
Mechatronica project.

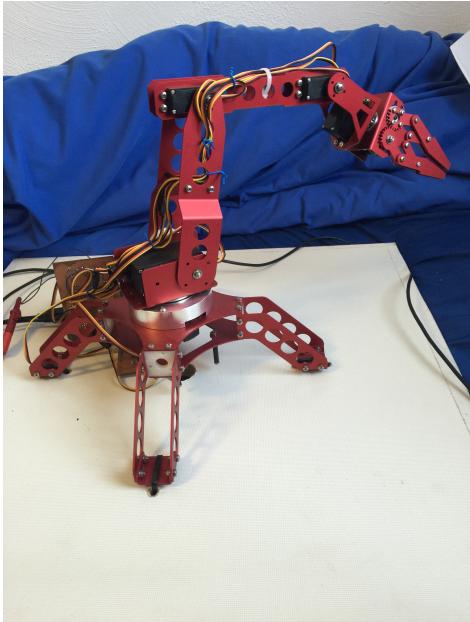
Robotarm

Nieskens Tom & Jooris Matthias
Docent: Vincent Claes

2Ea-ICT



Introductie



Voor het vak Mechatronica maken we per groepjes van 2 een project. Wij hebben het project gekregen van de Robotarm. Dit is een 4 assige robot met servo motoren. Buiten het bewegen van de assen zijn er ook nog 2 extra servo motoren voor het openen en sluiten van de grijparm en het draaien van deze arm. Het aansturen van deze robot zal gebeuren in Labview met behulp van de MyRio module. De motoren aansturen via een PWM is één ding. maar de opdracht was om in een X, Y en Z assenstelsel te werken. Hiervoor moeten we gebruik maken van Inverse Kinematics. Hier hebben we dan ook onderzoek naar gedaan.

MyRio

De Ni MyRio is een embedded design device met een dual-core ARM zinq processor en een FPGA. Een MyRio heeft 10 analoge ingangen, 6 analoge uitgangen en 40 digitale I/O lijnen. On board zijn er ook draadloze communicatiemethodes zoals wifi aanwezig, enkele leds en buttons. Ook heeft de MyRio een 3 assige accelerometer. Een MyRio is eigenlijk ontwikkeld voor studenten. Om op korte tijd een project op te starten (minder dan een semester) Dit is betaalbare hardware met eigenlijk alles in wat nodig is voor de meeste projecten en natuurlijk veel uitbreidingsmogelijkheden. Wijzelf gaan voor ons project enkel gebruik maken van de PWM uitgangen van de MyRio.



Labview

Om met de MyRio module te kunnen werken in Labview hebben we nog enkele uitbreidingen van Labview moeten installeren zoals:

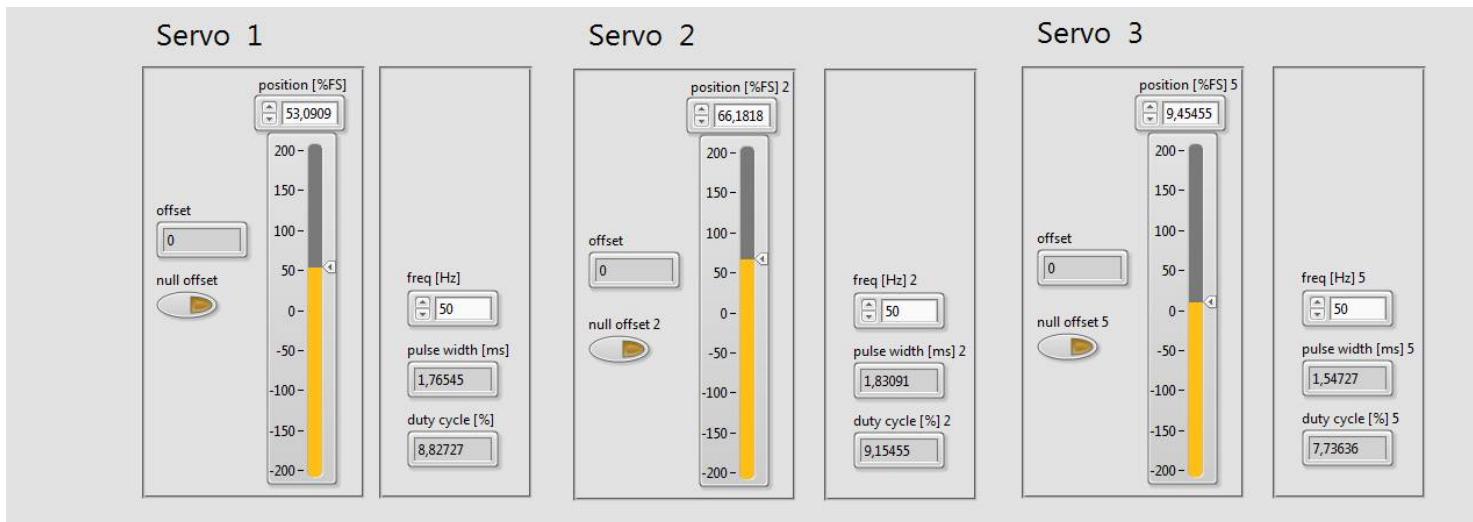
-MyRio toolkit: <http://www.ni.com/download/labview-myrio-toolkit-2014/4854/en/>

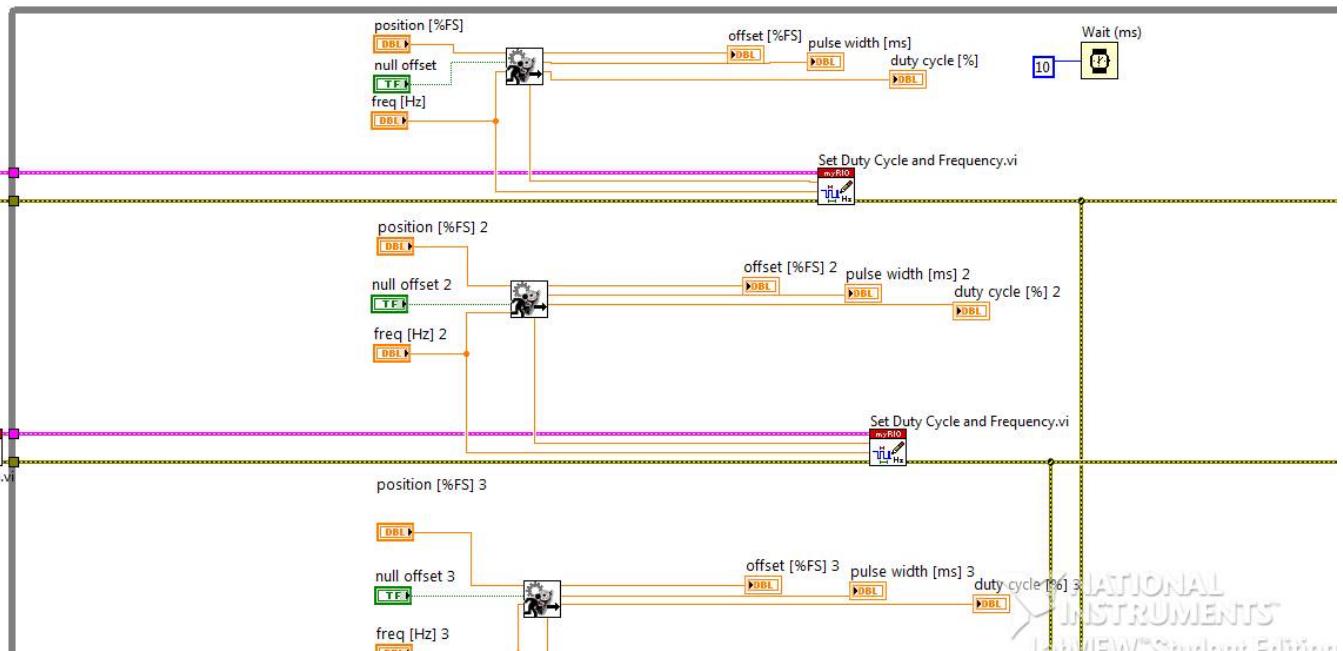
-Run-Time: <http://www.ni.com/download/labview-run-time-engine-2012/3433/en/>

