbud [6]. Ved å bruke denne verktøykassen kjøres MATLAB-programmene på PC som igjen kommuniserer med NXT-enheten enten via USB-kabel eller trådløst (blåtann). Den delen av MATLAB-koden som innebærer avlesning av sensorverdier og påtrykking av motorpådrag implementeres typisk i en *while*-løkke som gjør at MATLAB kommuniserer med NXT-enheten repetetivt og i sann tid. Tidsskrittet T_s som representer tiden for en gjennomkjøring i *while*-løkken, varierer typisk mellom 20 og 200 ms, avhengig av andre prosesser i MATLAB (for eksempel plotting) eller andre prosesser på PC-en.

Artikkelens videre innhold

I kapittel 2 beskrives hvordan prosjektet er organisert, hva som må til for at veiledningen skal være effektiv og hvordan man oppnår individuell læring i gruppearbeidet. I kapittel 3 beskrives den obligatoriske delen av prosjektet som alle gruppene må gjennomføre. Dette er forventet minimum, og tilsvarer karakteren E. For å illustrerer mangfoldet av læringsmuligheter med LEGO Mindstorms og MATLAB, er det i kapittel 4 gitt et utdrag av prosjektforslagene studentene kan velge blant. I kapittel 5 presenteres vurderingsform og eksamensresultatene for de 4 siste årene, og diskusjon/konklusjon blir gitt i kapittel 6.

2 Om prosjektet

Organisering

Antall data- og elektrostudenter som har tatt faget har økt fra 120 i 2012 til 160 i 2015. For at prosjektet skal være gjennomførbart med såpass mange studenter, blir de delt i inn egenkomponerte grupper på mellom 3 og 4 personer. De studentene som ikke finner noen å danne gruppe med, settes sammen i grupper av fagansvarlig. Dermed har det siden starten i 2012 vært fra 35 til 42 grupper som hvert år har fått utdelt et LEGO Mindstroms-sett og en styrestikke (*joystick*), se figur 1.

Figur 1: Styrestikken som brukes i prosjektet (Logitech).

Hensikten med å bruke en styrestikke er at dette er et velkjent grensesnitt for mange av studentene, og det forenkler interaksjonen med roboten i sann tid. Et alternativ er å bruke tastaturet som inndata, men det er ikke like funksjonelt. Styrestikken kan roteres omkring 3 akser og er utstyrt med 11 trykkbrytere og ett potensiometer. Det utdelte utstyret er studentene økonomisk ansvarlige for, og de kan enten ta det med seg hjem eller låse det inn i dedikerte skap på universitetet.

Studentene får videre utdelt et dokument (ca. 80 sider) som beskriver hva prosjektet går ut på. Her gis det også en innføring i de matematiske metodene som de skal implementere.

Prosjektet disponerer 2 laboratorierom med totalt 20 arbeidsbenker, og ved hver arbeidsbenk er det plassert én stasjonær PC og 4 stoler. Dette gjør at inntil 80 studenter kan jobbe med prosjektet samtidig, og veiledning blir til enhver tid utført av 4 studentassistenter. Hver gruppe har 6 timer hver uke med denne type veiledning, fordelt på 4 + 2 timers økter.

Veiledning med fagansvarlig

Gruppedynamikk

Erfaringene med tidligere introduksjonsfag til programmering har vist at det er nyttig med fysiske møter mellom gruppene og fagansvarlig. Den pedagogiske motivasjonen for dette er at fagansvarlig kan svare på spørsmål som studentassistentene ikke alltid har forutsetninger til å svare på, eller at fagansvarlig kan gi alternative svar i forhold til studentassistenenes svar. Videre gir møtene nyttig innsikt i gruppedynamikken og fremdriften, og møtene har derfor prinsipielle likheter med hvordan veiledningen til bachelor- og masteroppgaver fungerer.

