Aalborg University

# Teleoperation of a surgical robot using force feedback

Electronic & IT: Group:
Control and Automation: CA-735

Worksheet

20. December 2016

Dániel Bolgár
*dbolga16@student.aau.dk*

Simon Bjerre Krogh
*skrogh13@student.aau.dk*

Filip Marić
*fmaric16@student.aau.dk*

Nicolas Silviani
*nsilva16@student.aau.dk*

# Contents

# Introduction 1

The last decades were characterized by rapidly developing information technology and electronics. Today, we're on the brink of a fourth industrial revolution, which blurs the line between physical and computer systems [1]. One of the great beneficiary of this recent development was the health industry. The present project aims to analyze and improve one of the products of the fusion between control engineering and health industry, a minimally invasive surgical robot (MISR).The specimen being analyzed is a da Vinci robot manufactured by Intuitive Surgical.

The da Vinci robot is controlled by a specially trained surgeon. The robot extends the capabilities of the surgeon. It assists in performing minimally invasive surgeries (MIS) and minimizing the insertion cut inflicted on the patient. This results in faster recovery time for the patient[2]. The surgical tools are teleoperated by the surgeon using a control station equipped with spatial controllers and 3D video feed from the tools. The robot is able to improve on the surgeon's precision. A further advantage of the da Vinci robot is the possibility of remote control. The surgeon can be hundreds of kilometers away from the robot, still able to control it through the internet. This characteristic can prove to be quite useful in rural areas, where medical care is limited. Whenever an operation needs to be executed, a specialist from the specific field can sign into the robot and conduct a surgery. The technology can even be adjusted for military purposes. If an injured soldier needs treatment in a highly dangerous zone, the medic can send the robot to the soldier and conduct the required surgery, without risking their life. The main issue in increasing the distance is the delay in the communication also increases, which eventually can lead to an unstable behaviour in the system.

With all these advantages there comes a significant drawback compared to classical direct surgery. The operator has to solely rely on what is seen on the 3D video feed, since the controller lacks force feedback feature[3]. This means losing kinesthetic sense during operation, which sets back the accuracy of the surgeon and can lead to surgical accidents.

Studies show that an added force feedback feature increases the success rate of robotic surgeries[3]. Therefore the present project aims to implement the high demand feature of force feedback. The original da Vinci robot's controller station lacks any actuators, which means that the force feedback feature can not be implemented in it, thus another controller has to be used. This project utilizes a spatial haptic controller Geomagic Touch (GT) haptic device as the controller directing the da Vinci robot's end-manipulator called EndoWrist.

In order to provide the sensation of directly controlling the EndoWrist, the GT needs to output the force affecting the EndoWrist to the surgeon. Two aspect is needed to be covered. The first one is including the force fed back and the other one is the communication between the GT and the EndoWrist.

The main issue in feeding back the force is that the force cannot be directly measured. Regulations states that each surgical tool needs to undergo a sterilization process between

operations, where the tool is treated with high temperature detergents. Furthermore, The EndoWrist has to be discarded after a number of uses[4]. These two regulation makes it economically not worthwhile to implement expensive, high temperature tolerant force sensors on the end-manipulator, just to be discarded after a couple of uses. A different approach needs to be sought.
The current applied to the actuators of the EndoWrist is highly correlated with the force applied, but due to the fact that the tool is nonlinear the force calculated directly form the current will be misleading to fed back. Therefore a model of the EndoWrists dynamic including the actuators is needed to give a better estimate of the actual force applied to the EndoWrist.

It is commonly accepted that in order to have a realistic force feedback, the feedback loop should have a frequency equal or superior to 1000 Hz[5]. However, this value is widely debated as a frequency of 300 Hz has been proven to provide an acceptable estimation of the force[6]. Similar studies have shown that the minimal acceptable frequency would lie between 550 and 600 Hz [7]. The main difference in these studies may lie in what a "realistic" force is as it could denote an accurate estimation of the force or simply estimate the trend without great precision. In this study, the goal is to build a model that will estimate the trend of the force. Thus, it is not required to reach 1000 Hz. The available equipment set constraints on the frequency achievable, the present project aims at reaching the frequency of 550 Hz given by [7].

This project is based on a previous research. The previous setup could only reach 100 Hz in communication frequency[8], which does not satisfy the refresh rate demand of force feedback. Therefore this project aims at increasing the connection speed to the minimum requirement of a 1000 Hz. Furthermore this project aims at implementing a force feedback control loop utilizing the setup available at Aalborg university based on a dynamic model of the surgical tool.

### Summary

In this project two main challenges were addressed in order to implement force feedback:

- Increasing the refresh rate of the communication between the embedded system and the computer.
- Deriving and implementing a model that estimates the force applied to the end effector using only the data available on a da Vinci robot. A model that reliably reproduces the trend of the force without precisely giving back the absolute value is deemed to be sufficient.

# System overview 2

In this chapter a system description is given. This is made for the reader to get a better understanding of the setup which was available at Aalborg University. This description includes the following points and will be described in the same order, then a summary of the system overview is provided.

- The da Vinci robot,
- The EndoWrist,
- The mechanical test setup,
- The motors,
- The embedded system,
- The Geomagic Touch

## 2.1 Da Vinci robot

The da Vinci robot is a minimal invasive surgery (MSI) robot used in surgeries such as cardiac, coloretal and gynecologic surgeries[9]. The version available at Aalborg University is a member of the first generation of da Vinci robots, see *Figure 2.1*.

(a) The da Vinci robot or slave manipulator. (b) The master console for controlling the slave manipulator.

Figure 2.1: Full view of the da Vinci surgery system which include the slave manipulator and the master console

It consists of two main parts, a master console, see *Figure 2.1b*, and a slave manipulator, see *Figure 2.1a*.

- The surgeon uses the master console to control the slave manipulator. It consists of two eye pieces, which display a high resolution 3D view of the surgery for the surgeon.
- The slave manipulator is the robot which is controlled from the master console by the surgeon. It consists of four arms that with 6-7 actuated degrees of freedom (DOF) each, depending on the tool attached.

### Slave manipulator arm

The slave manipulator arm consists of three parts, arm, hand and tool, see *Figure 2.2*.

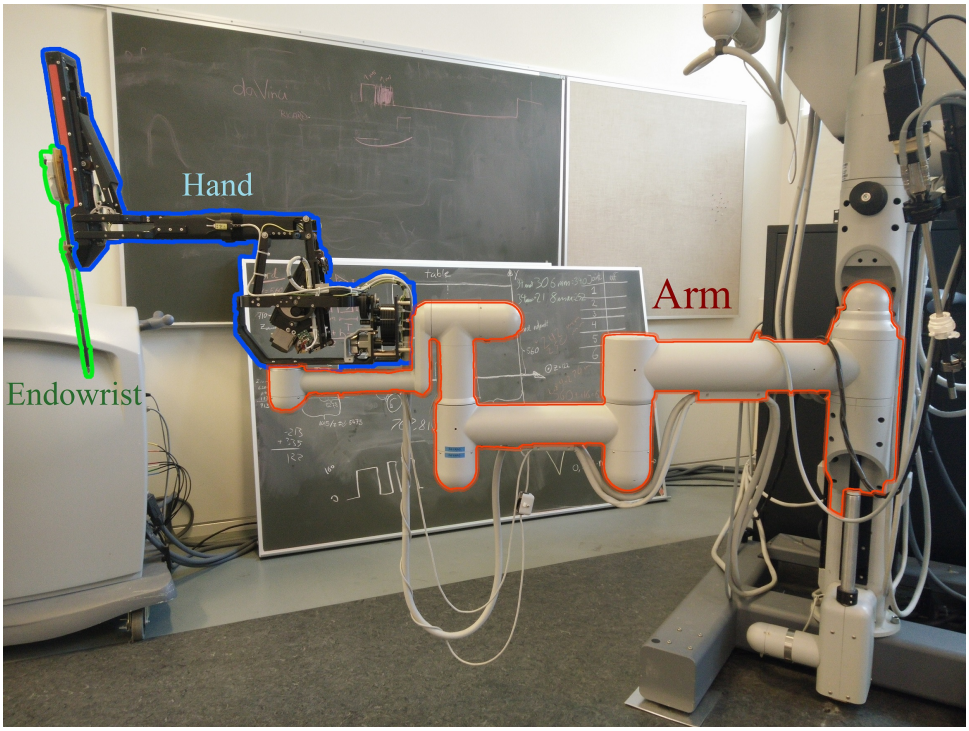



Figure 2.2: One extended arm of the da Vinci robot. The tool (EndoWrist), hand and arm are marked.

1. Arm:

   The arm of the da Vinci is the part being the furthest away from the patient. During surgery, it is locked to a specific position. Between surgeries, its 6 DOFs can be adjusted.
2. Hand

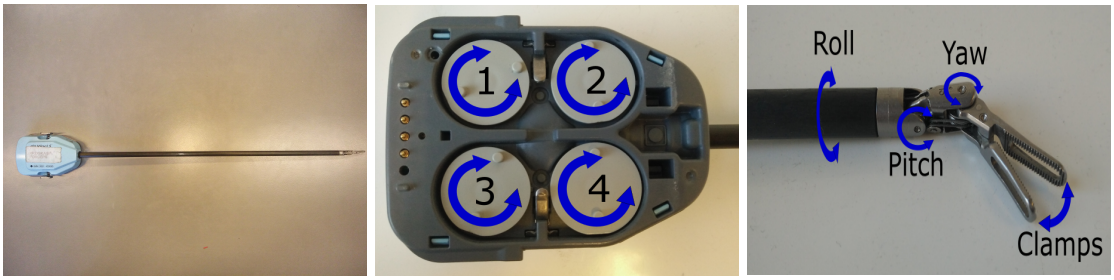
   The hand is at the end of the arm and has three DOF, two rotational and one translational, which can be actuated during surgery.
3. End-effector tool (EndoWrist)

   The end-effector tool is connected to the hand, it is only capable of doing rotational movement on its own. Together the tool and the hand have 6 - 7 movable DOF that can be actuated. The tool is the instrument which interacts with the patient during surgery. There are several different tools available, each with their own functionality.

The present project concentrates on implementing haptic feedback control for the end-effector called EndoWrist. Controlling the robot arm is not a subject of this project. The end-effector was separated from the da Vinci system. The da Vinci's master console controller lacks actuation. In order to be able to provide haptic feedback, a Geomagic Touch, see Section *2.6: Geomagic touch*, has been implemented as a replacement. The original master console will not be further analyzed.

## 2.2 EndoWrist

An EndoWrist, see *Figure 2.3a* is a surgical tool which can be manipulated in a similar manner as a human wrist with two fingers. Therefore it has four DOF, see *Figure 2.3c*. This enables roll, pitch, yaw movement and an open-closing mechanism that acts as the thumb and index finger of a hand. It is used in surgical procedures such as Laparoscopic surgeries, better known as minimally invasive surgery (MIS), where small incisions in the human body is made during the surgery. Since the incision cuts are small, blood loss during the surgery and the risk of infection is reduced. This has a positive effect on the recovery time for the patient.




(a) Full view of an EndoWrist (b) Actuator plates, which can alternate the end effector position (c) End-effector of the EndoWrist

Figure 2.3: The EndoWrist and how the interaction with it is made

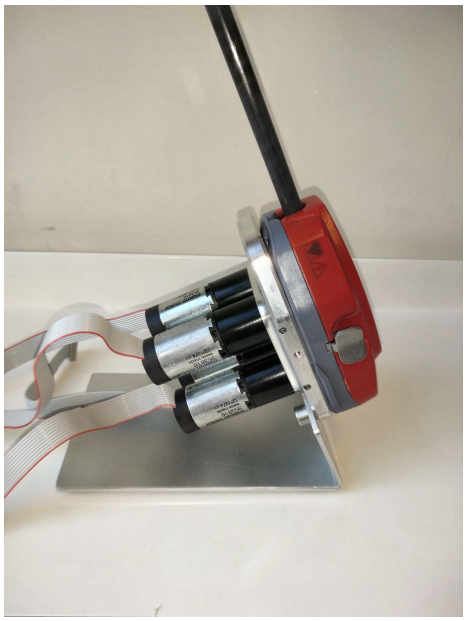
The end-effector is manipulated by the four wheels seen on *Figure 2.3b*. Wheel one and three define the movement of the yaw and the closing mechanism for the clamps. Wheel two and four moves the pitch and roll. The EndoWrist is cable driven, which provides the opportunity of making the EndoWrist small. The main drawback of the cable system is that it makes the system nonlinear, since the inherent frition absorbs a significant amount of the input power. Therefore the end-effector's power output is lower than the power input.

## 2.3 Mechanical test setup

The mechanical test setup available at Aalborg University can be seen on *Figure 2.4*. It includes four of the motors described in Section *2.4: Motor* attached to the four rotational plates seen on *Figure 2.4a* and *Figure 2.4b*. These plates connect the motors and the EndoWrist, such that the EndoWrist can be manipulated. See *Figure 2.4c* and *Figure 2.4d* for the attachment of the EndoWrist to the EndoWrist holder.

(a) EndoWrist holder. Front view

(b) EndoWrist holder with motors at the back. Side view



(c) EndoWrist holder with EndoWrist and motors. Side view

(d) Full view of the mechanical test setup

Figure 2.4: Four pictures of the mechanical test setup available at Aalborg University, which include motors, EndoWrist and EndoWrist holder

This setup can manipulate the EndoWrist in the same manner as if the tool was connected to the da Vinci robot, i.e. it is able to actuate the rotational DOFs of the end-effector.

