University of Twente

EEMCS / Electrical Engineering
Control Engineering

Characterization and modeling of a Dynamixel servo

Arno Mensink

Individuel Research Assignment

Supervisors:
prof.dr.ir. S. Stramigioli
ir. E.C. Dertien
ir. G. van Oort

November 2008

Report nr. 035CE2008
Control Engineering
EE-Math-CS
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands

Abstract

This individual research assignment was about characterization and simulation of the Dynamixel AX12+ servo. Due to limited time not all aspects were treated. In the characterization part, the working of the servo was investigated and the most important parameters were measured. In the simulation part a model of the servo was chosen and implemented with the measured parameters.

It turned out that the Dynamixel servo is position controlled, but with a maximum speed and maximum torque as main limiting factors. The maximum speed has been incorporated in the simulation model, but due to lack of information the torque has not. Main components in the model are the setpoint generator and the friction model. The setpoint generator was created to incorporate the maximum velocity in the model. Work has to be done on the friction model, because the model used for this in the 20Sim simulation software was not sufficient in this case.

As a conclusion, it can be said that the results are a good start, but the model can be made more accurate by incorporating more elements of the servo in the model.

Table of contents

1 Introduction

In this chapter a general overview is given over what this project is about. Also a brief description is presented over the servo system. At last, the report outline is given.

1.1 Project goal

The goal of this project is to characterize the 'Dynamixel AX12+' servo and make a simulation model in 20Sim. This means investigating the servo to see how it behaves and what it is able to do. Also measurements have been performed to obtain information. These measurements are then used in the simulation model.

1.2 The Dynamixel system

The Dynamixel AX12+ is a powerful servo which can be used to build robots. At the Control Engineering chair, they have bought 3 complete kits. Such a kit contains 18 servos and several kinds of brackets and screws to mount them together in numerous possible geometries. Separate units can also be bought. The servos are all identical. There is one sensor unit to let the system act on environmental inputs as a proper robot. The servos are daisy chained together with at the end the controller. This bus is used for both the data to the Dynamixels and for the power lines. The controller can be programmed with the computer via a supplied program. This program uses predefined building blocks which, when put in the right order, creates the program code lines.

Figure 1.1: Dynamixel servo

1.3 Report outline

The report is divided in the same two parts of the project goal. The first part is the analysis or characterization of the Dynamixel servo in chapter 2. In that chapter first the datasheet is being analyzed. After that reverse engineering has been done on the servo to get information of internal components of the Dynamixel. The last part of chapter 2 consists of measurements on several servo properties and this measurements are also being analyzed.

Chapter 3 handles about the second part, the design of the simulation model. This model is being composed of different components which are at first treated separately and then be merged into a complete model. Hereby the results from the analysis and measurements are used to set the correct value to the parameters of the model.

Chapter 4 treats the results of the analysis and the designed model, wheras in chapter 5 the conclusions are drawn from the results and recommendations are done. At the end there are some appendices with more background information about the Dynamixels.

2 Analysis

In this chapter an analysis of the Dynamixel servo is made. This involves getting information about the characteristics and measuring responses of the servo.

The Dynamixel servo gets its setpoints via a serial communication bus. These setpoints are then processed to drive the electric motor. In most servos a gearbox is attached to increase the output torque. This output then has generally a position sensor to measure its position for feedback to the controller. From this a IPM can be derived, which is shown in figure 2.1. This is not a very extensive model, but from this it can be seen what has to be known in order to be able to characterize the servo.

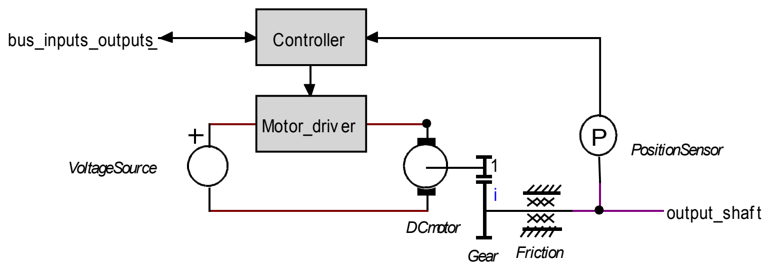


Figure 2.1: IPM of the Dynamixel

First some general info about the Dynamixel was looked up. This includes the voltage, maximum speed and others. Then the gear reduction was calculated by counting the teeth of the gears. Also the main electrical components in the servo were examined. After that, measurements were done on the in- and outputs to see how the Dynamixel reacts on the setpoints. As can be seen from the IPM, the friction has to be known. Therefore friction measurements have been performed. Finally an acceleration signal is calculated. This is because of the simulation model that had to be made. The controller in that simulation model can only handle one input, while the Dynamixel gets more then one from the bus. This can also be seen in the IPM. There are multiple bus inputs and only one input to the motor driver.

2.1 Data sheet

The first source of information is the data sheet of the Dynamixel servo [1]. There is not much information about the physical aspects, but still useful information.

On page 3, the data sheet lists some of the specifications of the Dynamixel. It says that the input voltage is 7-10V DC, recommended 9.6V which is normally produced by 8 rechargeable AA cells. At 7V, the Dynamixel has a maximum holding torque of 12 kgf.cm, that corresponds to 1.18 Nm in SI units. At 10V, the maximum torque is 16.5 kgf.cm, that is 1.62 Nm. The 60degree time is at 7V 0.269 s, at 10V 0.196 s. In table 1 this is summarized. The maximum speed is 114 RPM or 11.9 rad/s in wheel modus. In this modus the (software) angle limiter is removed to let the servo work as a normal electric motor. The servo is than velocity controlled instead of position controlled. In the datasheet [1] it is called endless turn. For more specifications, refer to the data sheet or appendix A.2.2.

	AX-12	
Input voltage (V)	7V	10V
Max. holding torque (Nm)	1.18 Nm	1.62 Nm
60degree time (s)	0.269 s	0.196 s

Table 1: Parameters of the Dynamixel AX-12 servo

In the data sheet is listed that the operating angle is 300º, or an endless turn can be implemented so that the servo acts just like a geared down motor. The number of positions is 1024, so the resolution is $\frac{300}{2^{10}-1}$ = 0.29º or $5.1 \cdot 10^{-3}$ rad. The data sheet lists a resolution of 0.35º. This is obtained when a complete rotation is used, $\frac{360}{2^{10}-1}$ = 0.35º. But then the datasheet is inconsistent, because it says that setting a value of 0x3FF (1023) moves the shaft to the position at 300º (page 16), whereas at page 3 the resolution is noted as 0.35º and a range of 300º. It is expected that the resolution value is incorrect, because the position sensor inside the Dynamixel has a measurable range of 320º due to physical limitations, and a linear position measurement of 300º [2].

2.2 Reverse engineering

Another possibility to know what is inside the black box (literally) is opening up the casing. In this section a description is given of the components inside.

At the mechanical side, there are the electric motor with a motor gear and six other gears. See figure 2.2. The largest of them is connected to the output shaft. Notable about these gears is that from the motor gear up, the gears become larger and the teeth become sturdier. This is done to handle the increasing torque on the gears. In the right picture of figure 2.2 the position sensor on the PCB can be seen at the arrow.

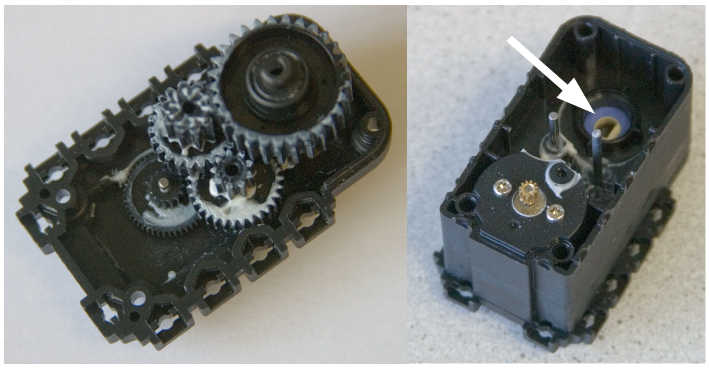


Figure 2.2: Mechanical parts inside the Dynamixel

To verify if the servo has a gear reduction of 254, as listed in the data sheet, the gear teeth are counted.

Gear #	Large sprocket teeth	Small sprocket teeth
1 (Motor gear)	-	11
2	48	15
3	38	10
4	31	9
5	23	10
6	29	1 (output axis)

Table 2: Gear teeth count of the Dynamixel

So the total gear reduction is $\frac{48}{11}\cdot\frac{38}{15}\cdot\frac{31}{10}\cdot\frac{23}{9}\cdot\frac{29}{10}$ = 253.97, which is the same as the data sheet states.

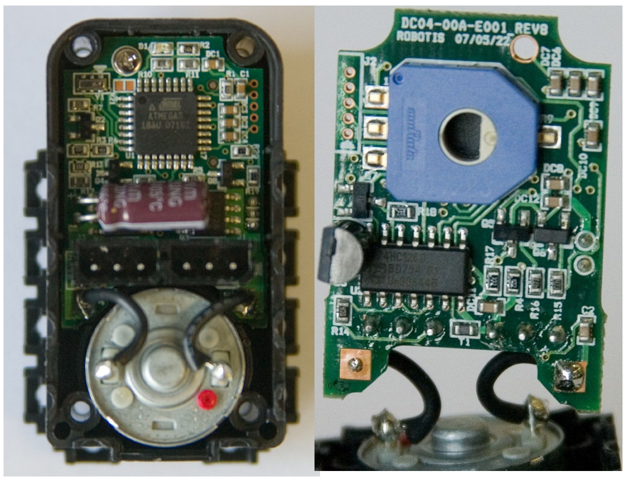


Figure 2.3: Electrical components inside the Dynamixel

On the electrical side of the servo, the components on the PCB are identified. Most noticeable is the Atmega8 microcontroller on one side and the muRata SV01 position sensor [2] on the other, figure 2.3. The microcontroller is an 8-bit controller. It has 10 bit A/D conversion and 10 bit PWM output. This explains the total of 2^{10} possible positions and 2^{10} possible speeds. The Dynamixel is also capable of measuring torque, but is not clear how this is done. It was expected to be done by means of current measurement, but there was no shunt resistor or other type of current measurement found. Maybe the torque is a software defined variable, or something on the PCB was looked over. The software defined variable may be derived from the PWM output of the microcontroller. This is a measure of the power needed to turn the output shaft. The position sensor is in fact a linear potentiometer of 10 kΩ. This sensor has a range of 320 degrees in which the output is considered linear. In upright position, which is in the Dynamixel coordinates 0 degrees, the output of the sensor is 50%. The chip (74HC126) below the position sensor contains 4 tri-state buffers, which are probably buffering the control bus lines. Under the electrolytic capacitor below the microcontroller there are two chips (4536) which are Dual MOSFET's. They are each used as half an H-bridge. This H-bridge is driven by the microcontroller. Next to the buffer IC there is a 5V, 100mA voltage regulator (M78L) in TO-92 package.