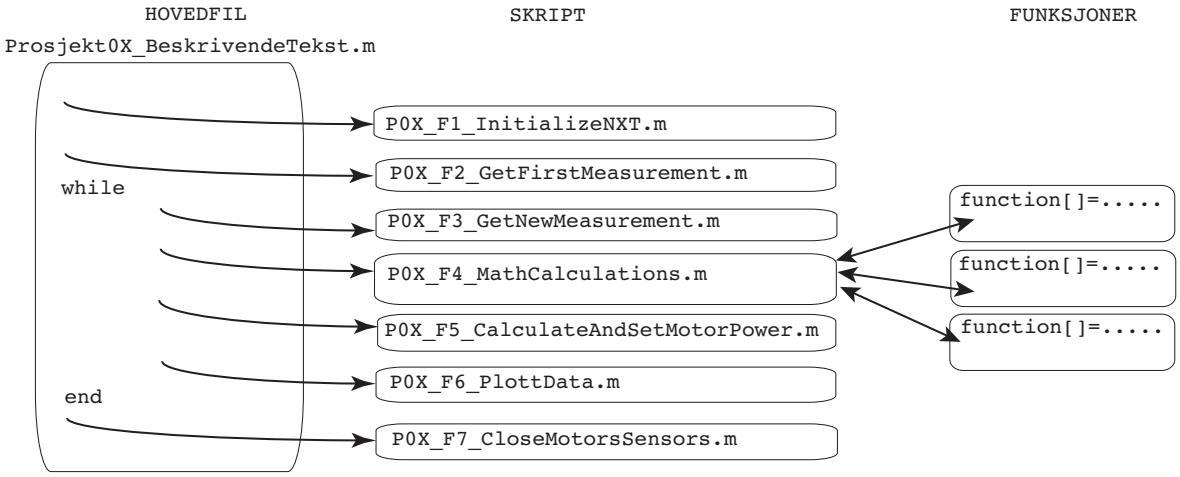
Basert på at fagansvarlig setter av en hel dag hver uke i prosjektperioden, har hver gruppe 3 veiledningsmøter á 15 minutter i løpet av prosjektet. Studentene har møteplikt og fagansvarlig er relativt direkte i kommunikasjonen i den forstand at forventingene til hver students bidrag innad i gruppen blir tydelige kommunisert. Beskjeden som formidles er at hver student skal bidra med utgangspunkt i sitt eget kompetansenivå og at gratispassasjerer er noe hele gruppen taper på. Siden svært få studenter ønsker å være til "byrde" for en gruppe, er denne metoden effektiv til å få de som i utgangspunktet har liten erfaring med programmering til å føle at de faktisk kan bidra (om nødvendig får de egne prosjekt som de jobber med). Videre er denne direkte kommunikasjonen effektiv i å avdekke potensielle konflikter slik at disse kan håndteres i en tidlig fase. Dersom det obligatoriske spørsmålet "Hvordan synes dere gruppen fungerer?" skaper betenkningstid eller at noen trekker litt på svaret, prøver fagansvarlig å nøste opp i hva som er det underliggende problemet. Ofte er problemet at en eller flere studenter er veldig passive, enten fordi programmering oppleves som kjedelig/vanskelig eller at de rett og slett ikke deltar i gruppearbeidet eller ikke møter opp til avtalte tider. I slike tilfeller blir disse studentene tildelt egne prosjekt og på den måten får de eneansvar for å produsere kode. Dette fører i noen tilfeller til at gruppen igjen blir samlet og gruppemedlemmene blir fornøyd med hverandres innsats. I de tilfellene hvor dette ikke skjer, er siste utvei å innføre individuell bedømming av vedkommende. Dette kan enten fagansvarlig bestemme eller studentene selv be om, og det innebærer at gruppen leverer 2 rapporter. Denne type utfordringer gjør veiledningen litt mer krevende og ca 10-15% av gruppene opplever dette årlig. Det hender også at studenter trekker seg fra faget fordi de selv innser at de (av forskjellige årsaker) ikke bidrar.