-Robotics: <http://digital.ni.com/demo.nsf/websearch/d2dea2cf5bc93049862576af00685960?opendocument>

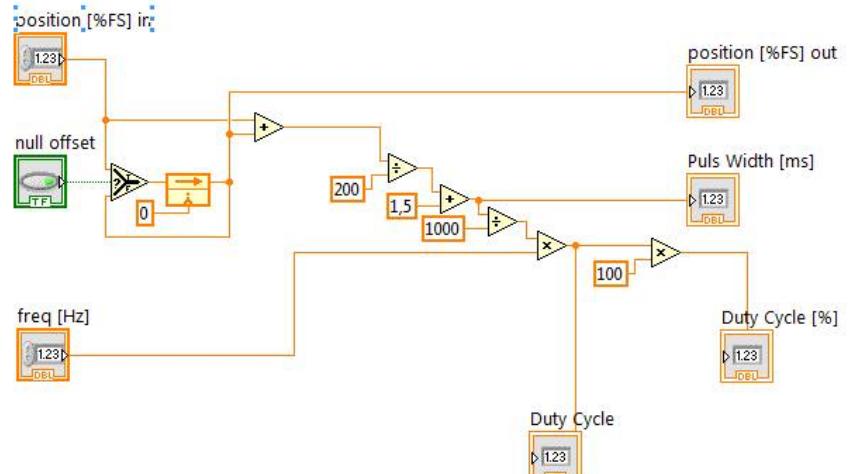
Deze modules zijn nodig om bepaalde blokken op te roepen die standaard niet in labview zitten. De belangrijkste voor ons zijn dan bijvoorbeeld de blokken die nodig zijn voor de pwm te gebruiken in de MyRio

Omdat we zelf nog geen enkele ervaring hadden met Labview zijn we niet meteen begonnen aan de volledige opdracht met kinematics. We hebben eerst een programma gemaakt waarin we iedere motor apart konden besturen. Zoals u in onderstaande afbeelding kan zien is er een aparte bediening voor iedere servo motor. Per servo is er de optie om de frequentie van de pwm in te stellen. Met de slider kan je de puls breedte en de duty cycle aan. Op die manier zal de servo bewegen.

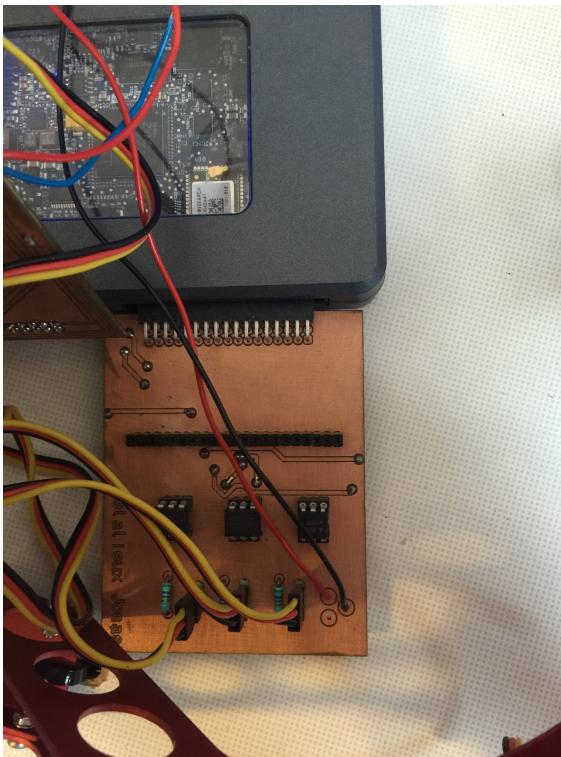




met de Open.vi laten we het programma weten met welke PWM uitgang we willen werken. Het blokje in het midden (met het tandwiel in) doet de berekening. Deze krijgt de frequentie, null offset en positie binnen en berekend dan de puls breedte en duty cycle die er nodig is om de arm tot op de juiste positie te brengen. De berekening zie je op de afbeelding rechts.