## 2.4 Motor

The four motors actuating the EndoWrist is sold as a bulk solution which include motor[10], gear[11] and a position encoder[12], see *Figure 2.5*. These motors are direct current (DC)

motors. The main parameters for this bulk solution are listed below:

- Torque constant: $10.9\frac{\text{mNm}}{\text{A}}$
- Nominal continues current: 0.681 A
- Gearing ratio: 1:19
- Encoder precision: 512 counts pr. rotation. With the proper programming, see section 5.4, 2048 positions can be distinguished in one rotation.


(a) The Maxon 110160 DC motor.

(b) The planetary gearhead, 110356, equipped with sleeve bearing.

(c) The encoder, 201937, used for getting angular data.

Figure 2.5: The combination gear disassembled

The control of these motors are done through the ESCON controller boards, connected to the sbRIO, as shown on *Figure 2.6*. A detailed description is provided in Section *2.5.2: ESCON 50/5 motor controller*.

## 2.5 Embedded system

The EndoWrist, see Section *2.2: EndoWrist*, is actuated by 4 Maxon DC motors, see Section *2.4: Motor*. Each motor is equipped with an ESCON motor controller. The ESCON controllers receive the control reference from a single-board reconfigurable input-output board (sbRIO). For a graphical illustration of the connection see *Figure 2.6*.

### 2.5.1 Interfacing

The interfacing between the subsystems have a linear flow, i.e. every data propagates through the same path, as shown on *Figure 2.6*. Each interface uses bidirectional communication. The communication between the Geomagic Touch and the computer, and the communication between the computer and the sbRIO both happen using UDP. The sbRIO's in-built microprocessor and FPGA communicates using onboard wires. The ESCON controller and the FPGA interfaces using electric wires. Information exchange between them happens using pulse width modulation (PWM) signals for numeric data and digital signal for boolean data.

### 2.5.2 ESCON 50/5 motor controller

For driving the motors, four ESCON motor drivers of the type 50/5 (part number 438725) have been implemented, see *Figure 2.7*.

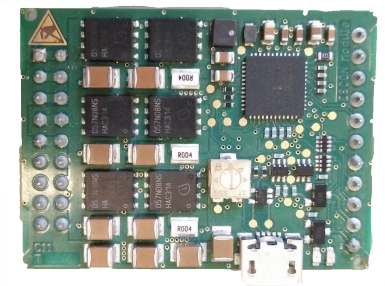


Figure 2.7: ESCON 50/5 DC motor controller[13]

These are controllers made for permanent magnet-activated brushed DC motors. They can deliver a power up to 250 Watt for the motor connected to the controller.

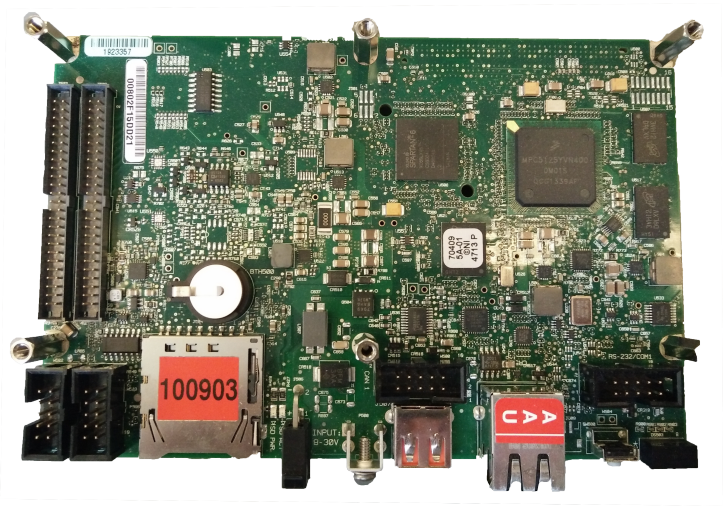
The ESCON 50/5 has three operating modes

- Current controller
- Speed controller (open loop)
- Speed controller (closed loop)

The reason we chose speed controller with inner loop current control is explained in 5.4. Each motor hosts an encoder that provides relative angular position information to the FPGA built into the sbRIO board. The ESCON controllers send the speed and current data of the motor to the sbRIO through analog outputs.

### 2.5.3 sbRIO

The embedded system controlling the EndoWrist contains a **Single Board Reconfigurable Input/Output 9636** (sbRIO) controller, see *Figure 2.8*. It is responsible for interfacing between the surgical robot and the computer running the robotic operating system[8], see Section *3.1.2: ROS*. It reads the data sent by the ESCON drivers, sends them to the PC and position controls the motors based on the received reference signals. The sbRIO has built in safety protocols that disable the motors in case of error.




<

Figure 2.8: The sbRIO 9636 board[14]

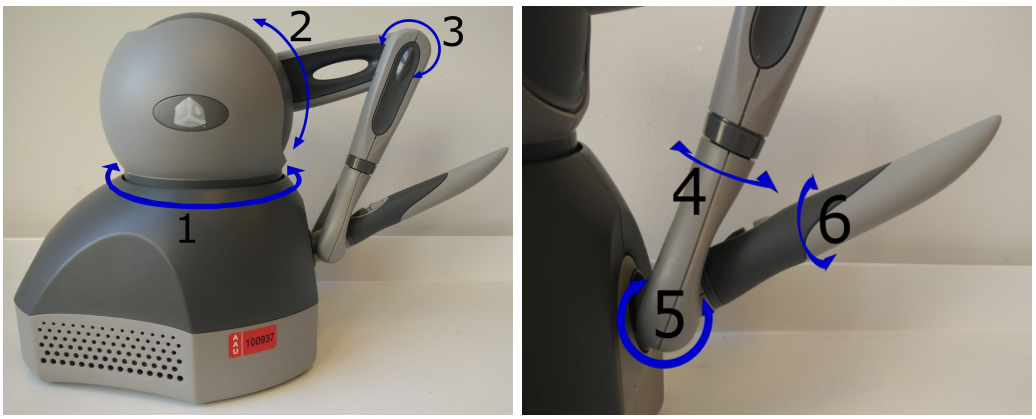
The controller consists of

- 400 MHz real time processor
- 256 MB of system memory and 512 MB nonvolatile memory
- Reconfigurable Xilinx Spartan-6 LX45 FPGA
- 16 bit analog and digital I/O
- Built in USB, CAN, 10/100 Mb/s Ethernet peripherials

The board is configured through an ethernet cable. The programs can be written using Labview, C or C++. In the present project, LabView will be used due to the simplicity of coding the FPGA through a high level language. The board is capable of running in real-time.

## 2.6 Geomagic touch

The Geomagic Touch(GT) is a haptic feedback controller device, that is able to output force to the user, and arbitrarily resist the user's attempts at moving the controller. The Geomagic Touch described in this section can be seen on *Figure 2.9*.




(a) Overview of the Geomagic Touch's first three joints.

(b) Overview of the Geomagic Touch's last three joints

Figure 2.9: Overview of all the Geomagic Touch's joints.

On *Figure 2.9*, it can be seen that the Geomagic Touch has six DOF, where only the first three can be actuated, see *Figure 2.9a*. This means that the device only has the ability to generate translational force feedback with three DOF.

The GT communicates with the computer through User Datagram Protocol (UDP). This communication can be established directly through an ethernet cable or by plugging the ethernet cable into USB converter. The interfacing is shown on *Figure 2.6*. The Geomagic Touch was programmed and the connection was established using a C++ based application programming interface (API). The API provides force rendering, virtual environment and position measurement features. The present project utilizes force rendering capabilities to provide force feedback and measures the GT's position as an input to the system.

## 2.7 Summary

As mentioned before, a fully featured DaVinci robot has four arms with 6-7 actuated DOF in total when the EndoWrist is included. Although the test setup only controls four motors, it can manipulate the surgical tool itself in the same way the da Vinci robot does. The missing features are the ones related to the hand of the robot holding the tool, see Section *2.1: Da Vinci robot*.

The sbRIO board controls the test setup and as such represents the onboard computer on the DaVinci robot. In order to perform higher level functions such as force feedback control, it is necessary to remotely handle data and send high-level commands. This is handled by an external computer system that is connected to the Geomagic Touch device.

The sbRIO board and the Geomagic Touch both communicate with the computer using UDP. The computer handles the force estimation using a dynamical model of the test setup (or EndoWrist, more precisely), this is vital for force feedback. Connecting software components responsible for communicating with hardware and the ones responsible for the control algorithm and estimation is necessary. For this purpose we use the Robot Operating System (ROS), which uses a network architecture to share data between components via data streams. Go to Section *F.3: Robotic operating system* for a short introduction to ROS.

A general overview of the system that will be developed can be seen on *Figure 2.10*.

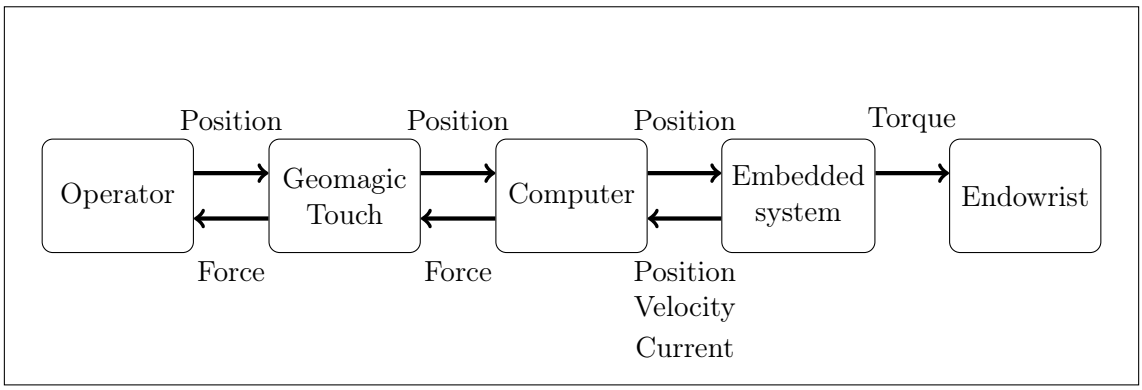



Figure 2.10: Overall system with feedback in both direction

The operator is setting the Geomagic Touch position, which then is send to the main computer. The computer sends the position to the embedded system, which handles the position of the EndoWrist.

The embedded system sends back the position, velocity and current applied to the motor. The computer computes a force from the received data and sends it to the Geomagic Touch, which generate a force in a counter direction of the force applied by the human.

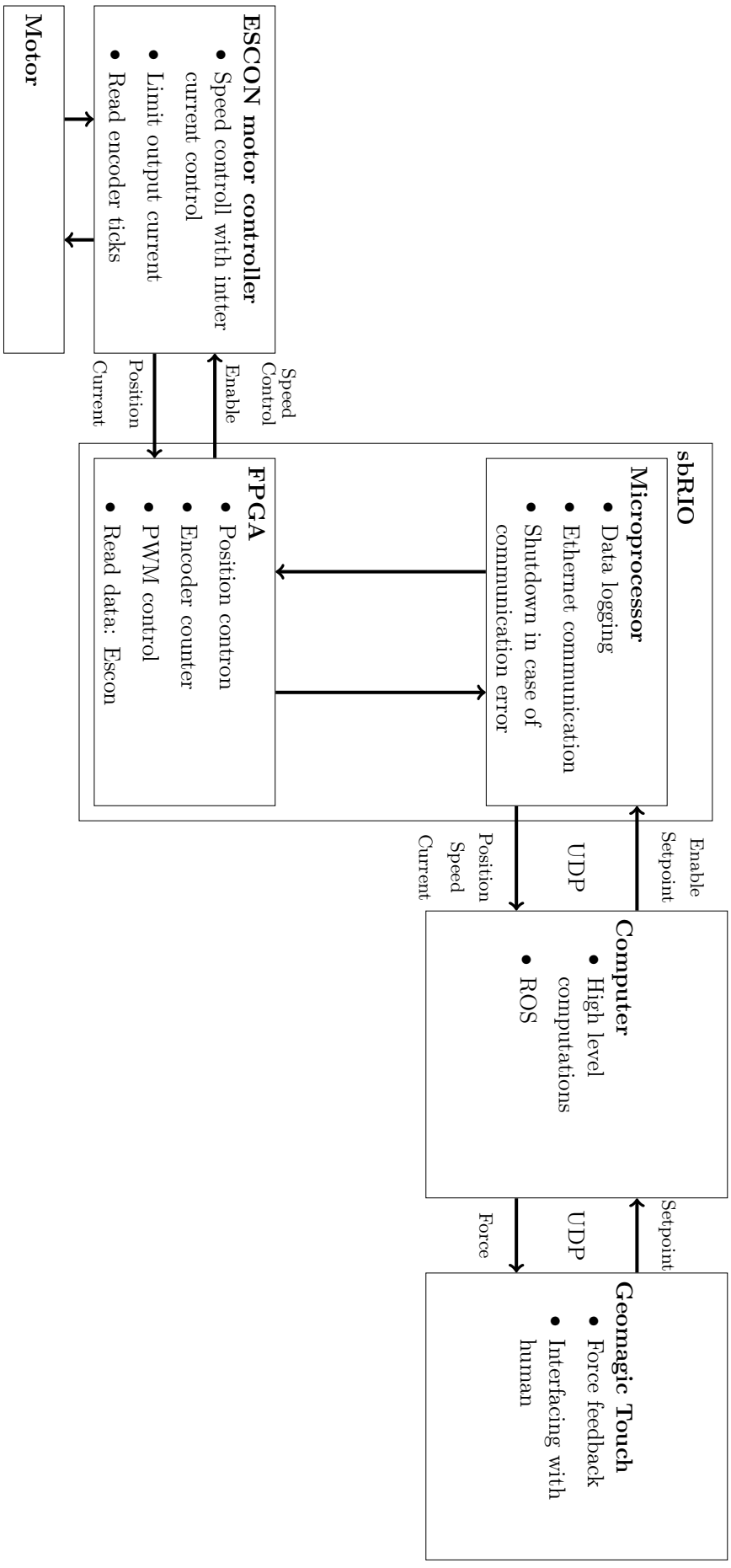



Figure 2.6: Interfacing between components

# Communication 3

As stated in the introduction, the force feedback loop should at least have a frequency of 550 Hz. This loop not only includes the communication between the Geomagic Touch and the computer but also the communication between the sbRIO and the computer and the computation time required for force estimation. In the current setup, the bottleneck is the communicating between the sbRIO and the computer as it can not exceed 100 Hz. As the drivers handling the communication with the Geomagic Touch have a refresh rate of 1000 Hz and the computation time should be small compared to the refresh rate, the frequency of the communication with the embedded system should be at least 1200 Hz. However, from experimentation it was found out that the built-in library for UDP on the sbRIO cannot handle a frequency higher than 1000 Hz. If this maximum frequency is reached, the feedback loop will have a refresh rate of 500 Hz, see *Figure 3.1*, which is still deemed acceptable according to [5]. This chapter focuses on improving this communication to reach 1000 Hz.