Faglig tilbakemelding

I tillegg til at møtene med fagansvarlig gir et innblikk i gruppedynamikken, får også gruppene *faglig* tilbakemelding i møtene. Til hvert møte tar gruppene med seg en utskrift av *all* kode de har laget, og etter et raskt overblikk, er fagansvarlig i stand til å gi faglige tilbakemeldinger. En forutsetning for å klare dette er at studentene benytter *programstrukturen* vist i figur 2.

Programstrukturen består en av relativt enkel hovedfil som i figur 2 kalles `ProsjektOX_BeskrivendeTekst.m` (studentene endrer dette filnavnet for hvert prosjekt). Denne er på ca. 20 kodelinjer, og består hovedsaklig av kall til følgende skript:

- `POX_F1_Initialize.m` som initialiserer NXT-enheten, motorene og sensorene.
- `POX_F2_GetFirstMeasurement.m` som henter første måling og definerer de datavektorene som trengs for lagring av måleverdier, pådragsverdier og beregnede verdier.



Figur 2: Generell programstruktur med hovedfil med tilhørende skript og funksjoner, hvor bokstaven `X` i filnavnene erstattes med et tall tilsvarende prosjektnummeret.

- `P0X_F3_GetNewMeasurement.m` som iterativt inne i *while*-løkken henter nye data fra styrestikken og nye målinger fra sensorene og lagrer disse i datavektorer.
- `P0X_F4_MathCalculations.m` som i utgangspunktet er tom, men hvor beregninger utføres og som kaller på forskjellige funksjoner.
- `P0X_F5_CalculateAndSetMotorPower.m` som beregner pådraget til motorene og sender dette til NXT-enheten.
- `P0X_F6_PlottData.m` som presenterer dataene fortløpende i figurer.
- `P0X_F7_CloseMotorsAndSensors.m` som stopper motorene, avslutter koplingen til styrestikken, lukker alle sensorene og til slutt avslutter koplingen mot NXT-enheten.

Studentene får utdelt denne strukturen med ferdig kode for å bruke styrestikken til å justere turtallet til én LEGO-motor. På den måten har alle gruppene et utgangspunkt i noe som virker, og ved å benytte denne strukturen for hvert nye prosjekt trenger fagansvarlig stort sett kun å fokusere på koden i `P0X_F4_MathCalculations.m` og `P0X_F5_CalculateAndSetMotorPower.m`, med tilhørende funksjonskall.

Ideen om å benytte en felles programstruktur kom som en følge av frustrasjonen en god del studenter opplevde de første par årene hvor det ikke var føringer for hvordan kode kan/bør organiseres. Studentene mistet rett og slett oversikten over egne program siden all kode var organisert i én fil, og denne filen ble kopiert fra prosjekt til prosjekt. Programmene ble på denne måten ofte veldig lange og inneholdt rester av gammel kode fra andre prosjekt. Det var også vanskelig for fagansvarlig å gi faglig tilbakemelding i løpet av 15 minutter på program som ikke hadde noen kjente strukturelle likhetstrekk.

Hvordan oppnå individuell læring i gruppearbeid?

Som de fleste som har programmert kjenner til, er programmering et fag som best læres ved å prøve og feile. Videre er det slik at i en gruppe på 4 studenter er det ofte den eller de som kan noe om programmering fra før som blir sittende med tastaturet

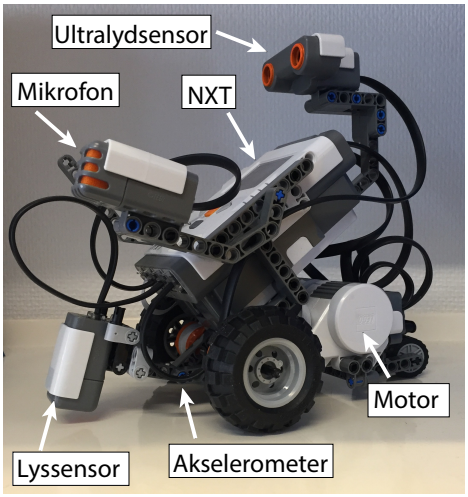
mens de andre ser på. Læringsutbytte for de som ser på er derfor tilnærmet lik null. For å bøte på dette er det gjort flere tiltak:

1. Studentassistentene er instruert til å observere gruppene med hensyn til hvem som sitter med tastaturet, og høflig henstille gruppene om å rotere på hvem som programmerer.
2. Alt installasjonsmateriell (MATLAB, LEGO Mindstorms Toolbox og driver til styrestikke) er gjort tilgjengelig slik at studentene kan installere dette på egne (bærbare) PC-er. Ved å benytte Boot Camp kan også Mac benyttes. Dette, sammen med at gruppemedlemmene kan ta med seg LEGO-settet hjem, gjør at hver student har mulighet til å jobbe på egenhånd med koden.
3. Det siste og viktigste tiltaket er muligheten for å kjøre koden uten at NXT-enheten er koplet til PC. Dette er nyttig i de tilfeller hvor kun måledata skal behandles (noe som gjelder for mange av prosjektene). Dersom koden beregner motorpådrag som igjen påvirker målingene, er ikke denne offline-måten mulig. For å jobbe offline, gjennomfører gruppen først det aktuelle eksperimentet, lagrer alle måledata og distribueres disse seg imellom. I hovedfilen settes deretter en parameter kalt `online` til `false` (er i utgangspunktet satt til `true`), og dette påvirker alle skriptene på en slik måte at det er mulig å benytte samme filstruktur til å utvikle kode uten at NXT-enheten er tilkoplet. Studentene kan da jobbe på egenhånd, enten hjemme eller på universitet. For å teste ferdig kode mot NXT-enheten settes `online=true`.

Disse tiltakene, kombinert med at arbeidsbenkene på universitetet er relativt store, resulterer ofte i at alle gruppemedlemmene sitter med hver sin bærebare PC og jobber individuelt på samme problemstilling.

3 Obligatorisk del

Prosjektet er delt opp i en obligatorisk del og en kreativ del. I den obligatoriske delen skal alle gruppene bygge det samme motoriserte kjøretøyet (også kalt roboten), se figur 3. Denne er inspirert av byggebeskrivelsen som følger LEGO Mindstorms-settet.

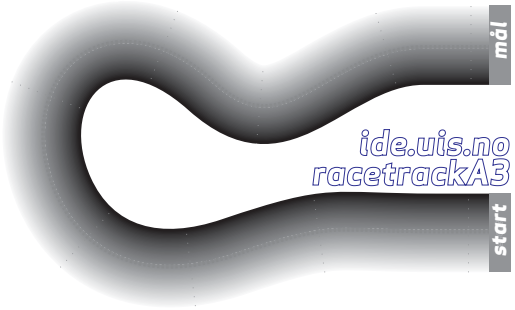


Figur 3: Kjøretøyet (roboten) som studentene bygger.

Roboten er instrumentert opp med lyssensor (består av en lyskilde og en mottaker som måler reflektert lys), ultralydssensor (består av en sender og en mottaker),

akselerometer og mikrofon, og den drives av to motorer som er koplet til hvert sitt forhjul. Et tredje hjul som roterer fritt er montert bakerst, og dette gjør at roboten kan svinge ved å anvende forskjellig motorkraft på motorene som styrer forhjulene.

Hensikten med å benytte et kjøretøy i den obligatoriske delen er at dette gir mange muligheter for studentene til å programmere egne løsninger på problemstillinger som mange har hørt om, slik som f.eks. Google Self-Driving Car Project [7]. Mange nye biler har nå utstyr som gjør at avstanden til bilen foran kan holdes konstant (både i ro og i fart), samt automatikk som sørger for å posisjonere den mellom gul og hvit stripe. For å anvende varianter av denne type teknologi på LEGO-roboten, får studentene utdelt en A3 versjon av baneprofilen vist i figur 4.



Figur 4: Bane for kjøring av robot. Selve banen er en 7cm bred strek som på tvers går fra hvitt til svart i økende gråtoner.