Al de nodige informatie wordt dan verzonden naar de Set duty cycle and frequency.vi. Deze blok zet dan de juiste pwm waarden op de vooraf gekozen pwm uitgang. Dat deel van het programma hebben we dan ongeveer letterlijk 6x gekopieerd om voor de 6 servos afzonderlijk te gebruiken. Met dit programma was de robot perfect te besturen. Maar de opdracht was om met X, Y en Z coordinaten te werken.



Omdat de MyRio zelf niet genoeg stroom kan leveren voor onze servo motoren aan te sturen maken we gebruik van printplaatjes met optocouplers. De MyRio stuurt deze optocouplers aan, achter de optocoupler hangen we dan 2 labo voedingen die wel genoeg stroom kunnen leveren voor de servo motoren aan te sturen.

Kinematics

Onze robot is een 4-assige robot. Op de afbeelding aan de rechterkant kan u zien hoe deze in elkaar zit. Het gele uiteinde zou dan een positie moeten krijgen in het X, Y en Z gebied. Om deze positie te berekenen moet je rekening houden met enkele variabele:

- De lengtes van de armen van de robot
- De X, en Y positie

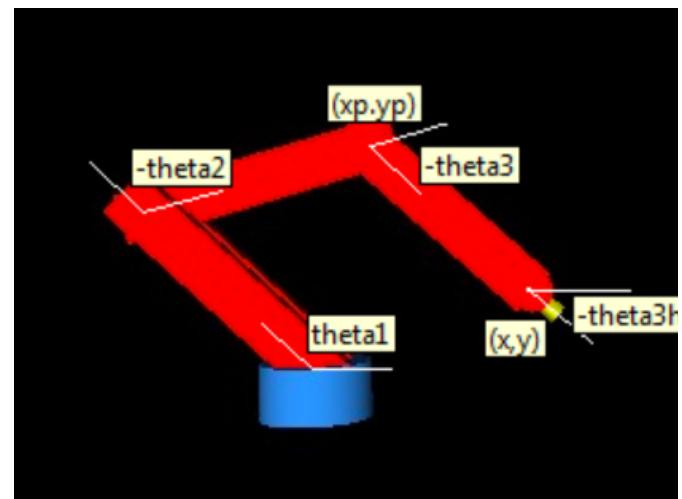
```
float c, xp, yp, thetai, theta3h, alpha;
x = x, y = y;
```

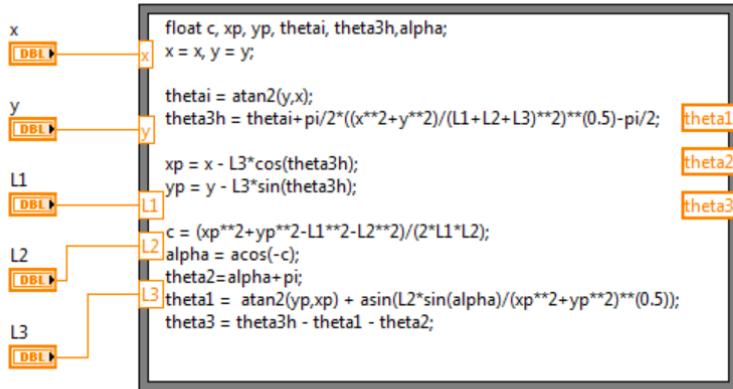
```
thetai = atan2(y,x);
```

```
theta3h = thetai+pi/2*((x**2+y**2)/(L1+L2+L3)**2)**(0.5)-pi/2;
```

```
xp = x - L3*cos(theta3h);
yp = y - L3*sin(theta3h);
```

```
c = (xp**2+yp**2-L1**2-L2**2)/(2*L1*L2);
alpha = acos(-c);
theta2=alpha+pi;
theta1 = atan2(yp,xp) + asin(L2*sin(alpha)/(xp**2+yp**2)**(0.5));
theta3 = theta3h - theta1 - theta2;
```