2.3 Measurements

A third way to gain information about the Dynamixel is doing measurements, this also is a form of reverse engineering. The signals on the bus were investigated and also Rob Reilink had done some measurements on the Dynamixel servo for test purposes on his work on stereo-vision for robots. This data was being analyzed to gain information about the physical properties of the Dynamixel.

2.3.1 Specification of the bus communication

According to the datasheet [1], the hardware used in the Bioloid communication between control unit and Dynamixels is a UART (Universal Asynchronous Receiver/Transmitter) [3]. The physical connection is a half duplex TTL level serial bus. As mentioned before, all devices are daisy-chained together. Because it is half-duplex, only one device can transmit at the same time, so the other devices have to listen. The communication goes over one wire, with GND as the reference.

If the CM-5 control unit wants to send an instruction packet to one of the Dynamixels, it sets its communication direction to output. After sending the instruction packet, the direction changes automatically to input.

The return of a status packet is programmable. There are three possibilities, never return a status packet, always return a status packet or return only if a READ DATA instruction was given. The standard mode is always returning.

When a Dynamixel receives an instruction packet with the corresponding ID, it waits a specified time (standard about 150µs) and then sends back a status packet.

The protocol used with the UART is an asynchronous serial start-stop communication protocol[1]. It specifies that every byte sent, has to begin with logic ‘0’ as a start bit. Then comes the eight bits of data, followed by logic ‘1’ as a stop bit. There is no parity bit and the data is sent LSB first. When no data is on the line, the line is pulled up to logic ‘1’.

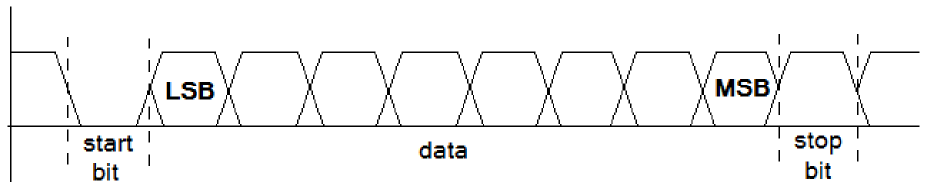


Figure 2.4: Communication over the data bus

Measuring of the communication on an oscilloscope reveals that the TTL levels in practice are the same as the CMOS levels. Logic ‘0’ is at the GND potential and logic ‘1’ is at 5V.

[1] In the datasheet of the 'Dynamixels' is sometimes the RS-485 protocol mentioned. This is a two wire protocol, where the 'Dynamixels' communicate over one. These two wires are the opposite of each other, so probable only the non-inverting signal is sent, where the inverting signal is recreated with a inverter in the 'Dynamixels'.

Rob has written a program that communicates with the Dynamixels over the bus with a special converter, the USB2Dynamixel [4]. This is a multi functional device that contains a USB to serial converter and it can communicate with the Dynamixels.

As the servo starts to move the actual position, speed and torque variable were sampled in with 100 Hz. It was done first without a load, second with a little web cam as a load and last with the camera and withstanding it with his hand as an extra load. The results from the measurement with the camera as a load are plotted in figure 2.5, the other measurement plots are in appendix A.5.

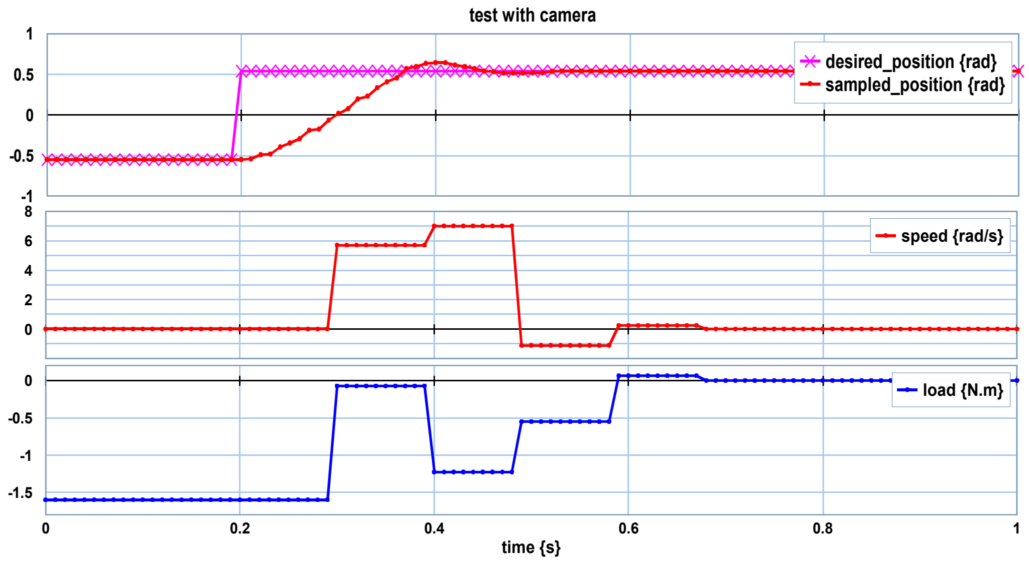


Figure 2.5: Measurements on the Dynamixel servo

A few things can be directly concluded from the figures. The shape of the waveforms is generally the same in all figures, but an applied load does affect the controller. In the measurement with extra friction, counteracting the servo by hand, it can be seen that the overshoot in the position which is present in the other figures, does not appear.

There is another effect that can be seen from the plots. The position is changing every sampling moment, but the speed and the load are always constant for 9 or 10 samples.

2.3.2 Measurement of the internal friction of the servo

As the Dynamixel has a gear reduction of 254:1, it is expected that friction in the servo can not be neglected. As no information was found, it had to be measured. In the translation domain, the friction curve is plotted as the force against velocity. In the rotation domain, forces become torques and velocities become angular velocities. The most extensive friction model in 20Sim was chosen to implement. This exists of 4 components, as there are static friction, coulomb friction, viscous friction and the Stribeck velocity. Not all this components can be (easily) measured, but can be estimated using measurements of other components.

The static friction is the friction at stand-still that has to be overcome before the servo starts to move. On the servo, this can be measured by attaching an arm to the axes of the servo and hanging a force meter at the end. When pulling at the other end, the force at which the servo starts to move, is a measure for the static friction. The results for the Dynamixel are as follows. A supplied bracket from the kit was used as an arm. The length from the axis of the servo to the end of the bracket was 54 mm. The force shown at the force meter was 2.4 N when the servo started to turn. Thus the torque due to static friction is: $\tau_{static} = F\,x = 2.4 * 0.054 = 0.13[Nm]$

For measuring the viscous friction, another setup is built. With the Bioloid kit come some wheels for building rolling robots. Such a wheel is mounted to the Dynamixel an a piece of string is wound around the wheel. At the end of the string a bucket with a mass is tied. The Dynamixel is clamped to a table and the time is measured that the mass need to fall down a certain distance. This is done several times with different masses. If the mass is not large enough to create a torque high enough to overcome the static friction, the servo is helped by hand to start turning. The procedure results in a constant force at the string and an average velocity over the length at which the mass has fallen. By the radius of the wheel the force is converted into a torque and the velocity into an angular velocity. When these values are plotted against each other, a linear curve is expected. From this curve, the viscous friction coefficient μ_v can be calculated.

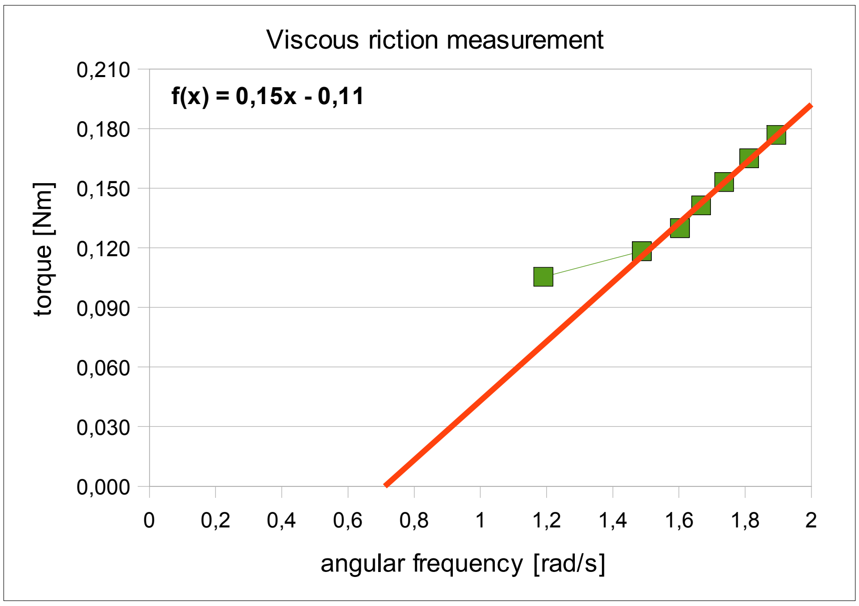


Figure 2.6: Viscous friction measurement

In figure 2.6 the executed measurements are plotted. The measurements are done in a small range, as the Dynamixel is capable of turning with an angular velocity of about 12 rad/s (appendix A.2.2). This is because the setup didn't suit for higher masses and thus higher velocities, and at lower velocities, the applied mass didn't make it the whole way down. The measurements do reveal a almost linear relation between torque and angular velocity. Therefore a regression line is drawn through this points to find the viscous friction coefficient. This coefficient indicates how much the friction increases per unit of velocity. This is thus the slope of the regression line. The function of this regression line is shown in the upper left corner of figure 2.6. The first measurement point is discarded for the linearization, because it is outside of the linear path of the rest of the measurements. At this point, also other friction types as coulomb friction did probably play a role. So, the viscous friction coefficient μ_v is 0.15 [Nm.s/rad]. The regression line does not go through the origin, which it should do if viscous friction was the only friction component. Most striking about this offset (translation over the torque axis) is that the offset term is negative instead of positive. If the term was positive, it would have been a measure for the coulomb friction. Unfortunately the setup did not work for measurements at lower speed, but the first measurement point can indicate that the viscous friction of the Dynamixel does not have a linear curve. This should also possibly (partially) solve the negative offset. But it can also be a faulty measurement.

