DOBOT MAGICIAN LITE

– EN MINIATURE INDUSTRIROBOT

Undervisningsforløb til matematik i gymnasiet

Patrick Liljegren Gregersen og Niels Erik Wegge, Birkerød Gymnasium.

Version 7

Indholdfortegnelse

Note til læreren:

- Undervisningsforløbet kan bruges som indledning til trigonometri i 1g eller 2g, idet trekantsberegninger og koordinatsystemer (2D og 3D) kan motiveres med ønsket om at kunne styre Dobottens arme.
- Undervisningsforløbet er beregnet på et enkelt modul (90 minutter), men det kan være nødvendigt at skubbe 2.1.4 og evt. noget af 3.1.2 til et senere matematikmodul.
- To elever per robot er ideelt. Del evt. forløbet over to moduler med en halv klasse pr. gang.
- Vi anbefaler at Dobottens styreprogram er installeret og testet på et passende antal computere før start.
- Øvelsesvejledningen bør uddeles i papirform: 1 stk. pr. gruppe (for at undgå for mange skærme).
- Efterfølgende opsamling / konsolidering: brug et modul på at gennemgå matematikdelen af de to "Hvad har vi lært?"-afsnit (eksempler og regneopgaver).

Introduktion til Dobot

af Patrick Liljegren Gregersen og Niels Erik Wegge
Birkerød Gymnasium

1.1 Indledning

Dobotten er en stationær industrirobot i miniatureformat. Den arbejder med høj præcision og bruges faktisk til industrielle processer, fx i medicinalindustrien og til andet laboratorie-arbejde. Typiske arbejdsopgaver er at løfte eller gribe objekter og føre dem fra et sted hen til et andet – som sagt med høj præcision. Vores Dobot kan også skrive og tegne.

Da Dobotten er stationær, er der en grænse for hvor langt og højt den kan strække sin gribearm og hvor langt den kan dreje sig rundt. Robotfolkene siger at arbejdsområdet er begrænset – og matematikerne siger at robotternes parametre er begrænset til nogle definitionsmængder. Det skal vi undersøge i denne opgave.

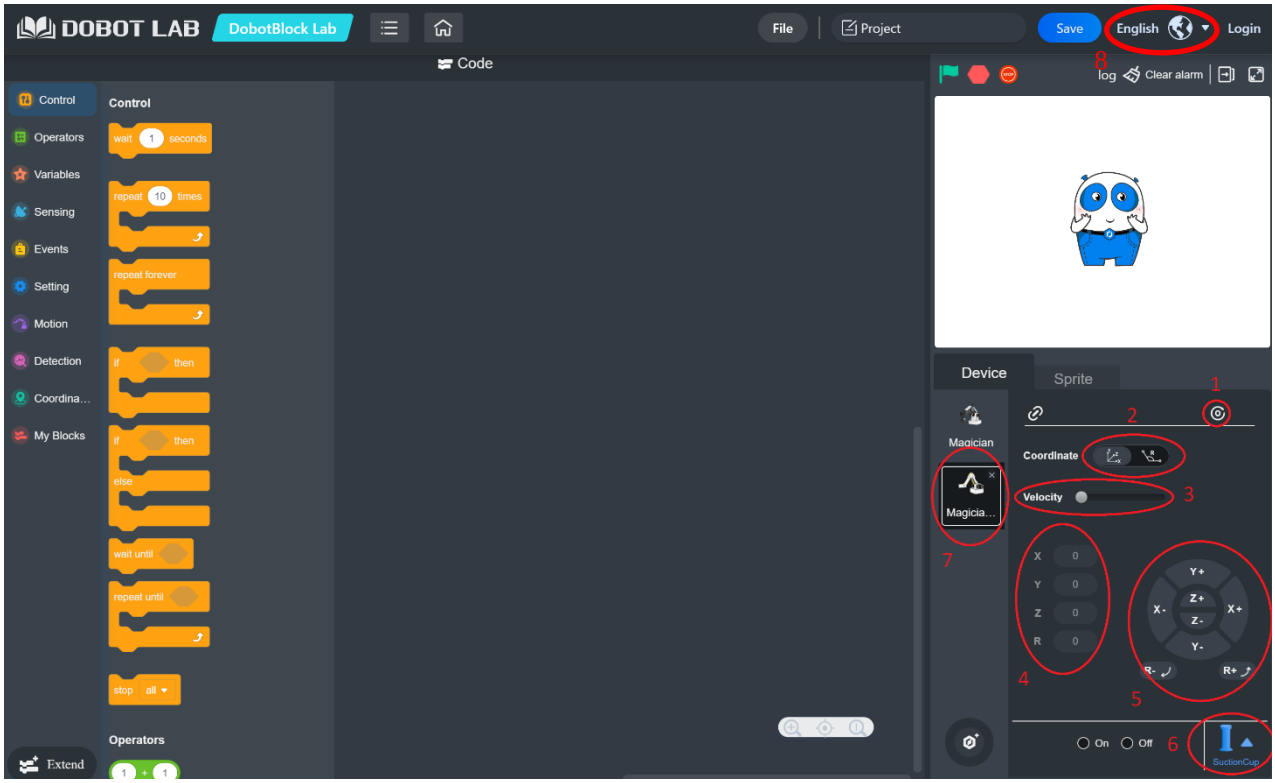
Vi skal også finde ud af hvordan man kan beskrive de punkter robotarmen skal bevæge sig hen til. Der er tre dimensioner, men hvilket koordinatsystem og hvilke slags koordinater bruger Dobotten? Vi skal også se på hvordan Dobotten bærer sig ad med at regne ud hvordan den flytter sin arm præcis derhen hvor den skal – det kræver nemlig ganske mange udregninger.