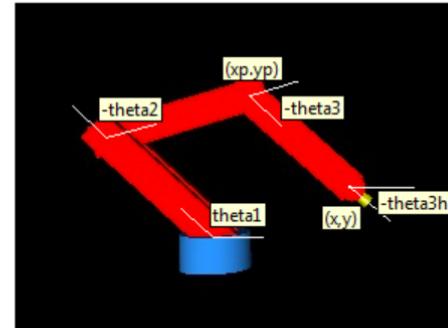
Inverse Kinematics for 4R Robot
- assume that the output is in cylindrical coordinates (x, y, theta0)

At this stage there are three angles to be determined but only two constraints x,y.

- define theta_i, the angle from the base to the end effector
- set theta_{3h} based on the distance from the base
 - + when the arm is at full extension theta_{3h}= theta_i
 - + when the arm is close to the base theta_{3h} approaches a right angle.
 - + any function that fits these characteristics can be considered.

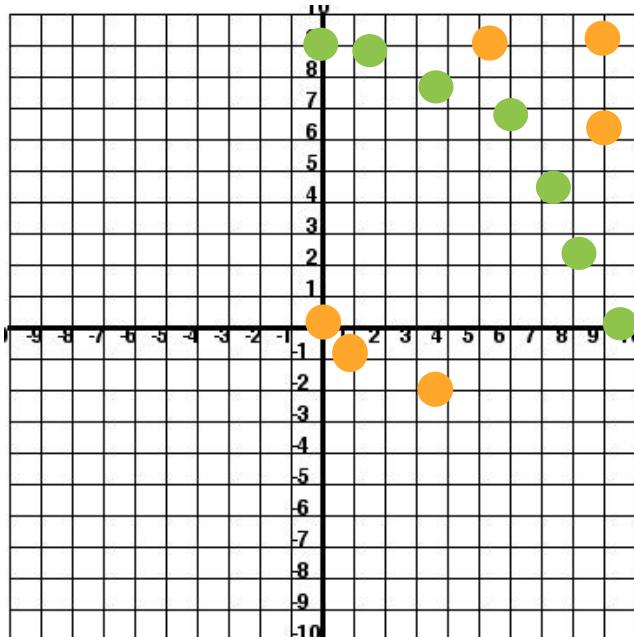
- define xp and yp as the proximal joint of the last link (see diagram)
- solve for theta₁ and theta₂ using the cosine rule.

The assumption defining theta_{3h} makes the system more easily solvable, but it constrains the workspace of the manipulator. It may also lead to unachievable large joint velocities near singularities.