The coulomb friction is the part of the friction that is independent of the (angular) velocity. It could be measured at very low speeds, where the viscous friction is very low. This can be done by applying a small torque and help the servo a little to start turning. Gradually apply a slightly larger torque, so the helping torque is less. When this helping torque is minimal, just enough to overcome the static friction, the applied torque is a measure for the coulomb friction. This method appeared to be quite inaccurate, because for repeated measurements it is not possible to give always the same helping torque. Also if the applied torque becomes to large, viscous friction will play a role as well. Because this inaccuracies, no useful results were obtained.

Eventually, the results are combined in a 20Sim friction model. The function used in this friction model is shown in figure 2.7. This is the standard function used in 20Sim which includes static friction. This curve is curve fitted in Matlab with the friction measurements done before. The resulting parameters are shown in figure 2.7, together with the used fitting curve. The resulting function is shown in figure 2.8.

```
General model:
fittedfriction2(x) =
(( mu_c + (0.13*abs(tanh(10000*x)) - mu_c) *
exp( -((x / v_st)^2 ) ) ) * sign(x) + mu_v*x);

     Coefficients (with 95% confidence bounds):
    mu_c =  -0.2937  (-3.865, 3.278)
    mu_v =   0.2424  (-1.523, 2.008)
    v_st =    1.025  (-1.224, 3.273)
```

Figure 2.7: Matlab curve fitting parameters

In figure 2.7 the several parameters of the model can be recognized. The static friction is inserted as a known and also the slope. This constant of 10 000 is a mathematical parameter to make the the friction curve continuous from the second to third quadrant (negative velocity means also negative friction). The other parameters are calculated by Matlab to give the best fit. Also here it can be seen that the outcome gives a negative value for `mu_c`, the coulomb friction. Also `mu_v` gives a much higher number than found at the viscous friction measurement. The Stribeck velocity `v_st` is a parameter that has to do with objects sliding with lubricants in between. This is not very important for this project, because it is assumed that the gears do not slide. This component was not measurable, but will be estimated by curve fitting.

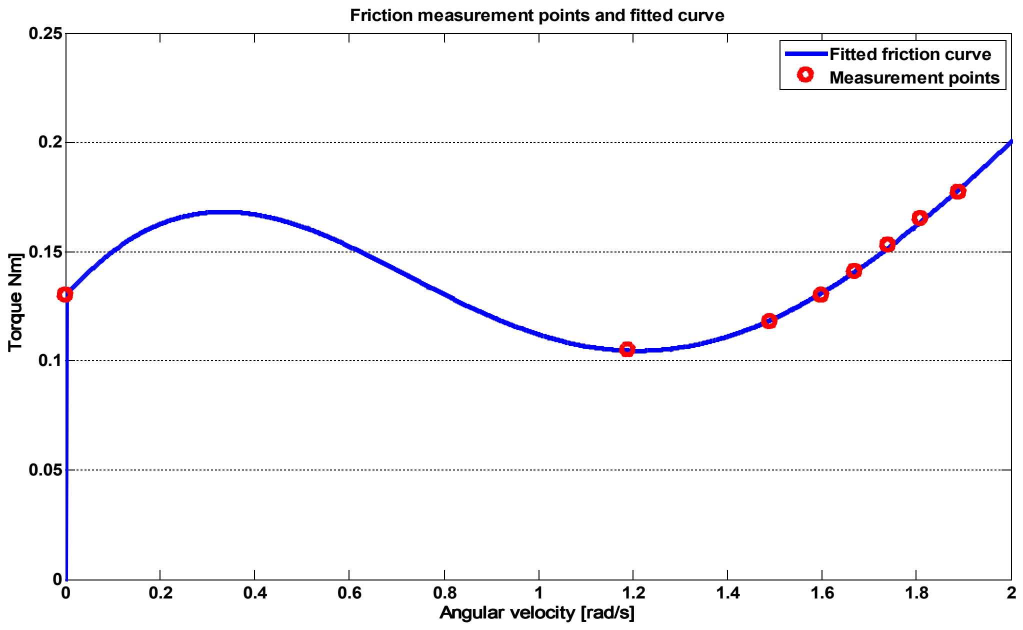


Figure 2.8: Friction torque as a function of angular velocity

But, as can be seen from 2.8, a curve can be fitted over the measurement points. Thus, although the parameters are not realistic, at least at the range of the measurements, the curve does show how the friction is in reality. Between the static measurement point and the other range there is however still a problem. Normally, the static friction point is a local maximum, so the curve should go down if the velocity becomes greater than zero. This is not the case in here. All this indicates that the function used as friction model is not completely suitable in this case.

2.3.3 Measurement of the torque constant

To model the electric motor in the Dynamixel, one has to know the value of the torque constant. To measure this constant, a bracket was mounted to the Dynamixel and the torque of the servo is turned on. At the axis of the servo an arm was attached. At this arm a force is applied perpendicular to the arm. Then for certain values of the force the current flowing through the Dynamixel was measured. The measured values are given in table 2. The length of the used arm was 0.16 m.

Since the servo has a static friction component, this influenced the measurements. By pulling on the axis of the servo with some force, the servo motor experiences less torque, because of the static friction. Therefore, the static friction component (0.13 Nm) is subtracted from the calculated torque values. This is shown in the column T_corr.

F (N)	T (Nm)	T_corr (Nm)	I (A)
0	0	0	0,037
1	0,16	0,03	0,095
2	0,32	0,19	0,097
3	0,48	0,35	0,110
4	0,64	0,51	0,134
5	0,8	0,67	0,169
6	0,96	0,83	0,186
7	1,12	0,99	0,208
8	1,28	1,15	0,229

Table 3: Measured values of torque constant measurement

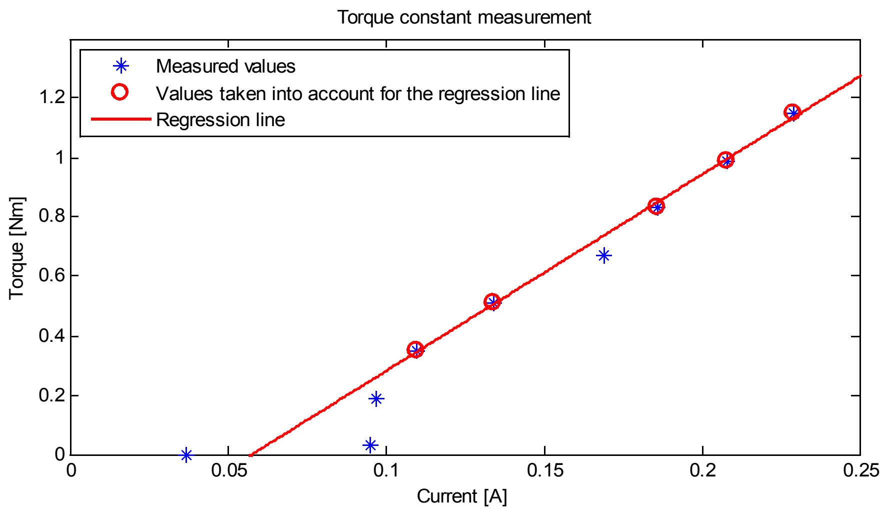


Figure 2.9: Measured data of the torque constant of the Dynamixel

The results are plotted in figure 2.9. In this plot the measured values of table 3 are shown graphically. With this measured data a regression line was created to determine the torque constant. Since the unit of the torque constant is [Nm/A], it is found as the slope of the regression line. Not all values are taken into account for this regression line. The values close to 0.1 A are the most deviating. These deviating values can be caused by the H-bridge, which, when low torques and thus low currents are needed, may not be very efficient and dissipate relatively more power. The reason why the first value at 0.04 A is on the left of the regression line may be that the H-bridge is completely turned off by the software. At 0.17 A there is another deviating value, but it is not as much as the other ones. This may be a measurement fault.

The remaining measurement values were used for the regression line and the outcome is as follows:

```
f(x) = 6.624*x - 0.3826
R²: 0.9984
```

The correlation coefficient R^2 is nearly 1, so the measurement values fit very good on a linear line. The torque constant is, as mentioned, the slope of the linear function, so it is found as 6.624 N/A. This is the torque constant of the electric motor with the gearbox inclusive. This torque constant will be used as the gyration constant of the gyrator in the simulation model.

2.3.4 Retrieving acceleration from measurement data

The measurements are used to retrieve the acceleration of the Dynamixel, which is needed by the setpoint generator.

The acceleration is obtained by differentiating the velocity once or the position twice. But if this signals are noisy, the differentiating operation amplifies the noise. As the position signal is noisy, a state variable filter is used to filter the noise out of the signals. The coefficients are chosen such that the filter behaves like a low pass filter. The transfer function of the filter is given as:

$$H(z)=\frac{k_2}{(z^2+k_1 z+k_2)} \quad H(z)=\frac{\omega_n^2}{(z^2+2\omega_n z+\omega_n^2)}$$

The constants k_1 and k_2 are defined as $2\omega_n$ and ω_n^2 respectively, where ω_n is the cut-off frequency. With these constants, the filter is critically damped.