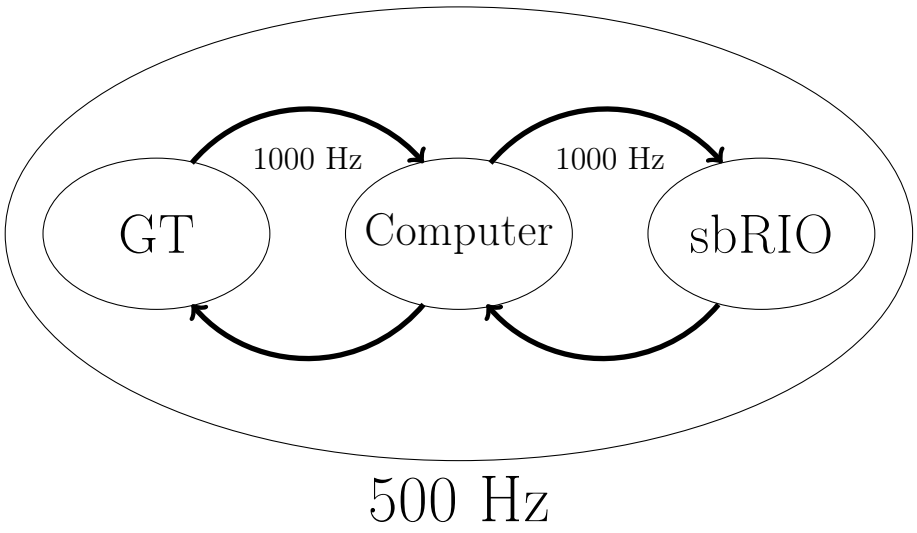



Figure 3.1: Communication loops inside the feedback loop

## 3.1 Previous communication

In the former system setup the sbRIO board and the computer running ROS are communicating through an ethernet cable using TCP. For short introduction to TCP, go to Section *F.1: TCP/IP*.

The speed of the communication between the sbRIO board and the computer running ROS was set to 100 Hz. This was done as a trade off between being fast and for avoiding bugs which where bandwidth related[8].

### 3.1.1 JavaScript Object Notation

In the former system the data exchanged through a TCP connection follow the JavaScript Objet Notation (JSON) format. JSON is a "*text format for the serialization of structured data*"[15] which mean that it defines a syntax and a structure that can be used to exchange data. The structure used by JSON associate a name to a value. The value can be a number, string, array, object, boolean or null. It is also worth mentioning that a JSON file only contains characters, if a decimal number is to be sent, each digit will be represented by a character instead of having the bitcode representation of a double or a float for example. The former message from the sbRIO to ROS that would contain the information regarding the motors for the Endowrist followed the structure on *Listings: 3.1.*

```
"p4_primary":{
        "position":[0.1818916529184269852691 3,0.0001308995560975745320 3,
        -0.0001266769975192886591,-0.1987191723699879841951,
        0.0054489523589498324 79,-0.0755423981233548954 23,0.3497519786326297133317 9],
        "velocity":[0.0245981621957453203,0.0019754796320965562379,
        0.0015707947313785552979,0.0001308995560975745320 3,-0.08079910119712697,
        0.0017923072681865736513 2,0.3684698759874165565320 3],
        "effort":[-0.48593282699584960938,-0.31835031509399414063,
        -0.36891269683837890625,-0.39591121673583984375,0.1524524863245698532566 3
        ,0.32147856985321569887862,0.24789/86568745221455553]
        }
```

Listing 3.1: Example of the previous JSON string.

As it can be seen in this example, the structure associate another structure to the element called "p4_primary" which designates the Endowrist. This second structure contains arrays of positions, velocities and efforts with values for each of the seven motors controlling the EndoWrist. When controlling four motors like it is done in this study, 346 bytes are required to store the file, see *Figure 3.2.* The communication that was implemented is

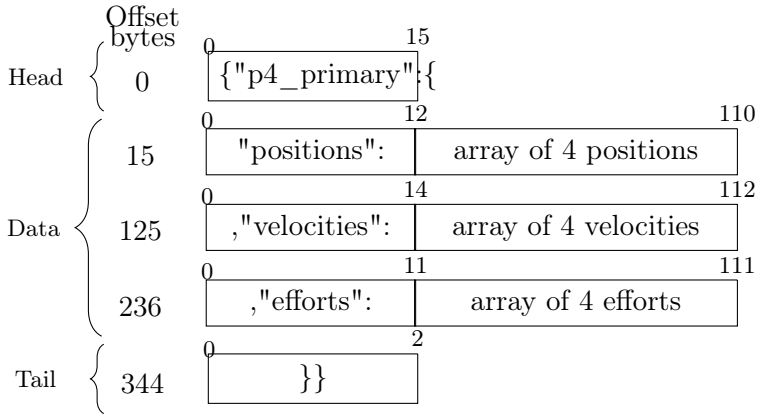



Figure 3.2: Packet using JSON

described in Section *3.1.2: ROS*.

### 3.1.2 ROS

In order to communicate between ROS and the sbRIO board a driver named davinci_driver[16] was used. It is composed of three parts:

- Low level driver
- Middle level driver

- High level driver

The low level driver communicates directly with the sbRIO board. The high level driver handles the communication between ROS nodes. It updates the data for the nodes and transmits the setpoints to the low level driver through the middle driver. The name "setpoint" is used to designate the next position a motor should take.

The middle level driver handles both the creation of the low lever driver(s) and the communication between the low and high level driver so that there is no need for the client, ROS in our system, to acquire mutexes. Since it is possible to create more than one low level driver it is possible to communicate with more than one arm of the DaVinci robot.

The low level driver connects to one sbRIO board using a TCP socket and exchange data with it using JSON files, see Section *3.1.1: JavaScript Object Notation*. In *Figure 3.3* the flow of the driver can be seen. The driver begins with an initialization of the connection and then start a loop. This loop is started in a thread and will run as long as the node is running. At each iteration of the loop, if some new data are present in the stream they are read and an update function is called to make them available to the higher levels. If there is some new setpoints or motor enable they are sent to the sbRIO. Finally, the thread goes to sleep before the next iteration of the loop. This sleeping period is introduced as to not overload the communication channel or the processor and this is the only control available on the speed of the communication.

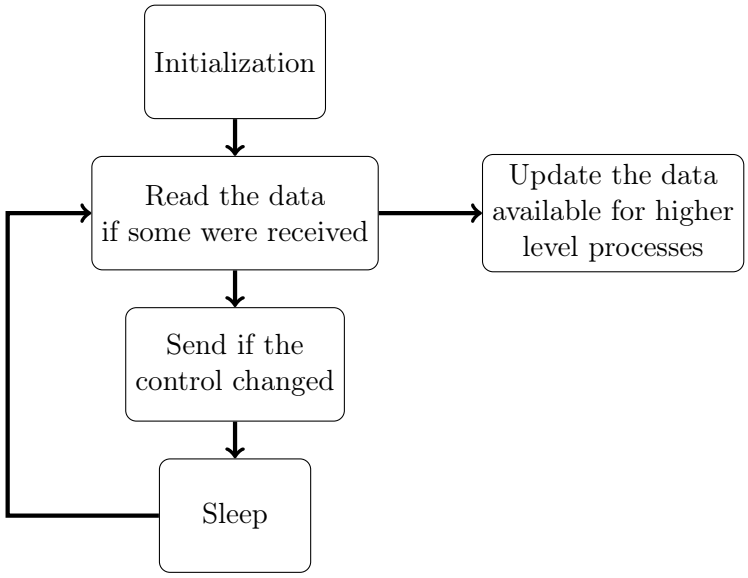



Figure 3.3: Structure of the original sbRIO driver

### 3.1.3 sbRIO

The sbRIO code running in the former setup has four code loops running in parallel. Since the sbRIO uses a single-core processor, the execution order depends on the operating system's decision, and can have stochastic characteristics. The four separated loops have distinct functions:

- TCP connection loop for communicating with the ROS node
- Data out loop for reading data out of the FPGA
- Data in loop for setting FPGA parameters according to the received JSON packet
- Debug/data logging loop

The debug/data logging loop is not used during normal execution. The TCP loop communicates with the data loops through queues. It uses one queue for the received data, one for data out. The queues have only one element. When an enque attempt is made, the previous element is discarded and the newer one takes its place, ensuring that the queue always has the most up-to-date element.

The TCP communication is set up serially, i.e. the sbRIO sends data then waits for incoming data. It is a timed loop, thus the code attempts to run once in each specified period. When the loop is executing for the first time or when it experiences error, it creates/recreates a TCP connection using a TCP listener. After that, the loop reads the data out queue, if it finds an element in it, it sends it through the TCP connection. Then it polls for incoming data. When it arrives, the loop enques the data in the received data queue. Then it restarts the process.

When the user stops the execution, the connection is terminated.

## 3.2 Communication improvements

This section focuses on studying the different ways available to increase the refresh rate of the communication between the embedded system and the computer. Speeding up the communication could be done in different ways, where three are stated below

- Find a faster communication protocol,
- Minimize the amount of data transmitted,
- Implement faster hardware.

Implementing faster hardware is not a possibility in this case as one of the goals for this project is to only use the existing hardware. However implementing a new communication protocol and minimizing the package size is possible. First, we will describe the process leading to the choice of the new transport protocol. Then the minimization of the exchanged data is studied.

### 3.2.1 Protocol

Finding a communication protocol faster than TCP is possible as the safeties in the protocol tend to slow down the overall speed of the communication by retransmitting packets or reducing the sending frequency[17]. Furthermore, some of the safeties like congestion control are irrelevant in our case since we are directly connected to the destination. Flow control is not necessary as the need for flow control implies that a delay is created on the receiving side which would induce delay in the entire system. Reliable transport and ordered delivery are a problem, since retransmitting data is useless in our case as the data can be deprecated. Instead, it is much more interesting to receive new data.

The protocol chosen for this project is User Datagram Protocol (UDP). This is done because its a faster protocol than TCP and the project goal is to reduce the latency in communication between the computer running ROS and the sbRIO board. It is however less reliable than TCP but it is deemed irrelevant to some extent as a new package with data should be send as soon as possible. This also solve the problem with storing old data in the transmitter buffer which could be obsolete. Even though it is deem irrelevant that UDP is a less reliable protocol and package lose can appear, it is still necessary to include safety in the system if an connection error should appear.

### 3.2.2 Minimizing the size of the transmitted messages

There are two elements that impact greatly the size of the exchanged messages: the number of character added by the JSON format and the length of the numbers. As detailed in Section *3.1.1: JavaScript Object Notation*, the JSON format add numerous characters for the structure and require a name for each value. The first modification made in order to minimize the size of the messages was to remove the JSON format. One way of doing this is to define a fixed format for the messages. For simplicity, it was decided to keep a format similar to the original one, composed of a vectors of positions, velocities and efforts. All of these vectors contain the same number of elements and each elements must have the same length to make it possible to read. With the original JSON format it is possible to specify the content of the file using the name of the first element in it. In addition to the messages containing the positions, velocities and efforts, there is messages that can be sent to indicate which motors are active or not. In order to implement that feature in the new format a byte is added at the end of the packet, this byte contains a boolean for each motors indicating if the motor is active or not.

As mentioned, all the numbers in the vectors should have the same length. In the original format, not only the numbers were long (18 digits) but the length would vary from one message to another. Different solutions were considered. The first one would be to fix the number of digits but that would still imply a conversion to string. The second solution considered was the compression of the characters sent.

However the compression rate generally is not higher than 50% for random data and the computation time induced by compression would results in an additional delay instead of decreasing the existing one [18][19]. The third and last option was to send the binary representation directly without any encoding. The numerical values being stored as floats, each of them requires 4 bytes. It have the advantage of having a fixed length without having to round the value. The last solution was chosen as it has the lowest computation time, since no encoding is used. The only issue is that the endianness of both devices is not the same, this is corrected in the implementation on the ROS side. The packets obtained with the new format are shown in *Figure 3.4*.

With the original format, the packet described in Section *3.1.1: JavaScript Object Notation* contained 346 characters. With the new format, the packet that contains the same information is 49 bytes. Therefore, the packets following the new format uses 15% of the original size.