Endelig skal vi lære at kommunikere med Dobotten. Det sker med hjælp af programmeringssproget Python, som er et af de mest udbredte programmeringssprog både for robotter og for videnskabeligt arbejde. Det kan være svært at skrive programkoden helt grammatisk korrekt, men heldigvis arbejder Dobotten med blok-programmering, og så er det meget nemmere. Det er lidt ligesom at have stavekontrol på sin computer.

Husk at Dobotten er kostbar (den koster 10.000 kroner), så man skal ikke hive og flå i den!

1.2 Få forbindelse til Dobotten

1) Dobotten styres gennem programmet *Dobot Lab*  . Installér det på din computer!
2) Åben *Dobot lab* og tryk på *DobotBlock Lab*. Tænd for Dobotten. Sørg for at der er strøm til og at Dobotten er forbundet til din computer med USB-kablet.
3) Vælg *Magician Lite* (knap 7 på figuren næste side).
4) Vælg også *English language* – de danske oversættelser er ret håbløse. Knap 8!
5) Tryk på Hjem-knappen *Home* (knap 1). Det kan være nødvendigt at vælge "connect" i pop-up vinduet og lukke nogle andre pop-ups. Du skal **ikke** begynde at opdatere software, selvom robotten foreslår det.
6) Tryk Home en gang til – nu er du forbundet til robotten!



Her er en oversigt over en række vigtige ting på skærmen. Tallene henviser til de røde ellipser på figuren ovenfor.

1) ”Home”-knappen. Den bruger vi til at forbinde til Dobotten og til at nulstille positionen af armen.
2) Vi kan skifte mellem koordinater for enden af armen og vinkler på delene af armen.
3) Hastigheden kan ændres med ”Velocity”-skyderen.
4) Koordinaterne (eller vinklerne) for robotten i den pågældende tilstand.
5) Styreknapper til robotarmen.
6) Vi kan indstille, hvilket redskab vi har sat på enden af Dobottens arm.
7) Vores robot er en *Magician Lite.*
8) Her kan vi skifte mellem engelsk og dansk. Alle efterfølgende skærmbilleder fra programmet vil være på engelsk.

1.3 Fri leg

Hvad gør de forskellige knapper i *DobotLab*? Prøv dig frem!

2.1 Dobottens arbejdsområde og definitionsmængde

Nu går vi i gang med en struktureret undersøgelse af Dobottens virkemåde og matematik. Formålet er at forstå de to forskellige koordinatsystemer, som Dobotten bruger.

2.1.2 Styr Dobotten med x-y-z-knapperne

Prøv at flytte Dobottens arm forskellige steder hen ved at trykke på x-y-z-knapperne. Bemærk at du kan ændre på hastigheden (Velocity, knap 3).

1) Hvilken af knapperne får spidsen af armen til at flytte sig op og ned? x, y eller z?
2) Hvad gør de to andre knapper?

2.1.2 Flyt Dobottens arm manuelt

Ude på enden af Dobottens arm (for oven) sidder der en lille knap med et hængelås-ikon. Når man holder den nede, frigøres armen, så man selv kan flytte den derhen, hvor man vil. Der skal ikke bruges vold til at rykke robotarmen, og det er forbudt at prøve at flytte armen manuelt, hvis ikke hængelås-knappen er aktiveret!