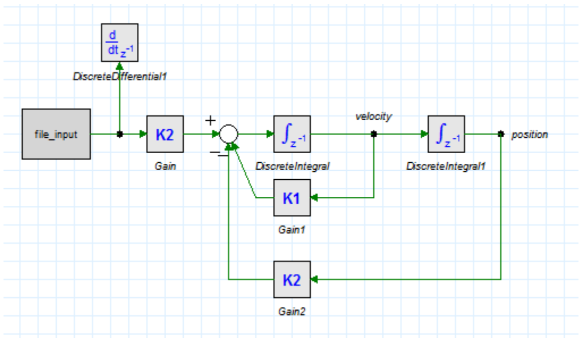


Figure 2.10: State variable filter

In figure 2.11, the cut-off frequency is set at 30 rad/s.

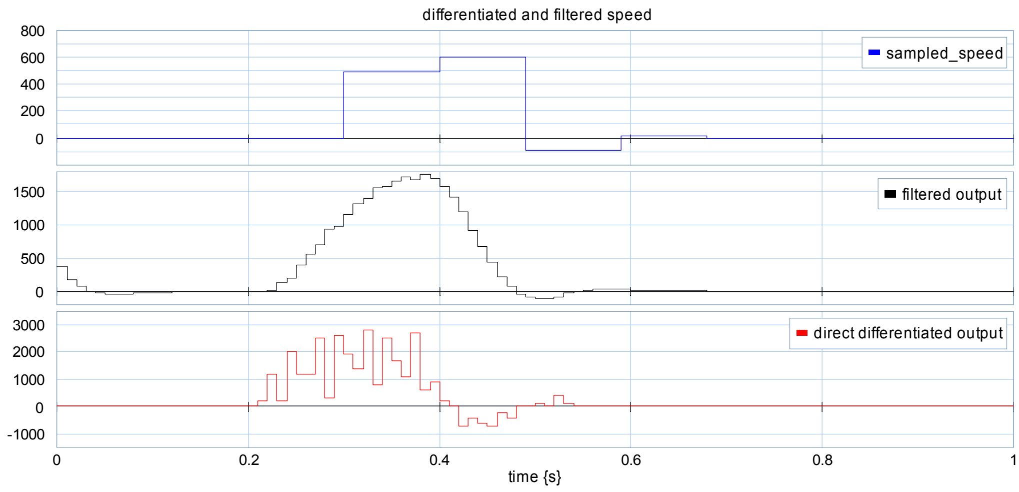


Figure 2.11: Simulation plot of the sampled, filtered and differentiated speed

The filter does improve the curve of the velocity compared to direct differentiation of the position. It does reduce the noise of the signal, but it also introduces a time delay. Further, it changes the shape of the response. This is of cause unwanted.

Another approach is to use curve fitting techniques to create a curve that fits best for the measurement points of the differentiated position. This is done in Matlab, with the curve fitting toolbox. As a type of fitting curve the 4th order Fourier series is found as best fit. Matlab creates that function with the best fitting coefficients, and than it can be used to for further processing. Figure 2.12 shows the outcome of the fitting tool. This outcome is based on the measurement of the Dynamixel at no load.

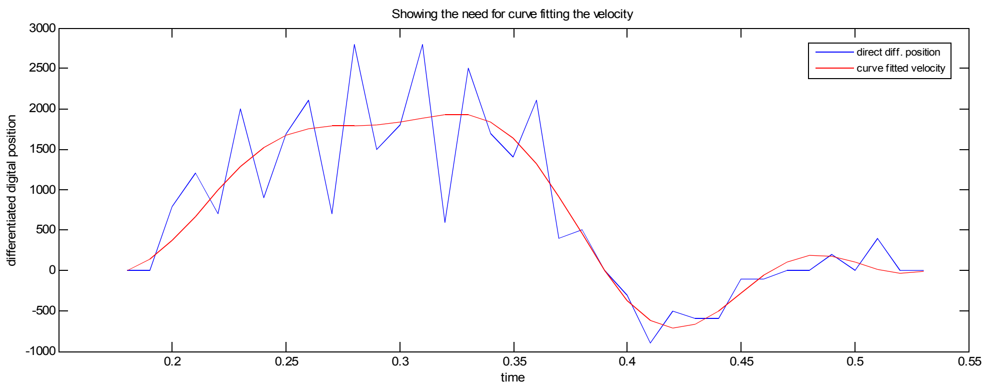


Figure 2.12: Curve fitting function

Next some correction has to be done on the unit of the function. The differentiated unit is ['digital position steps'/second]. As the velocity output of the Dynamixel is 2^{10} steps over a range of 114 RPM, the result of the curve fitting has to become the same unit in order to be able to compare the result of the curve quantitative.

$$[\frac{'digital\ steps'}{s}]*(\frac{300}{2^{10}})[\frac{degree}{step}]=[\frac{degree}{s}],\quad \frac{[degree/s]}{360}=[\frac{rotations}{s}]*60=[RPM]$$

$$[RPM]*(\frac{2^{10}}{114})=['digital\ RPM']$$

The result is shown in figure 2.13, the second plot. It can be seen that the amplitudes of the fitted curve are equal to the output of the Dynamixel, with the notice that the fitted curve isn't the real velocity, but a processed one that fits reasonably. The fitted velocity starts before the position actually changes. This is anti-causal, but it is a consequence of the curve fitting.

For the acceleration the [rotation/s] of the above formula is taken and multiplied by 2π to get the angular velocity. Then it is differentiated to obtain an acceleration in [rad/s^2]. This is plotted in the lower plot of figure 2.13. See further Appendix 7.5. From this plot is seen that the maximum acceleration of the Dynamixel is about 200 radians per second.

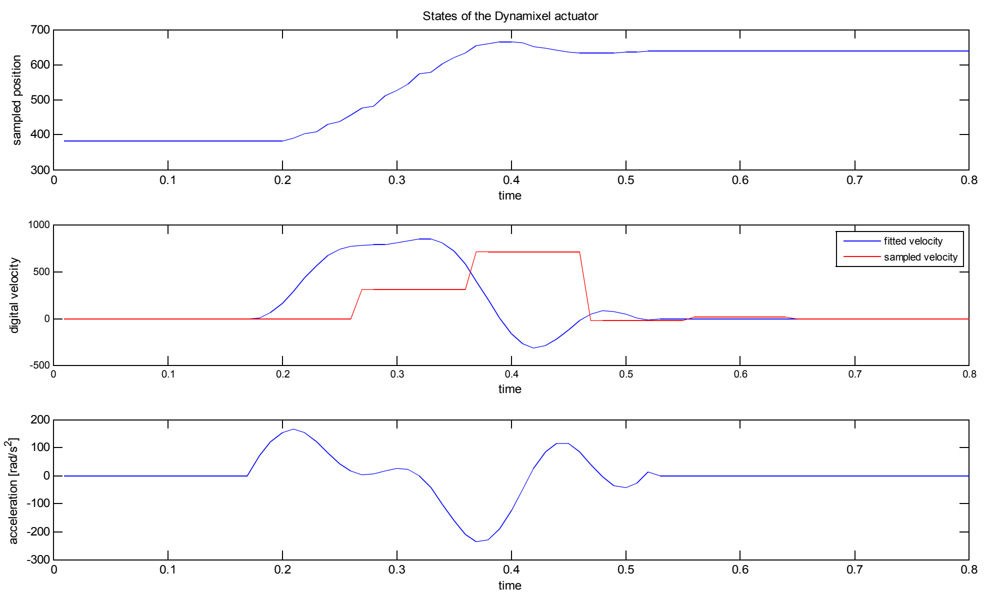


Figure 2.13: States of the Dynamixel servo obtained with curve fitting

3 Simulation model

The general setup for the simulation model of the Dynamixel consists of several parts. At first, there are the inputs from the main controller. These commands are given to the Dynamixels over the serial bus. The controller outputs the desired values to the servos, from which the servos themselves calculate how to get there. So the inputs for the Dynamixels are step functions. As both the final position and the maximum velocity can be given as a setpoint, and the simulation controller can handle only one input or error signal, a setpoint generator has to be built, that processes these inputs to one signal that can be fed into the controller. Also a model of the electric motor has to be made that drives the load.

3.1 Setpoint generator

Via the command bus the Dynamixel gets data from the main controller. The most important is the position setpoint, but as mentioned before, also a maximum velocity can be given as a constraint. The setpoint generator generates for every simulation step a setpoint for the internal controller using these two parameters. This generator should be able to handle changing inputs even if the previous goal wasn't reached yet.

To build the setpoint generator, the final position and the maximum velocity have to be merged into a single output signal. This can be done by integration. If a certain acceleration is taken and integrated, the corresponding velocity is obtained. If this velocity is integrated, the position is obtained. If the acceleration is assumed to be constant, ie. It is assumed that the servo always runs as quickly as possible behold of the given constraints, the velocity will change linearly and the position quadratically. For each simulation time step, a state machine determines what the acceleration has to be in order to reach the desired position and not exceed the maximum velocity. Then this acceleration value is integrated twice to obtain both the velocity and the position. The state machine for the setpoint generator is shown in figure 3.1.

As an example, if the servo is at standstill, it is in the 'rest'-state (state 1 in figure 3.1). This means that both the acceleration and velocity are zero. If a setpoint is given to move to another position, the setpoint will switch to state 2 or 4, dependent of the final position lies clockwise or counterclockwise of the current position. In this states the acceleration is set its maximum. The setpoint generator will stay in this state until the maximum velocity is reached or if the position error or traveling distance is smaller than the braking distance `x_b`. This distance is calculated to determine when a state change to negative acceleration is needed in order to achieve standstill at the desired position. If the maximum velocity is reached, there will be a state change to state 3 or 5, where the setpoint generator again waits until the position error is smaller than the braking distance to change to negative acceleration.