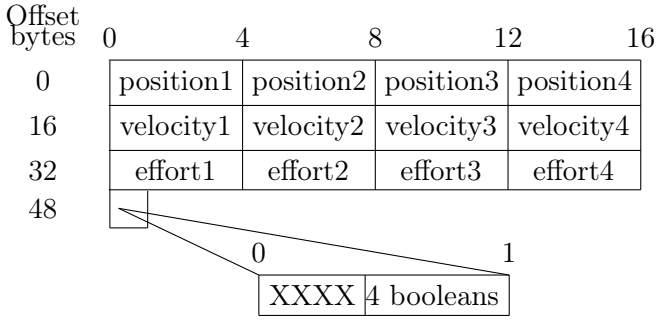



Figure 3.4: Packet , built using the binary representation

## 3.3 Implementation

For the modifications suggested in Section *3.2: Communication improvements* to be effective, they need to be implemented on both the computer running ROS and the sbRIO board. As the two systems are programed in very different ways, i.e. the computer can be programed through traditional means, in our case through C++ and the sbRIO has to be programed through Labview.

### 3.3.1 ROS side

In order to implement in ROS the improvements described to increase the speed of the communication it is necessary to write a new driver. As previously explained, the new driver should use UDP instead of TCP/IP. It should also transmit the actual data without using a JSON Stream so that the size of each packets is reduced. In addition to those objectives it would be better to modify as little as possible the current structure of the ROS node so that it does not have to be fully rewritten. Three goals that need to be fulfilled when writing the new driver were defined. The goals are stated below:

1. Replace TCP by UDP
2. Replace the JSON Stream by the numerical values
3. Modify as little as possible the original driver

The modifications that need to be made in order to carry duty 1. and 2. are related to the communication protocol and the format of the transmitted data which are both handled by the low level driver. In order to fulfill goal 3. only the low level driver should be modified without having consequences on the higher level drivers. This means that the public functions should keep the same prototype and that the format of the data available for them should stay the same (i.e vectors of double). The boost library should be used as it was in the original driver, to handle threads and mutexes.

The socket initialization for the UDP protocol is very close to the one made for TCP protocol, thanks to this the establishment of the communication with the sbRIO is similar in both the old and new drivers. However, the UDP being connectionless it is not possible to use the same method as before to read incoming data, i.e it is not possible to look if some new data arrived, with UDP it is necessary to constantly be listening for new incoming data. In a first time, a new code structure was implemented. This new structure ran two threads, one for receiving data and one for sending. However when the frequency of the communication increased towards 1000Hz the packets were sent by two and received by two instead of sending one and receiving one. In order to cancel this it was decided to use a single thread as in the original driver.

To use a single thread it is necessary to modify the receive function as the default one for UDP is blocking, i.e. it waits until it receives a packet before the code can continue. Thus, a new funtion was defined, that function blocks only for a specified amount of time to received a packet, if nothing is received the loop can continue. With this new function it is possible to use a single thread and thus to keep full control over the order in which the packets are sent and received.

As shown in *Figure 3.5*, the new driver starts with an initialization of the connection and of the data structure and then starts a loop. As in the original driver this loop will run as long as the driver is running. In each iteration, if there is some new setpoints or motor enables, a message is sent to the sbRIO. Then, the new receive function is called, if a message is received the bitcode is converted to the actual values and made available to the higher levels, if no message is received before the deadline, the code continues. Finally, the

thread goes to sleep as the original one did. Once again this sleep controls the frequency of the communication, however, in this new driver the waiting time of the receive function and the Round Time Trip (RTT) delay also impact the frequency.

As explained before, when the JSON Stream was removed it was decided that only send numerical data would be sent. And as goal 3. is kept in mind, it is needed to receive the position, velocity and effort as float and one boolean for each motor and to send a setpoint as a double and a boolean for each motor. The received values must be made available as double for the higher levels by storing them in the vector of doubles. For simplicity it was decided that the packet should be following the structure described in *Figure 3.4*. To reach the desired vectors, the received bitcode stored in an array of char is copied in a float variable that is converted in a double and then the value in the vector is updated using this double. As the endianness of the sbRIO is not the same as the one used on most computer, when the bitcode is copied to the float variable the order of the bytes is reversed (the first byte becomes the last one).

The sent packets follow the same structure as the received packets, however only one vector of numerical values containing the setpoints is required. The bitcode of the numerical values is copied into an array and the array is sent. In regards to the boolean values, they are stored in a byte with 1 bit per value which is possible only because the number of motors for one arm is 7 (i.e inferior to 8 which is the maximum amount of booleans that could be stored in 1 byte).

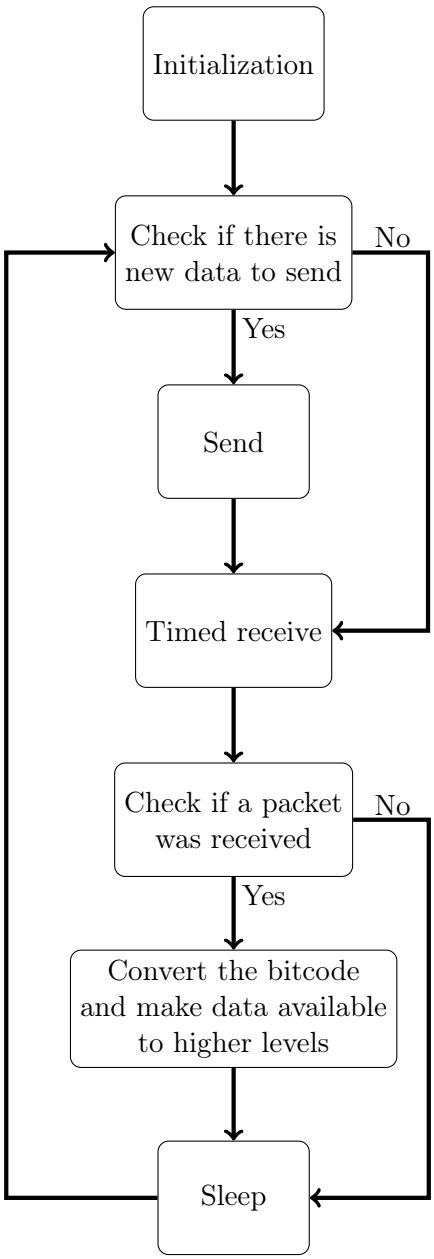



Figure 3.5: Structure of the new low level driver

The driver described is functional, however there is one feature of the TCP/IP that was useful to our system and that we lost by switching to UDP: the connection timeout detection. As safety is essential in this system, it is necessary to implement an additional layer to the communication to make it more reliable. To implement this safety we have two options: using a free library or implement our own safety protocol. The free libraries that can be found include UDT[20] and ENet[21]. Those libraries provide a reliable option for long distances communication by adding a number of features. However on the ROS side, it is only necessary to alarm the user of the connection timeout. For such a simple purpose, the free libraries would add to many features that are superfluous in this system and thus slow down the communication. That is why we chose to implement our own safety protocol.

In order to do so, we decided to add a counter to the current driver. Every time the receive function reaches its deadline, the counter is incremented. Every time a packet is

received, the counter is reseted. When the receive function reaches its deadline, in addition to the incrementation, the value of the counter is tested. If the value is superior to 10 (this value is arbitrary), en error is thrown and displayed to the user to notify of the connection timeout. The new code structure is shown in *Figure 3.6*

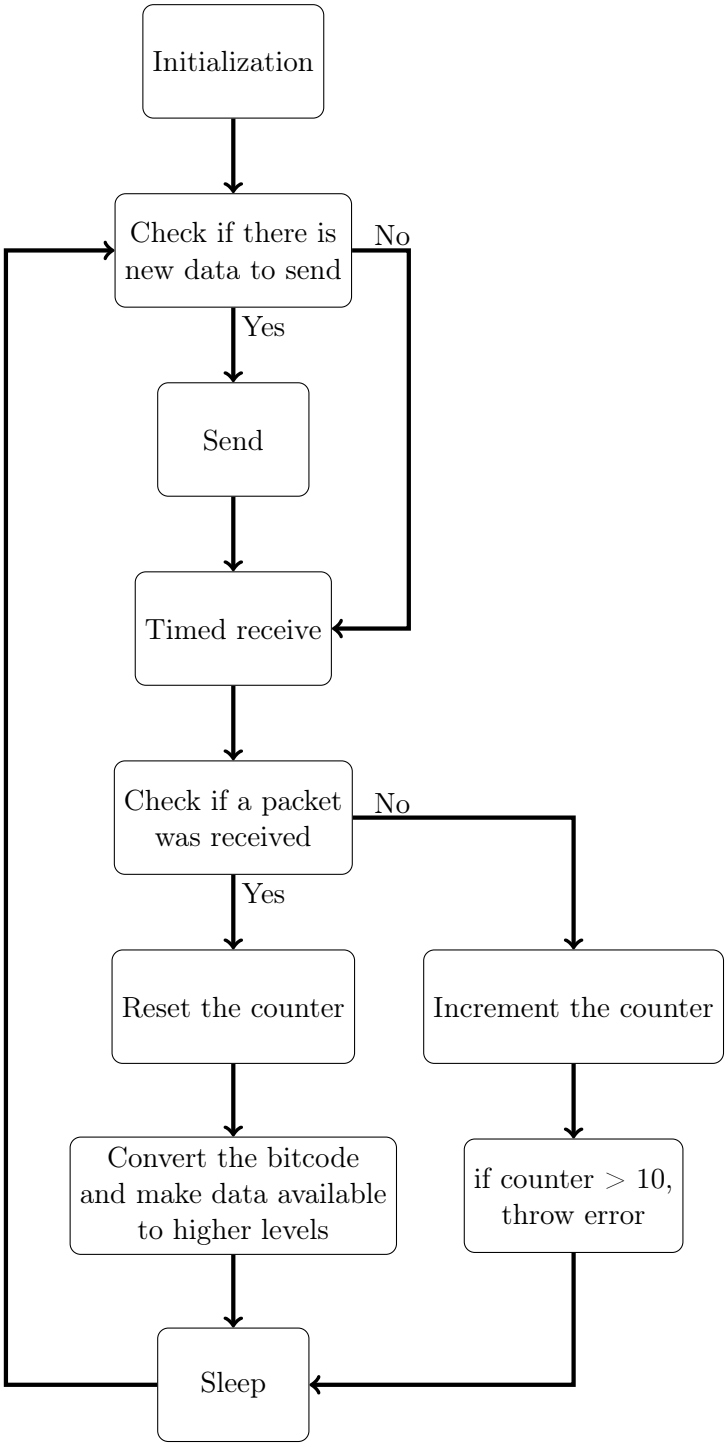



Figure 3.6: Structure of the new low level driver with connection timeout detection

### 3.3.2 Embedded sbRIO Microprocessor

The processor on the sbRIO board is running at a 400 MHz clockrate, while our target communication frequency is 1 kHz. This means that the code running on the processor

has to be optimized in order to achieve the set goals. The fundamental UDP functions of the sbRIO are the following:

- Read data from the FPGA, converting position values based on the gearing
- Encode the data into a string
- Send the string through a UDP port
- Fix communication errors
- Poll incoming UDP packets, send the decoded values to the FPGA
- Emergency shutdown

The code includes tools for debugging and measurement, these have a substantial impact on the communication performance. They are disabled during normal operation. LabView's array handling is far from optimal, further speed increase could be achieved by compiling a more optimized C code into a DLL file and calling it in the LabView code. The extra functionalities are the following:

- Timestamp sender through UDP
- Time logging for packet departure and arrival to csv file
- Front Panel PID gain adjustment, motor enabler
- Manual, sinusoidal, squarewave signal generator for the motor
- Motor data logging to csv file

We need to decide how to handle connection loss. Either we turn of the motors or we stay in position.

### 3.3.3 Software architecture

In a high level overview, see *Figure 3.7*, our system could structurally be represented in several components. The sBRio board controls the actuators on the DaVinci robot arm and the rest of the system communicates with it via drivers contained in the Davinci_ros package. The Phantom Omni communicates with the system via the Open Haptics API, and connects to ROS through the phantom_omni package.

Connecting all the system components requires an architecture based on a central node which handles communication between the two packages and has access to all the other methods required for control. We have developed the Endohap package exactly for this purpose. The endohap_node node contained in the package connects all the main system components using protocols either defined by ROS (TCP/IP) or externally (UDP).

Communication with the Endowrist test setup is handled by the boost libraries in the davinci_drivers package. The sbrio_driver program transforms the data received by UDP and propagates it through the system using internal ROS communication protocols and vice versa. More precisely, the motor state data is poblished to the joint_states ROS topic, which the central (endohap) node is subscribed to. Since this makes it is possible to use a separate (non-ROS) protocol for communicating with the test setup while maintaining ROS protocols for the internal system, we conclude that this approach gives a higher degree of control in terms of communication speed.

The motor state data is then used by the endohap node for Endowrist external force estimation and feedback generation. After this is done, the resulting force feedback reference needs to be sent to the phantom_omni package, from which the endohap node

also recieves the user input for the Endowrist. While the phantom_ omni package handles hardware communication via external TCP/IP protocols, the endohap node only publishes and subscribes to appropriate topics in order to communicate with that part of the system.

All the calculation required for controlling the endowrist and giving force feedback to the phantom omni is also done within the endohap node. Forward kinematics are handled by the tf ROS package, which utilizes URDF description files combined with current joint position information published to ROS. This consequently allows for other useful features such as visualization in Rviz or various kinematics calculations. Forming the node in an object-oriented manner gives the added advantage of modularity, allowing for implementation of additional features as they become necessary.