1) Læg mærke til koordinat-tallene for x-y-z for forskellige positioner af armen. Hvor skal man flytte armen hen for at få positive / negative…
    - …z-værdier?
    - …y-værdier?
    - …x-værdier? (ja, x-værdien kan faktisk godt være negativ).
2) Flyt robotarmen hen i nærheden af koordinaterne $(x, y, z) = (200, -200, 0)$. Du kan finjustere med x-y-z-knapperne på skærmen, men det gør ikke noget at tallene ikke er helt præcise. Peg på det sted hvor du tror punktet med koordinater $(x, y, z) = (200, 200, 0)$ ligger – og brug så y-knappen til at bevæge spidsen af robotarmen derhen. Havde du ret?

2.1.3 Begrænsninger for x-y-z koordinaterne

Vi skal nu undersøge i hvilket område for x-y-z-koordinaterne robotten kan arbejde.

1) Flyt robotarmen hen tæt på koordinaterne $(200,0,0)$. Kør spidsen af robotarmen op og ned med z knappen. Hvorfor kan den ikke komme helt op og ned?
2) Flyt robotarmen tilbage til koordinaterne $(200,0,0)$. Kør spidsen af robotarmen frem og tilbage med x-knappen. Hvad er den højeste og laveste værdi du kan få for x? Afhænger intervallets størrelse af, hvor du startede? (Ja!)

Nu har du set, at definitionsintervallerne for de forskellige koordinater afhænger af, hvilke værdier de andre koordinater har. Det skyldes selvfølgelig den måde Dobotten er skruet sammen på rent mekanisk – og det er med til at gøre det vanskeligt at styre robotten. Når armen flytter sig fra et sted til et andet, skal hver af de tre koordinater hele tiden holdes indenfor deres definitionsområder, og disse definitionsområder ændrer sig undervejs i bevægelsen. Heldigvis bliver det meget nemmere når man arbejder med vinkler i stedet for x-y-z-koordinater, som vi skal se på senere.

2.1.4 Definitionsmængder i matematikken

I 2.1.3 fandt du nok ud af, at Dobottens arbejdsområde er kompliceret: de tilladte værdier for hver af de variable koordinater afhænger af værdien af de andre to koordinater. Arbejdsområdet – eller definitionsmængden – for Dobotten er nærmest umulig at skrive ned i x-y-z koordinater.

De funktioner, du kommer til at møde i matematikundervisningen, har meget simplere definitionsmængder. Det hænger ikke mindst sammen med at funktionerne som regel kun har én variabel. Eksemplerne og opgaverne nedenfor tager vi i næste matematiktime (medmindre din gruppe er nået hertil hurtigt).

1) Betragt den lineære funktion $f(x) = 2x - 3$. Der er ikke umiddelbart nogen begrænsning på hvilke x-værdier man kan bruge, så definitionsmængden er som udgangspunkt $\mathbb{R}$, alle de reelle tal. Men hvis Kurt ønsker at $f(x)$ skal ligge i intervallet fra -3 til 5, hvad skal definitionsmængden for f så være?
2) Betragt funktionen $f(x) = \sqrt{4 - x^2}$. For denne funktion gælder der at x kun kan have værdier fra -2 til 2, altså $x \in [-2; 2]$. Det er fordi vi ikke umiddelbart kan tage kvadratroden af et negativt tal.
    a. Hvad er definitionsmængden for funktionen $g(x) = \sqrt{100 - x^2}$?
    b. Hvad er definitionsmængden for funktionen $L(x) = \sqrt{x^2 - 100}$?
    c. Hvad er definitionsmængden for funktionen $h(x) = \sqrt{64 - (2x)^2}$?
    d. Hvad er definitionsmængden for funktionen $p(x) = \sqrt{27 - x^3}$?
    e. Hvad er definitionsmængden for funktionen $s(x) = \sqrt{a - x^n}$?
3) Udfordring: Funktionen $f(x, y) = \sqrt{xy}$ har to variable. Prøv at beskrive definitionsmængden for f.

Bemærk at definitionsmængderne i opgave 1) og 2) er givet ved faste intervaller, mens definitionsmængden for hhv. x og y i 3) afhænger af hinanden. Det minder lidt om, hvordan Dobottens robotarm er begrænset.

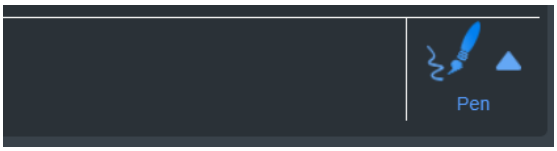
2.2 Dobottens koordinatsystem

Nu skal vi finde ud af præcist hvad x-y-z-tallene i *DobotLab* betyder. Med andre ord: hvilket koordinatsystem bruger Dobotten? Først når vi kender svaret på det, kan vi få robotarmen til at bevæge sig præcist derhen, hvor vi vil have den.