As mentioned before, the setpoint generator should also be able to handle new data while the previous goal position was not reached. This requires in every state checking all possibilities in order to go to the correct state to be able to reach the new position. If for instance the state machine is in the state for maximum speed (3) and the maximum speed changes to for instance a lower value, the state machine will switch to negative acceleration (4) to slow down as much is needed and then switches back to (3).

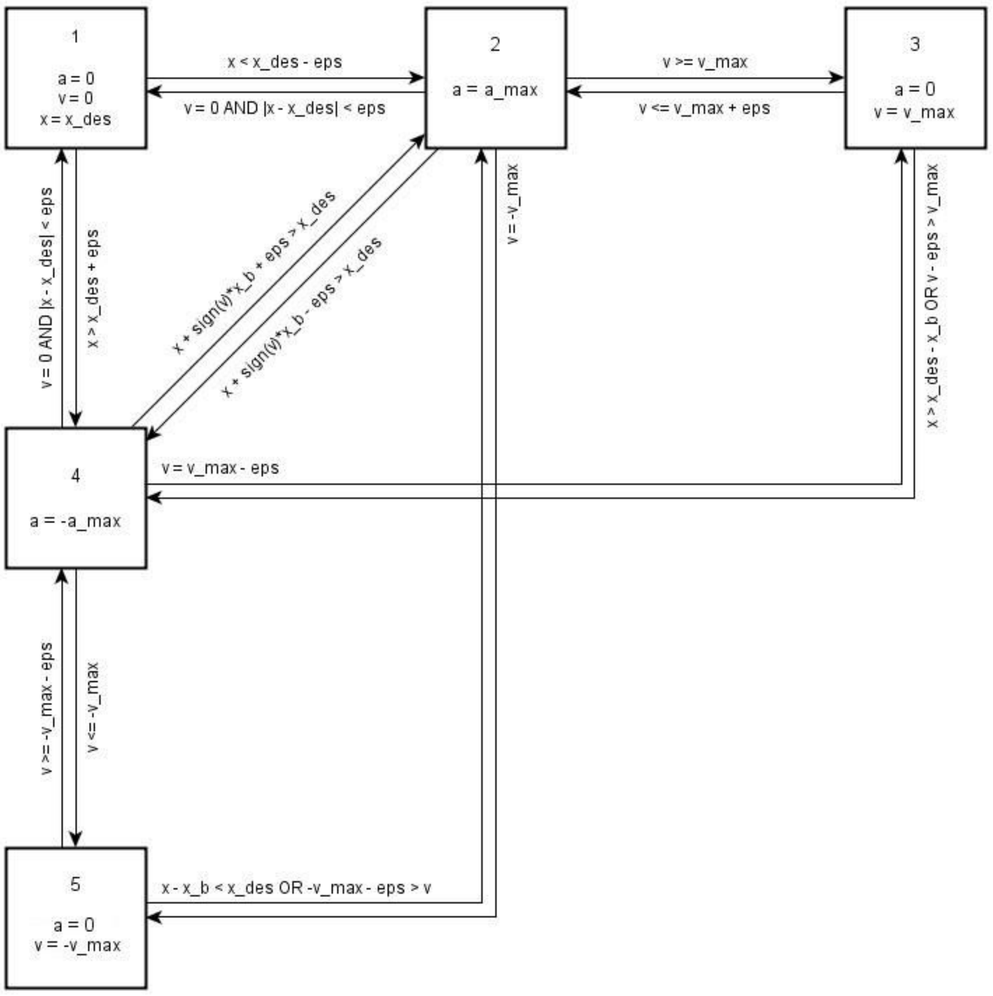

1
a = 0
v = 0
x = x_des
2
a = a_max
3
a = 0
v = v_max
4
a = -a_max
5
a = 0
v = -v_max
x < x_des - eps
v = 0 AND |x - x_des| < eps
v >= v_max
v <= v_max + eps
v = 0 AND |x - x_des| < eps
x > x_des + eps
x + sign(v)*x_b + eps > x_des
x + sign(v)*x_b - eps > x_des
v = -v_max
x > x_des - x_b OR v - eps > v_max
v = v_max - eps
v >= -v_max - eps
v <= -v_max
x - x_b < x_des OR -v_max - eps > v

Figure 3.1: State machine used in the setpoint generator

In figure 3.1 also a constant with the name `eps` can be noticed. This is needed due to simulation accuracy. If the state machine has decided that the position error is smaller than the braking distance, it changes state. At the next time step, the position error should still be smaller than the braking distance, but can be larger again due to the limited accuracy of 20Sim. This causes the state machine to switch back to the previous state and the state machine starts to oscillate. The constant `eps` has been used to prevent this. This constant is chosen several orders larger then the simulation accuracy. If the state machine has changed state because of the braking distance, the constant `eps` is added to the braking distance. Therefore the state machine cannot switch back until the position error is larger than the braking distance plus `eps`. However, the constant should not be chosen too large because then the accuracy of the state machine will become affected. If a new data comes in and the difference between the new desired position and the current position is smaller than `eps`, the state machine can't cope with this new position. In fact, the `eps` constant creates a dead zone over the position. In the simulations the `eps` constant is given a value of 0.001 rad, which is large enough to solve the oscillating problem, but still smaller than the resolution of the Dynamixel. In the results section 4.1, a figure of the correct working is included.

3.2 Controller

The type of controller in the Dynamixel is nowhere specified. So, based on the behaviour, the assumption is made that a PID controller is used. As the measurements reveal an overshoot of the position, a proportional gain with an integrating action is needed. This results in the overshoot with the correct frequency, but this oscillation has to be damped. Therefore a differentiating action is needed, but this influences also the frequency. With iterative steps, the best fitting parameters were found. The controller parameters were determined via simulations and compared with the measurements. The results are treated in chapter 4.

3.3 Model of the electric motor

As in every servo, the Dynamixel has an electric motor, which have to be modeled for the characterization. Figure 3.2 shows an extended model of this motor. This electric motor is driven by a PWM source. Such a source can be modeled as a current source.

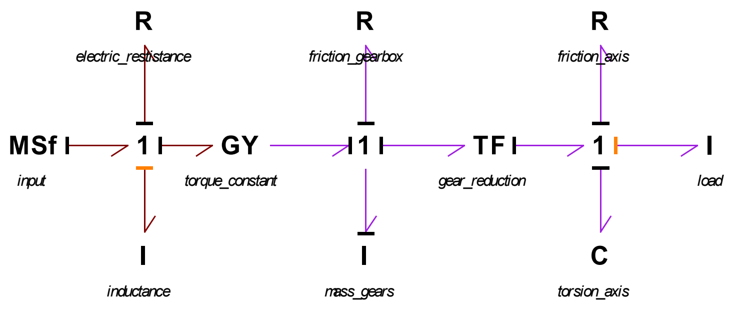


Figure 3.2: Extensive model of an electric motor

This means that the inductance of the windings of the rotor gets differential causality and thus doesn't play a role in the dynamic behaviour of the motor. On the mechanical side the torsion of the axis is being neglected. The transformer, which represents the gear reduction, is eliminated by adjusting the torque constant. In fact, this adjusted constant is the constant measured in section 2.3.2. The two frictions are combined into one friction component. Also the mass of the gears are not taken into account. This results in the reduced model of figure 3.3, with the corresponding parameter values in the table. The load is the camera, of which the moment of inertia is calculated.

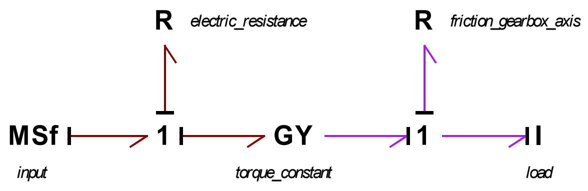


Figure 3.3: Reduced model of an electric motor

Electric resistance	6.3	Ohm
Torque constant	6.5	N/A
Friction	20Sim model, parameters in 2.3.2	-
Load	$1.35e^{-4}$	Kg m^2

4 Results

In this chapter the results of the designed components are presented. First, the output of the setpoint generator is shown, and second the results of the controller are given.

4.1 Results from the setpoint generator

In figure 4.1 the results of an arbitrary setpoint waveform are plotted. The amplitudes of this run are test values and not realistic values of the Dynamixel, but give a better view in the working of the setpoint generator. In figure 4.2 the position waveform is taken out of figure 4.1. In this plot it can be seen that the setpoint controller is able to handle changing inputs correctly while the desired position or velocity has not been reached yet. The states waveform shows the current state of the state machine.