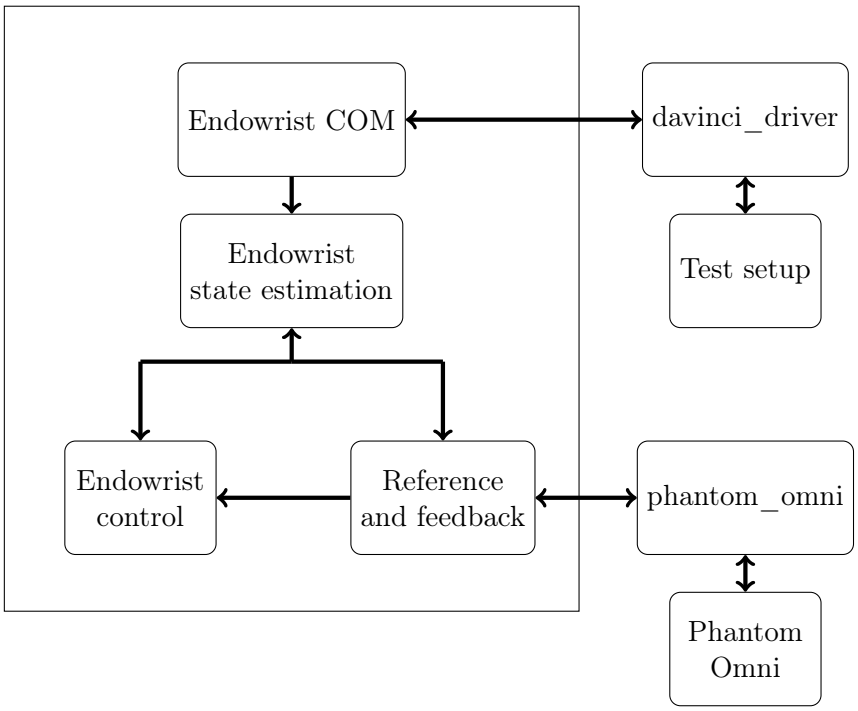



Figure 3.7: Overview of the software architecture

## 3.4 Measurements

In order to evaluate the performance of the new communication protocol, measurements of jitter and packet loss were made for different frequencies. To quantify the improvements an attempt was made to measure the performances of the previous communication. However, as the communication does not have the expected behaviour when increasing the refresh rate, the results could not be used, details are available in Section *E.2.1: TCP measurements*. The only comparison between the two protocols that can be made with available data is between the maximum refresh rate that can be reached.

The data resulting from the measurements of the designed communication protocol are gathered in *Table: 3.1.*

| Frequency (Hz) | delay (ms) | Jitter ($\mu$s) | Packet loss (%) |
|---|---|---|---|
| 99 | 10.1 | 4.66E-2 | 0 |
| 474 | 2.1 | 5.51E-2 | 0.2 |
| 638 | 1.6 | 1.16E-2 | 1.2 |

Table 3.1: Measurements of the performances for the new communication protocol

From a frequency of 99 Hz to 474 Hz, it can be seen that the jitter, as well as the packet loss increase. As both of them are mainly consequences of congestion on the network[22], this phenomenon was expected. However, when the frequency is increased to 638 Hz, the jitter decreases and the packet loss keeps increasing. The cause is that the jitter increases but the driver only waits 1 ms to receive a packet from the sbRIO after sending and thus rejects the packets coming after that deadline. The packet loss could be reduced by increasing the waiting time but that would reduce the frequency. A trade-off between frequency and error rate has to be made. The data for the maximum frequency of the driver could not be collected as the congestion on the network becomes too significant. When that happens, the communication does not behave as intended and each device sends more than one packet in a row.

The communication cannot reach frequencies higher than 638 Hz due to congestion, thus not meeting the requirements. However by implementing the new communication protocol the refresh was increased from 100 Hz to 638 Hz. Solutions to further improve the refresh rate are discussed in Chapter *6: Results and Discussion*.

# Force Estimation 4

In order to have a representation of the reaction force on the EndoWrist, estimation is needed. In Section *1: Introduction*, it is stated why force measurement can not be directly measured due to the lack of sensors. Thus the force has to be estimated by mathematical models as functions of actuator measurements.

The main challenge faced in making a model lies in the fact that the pulley system on the EndoWrist is nonlinear, and thus its full dynamics cannot be modeled in a straightforward manner. In other words, to have an accurate representation of Cartesian force a nonlinear model is required.

One method of tackling this problem is to create multiple mathematical models pertaining to forces output by actions performed with the EndoWrist. In this manner, the feedback vector is transformed from Cartesian space to a task space in which the chosen action forms a basis.
Each element of the new feedback vector corresponds to an actuated axis of the Geomagic Touch.

For the purpose of this project, we choose to feedback the yaw force generated by the grip action of the clamp, the torque generated by the roll actuator and force exerted by the clamps pitch movement.

## 4.1 System Identification

System identification is performed by fitting input and output data of the system to a known (grey-box) or unknown (black-box) mathematical model, generated either using theoretcal knowledge or data-fitting algorithms. This is done by changing the parameters of the model chosen, in such a way that the models error is minimized.

For this project, it is required that the model outputs the force exerted by the EndoWrist. Ideally a mathematical model derived from classical mechanics would be used to describe the dynamics involved in EndoWrist movement. However, deriving this model precisely enough for grey-box identification has been proven difficult and time consuming due to the nonlinear nature of the dynamics[23].

The nonlinearity of the EndoWrist dynamics emerges from friction forces between the pulley strings controlling the end-effector. This causes multiple pulleys to move in cases where only one actuator is actuated. Additionally, the strings themselves are elastic, which adds another dimension of nonlinearity.

For these reasons it was decided to use a black box identification algorithms which only provide a general model structure where parameters do not have any physical meaning. Parameters are then identified using different variations of these algorithms and checked for accuracy.

The system identification process used in this project can be described in three steps:

1. Picking a model structure.
2. Identifying parameters.
3. Model validation.

### 4.1.1 Model structure

The first choice regarding model structure pertains to linearity. Since the system is nonlinear, a straightforward approach would involve choosing a nonlinear model structure for identification. On the other hand, stability analysis of nonlinear models is difficult. Since this project only aims for a trend representation of the force applied to the EndoWrist, it was decided to identify a linear state-space model which is used as a part of Hammerstein-Weiner nonlinear model [24].

#### State-space model

A state-space model separates the dynamics of a system into a set of first-order differential equations. Additionally, if the dynamical system is linear, time-invariant, and finite-dimensional, then the differential and algebraic equations may be written in matrix form. The general structure of such models is presented in *Equation: (4.1)* and *Equation: (4.2)* for the continuous and discrete variants, respectively.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{4.1}$$

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k+1) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \end{aligned} \tag{4.2}$$

This representation of a system is especially useful in linear MIMO systems as it maintains the same form for any number of inputs and outputs, as opposed to transfer functions.

#### Hammerstein-Wiener model

An identified state-space model can be used as the linear part of an Hammerstein-Weiner model, see *Figure 4.1*. Hammerstein-Wiener models are useful when the output of the system depends nonlinearly on its inputs, in this case it is possible to decompose the system into linear and nonlinear parts. The nonlinearity estimators at the inputs and outputs of the system can be modeled by a variety of nonlinear systems, e.g deadzones, saturations or neural networks.

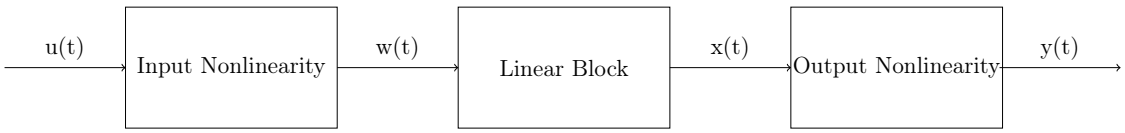



Figure 4.1: Block diagram of a Hammerstein-Wiener model.

A deadzone nonlinearity is used for this project to represent the friction effect on the force. Different output nonlinearities are also utilized and compared. Different output nonlinearities are also utilized and compared.

**Inputs and outputs**

Multiple approaches have been considered when taking into account inputs and outputs of the system.

The most direct approach would have all the available data as inputs to a single output system, which outputs a force estimate. Ideally, this would provide enough information to identify an accurate force model.
However, the imperfect nature of measuring equipment and the experiments performed, as well as limits in computing power and optimization algorithms prevent a fully utilization of this information.

A more robust approach could be a system that estimates some measurable values in addition to force.If the system is observable and the model captures the dynamics sufficiently, it could be possible to correct the model states using output errors.

Both of these approaches are examined and simulated in this chapter.

The same modeling process was used for both yaw and pitch force models, while the roll torque model is explained in section 4.3. For clarity, the results of each step in the process will only be presented for the yaw model, while the final results will be presented for both the yaw and pitch models.

## 4.2 Parameter identification

For parameter identification to be performed on the Hammerstein-Weiner model, a linear model needs to be defined. This means that it is first necessary to identify the linear state-space model of the system. Linear state-space model fitting can be done using various methods , but for this purpose it has been chosen to to use a subspace identification algorithm.

Subspace identification is an algorithm that combines concepts from system theory, linear algebra and statistics in order to provide a state-space model of the system, which makes it useful for MIMO system identification [25]. The most important achievement in subspace identification is that Kalman filter states can be obtained from input-output data using linear algebra methods. Although the algorithm itself is complex, it is implemented in the System Identification Toolbox as a part of MATLAB.

The order of the identified model was picked using Hankel singular values [26], which determine the amount of data dynamics a model can describe at a given order. As seen on *Figure 4.2*, the diminishing returns of increasing the state-space order are visible for different subspace identification algorithms.

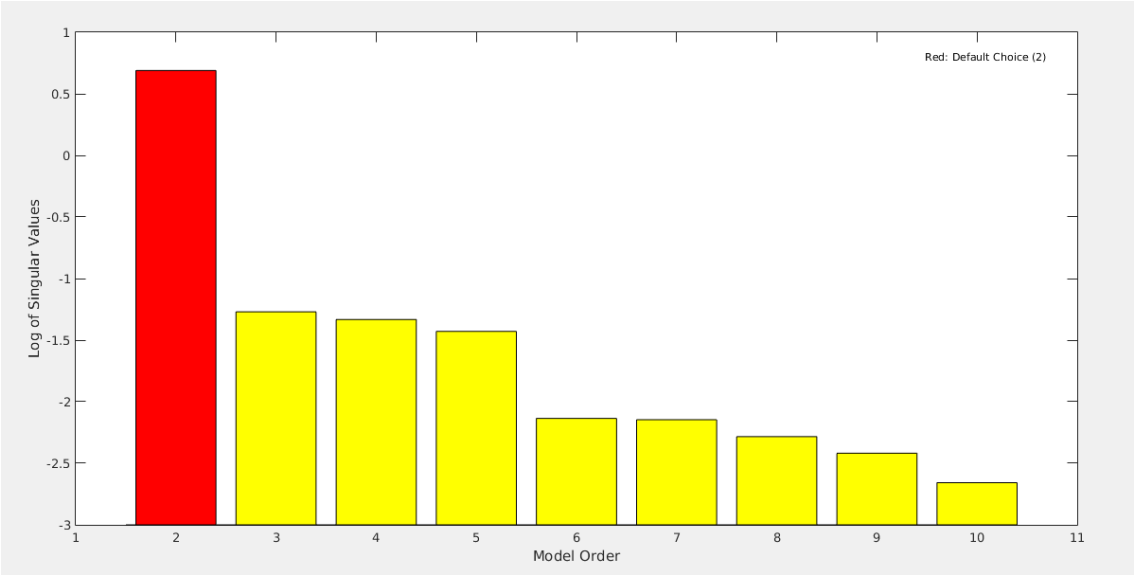



Figure 4.2: Comparison of Hankel sigular values for different system orders.

### 4.2.1 Identification data

Both approaches discussed in Section *4.2: Parameter identification* are possible to realize on the data sets acquired in Chapter *E: Measurement*. However, the data contains various irregularities caused by the imperfect measuring process as well as the hardware. For this reason, parts of the data needed to be removed before identification, which required interpretation of results.

Noise is also present and can be removed from the identification data using a low pass filter. Since most of the dynamics are captured in the frequency band of 0 to 20 Hz, this frequency range was chosen for identification. Offsets were also removed from the output data as they reduced model quality. A sample of original and modified data can be seen on *Figure 4.3*.

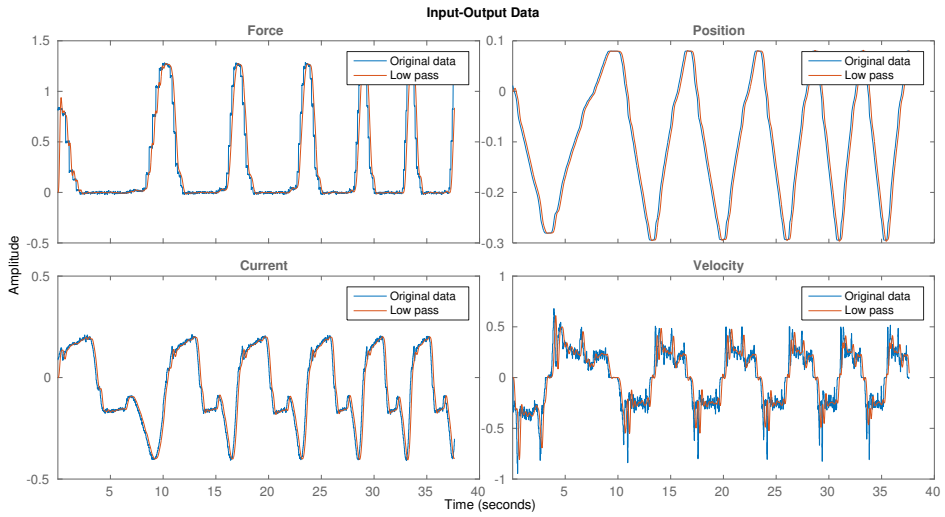



Figure 4.3: Comparison of original and low-pass filtered data.

Linear models can also be fitted in the frequency domain so the data is transformed using

fast Fourier transform [27] and used in the identification process.