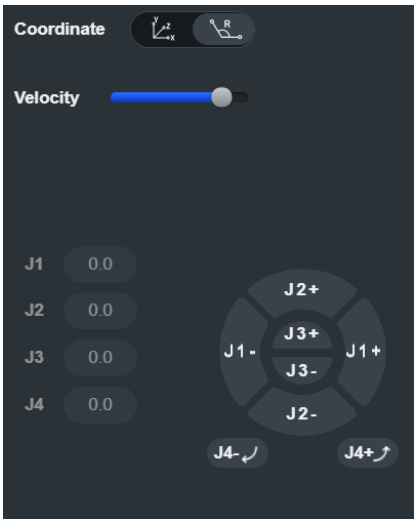
Vi starter med at finde koordinatsystemets centrum. Til det får vi brug for robotarmens vinkler.

2.2.1 Robotarmens vinkler

1) Start med at montere penne-holderen og indstil *DobotLab* til ”Pen”. Det er knap nr. 6 på figuren side 3.
   Men der skal *ikke* være nogen pen i penneholderen endnu.



2) Nulstil robotarmens position med ”Home”-knappen (knap nr. 1). Nu kører armen lige så stille tilbage i sin udgangsposition. Prøv det!
   a. Giver det mening, at y nu er præcis 0?
   b. Hvilken z-værdi har du fået? Hvis det angiver et antal millimeter, hvor er nulpunktet på z-aksen så henne? Er det nede i bord-højde?
   c. Se på x-koordinaten. Hvor må nulpunktet på x-aksen ligge, sådan cirka?
   d. Du har fået skæve koordinater til x og z – men se på ”Joint”-koordinaterne. Hvad er det egentlig ”Home”-knappen nulstiller?



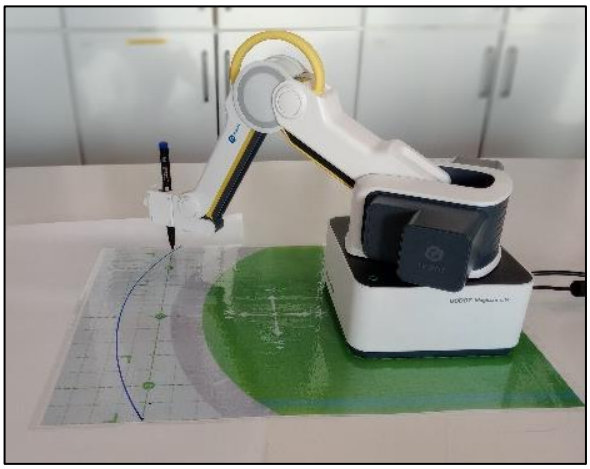
3) ”Joint” betyder ”led” på engelsk, og tallene J1, J2 og J3 beskriver den vinkel, Dobottens tre led står i. Prøv at gennemskue, hvor på Dobotten de forskellige vinkler er ved at flytte robotarmen manuelt (husk hængelåsknappen!). Undersøg om dine antagelser passer ved at køre på joint-kontrolknapperne i *DobotLab*, én ad gangen.
4) Kunne du ikke finde ud af hvad vinklen J4 gør? Det er der en god grund til. Denne vinkel bruges ikke, når penne-holderen er monteret. J4 bruges nemlig til at rotere den sugekop eller gribeklo, som også kan monteres på armen.

Nu har du forhåbentlig set hvordan vinklerne kan bruges til at beskrive robotarmens position og at deres definitionsmængder er noget simplere end x-y-z-koordinaternes. Faktisk er det vinklerne som robottens motorer kan indstille til bestemte værdier, ikke x-y-z-koordinaterne. Hvis man vil hen til et bestemt (x, y, z)-koordinatsæt er robotten derfor nødt til at omregne x-y-z-tallene til tre passende vinkler. Denne ikke helt simple matematikopgave vender vi tilbage til i et andet forløb!

2.2.2 Koordinatsystemets nulpunkt

I 2.2.1 fandt du sikkert ud af at punktet $(x, y, z) = (0,0,0)$ må ligge et sted cirka midt i robotten. Vi kalder dette punkt for koordinatsystemets nulpunkt eller *Origo*. Desværre kan vi jo ikke flytte robotarmen hen til dette nulpunkt (hvorfor ikke?), så vi må måle os frem til hvor det ligger. Vi gør det på en elegant måde, der involverer nogle matematiske egenskaber ved cirkler.