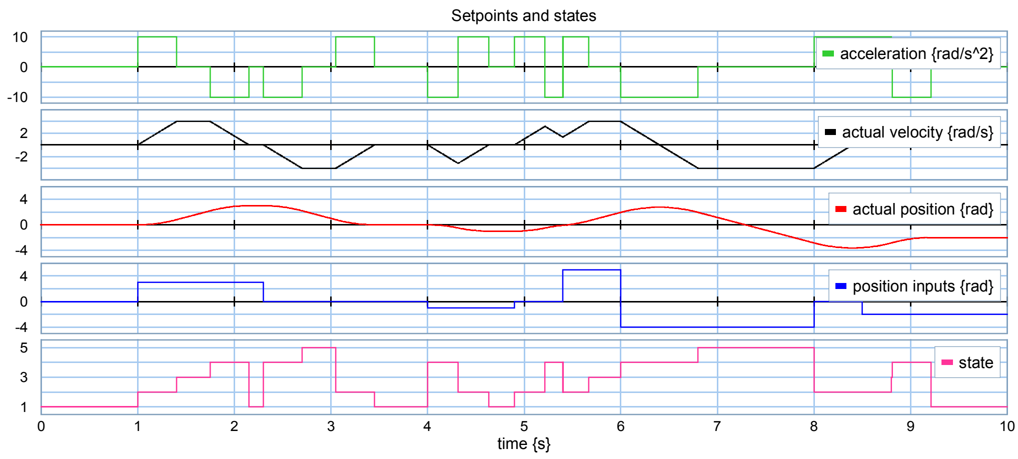


Figure 4.1: Setpoints and calculated states

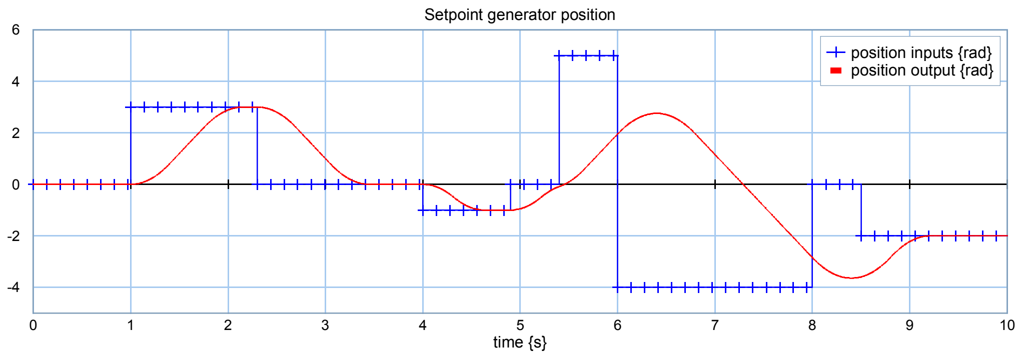


Figure 4.2: Position setpoints and output

4.2 Results from the controller

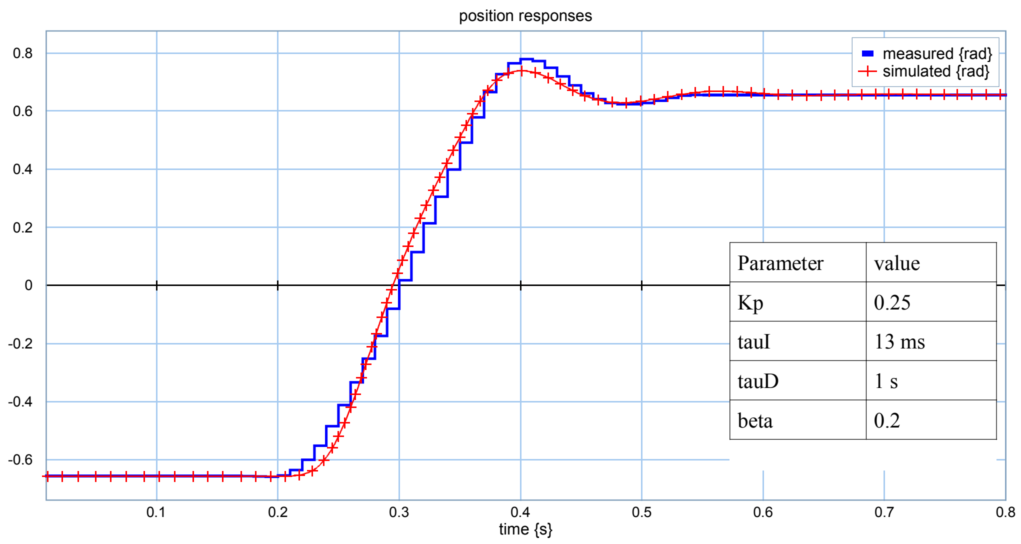
The parameters for the PID controller were obtained by comparing the outcome of the controller with a position curve from a previous measurement. The adjusting was done with the model of the electric motor included.

First the maximum value of the speed setpoint in the setpoint controller had to be found. This is needed because in the measured plots it can be seen that there is a linear region, indicating that the speed limiter is active or the servo operates at its physical maximum. $\frac{0.597--0.333}{0.36-0.26}=9.12[rad/s]$

$$\frac{9.12}{11.9}*1023=785[digital]$$

This reveals that the maximum speed set is about 75% of the physical maximum. This is used for the v_max setpoint of the setpoint generator. Then the parameters for the controller were found.

First was tried to find a proportional gain for which an overshoot occurred. Then the integrating time constant was tuned to get the correct oscillating frequency. This was damped by the differential time constant, but also influences the oscillating frequency, so iterations were needed to come to a solution. The results for the position are in figure 4.3.



Parameter	value
Kp	0.25
tauI	13 ms
tauD	1 s
beta	0.2

Figure 4.3: Measured and simulated position responses

In figure 4.4 the velocity responses are plotted. The digital values were converted to rad/s for comparison. As can be seen, the curves are very different. This was also noticed in section 3.2.3 where was tried to retrieve an acceleration signal from the measurement data. There is a extra top on the simulation response that goes over the 9.12 rad/s as calculated before. This is because of the differentiating action. It can also be seen in the position plot, where at about 0.27 s, the curve is steeper than the measured position response, which corresponds to a higher velocity.

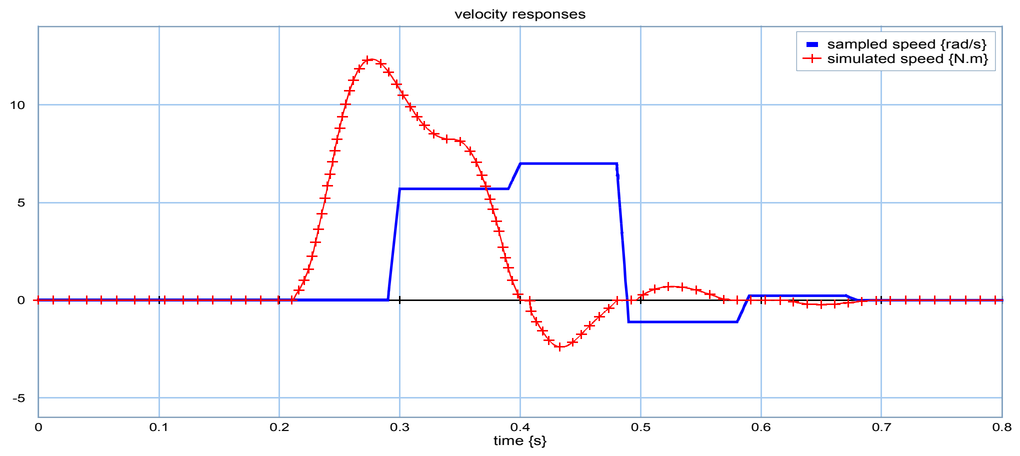


Figure 4.4: Sampled and modelled velocities

The load curve was also sampled from the Dynamixel. This is plotted in figure 4.5, together with the shaft torque of the Dynamixel. This load is not converted to digital values, because it was not known what exactly the sampled torque curve was. It can be seen from the figure that is not directly the shaft torque. It can be the maximum allowed torque at a certain time. This is a parameter which can be set in the Dynamixel servo as a function of the position error. This is shown in figure 4.6. The letters A to E represent parameters which can be sent to the Dynamixel to change the curve. If it is this curve what is plotted in figure 4.5, then there is probably a wrongly captured sample value at 0.3s. In the output torque there are spikes at every time the velocity is zero. This is due to the static friction. For this plot, the static friction was reduced, because otherwise the curve was not visible because of the amplitude of the spikes.

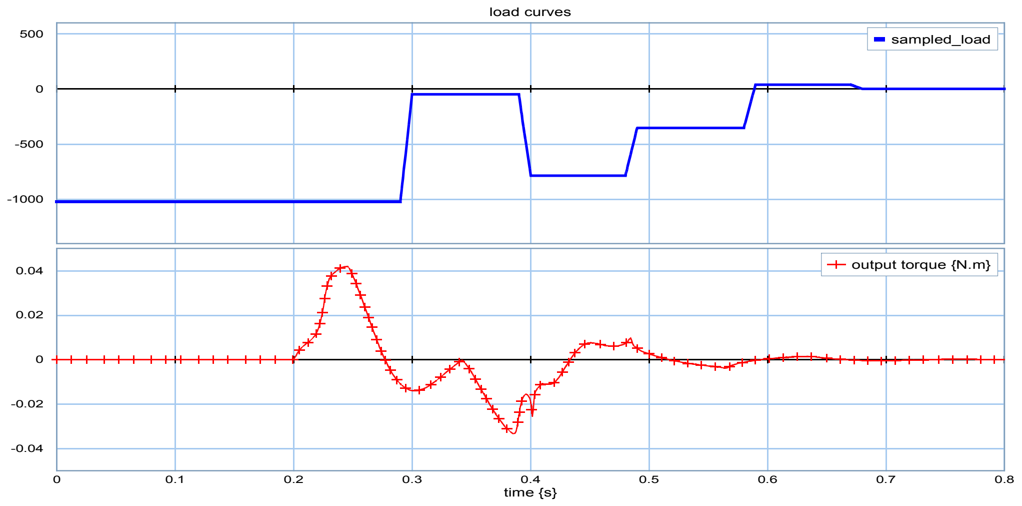


Figure 4.5: Sampled torque and model shaft torque

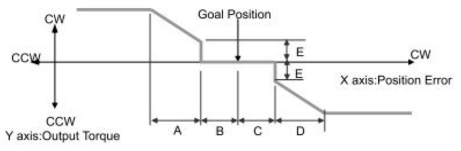


Figure 4.6: Torque curve from the Dynamixel datasheet

4.3 Combined results

In the above sections the results of the main submodels are treated. In this section the results of the whole model are presented. The 20Sim model is shown in figure 4.7. The main components can be identified to be the setpoint generator, the controller and the electric motor (with gearbox included). As the model works analogous and the inputs and outputs are discrete, D/A and A/D converters are inserted. In the real Dynamixel there is no velocity sensor, but due to algebraic variables the flow sensor is inserted to measure the shaft velocity.