### 4.2.2 Linear model identification

The System Identification Toolbox provides 3 different subspace identification algorithms which often provide different results depending on the chosen system order [28]. For this reason all 3 algorithms are attempted in the identification process, and the best one is chosen depending on the average fit. The algorithm which provides the best result varies heavily on the data set and the chosen order of the model, see *Figure 4.4*. It can be see that second order models have slightly different dynamics depending on the algorithms, while in certain cases some models lose stability.

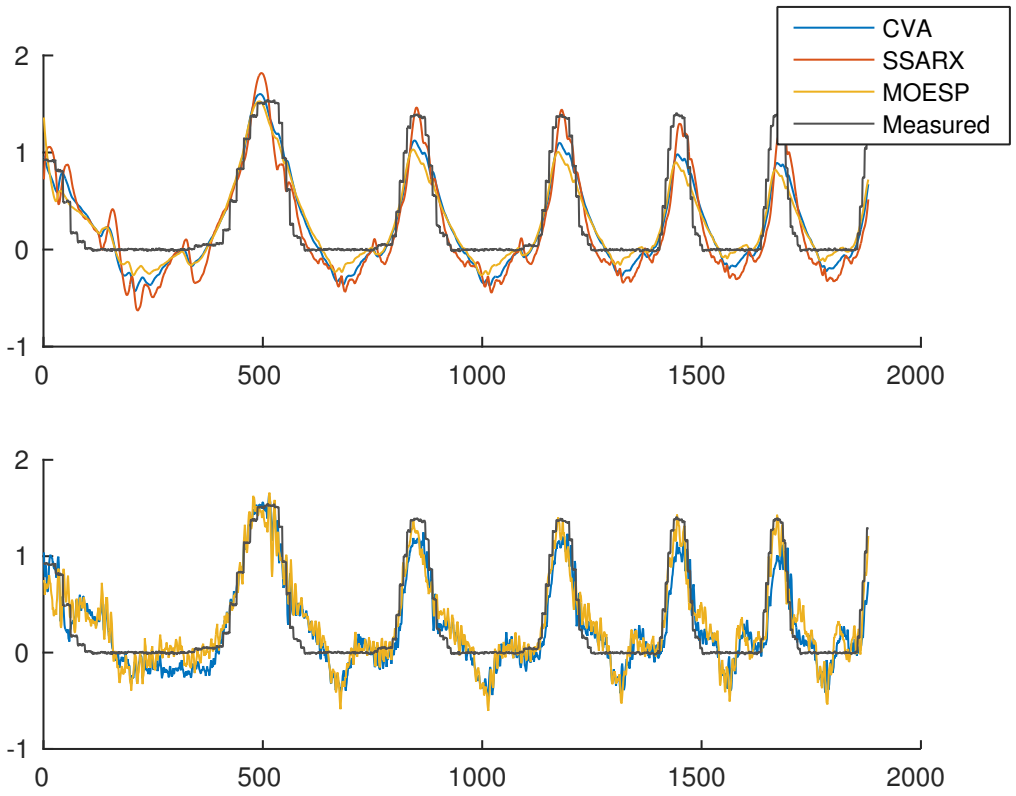



Figure 4.4: Second and sixth order yaw force models identified by different subspace identification algorithms. The lower plot shows only two methods as the SSARX method is not stable for this order and data set.

The choice of inputs, outputs and the domain of identification data heavily affects the model quality in a unpredictable way, see *Figure 4.5*. It is assumed that the cause of this is a limited number of iterations, since a higher order model should always represent dynamics better than a lower order one.

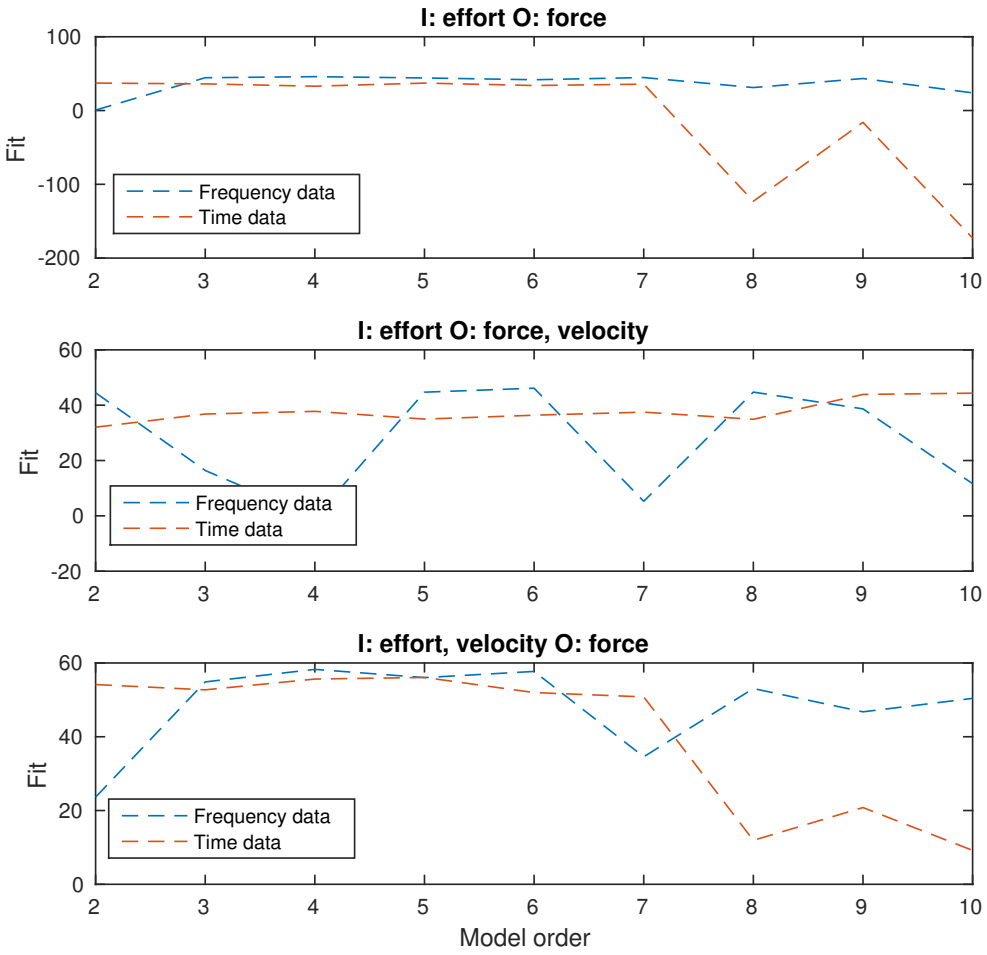



Figure 4.5: Best average fit of models to validation data depending on the order and domain of data used in identification.

This fact makes an approach consisting of testing various inputs and outputs somewhat necessary in picking the best model for estimation. Comparing the responses of these models it is clear that having effort and velocity as inputs provides a much better estimate than when the only input is effort. As seen in *Figure 4.5* his is true for both the time and frequency data sets.

### 4.2.3 Hammerstein-Weiner model identification

A linear model is identified and the estimate of the deadzone nonlinearities with the Hammerstein-Wiener model can be made.

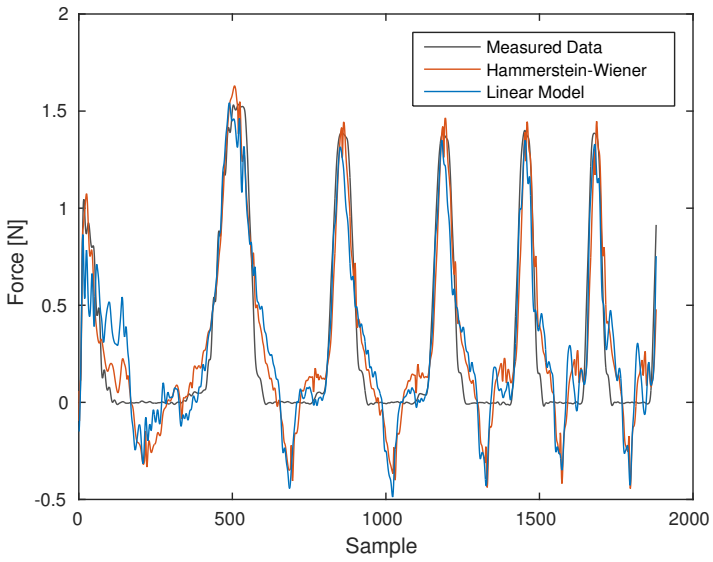



Figure 4.6: Hammerstein-Wiener model with deadzone nonlinearity responses compared to measured data for yaw force.

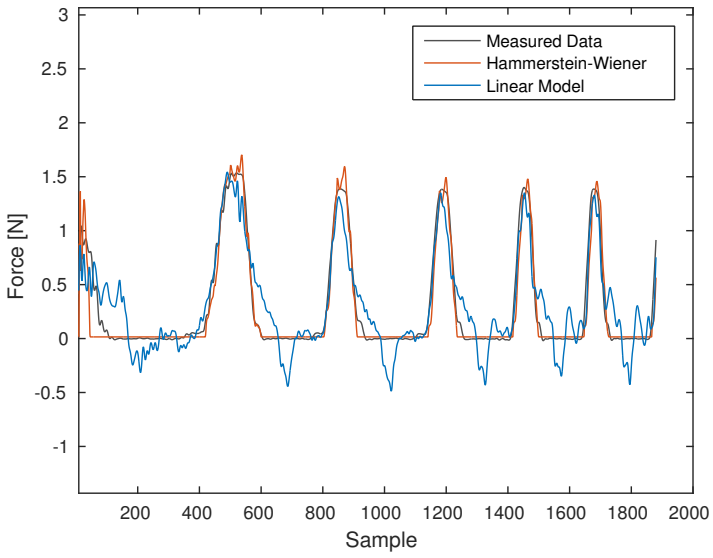



Figure 4.7: Hammerstein-Wiener model with deadzone input nonlinearity and saturation output nonlinearity responses compared to measured data for yaw force.

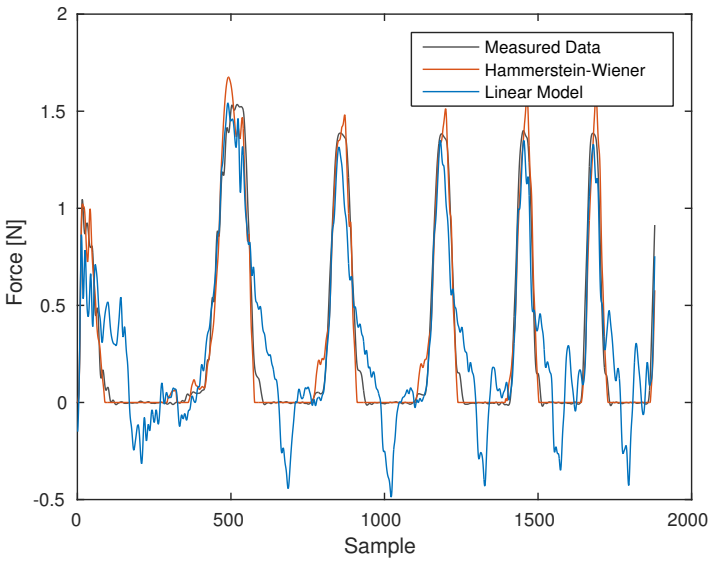



Figure 4.8: Hammerstein-Wiener model with deadzone nonlinearities on the output.

From *Figure 4.6*, *Figure 4.7* and *Figure 4.8* it should be noticed that the deadzone nonlinearity on the inputs significantly improves the linear model and as such makes for a good yaw force model. This serves as a proof of concept for our own friction estimating nonlinearity based on measured data described in the control chapter.

## 4.3 Model validation

In this section, the model derived in Section *4.1.1: Hammerstein-Wiener model* is tested. This test is done on new data, which the model has not been fitted to.

The model is expected to capture the general force dynamics in a way proportional to the actual force. If this is to be true, the represented force estimate can be send back to the Geomagic touch as force feedback.

It should be noted that the model is fitted to dynamics of a clamp gripping a load-cell, which has a hard surface, thus having a steep force gradient. It is expected that soft tissue would have a much smaller force gradient, thus it is expected that the model response overestimates the force slightly. Nonetheless, through the use of the system we came to a conclusion that the response is satisfactory for the degree of accuracy provided by the Geomagic Touch force feedback.

In *Figure 4.9* and *Figure 4.10* the response to different types of input data is examined. The data was gathered in measurements described in chapter E, but wasn't used in the fitting process.