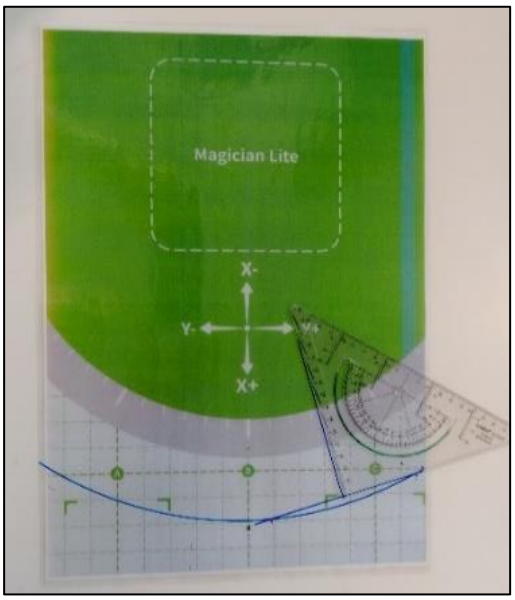
Materialer: Til denne opgave skal du bruge det laminerede underlag og en sprittusch med blød spids.



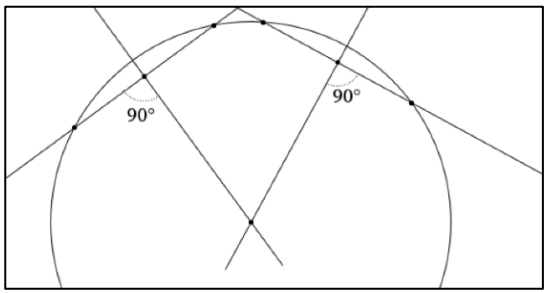
1) Placér Dobotten præcist i den stiplede hvide firkant som vist. Sørg for at have papir i siderne så du ikke kommer til at tegne på bordet.

2) Få Dobotten til at tegne en cirkelbue på det laminerede underlag (se billedet) og flyt så Dobotten helt væk.

3) Koordinatsystemets x-y-nulpunkt må være centrum af cirklen – men hvor er det? Vi har jo ikke hele cirklen, kun en lille cirkelbue. Men centrum ligger der hvor diametrene skærer hinanden, så vi skal "bare" tegne to diametre. Den første diameter tegnes sådan:
    a. Markér to tilfældige punkter på cirkelbuen.
    b. Tegn en ret linje gennem de to punkter (med lineal). Den kalder man en *sekant*.
    c. Mål dig frem til det punkt på sekanten som ligger midt mellem de to punkter.
    d. Tegn en linje gennem dette midtpunkt og vinkelret på sekanten. Denne vinkelrette linje vil være en diameter, så den vil gå igennem cirklens centrum.

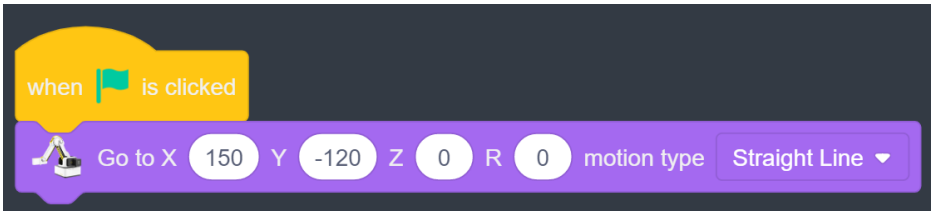


4) Den anden diameter tegnes på samme måde.
5) Nu har du fundet cirklens centrum: dér hvor de to diagonaler skærer hinanden. Og det skulle være lige i midten af den stiplede firkant på underlaget.
6) Vis det til din lærer, før du fortsætter 😊
7) De vinkelrette halveringslinjer kaldes *midtnormaler*, og man kan faktisk konstruere dem uden at måle sig frem – spørg evt. din lærer om hvordan man kan gøre det ved hjælp af bare en lineal *uden tal på* og en passer.



Nu har vi en god idé om, hvor centrum af Dobottens koordinatsystem er. Lad os teste det ved at gå til nogle præcise koordinater. Det betyder at I skal til at programmere!

8) Lad os sige at vi gerne vil have robotarmen til at gå til koordinaterne $(x, y, z) = (150, -120, 0)$. Mon tallene er i millimeter? Mål dig frem til x-y-placeringen af dette punkt og marker det på det laminerede underlag.
9) Sæt robotten tilbage i firkanten og lav følgende programmerings-blokke:



10) Tryk på knappen "Play"  i øverste højre hjørne – går robotarmen det rigtige sted hen?
11) Fjern robotten igen – og brug en lineal til at måle (x, y)-koordinaterne for det punkt der hedder A på det laminerede underlag. Kan du indstille robotarmen til at gå hen til A? Hvor mange forsøg brugte du?