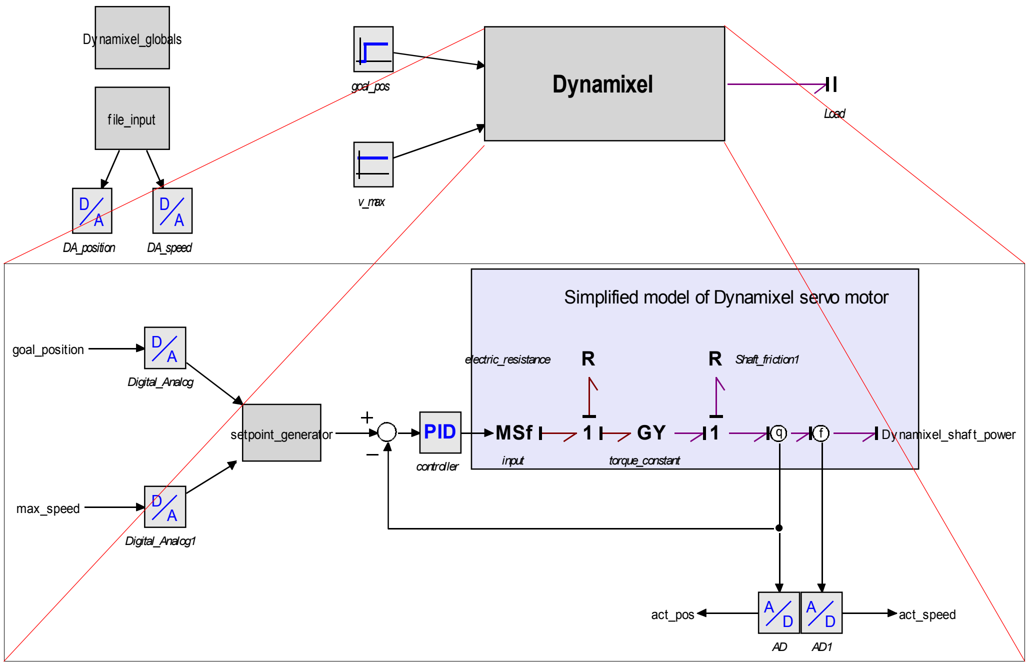


Figure 4.7: Model of the Dynamixel

In the figure only inputs and outputs for position and velocity are present. There is another main variable which is not implemented, the torque. It is possible to set a list of constraints on the torque as well, so it is quite important. The main reason for having this not being implemented are the torque curves in figure 4.5.

5 Conclusions and recommendations

The assignment was to characterize the Robotis Dynamixel servo or servo and make a simulation model in 20Sim. In this chapter conclusions are drawn from the results and recommendations are given.

To be able to characterize the Dynamixel, information about this servo was collected. Therefore the datasheet and the supplied kit books were examined. Further the servo was opened to see what is inside and how that works. Also measurements were done on the mechanical and electrical port.

The mechanics inside did not reveal surprises. The mechanical assembly consists of an small electric motor with 5 gears to create a total gear reduction of 254:1, so the datasheet is correct, as this number first seemed to be very large. The gears are made thicker and tougher towards the output shaft to be able to handle the increasing torque.

The position tracking of the output shaft is done with a position sensor. This sensor is a linear potentiometer that probably goes to the A/D converter of the micro-controller. This means that the velocity is obtained by differentiation. It is still unclear how the Dynamixel measures its load at the shaft. This is because the PCB is not reverse engineered to obtain the schematic. It is most likely that there is a current measuring circuit on the PCB.

Also measurements are performed to gain information of the behaviour of the Dynamixel. Rob Reilink had done some movement measurements, which resulted in movement profiles. As he had written the measurement program for a Mac, it could not be used for a computer. I am quite bad at programming and have only experience with microcontroller programming. Therefore, I have used his results. These measurements are used to extract the acceleration of the Dynamixel and as comparison for the simulation model that has been made. As can be seen in figure 2.5, not all variables are updated at the same frequency. Rob has sampled this variables at 100 Hz, but only the position variable is updated at at least this frequency. The speed and and load variable are apparently updated at 10 Hz, as they have the same value for ten sample times. The velocity also lags behind the position. This may be caused by the differentiation of the position, which can be demanding for the 8-bit controller. The servo output axis experiences some overshoot. This may have been done to let the servo reach its position faster.

Further, measurements has been done to obtain a curve for the friction against radian velocity. Static friction is measured as well as viscous friction. With the measurements a friction curve (figure 2.8) is created. It is recommended to improve this friction curve, because it is not completely correct. The static friction should be a local maximum in the friction curve, but in this case it is not. The function still increases for very low angular velocities, where it should decrease to a minimum, after which the viscous friction becomes dominant. This is caused by the lack of measurement points in this area, due to setup limitations. Also the Coulomb friction returned form the fitting toolbox was negative. This also indicates the incorrectness of the function. The function is still used in the model, because the order of magnitude is similar and therefore the function is usable. It is recommended that more measurements should be done over the total operating region (0 ~ 12 rad/s) to provide a better friction curve.

The torque constant for the gyrating transducer is also measured. This did not reveal a linear relation between current and torque. A reason can be the presence of the control circuit that was in between. The measurements were taken with the instruments at the power lines and not directly at the motor. This shows a torque constant of 6.55. This seems high, but this is the geared down constant. Correcting the factor for the gearbox means dividing this number by the gear reduction. This gives a number of 25 mN/A for the motor itself. This is a common figure for small DC motors.

With the performed measurements and information about the datasheet a simulation model has been built that generally behaves like the actual Dynamixel. This behaviour is not perfect as assumptions and linearizations were made.

The Dynamixel servo has multiple inputs that all act on the same output axis. As the controller in the simulation model can handle only one signal, these multiple inputs had to be combined to produce one input signal for the controller. This was done with a setpoint generator. This setpoint generator is basically a state machine that determines the output based on the input signals. As the two inputs velocity and position have a direct (integral) relation, this is used in the setpoint generator. Started is with a constant maximum acceleration. This is the acceleration measured before. This is integrated to obtain the velocity and integrated again to obtain the position. The state machine checks if this variables do not exceed the input values and changes state accordingly to keep the output within the boundaries of the inputs. The position signal is then transferred to the controller. This position signal thus is one signal that incorporates the velocity, because the maximum rate of change of this signal equals the maximum velocity set as input for the Dynamixel.

The setpoint generator starts with an acceleration and is then integrated twice. This acceleration signal is tried to extract from the measurements. This was done by differentiating position measurement twice. The position was differentiated and the obtained velocity was curve fitted to get a smooth curve. This curve fitted velocity was differentiated to get the acceleration. This does also cause a time delay, but it is only one sample time. This method also introduces errors in the curve fitting, but these are small if the fitting is good. Afterwards, it was better to curve fit the position directly, because the errors are not amplified then by differentiating, and the curve fitting will be more precise. If this fitted position is differentiated, the error should be lower than with the fitted velocity.

The controller in this simulation model is of the PID type. This controller was chosen based on the measured movement profiles. With the measured constants this controller variables are tuned to match the movement profiles as good as possible.

The controller then tries to keep the error to a minimum and its output is the input for the simulated electric motor. But there was a peculiarity: The controller is mostly designed to have no overshoot or very little overshoot, but in this case there should be overshoot, because the measurements showed such.

The parameters for this controller could be found to match quite well. During movement the error is less then 0.1 rad as can be seen in figure 4.3. Also the steady state error is very small. This would not be seen in the digital output value. The start of the controller is a little too slow, this is because of the friction model and the differential gain, which has a slow starting response. Also because of this differential action is the steep section in the position response. This causes the extra top in the velocity response from figure 4.4. The measured velocity response is in any case difficult to compare with, because of the delay and the few sample values. Globally the shape is similar, which shows that the model is implemented correctly.

The maximum torque curve is not implemented in the simulation model, because it was first not known what that curve meant. It is still not very clear, but it could represent the maximum torque curve in the datasheet which is shown in figure 4.6. This is one of the main things that should be implemented as future work, as this is used for robot arm grippers and so on.

Other items that should be done in future work are the detailing of the model by adding more parameters of the Dynamixel servo into the model. Also the digital interface from and to the Dynamixel has to be developed. For this some other network description tools are needed because 20Sim cannot simulate bi-directional signal flows. Maybe multiports can be used, but then you still end up with two single-directional signal lines.

A Appendices

A.1 Information about the Robotis Bioloid kits

The Bioloid robot kits contain all bits and pieces with which one can build their own robot. The most important are the servos, called Dynamixels, a sensor module and the main control unit. The servos/servos are controlled by the control unit via a bus. For programming the robot, there are three/four tools at hand: a graphical programming tool, a motion editor for programming standard poses to the main controller and a robot terminal where one can directly control the robot from the computer or one can load their own C-coded program into the robot.

A.2 Hardware

Below is a listing of the contents of the comprehensive robot kit and the specifications of the main parts.

A.2.1 Comprehensive kit contents

- CM-5 Main control unit
- AX-S1 sensor module
- 18 AX-12+ servo Modules
- 25 connecting cables
- Battery pack
- Power supply/Charging unit
- Variety of connecting brackets
- Lots of screws and nuts
- Serial programming cable
- CD with programming tools
- User's guide
- Quick start guide

A.2.2 Specifications

- CM-5 Main control unit

Processor:	ATmega128
Bus protocol:	half duplex UART
Bus speed:	max. 1 Mbps
Bus connections:	4
ID range:	0-253
Max. ID's conn'd:	30

- AX-S1 sensor module

Processor:	ATmega8
Bus connections:	2
Voltage:	7V-10V (recommended 9.6V)
Supply Current:	40mA
Feedback:	- Infra-red distance measurements (left, center, right)

	- Brightness measurement
	- IR remote control Tx/Rx
	- Microphone input
	- Input voltage, temperature, etc.
Output:	Speaker

- AX-12+ servo Module

Processor:	ATmega8
Bus connections:	2
Voltage:	7V-10V (recommended 9.6V)
Sink Current:	900mA max
Operating angle:	300°, endless turn
Resolution:	0.29°
Max. Holding Torque:	1.176Nm @ 7V, 1.617Nm @ 10V
Max speed[2]:	114RPM

- 25 connecting cables

Length:	6cm-20cm
Wires:	3
Connector[3]:	Molex SPOX 2.5 mm

- Battery pack

Type:	NiMH
Nr. of cells:	8
Voltage:	9.6V

- Lots of screws and nuts

M2 screws:	540
Length:	6mm-30mm
M2 nuts:	400
M3 screws:	30
Length:	10mm
M3 nuts:	10

A.2.3 USB2Dynamixel

An optional part that can be bought for the Bioloid Robots is the USB2Dynamixel converter. This converter is a USB to serial converter for computers which

[2] As listed on page 17 of the AX-12+ data sheet. On page 3 in the main specifications is a parameter sec/60degree. This corresponds with 37.2RPM @ 7V, 51RPM @ 10V. This parameter may be the response time of the 'Dynamixel' to turn 60 degrees.