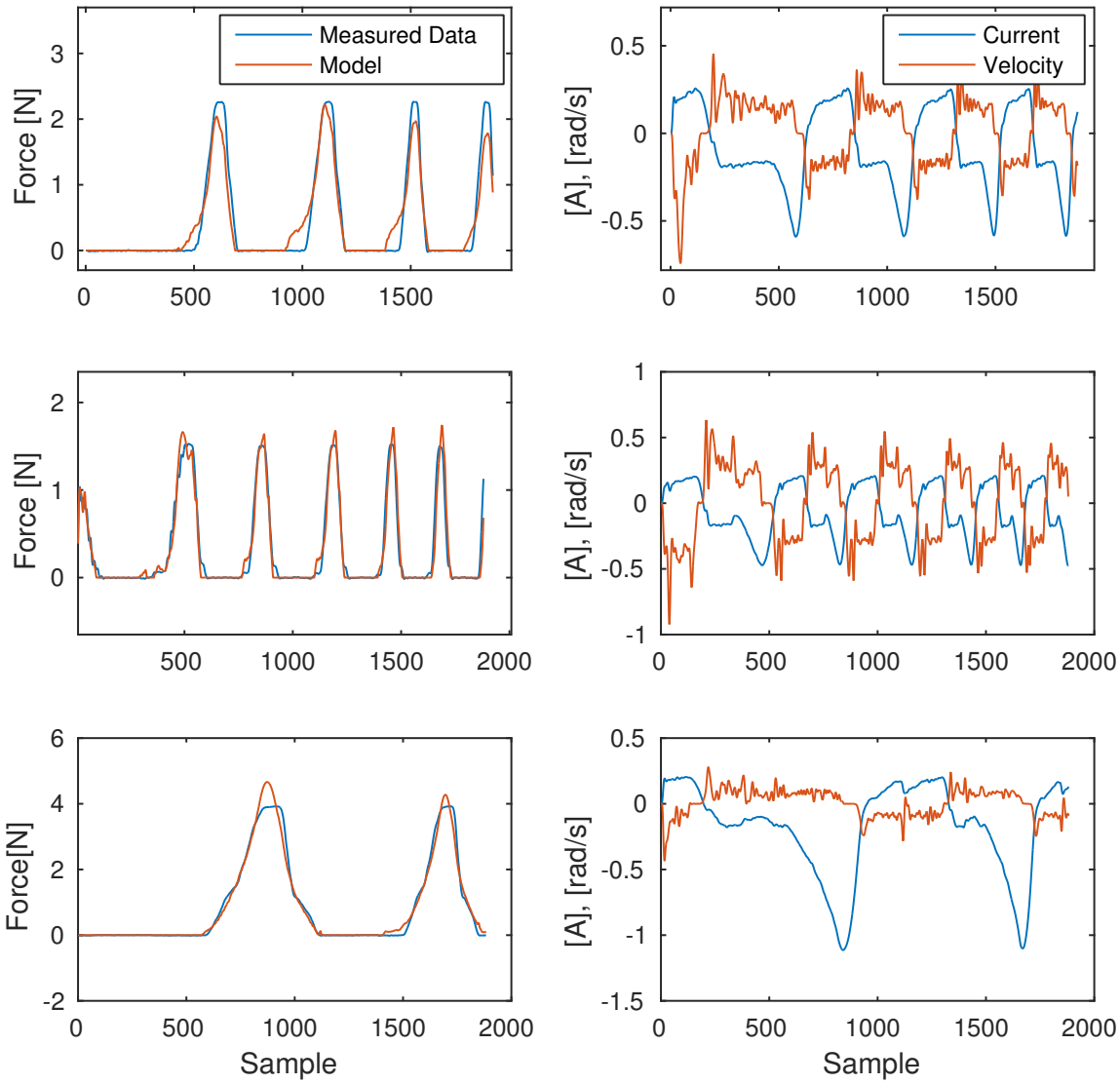



Figure 4.9: Yaw model excited with different types of inputs.

Unlike the yaw model, the nonlinear pitch model had large jumps in value and was not useful for feedback. Thus, the linear model is used in estimation until a better model is developed.

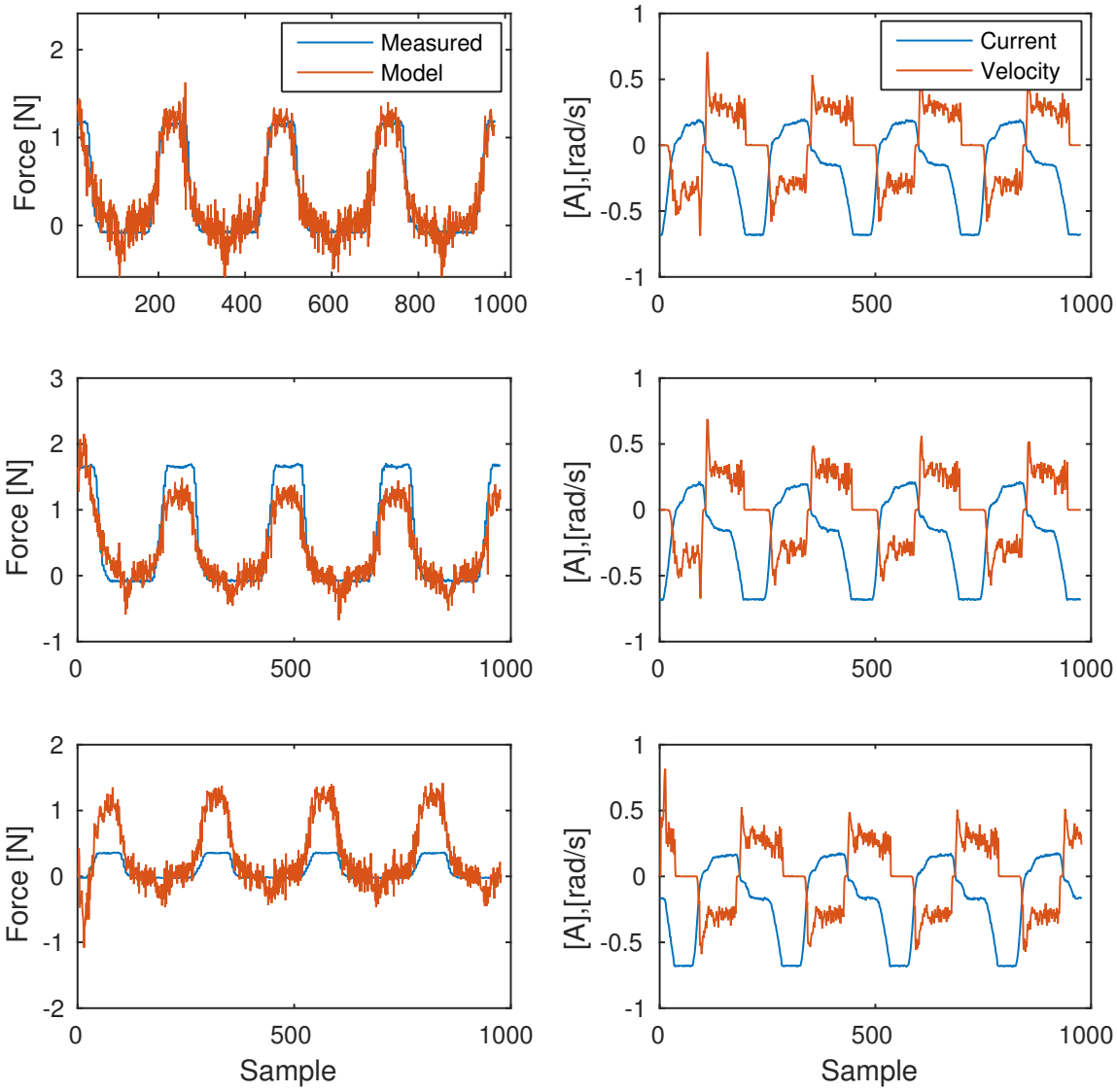



Figure 4.10: Pitch model excited with different types of inputs.

The roll torque, which is determined by the roll actuator and directly rotates the entire tool, making it independent to the rest of the system. We can model the roll torque as a state that only depends on input in the state-space model of the system 4.3. The final model is derived by simple measurement, as described Chapter *E: Measurement*.

A useful property of viewing the linear parts of models in the state-space is simple parallel composition. This means that the entire dynamic model of the system can be viewed as one state-space matrix (4.3).

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{pitch} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{yaw} \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{B}_{roll} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{pitch} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{yaw} \end{bmatrix} \mathbf{u}(k) \tag{4.3}$$

$$\mathbf{y}(k+1) = \begin{bmatrix} \mathbf{C}_{roll} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{pitch} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_{yaw} \end{bmatrix} \mathbf{x}(k) \tag{4.4}$$

This is a useful way to view the system since the force feedback loop can be veiwed as a single system with state feedback and input/output nonlinearities.

## 4.4 State estimator design

As mentioned in section Section *4.1.1: Model structure*, an approach was taken towards designing a state estimator which could correct the force estimates. This method would involve designing a steady-state Kalman filter [29] and/or a Luenberger observer [30] for a model that outputs both force and velocity.

The difference between the Kalman filter and the Luenberger observer is based on the fact that the Kalman filter takes into account the stochastic nature of measurements and processes. The Luenberger observer is based on deterministic systems and does not "weigh" data according to reliability.

Before implementing an observer/estimator, it is necessary to check the systems observability. Formally, a system is said to be observable if, for any possible sequence of state and control vectors, the current state can be determined in finite time. The observability is found by ascertaining that the rank of the observability matrix *Equation: (4.5)* is equal to the rank of $\mathbf{A}$.

$$\mathcal{O}_v = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{v-1} \end{bmatrix} \tag{4.5}$$

Checking the matrices of the linear parts of our models, we find that this is indeed true and an observer gain exists.

### 4.4.1 Kalman filter

The Kalman filter is an estimation algorithm that uses a series of measurements observed over time to produce estimates of unknown variables in a system. Estimation is carried out in two steps:

- Prediction step.
- Update step.

The prediction step consists of calculating *a priori* estimates of the current state and covariance using the updated state estimates from the last step. The $\mathbf{P}$ matrix is the so

called covariance matrix and represents the estimated accuracy of the state estimate. The **Q** and **R** matrices represent the covariances of the measurement noise, respectively.

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{4.6}$$
$$\mathbf{P}_{k|k-1} = \mathbf{A}_k \mathbf{P}_{k-1|k-1} \mathbf{A}_k^{\mathrm{T}} + \mathbf{Q}_k \tag{4.7}$$

In the update step, the Kalman gain is calculated, which is used to multiply the output differences in order to update the state and **P** estimates.

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \tag{4.8}$$
$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{4.9}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{4.10}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{4.11}$$
$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{4.12}$$

If the noise characteristic does not change with time, the Kalman gain reaches a steady state. This Kalman gain can then be used without the need for recalculation, this gives the steady state Klaman filter.

### 4.4.2 State estimation

For a model with a sufficiently accurate description of the system dynamics, a steady-state Kalman filter should provide a powerful state estimation capability. Moreover, the stochastic nature of measurements can be compensated for, thus providing an even better estimate.

In order for the Kalman filter to work on the system it needs to use the available measurments. The **R** matrix requires the estimates of covariances in the output measurements, which are easily calculated from the experiment data. On the other hand, the process noise **Q** cannot be determined, as it's value is mostly used to change the degree of trust we have in the model. Higher values in the **Q** matrix indicate that the Kalman gain should be calculated in a way that the system outputs follow the measurements to a greater degree than the model.

A model has been generated that estimates force and the position error as a function of effort and velocity. To validate the theory, it was assumed that force measurements were avilable for our system. This situation was simulated and compared with the results of the open-loop model (*Figure 4.11*).

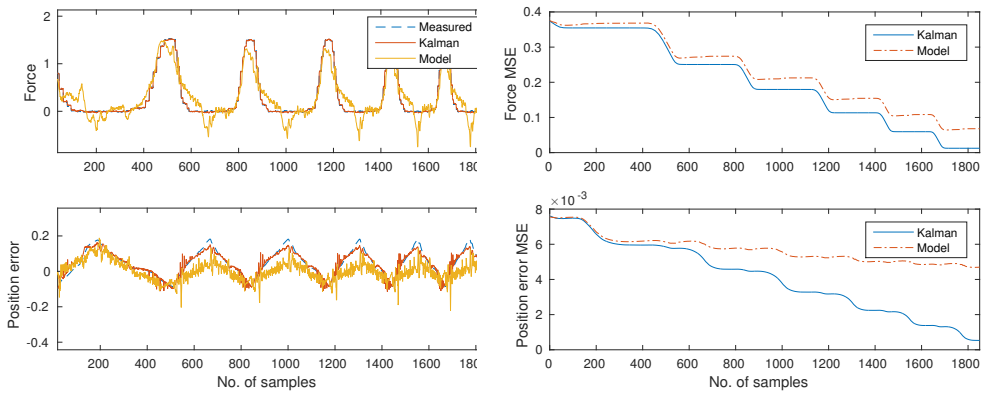



Figure 4.11: Left: Response of the model and the filter estimate compared to measurments. Right: Comparison of mean-square errors.

As seen on *Figure 4.11*, the Kalman filter surpasses the open-loop model when all the measurements are available. Both the force and position error estimates havea really low error which can be reduced further by recalculating the gain for higher values in the **Q** matrix. With this proof of concept, the same simulation is attempted for only the position error measurement.

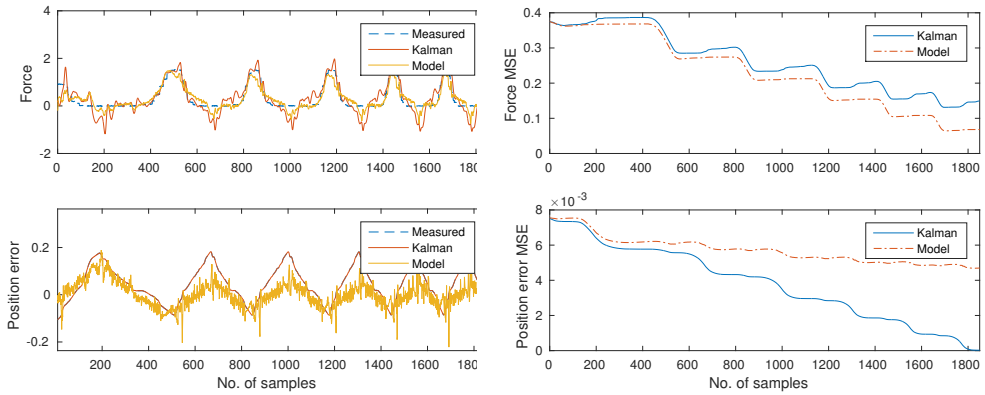



Figure 4.12: Left: Response of the model and the filter estimate compared to measurments. Right: Comparison of mean-square errors.