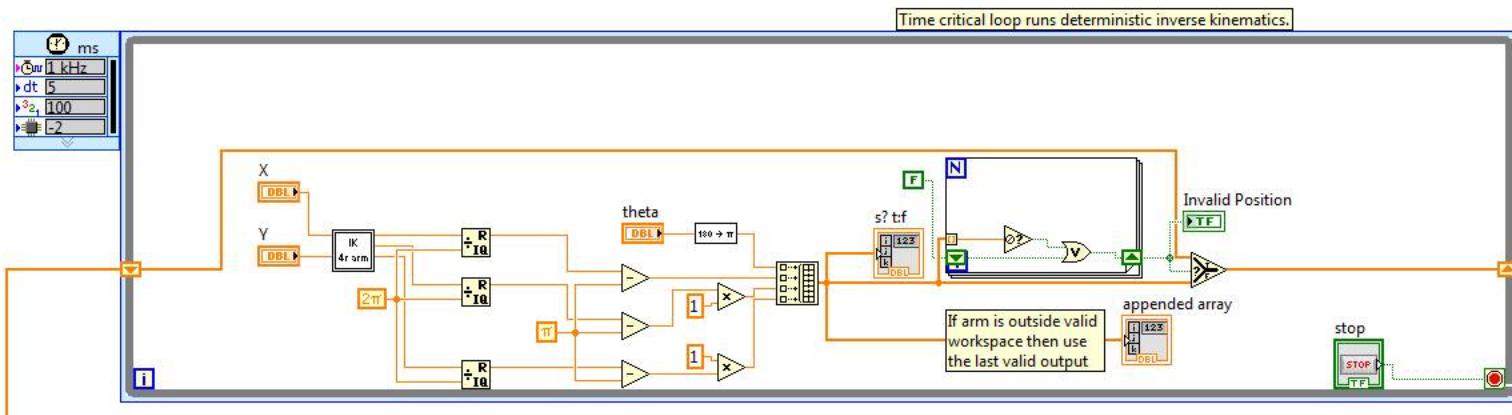
Note that the angles are measured anti-clockwise therefore some of the angles shown above are negative

Deze berekening geeft de hoeken weer in radialen. Later in ons programma vormen wij deze dan om naar graden. Omdat er in de berekening geen rekening gehouden wordt met onmogelijke posities hebben we hier nog wat problemen mee. Onder de grond gaan is natuurlijk niet mogelijk en door de robot arm zelf gaan ook niet. Posities gelijk X=max en Y= max is ook onmogelijk omdat de arm daar niet lang genoeg voor is



Zoals hier links te zien is, de groene punten zijn allemaal haalbare posities met de robot. De oranje punten daarentegen zijn in de werkelijkheid onhaalbaar. Echter wiskundig gezien zijn deze punten wel haalbaar. Het is dan ook moeilijk om in de software een onderscheid te maken tussen wat haalbaar is en wat niet. Zodra je op een onhaalbaar punt komt gaat de robot ook niet meer doen wat we willen dat er zou gebeuren.