[3] Below are the order numbers of the connectors for reordering

Male (header):	Molex nr. 22035035 (vertical)	Farnell nr. 9979620
	Molex nr. 22057035 (right angle)	Farnell nr. 9979689
Female (housing):	Molex nr. 50375033	Farnell nr. 9979557
Crimp terminals:	Molex nr. 8701040	Farnell nr. 9979530

don't have a serial port. But the most important thing is one is able to control Dynamixels directly from the computer without the CM-5 control unit.

A.3 Software

Three programming tools are supplied with the Bioloid Robot kit. First is the Behavior Control Program. Then, the Motion Editor en third is the Robot Terminal. They are explained briefly.

A.3.1 Behavior Control Program

The Behavior Control Program (BCP) is a simple graphical programming environment. All commands are visualized as blocks that are put on a line. This works intuitively, but overview is easily lost as programs grow. Due to the size of the blocks, only 16 or so lines are visible on the monitor. The scroll function doesn't work fine as well.

The number of commands is also very limited. These are all the commands:

program control commands (`START, END`),
conditional branching commands (`IF,ELSE IF,ELSE,CONT IF`) with conditional operations (`=, >, >=, <, <=, !=`),
program sequencing commands (`JUMP & CALL/RETURN`),
numeric commands (`COMPUTE`) with operands (`+,-,*,/` and `&(and),|(or)`)
assignment commands (`LOAD`).

Almost every address in the Dynamixels that can be written, has got a graphical symbol, i.e. the moving speed of the Dynamixel is programmed to address 32. Then there is a symbol which says SPEED with the ID of the Dynamixel and the number behind is the value that has to be written to it. These symbols can also be used for the IF statements, then the value of the corresponding address is read.

A.3.2 Motion Editor

The Motion Editor is a package that allows the user to move the motors of a robot simply by increasing or decreasing the number that describes the motors current position.

Motions are built up frame-by-frame - very similar to a story board in an animation sequence. This allows quite complicated 'animations' to be quickly programmed and tested. Once a motion has been defined it can then be downloaded into the Bioloid's memory and called from the Behaviour Control Program.

A.3.3 Robot Terminal

This small application allows the user to directly give commands to the CM-5 control unit. The program connects the PC's keyboard and monitor to the CM-5. Text outputs from the control unit appear on the monitor and commands from the keyboard are via the serial cable transferred to the control unit. One is than able to control the Dynamixels via the Robot Terminal and the CM-5 with the keyboard. The monitor is displaying what's going on.

A.4 Web resources

Homepage manufacturer
www.robotis.com

General info
http://robosavvy.com/site/index.php?option=com_content&task=view&id=82&Itemid=81?

http://robosavvy.com/site/index.php?option=com_openwiki&Itemid=&id=robotis_bioloid

http://www.bioloid.info/tiki/tiki-index.php

About AX-12+
http://www.bioloid.info/tiki/tiki-index.php?page=AX-12+page

About CM-5
http://www.bioloid.info/tiki/tiki-index.php?page=CM-5+page

Forums
http://forums.tribotix.info/forum1/
http://robosavvy.com/forum/viewforum.php?f=5

Online Shop
http://robosavvy.com/store/

A.5 Measured responses from the Dynamixel

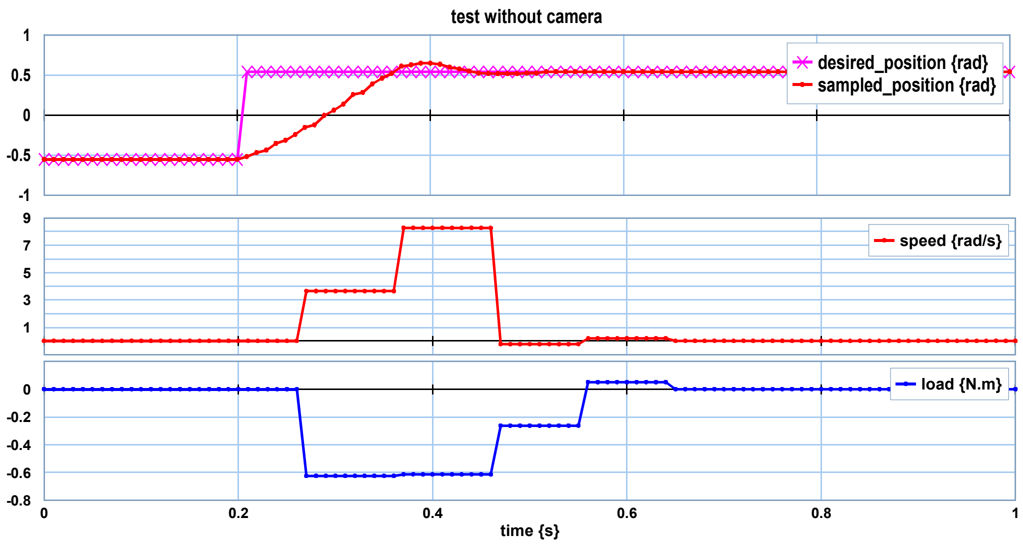


Figure A.1: Responses without load

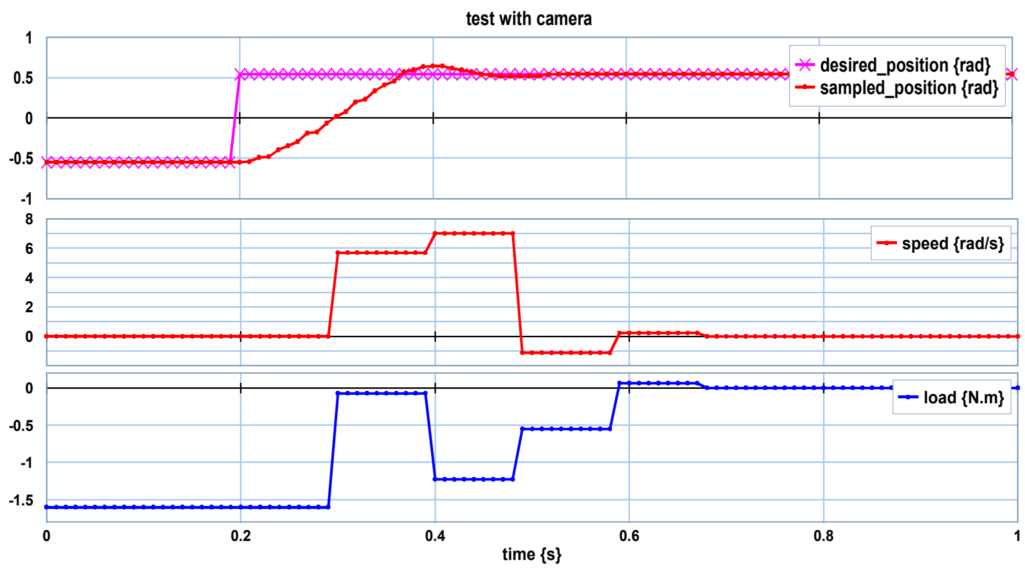


Figure A.2: Responses with camera as load

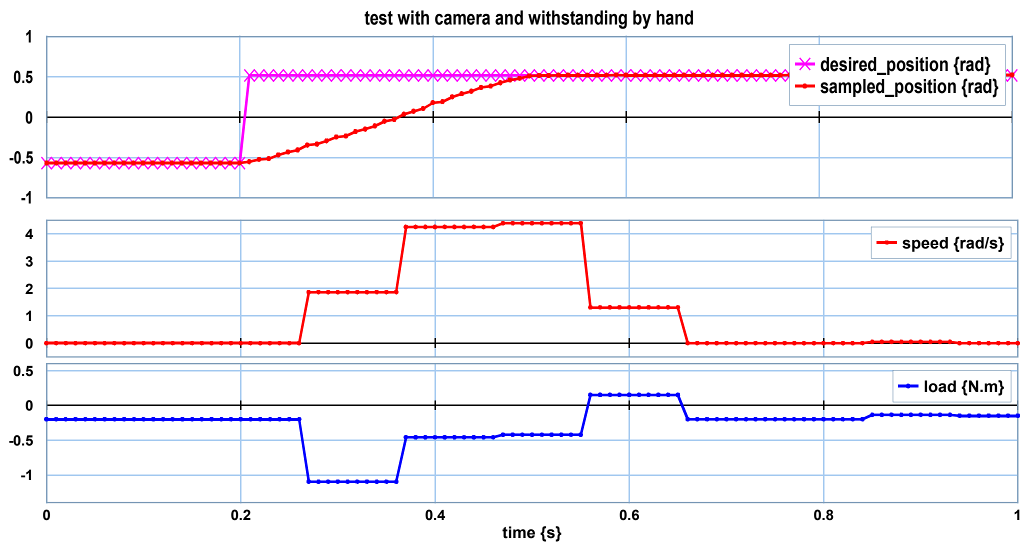


Figure A.3: Responses with camera and extra friction

References

[1] Datasheet Dynamixel AX-12+
http://www.robotis.com/zbxe/5419

[2] Datasheet muRata SV01 position sensor
http://www.murata.com/catalog/r50/r50e15_l0595.pdf

[3] Wikipedia.org: Specification of UART
http://en.wikipedia.org/wiki/UART

[4] Manual USB2Dynamixel
http://www.robotis.com/zbxe/6255