Ved hjelp av styrestikken skal studentene kjøre roboten fra start til mål på kortest mulig tid og med minst mulig avvik fra den *midterste* gråtonen. For å finne et kvantitativt mål på hvor bra hver student kjører, benyttes lyssensoren (se figur 3) til å avlese reflektert lysmengde (hvitt tilsvarer verdien 1023 og svart tilsvarer 0, noen som tilsvarer 10 bits oppløsning). Ved å plassere roboten ca. midt på banen på startfeltet og definere den første avleste gråtonen som et nullpunkt, kan absoluttverdien av avviket fra nullpunktet bli numerisk integrert mens roboten kjører, og dette er med på å definere hvor bra kjøringen er. Den numeriske integrasjonsmetoden som studentene presenteres for er Eulers forovermetode, men de oppfordres til å teste ut og sammenligne med alternative integrasjonsrutiner (for eksempel trapesmetoden).

I tillegg til å programmere rutiner for numerisk integrasjon, skal studentene også lage rutiner for numerisk derivasjon. Den pedagogiske motivasjonen for dette er at programmet skal finne ut hvorvidt roboten er på vei ut i "grøfta". Dersom den er på vei ut mot høyre, det vil si mot mørkere gråtoner og lavere lysverdier, vil den deriverte av lysverdiene være negativ og en høyrepil skal dukke på skjermen (og en tilsvarende venstrepil for når den svinger til venstre).

Siden lysverdiene er befengt med målestøy vil beregningen av den deriverte være direkte påvirket av støyen og dermed relativt ubrukelig. For å gjøre signalet mer egnet for derivering, må studentene først lage rutiner for filtrering. I dokumentet de får utdelt introduseres de for FIR-filter og første orden IIR-filter.

Ved å lage funksjoner for numerisk integrasjon, filtrering, og numerisk derivasjon, samt å kjøre roboten manuelt gjennom banen, har gruppen oppnådd karakteren E. Karakteren kan forbedres ved å gjøre flere prosjekter den i kreative delen.

4 Kreativ del

I den kreative delen skal studentene lage egne prosjekt eller velge blant en liste på ca 30 prosjektforslag. Typisk for prosjektforslagene er at de gjenbruker funksjonene for numerisk integrasjon, filtrering og derivasjon, og et lite utvalg av forslagene er presentert under.

- Få roboten til å kjøre automatisk langs baneprofilen (f.eks. ved å bruke en PID-regulator).
- Bruk lysmåleren til å estimer hastigheten på roboten ved å kjøre over et ark med svarte/hvite streker (som et fortau). Sammenlign med virkelig hastighet beregnet fra avlest vinkelposisjon til motoren som styrer hjulene.
- La roboten følge etter håndflaten i en fast avstand (f.eks. 10 cm), uavhengig av hastighet på hånden.
- Vis nedfoldingsfenomenet og relater dette til samplingsfrekvensen og Nyquist-frekvensen.
- Lag en hjemmelaget styrestikke ved hjelp av akselerometeret.
- La roboten kjøre mot en papplate og estimer hastigheten til roboten ved å integrere akselerometermålingen samt å derivere ultralydmålingen. Sammenlign med virkelig hastighet (avlest vinkelposisjon til motorene).
- Beregn pappeskeareal basert på en ultralydsensor som ved hjelp av en motor (hvor vinkelposisjonen avleses) roterer rundt inni en pappeske. Tegn opp pappeskens utforming (kan være for eksempel firkantet, sekskantet, rund, osv.). Sammenlign med virkelig areal.
- Lag en sykkelcomputer som viser akselerasjon, hastighet, gjennomsnittshastighet og tilbakelagt strekning. Bygg et sykkelhjul som styres av en LEGO-motor og hvor lyssensoren registrerer at hjulet roterer (tilsvarer magnetsensoren i en virkelig sykkelcomputer).
- Bygg en katapult som benyttes til å finne sammenhengen mellom hastighet/utgangsvinkel (avlest med endring av vinkelposisjon) og kastelengden målt med metermål. Sammenlign med teorien fra fysikken.