2.3 Hvad har vi lært?

Om robotten:

- De brugbare intervaller for hver af x, y og z afhænger af hvilken værdi de to andre har. Dobottens arbejdsrum (definitionsmængde) er kompliceret!
- Nulpunktet for x-y-z-koordinatsystemet ligger midt inde i robotten.
- Robotten måler afstande i millimeter og vinkler i grader.
- Positionen for "Home" er kompliceret for x-y-z-koordinater, men simpel for vinkler.

Om matematik:

- Definitionsmængder for lineære funktioner og kvadratrodsfunktioner.
- Tegne sekanter på en cirkelbue.
- Bruge sekanter til at finde centrum af en cirkel.
- Evt. konstruktion af midtnormaler vha. passer og lineal (uden tal på).

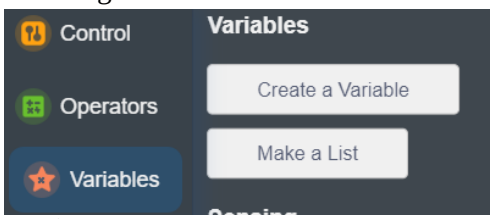
3.1 Dobotten tegner firkanter og trekanter

Nu skal vi til at programmere robotten. Vi starter med at få den til at tegne simple geometriske figurer med bestemte mål. Programmerne bruger variable og bruger også matematiske udregninger til at få robotarmen til at flytte sig derhen, hvor den skal.

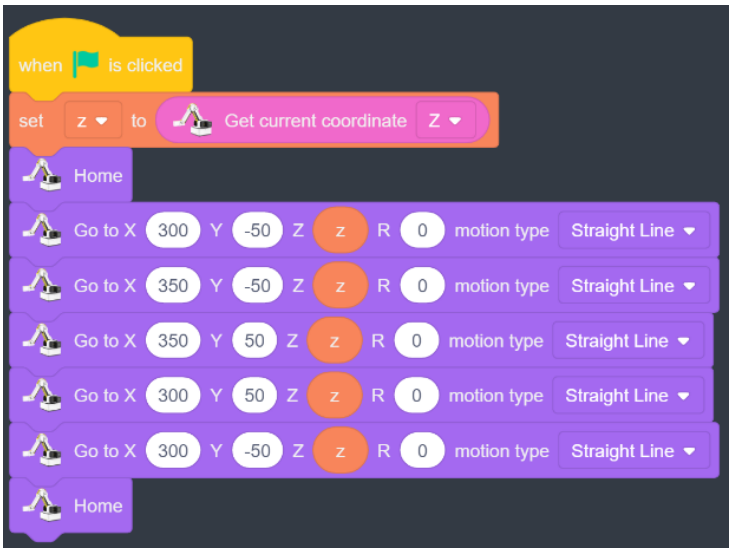
3.1.1 Tegne et rektangel på to forskellige måder

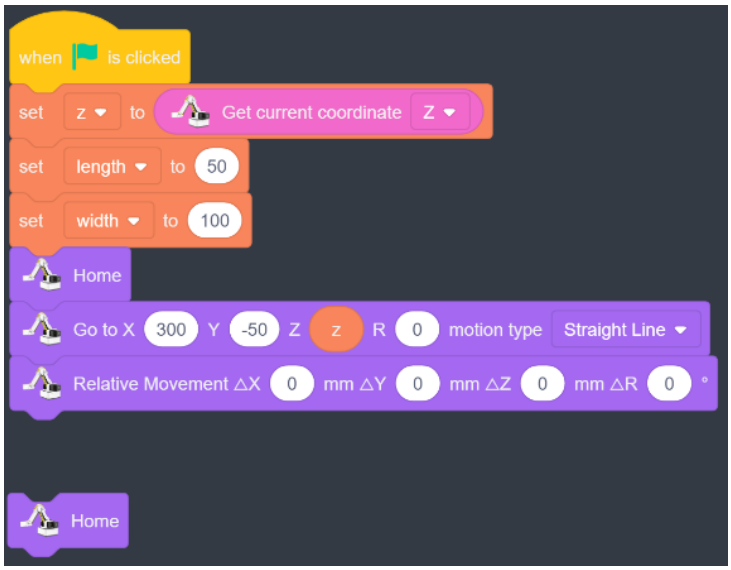
I denne opgave vil vi få Dobotten til at tegne en firkant med længde 10 cm og bredde 5 cm.