labview programma met kinematics



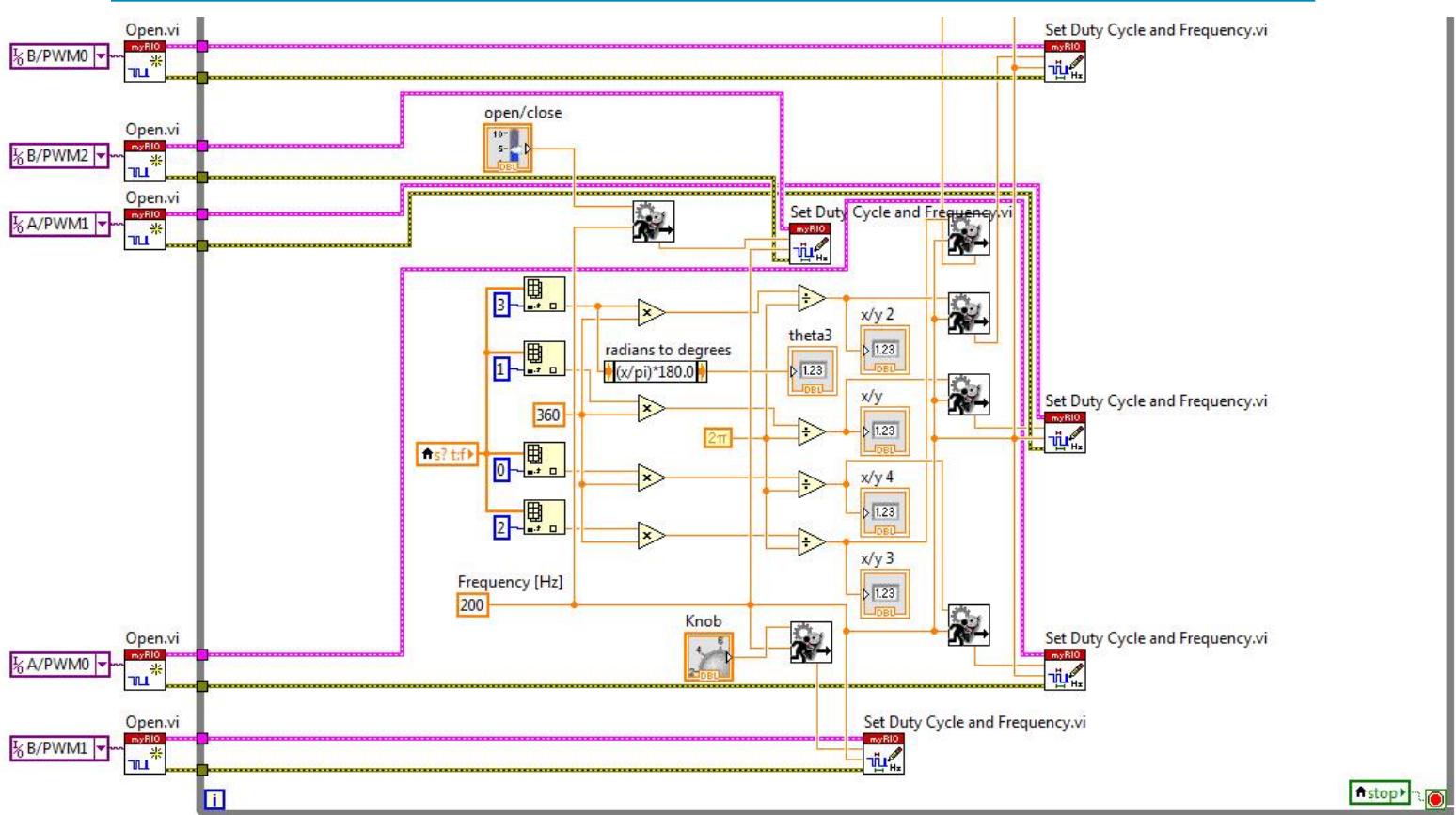
Eerst worden de X en Y positie uitgelezen. Deze zijn ingangen in de kinematics berekening van op de vorige pagina (deze, samen met de lengtes van de armen worden gebruikt in de berekening).

Omdat we niet meer dan 180 graden mogen hebben (servo motor kan niet verder draaien) delen we de waarde die uit de berekening komt door 2π , de restwaarde houden we bij. Op die manier sluiten we alle kerken n^*360 uit.

Van die restwaarde trekken we π af zodat ons getal altijd tussen het bereik van $-\pi$ en π zal liggen (respectievelijk -180 en 180 graden).

De Theta is de hoek van de basis, deze wordt omgezet naar radialen zodat alle waardes hetzelfde zijn.

Al de waardes die we nu over hebben slaan we op in een array. Deze array wordt opnieuw uitgelezen in de volgende loop.



De Open.vi blokken net zoals in ons vorig beschreven programma bepalen welke pwm output op de MyRio gebruikt zullen worden.

de s?t:f is de array waarin we de waardes in de vorige loop hebben opgeslagen. Deze gaan we nu uitlezen en verdelen onder de 4 motoren die aangestuurd moeten worden. Al deze waardes doen we maal 360 en delen door 2π om ze weer om te vormen naar graden. In de blokjes met het tandwiel wordt weer de waarde van de pwm (duty cycle en de pulsbreedte.), deze worden dan via de MyRio verzonden om de servo motoren aan te sturen.

Dan zijn er nog 3 aparte sturingen die niet in de kinematica berekening zitten, deze dienen om de robot te laten draaien, de bek open en toe te doen en om de bek zelf te laten draaien.

Github: <https://github.com/tomnieskens/Machatronica-4R-robot-PXL>