5 Eksaminering og resultater

Vurderingsform

Hver gruppe leverer inn til vurdering en rapport som beskriver prosjektene de har jobbet med. I tillegg leverer de inn en redigert film på inntil 3 minutt som viser alle prosjektene (hvert prosjekt blir filmet/dokumentert med mobiltelefonkamera). Eksempler på hva studentene har laget finnes på YouTube, se [8] og [9]. De faktorene som teller i vurderingen er

- antall studenter i gruppa
- antall prosjekter de har jobbet med
- prosjektenes kompleksitet
- rapportens kvalitet

Prosjektet blir gitt bokstavkarakteren A, B, C, D, E eller F. Prosjektet teller 60% av sluttkarakteren, og skriftlig eksamen i fellesdelen 40%. For å få *utdelt* karakteren må gruppa møte til en 15 minutters utspørring hvor gruppa først viser filmen og deretter svarer på spørsmål (denne utspørringen teller ikke på karakteren). 40 grupper blir på denne måten eksaminert i løpet av to intensive dager.

Karakterfordeling

I løpet av perioden siden faget ble startet opp i 2012 har ingen grupper strøket i prosjektdelen av faget, men flere grupper har hatt studenter som har sluttet underveis i prosjektet. Karakterfordelingen for gruppene i perioden 2012-2015 er vist i tabell 1. Antall grupper har i perioden vært mellom 35 og 42 pr. år, men siden det har vært tilfeller av individuell bedømming, er summen av antall karakterer noe større. Flest individuelle bedømminger var det i 2014.

Tabell 1: Eksamensresultater i perioden 2012-2015.

Karakter	År			
(antall/prosent)	2012	2013	2014	2015
A	11/29%	17/40%	10/18%	10/22%
B	13/34%	6/15%	13/25%	12/26%
C	9/24%	17/40%	17/32%	14/30%
D	5/13%	2/5%	11/21%	3/7%
E	0/0%	0/0%	2/4%	7/15%
# karakter	46/100%	42/100%	53/100%	46/100%

6 Diskusjon og konklusjon

Faget blir evaluert hvert år ved at studentene vurderer seg enig/ikke enig i 15 påstander om læringsutbytte, nytteverdi, gjennomføring og veiledning i faget. I tillegg oppfordres de til å gi generelle kommentarer og ris/ros om faget. Basert på disse tilbakemeldingene så opplever de fleste studentene faget som inspirerende, motiverende og ikke minst gøyalt. Det å anvende matematikk og fysikk i programmeringen har i tillegg gjort at mange studenter har fått en praktisk vinkling på integrasjon og derivasjon, og mange har fått en stor aha-opplevelse av å se hvordan integralet av en konstant a er blir til den stigende linjen $a{\cdot}t$. Fordelen med å jobbe med tiden t som abscisse istedenfor den generelle variabelen x som ofte brukes i matematikken, er at det tidsavhengige resultatet som presenteres i sann tid ofte bidrar til en alternativ forståelse av de matematiske begrepene. Etter at studentene har knekt denne koden, er det mange grupper som ved hjelp av lyssensoren og gråtonearket lager andre tidsvarierende funksjoner som de deretter integrerer og deriverer. Eksempler på slike funksjoner er $f(t){=}a{\cdot}t{+}b$, $f(t){=}a{\cdot}t^2$, $f(t){=}a{\cdot}\sin(b{\cdot}t)$ og $f(t) = \sin(t^{-1})$. En annen fordel med tiden t som abscisse er at læringen i dette faget kan knyttes direkte opp mot begrepet *dynamikk* som er svært viktig i fagene Elektroteknikk og Reguleringsteknikk.