I *DobotBlock Lab* kan vi også definere variable vi gerne vil bruge flere gange i løbet af programmet. Dette gøres under ”Variables” og ”Create a Variable” →



1) Start med at lave variablen *z*. (Der kommer flere variable senere.)
2) Rengør det laminerede underlag med noget sprit.
3) Lav blok-programmet til højre →
4) Placer robotarmen så tuschens spids lige rører ved det laminerede underlag (et eller andet sted).
5) Kør programmet!
6) Mål efter med en lineal om Dobotten har tegnet den ønskede firkant!
7) Hvad sker der, hvis du fjerner en af de tre midterste ”Go to” kommandoerne?
8) Hvorfor tror du vi startede programmet med at sætte ”z” til ”Get current coordinate”?
9) Vi indfører nu to nye variable ”length” og ”width”. Prøv at lave samme firkant, hvor du bruger disse variable i stedet for konkrete tal! Starten af programmet er vist til højre →
10) Prøv at se, hvor stor en firkant du kan få tegnet (forbudt at tegne på bordet).

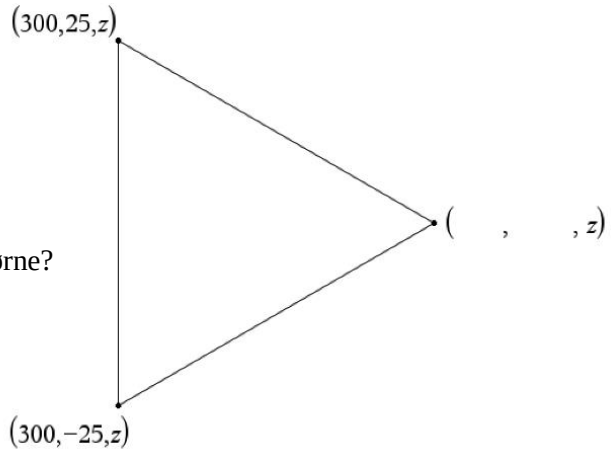




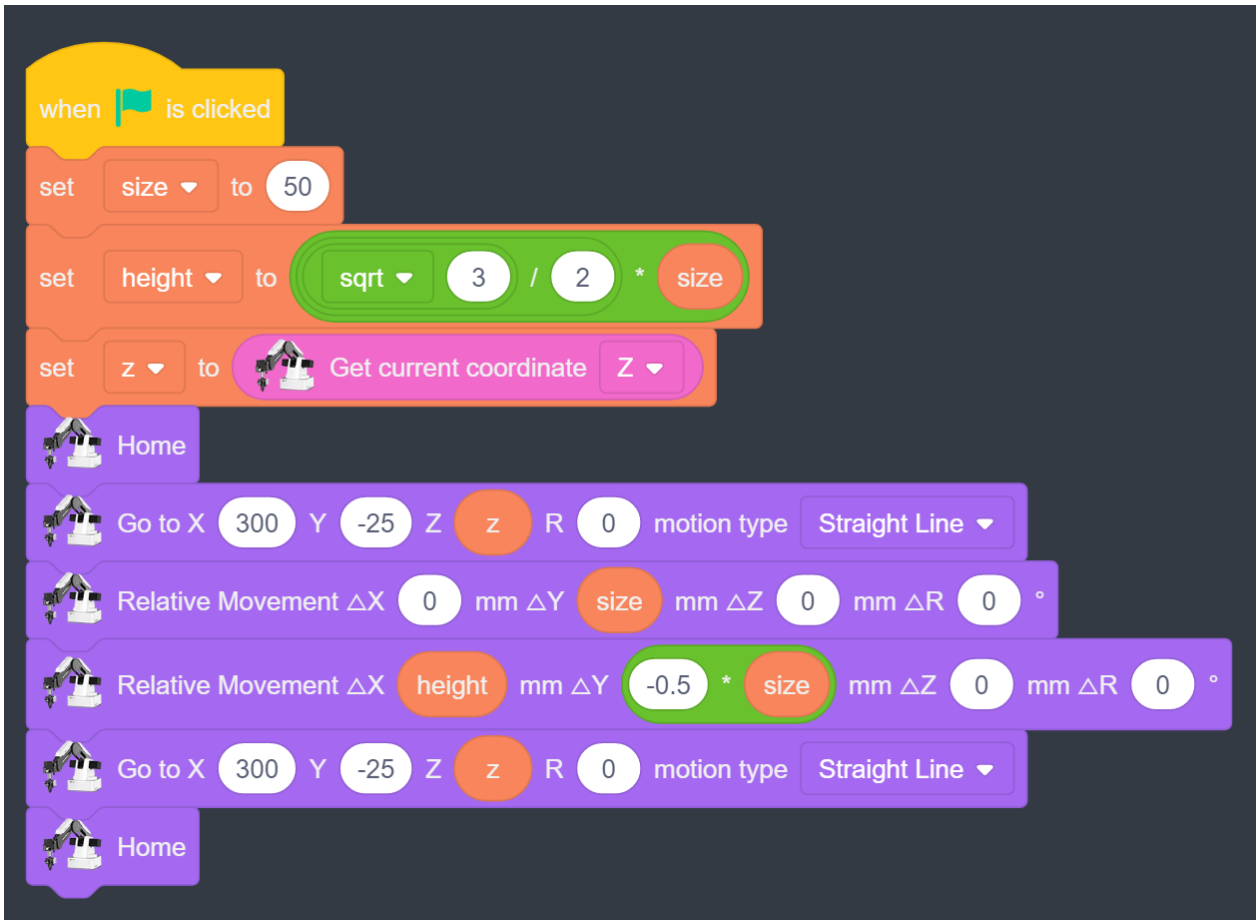
3.1.2 Tegne trekanter

I 3.1.1 kom du måske frem til, at vi i nogle tilfælde får tegnet en trekant. Vi skal nu se på, hvordan vi kan tegne en ligesidet trekant.

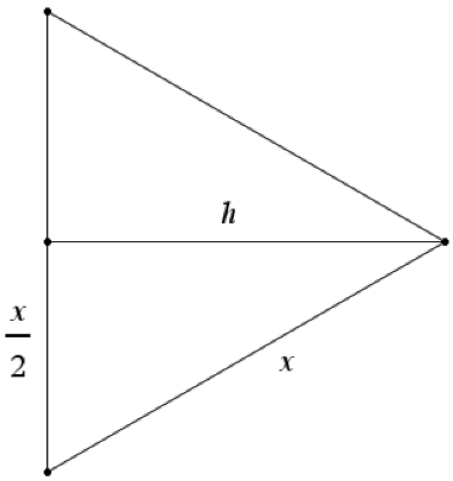
1) Vi ønsker en ligesidet trekant hvor to af hjørnerne ligger i $(300, -25, z)$ og $(300, 25, z)$, hvor x, y og z er målt i millimeter.
   a. Hvor lang er hver side? (husk enhed)
   b. Ved hvilke koordinater finder vi det sidste hjørne? [Hint: Pythagoras…]



2) Lav et program, der kan få Dobotten til at tegne trekanten. Mål efter med en lineal: Er de tre sider lige lange? Og er sidelængden hvad den skulle være?