I forhold til prosjektet ved RWTH Aachen beskrevet innledningsvis, så har det pågått siden 2007 [10]. Prosjektet har blitt beskrevet i flere artikler [11, 12], og innhold, resultat og den praktiske gjennomføringen av prosjektet har klare likhetstrekk med vårt ING100 prosjekt. Det som allikevel skiller seg ut som de største forskjellene er selve gjennomføringen og omfanget av veiledningen. Ved

RWTH [11] rapporteres det at prosjektet ble gjennomført i løpet av 8 hele dager (6 timer pr. dag), noe som gir totalt 48 timer med intensivt prosjekt, hvorpå de andre første-semesterfagene tok en pause i denne perioden. For å veilede de 309 studentene som tok faget brukte de 60 studentassistenter. Dette gir totalt ca. 2900 studentassistent-timer for hele prosjektet. For ING100-faget veiledes 150 studenter av kun 8-9 studentassistenter i løpet av den 7 uker lange prosjektperioden, noe som tilsvarer ca. 500 studentassistent-timer. Derimot har ING100-studentene 2 undervisningstimer i uka hvor fagansvarlig går gjennom enkle eksempler og avklarer typiske programmeringsmessige misforståelser, samt at de i tillegg har 3 veiledningsmøter med fagansvarlig. Noe tilsvarende beskrives ikke i [11]. Dersom prosjektet skulle vært gjennomført på samme måte som beskrevet i [11], ville dette krevd mer koordinering med andre fag samt at det kapasitetsmessig ikke ville vært gjennomførbart. Slik sett er det en fordel å gjennomføre prosjektet over lengre tid, og pedagogisk mener vi det er bedre siden programmering er et modningsfag hvor det å ha god tid til prøving og feiling er en vesentlig del av læringen. Ved at prosjektet går over 7 uker er forventet arbeidsinnsats fra hver student ca. 100 timer, og basert på timelistene som studetene leverer inn er det en klar sammenheng mellom innsats i form av timer og karakter i faget.

I prosjektet benyttes flere pedagogiske virkemidler som gjør det mulig for én vitenskapelig ansatt og 8-9 studentassistenter å gjennomføre et såpass omfattende prosjekt. De viktigste virkemidlene er som følger:

- Studentene får utdelt et lite eksempelprosjekt som fungerer, og de trenger derfor ikke bruke verdifull tid på å lese seg opp på detaljer i kommunikasjonsrutinene mellom NXT-enheten og MATLAB. Det å vurdere hvor mye studentene skal få av ferdig kode på forhånd er alltid vanskelig. Dersom det stilles for store forventninger til hva de skal klare på egenhånd, kan effekten fort bli det motsatte av hva vi ønsker, nemlig at motivasjonen synker som følge av at programmering oppleves som vanskelig og uangripelig.
- Studentene får utdelt et dokument som beskriver den matematiske teorien bak numerisk integrasjon, derivasjon og filtrering, og disse teknikkene blir knyttet opp mot LEGO-utstyret samt praktiske og gjenkjennbare eksempler fra fysikk og matematikk.
- Det at alle prosjektene er organisert med *identisk* filstruktur gjør veiledningsjobben for studentassistenter og fagansvarlige mye enklere.
- Ved å lagre loggede data fra LEGO-enheten kan studentene jobbe i MATLAB mot disse dataene i offline-modus og utvikle kode på egenhånd i ro og mak.
- De fysiske møtene som studentene har med fagansvarlig er med på å ansvarliggjør studentene. I tillegg fungerer møtene slik at hver enkelt student blir sett, noe som er med på å bryte ned følelsen av å være én blant mange.

Ut fra denne listen over pedagogiske virkemidler, er det naturlig å spørre seg hvorvidt hele eller deler av den er overførbar til andre fag. Opplegget er på mange måter skreddersydd for programmeringsfag, men noen av virkemidlene er relativt universelle, som for eksempel det å gi uferdige filer som utgangspunkt for videre programmering, samt det å jobbe offline mot en problemstilling for deretter å teste online. Den type virkemidler er allerede i bruk i mange fag ved Institutt for data- og elektroteknikk, som f.eks. Elektroteknikk, Styringsteknikk og Reguleringsteknikk.

Det virkemiddelet som antagelig ikke er så lett å overføre til andre fag er det å organisere all kode inn i samme predefinerte filstruktur. Selv om dette forenkler veiledningen betydelig, kan man hevde at det er uheldig at studentene ikke gis mulighet til å strukturere sitt eget arbeid. Bakgrunnen for at vi allikevel har valgt å gjøre det på denne måten er at studentene (på dette nivået) ikke har nok erfaring til å strukturere sin egen kode på en god måte. Vi mener derfor det er mer pedagogisk å presentere dem for en struktur som hjelper dem med å holde orden i programmeringen. Det å ha møter mellom fagansvarlig og studenter/studentgrupper er i utgangspunktet gjennomførbart i ethvert fag, men siden det er en relativt tidkrevende prosess blir det i mange tilfeller ikke gjort. Vår erfaring er at det er nyttig å ha disse møtene, både som et middel for å få innsikt i gruppedynamikk og som et middel til å gi faglige tilbakemeldinger. Siden studentassistentene ofte gir forskjellige (og av og til forvirrende) svar på studentenes spørsmål, kan fagansvarlig benytte møtene til å avklare misforståelser.

Oppsummert så har denne artikkelen presentert erfaringene med og innholdet i den prosjektbaserte undervisningen i Ingeniørfaglig innføringsemne ved Universitet i Stavanger. Det er vist hva som er innholdet, hvordan det er organisert for å oppnå individuell læring samt hvilke forutsetninger som må til for at én vitenskapelig ansatt skal kunne veilede 160 studenter.

Referanser

[1] MATLAB. The Language of Technical Computing. MathWorks. (Accessed: 2016-05-30). [Online]. Available: http://se.mathworks.com/products/matlab/

[2] T. Ryen, "Mikroforelesninger med programmering i MATLAB - Blandet læring i Ingeniørfaglig innføringsemne," in *Proc of NIK2015*, 2015.

[3] FIRST LEGO League Scandinavia. (Accessed: 2016-05-30). [Online]. Available: https://hjernekraft.org/

[4] Om EV3 - Mindstorms LEGO.com. LEGO. (Accessed: 2016-05-30). [Online]. Available: http://www.lego.com/nb-no/mindstorms/about-ev3

[5] RWTH - Mindstorms NXT Toolbox. MathWorks. (Accessed: 2016-05-30). [Online]. Available: http://www.mindstorms.rwth-aachen.de/

[6] LEGO MINDSTORMS NXT Support from MATLAB - Hardware Support - MathWorks Nordic. MathWorks. (Accessed: 2016-05-30). [Online]. Available: http://se.mathworks.com/hardware-support/lego-mindstorms-matlab.html

[7] Google Self-Driving Car Project. [Online]. Available: https://www.google.com/selfdrivingcar/

[8] Matlab Lego NXT Project UiS - YouTube. YouTube. (Accessed: 2016-05-30). [Online]. Available: https://www.youtube.com/watch?v=RBPjeNTwY1Y

[9] Lego-prosjekt 1522 - YouTube. YouTube. (Accessed: 2016-05-30). [Online]. Available: https://www.youtube.com/watch?v=OMMZdNFzwJ0

[10] News - RWTH - Mindstorms NXT Toolbox. MathWorks. (Accessed: 2016-05-30). [Online]. Available: http://www.mindstorms.rwth-aachen.de/trac/wiki/News

[11] A. Behrens, L. Atorf, R. Schwann, J. Balle, T. Herold, and A. Telle, "First steps into practical engineering for freshman students using matlab and lego mindstorms robots," *Acta Polytechnia*, vol. 48, no. 3, pp. 44–49, 2008.

[12] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Schnitzler, J. Balle, T. Herold, A. Telle, T. G. Noll, K. Hameyer, and T. Aach, "Matlab meets lego mindstorms – a freshman introduction course into practical engineering," *Education, IEEE Transactions on*, vol. 53, no. 2, pp. 306–317, May 2010.