3) Det ville være rart at programmet selv regnede ud hvor det sidste hjørne skal placeres. Et program der kan tegne vilkårlige ligesidede trekanter ses nedenfor. Prøv det!



4) Hvordan virker matematikken i programmet? Sådan her!



Til højre ses en ligesidet trekant med sidelængderne x.
Som alle ligesidede trekanter kan den deles op i to retvinklede trekanter. Hver af disse har kateterne h og $\frac{x}{2}$ og hypotenusen x.
Pythagoras giver

$$h^2 + \left(\frac{x}{2}\right)^2 = x^2.$$

Vi isolerer h:

$$h = \pm\sqrt{x^2 - \left(\frac{x}{2}\right)^2}$$

Da højden h ikke kan være negativ, får vi

$$h = \sqrt{x^2 - \left(\frac{x}{2}\right)^2} = \sqrt{\frac{3x^2}{4}} = \frac{\sqrt{3}}{2}x$$

5) Analysér nu programmet fra 3) – kan du se hvordan programmeringen af højde-formlen er lavet? Og hvordan trekanten tegnes fra punkt til punkt?
6) Eksperimentér med ligesidede trekanter af forskellige størrelser – hvor store kan Dobotten tegne dem?
7) Eksperimentér med andre geometriske figurer!

3.2 Hvad har vi lært?

Om robotten:

- Brug af variable i programmer.
- Matematiske beregninger i Block programmering.

Om matematik:

- Beregning af højden i en ligesidet trekant.
- Beregning af koordinaterne for spidserne i en ligesidet trekant.

4.1 What's next?

Der bliver udviklet videre på undervisningsforløb omkring:

- Omregninger mellem x-y-z-koordinater og vinkelkoordinater. Beregninger med cosinus og sinus.
- Konstruktioner af regulære polygoner. For eksempel en 17-kant.
- Anvendelse af Dobotten som industrirobot til at flytte og sortere ting.