As seen on *Figure 4.12*, when the only measurement is the position error, system state estimation isn't nearly as accurate. The states are estimated in a way that makes the position error output converge twoards the measurments, but it doesn't reduce (or significantly increase) the force estimate error. It is concluded that this is mostly to the lack of accuracy in the model itself, and as such a Kalman filter wouldn't bring any advantages to the system. Nontheless, the Kalman filter is included in the software implementation, so that higher accuracy models can be tested.

# Models and control 5

In this chapter the derivation of the models used for this project is described. Furthermore the controller implemented will be described after the models have been introduced. The overview of the control system is shown on *Figure 5.2* and *Figure 5.1*.

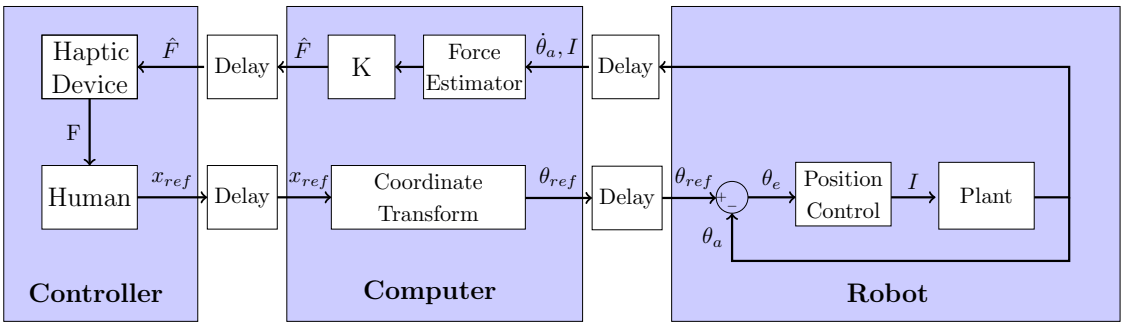



Figure 5.1: Simplified block diagram of the control design.

The implemented control loop is made up of two closed loops updating the references of each other. The two main loops are the position control of the robot and the force control of the haptic device. The haptic device's manufacturer did not publish the control architecture of the GT, thus it is treated as a black box which tracks the force references sent to it. The surgeon controlling the GT sets the references for the position loop. The position controller tracks the setpoints and propagate the measured currents to the force estimator. The estimated force is then sent back to the GT as a force reference.

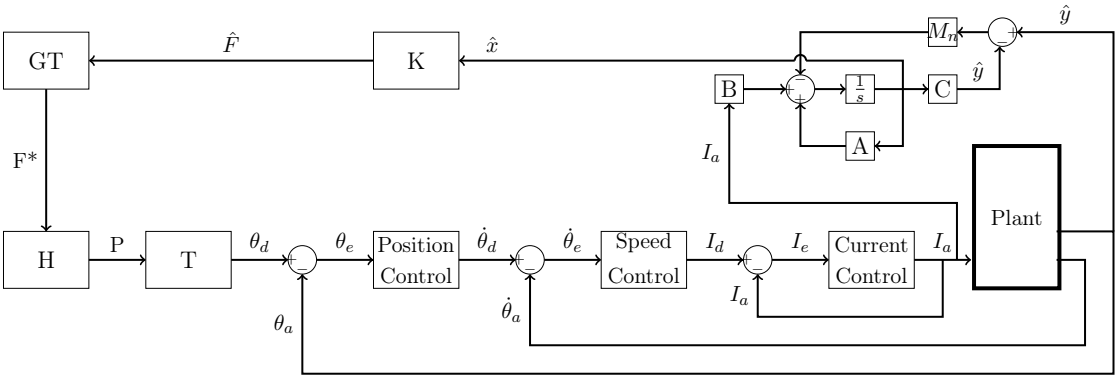



Figure 5.2: Overview of the control system

## 5.1 Human model

In this section a model of a human arm and hand are derived. This model is required to make a simulation of the entire system studied in this project. Most model built for

teleoperation exclude a part of the arm-hand system[31] or limit the number of DOF[32]. For surgical application, the surgeon makes delicate movements relying on the fingers, thus it was decided to use a model including both the arm and the hand.

The human model can be describe from a mass-spring-damper model[33], see *Figure 5.3*. This model has the base position in the armpit of the operator. The constants $K_2$, $b_2$ and $K_1$, $b_1$ are the damper and spring constant for the arm and hand respectively. The mass, M, is the arm of the operator and the force, F, is the force between the hand and the manipulator, in this case the Geomagic touch.

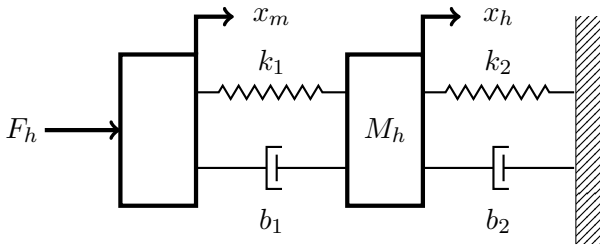



Figure 5.3: Simple human arm/hand dynamical model.

From *Figure 5.3* the dynamic equations can be derived as *Equation: (5.1)* and *Equation: (5.2)*.

$$F_h = k_1(x_m - x_h) + b_1(x_m - x_h)s \tag{5.1}$$

$$m_h x_h s^2 = k_1(x_m - x_h) + b_1(x_m - x_h)s - k_2 x_h - b_2 x_h s \tag{5.2}$$

By substituting the equations into each other the transfer function for the dynamic model can be found. In this case the transfer function is made for the force between the hand and the Geomagic touch and its position, see *Equation: (5.3)*

$$\frac{x_m}{F_h} = \frac{m_h s^2 + (b_2 + b_1)s + (k_2 + k_1)}{m_h b_1 s^3 + (m_h k_1 + b_1 b_2)s^2 + (k_1 b_2 + b_1 k_2)s + k_2 k_1} \tag{5.3}$$

The parameters used for simulation are obtained from [33] and displayed in *Table: 5.1*.

| Variable | Value | Unit |
|---|---|---|
| $b_1$ | 4.5 | N s/m |
| $b_2$ | 7.9 | N s/m |
| $k_1$ | 48.8 | N/m |
| $k_2$ | 375 | N/m |
| $m_h$ | 1.46 | kg |

Table 5.1: Parameters of the humand arm-hand model

## 5.2 Friction model

The current effort measured by the ESCON controller is proportional to the sum of the forces required to overcome the nonlinearities of the EndoWrist and the external forces applied to the end-effector. One of the main nonlinearities is the friction. For the system to be transparent, the force required to overcome friction should not be fed back as this force would not be felt if the teleoperator was holding the tool directly. In order to isolate the external forces, a friction model is required. The model built is based on the one derived in [34] that describe the friction in a motor. However in the model it was decided to consider the motor and the EndoWrist as one element as no dynamic model for the motor alone is derived. Thus, the model estimates the sum of the friction in the EndoWrist and in the motor.

### 5.2.1 Model

The total friction acting on the actuator can be written as:

$$\tau_f = \tau_v + \tau_c + \tau_s \tag{5.4}$$

with:
$\tau_v$ viscous friction
$\tau_c$ coulomb friction
$\tau_s$ static friction

The viscous friction is proportional to the opposite of the velocity:

$$\tau_v = F_\mu \cdot \omega \tag{5.5}$$

with:
$\omega$ the angular velocity
$F_\mu$ a negative coefficient

The coefficient $F_\mu$ can be computed from the measurements made on the setup by plotting the effort depending on the velocity.

The stiction or static friction is the amount of effort required for the object to start moving when its velocity is zero. As such it can be described as:

$$\tau_s = \begin{cases} K_s, & \text{if } \omega = 0 \\ 0, & \text{else} \end{cases} \tag{5.6}$$

with:
$\omega$ the angular velocity
$K_s$ a constant

The coulomb friction occurs as a constant force opposing the movement:

$$\tau_c = sign(\omega) \cdot K_c \tag{5.7}$$

with:
$\omega$ the angular velocity

$K_c$ a constant determined experimentally

the *sign* function is defined as $sign(\omega) = \begin{cases} 1, & \text{if } \omega > 0 \\ 0, & \text{if } \omega == 0 \\ -1, & \text{if } \omega < 0 \end{cases}$

From *Equation: (5.5)* to *Equation: (5.7)*, the total friction given by *Equation: (5.4)* can be plotted as:

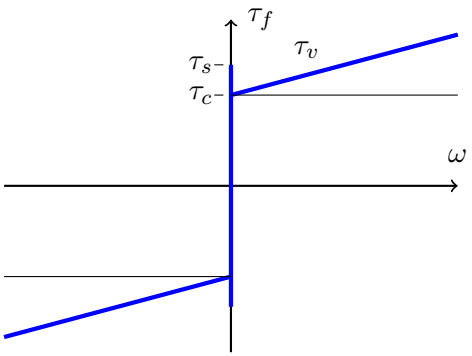



Figure 5.4: friction model

### 5.2.2 Measurements

For the model to be complete, each type of friction requries a value to be measured. These measurements need to be made for each joint as the nonlinearities in the EndoWrist varies greatly depending on the joints.

#### Test equipment:

- Endowrist model 420093 (AAU number: #4).
- Maxon 110160 motor with attached Maxon gearhead 110356 and Maxon encoder 201937.
- sbRIO board.

#### Procedure:

The viscous friction coefficient $F$ can be calculated by measuring the current for different speeds. From these values, an affine function can be computed, the slope of which will be the coefficient. The constant $K_c$ for the Coulomb friction can be measured by setting the motor in motion and decreasing the current until it stops moving. The value of the current at that time is the value of $K_c$. The constant $K_s$ for the static friction is measured by increasing the current sent to the motor until it starts moving. The value of the current at that time is the value of $K_s$.

This procedure is repeated for each motors.

**Measuring data:**

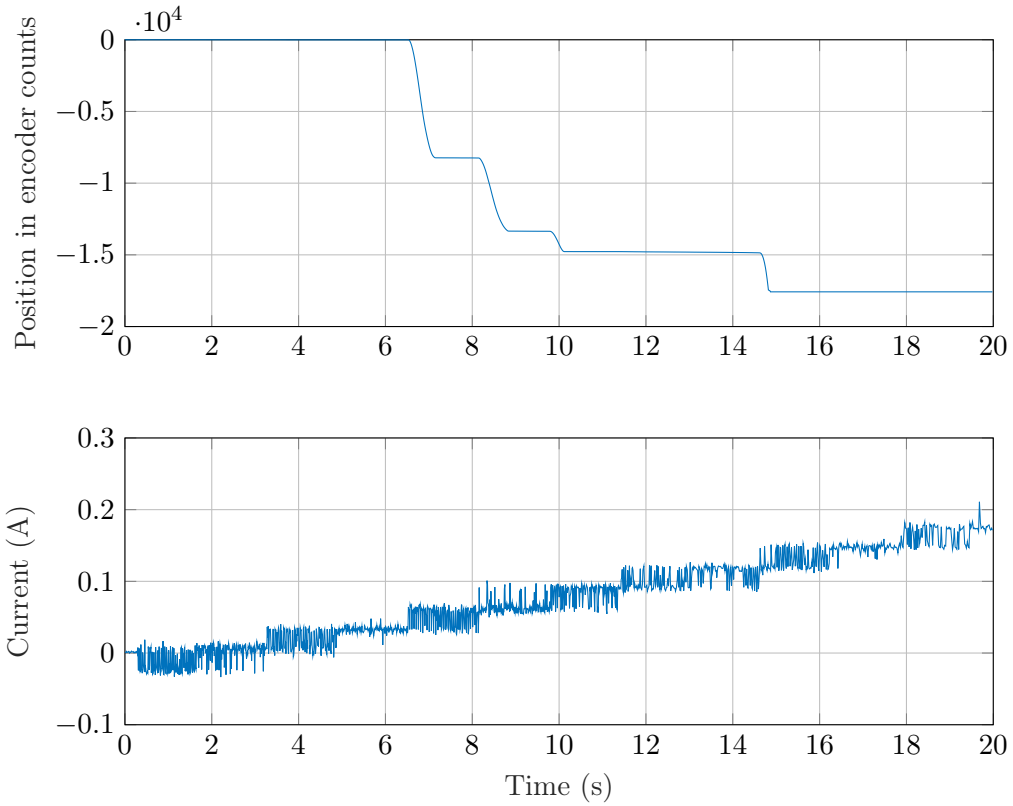



Figure 5.5: Measurement of the static friction coefficient for the roll

| Joint | static friction coefficient [A] |
|---|---|
| roll | 0.7 |
| pitch | 0.17 |
| yaw | 0.21 |
| clamp | 0.20 |

Table 5.2: Static friction coefficients

**Results:**

As it can be seen in *Figure 5.5*, not only is the current measurement noisy but the position does not have a clearly distinguishable point where it starts increasing. This behavior of the position is due to the play between the motor and the EndoWrist and to the nonlinearities of the EndoWrist. Because of this it is impossible to obtain proper measurements of all coefficients with the equipement available for this project. As the goal for this project is not accuracy in the force feedback, it was decided to simplify the friction model.

In the new model, viscous friction is neglected as the end effector cannot move at high speed. In addition, the coulomb constant $K_c$ is chosen to be equal to $K_s$, by doing so the overall friction is slightly overestimated and the friction model removes the nonlinearities and insignificant external forces from the feedback. The values used for $K_s$ are gathered in *Figure 5.2*

**Conclusion:**

The values for the static friction constants have been measured and used to build a new simplified model of friction that still suits the requirements this project. The new model is represented in *Figure 5.6*. It is used in combination with the force estimation to increase the overall transparency.

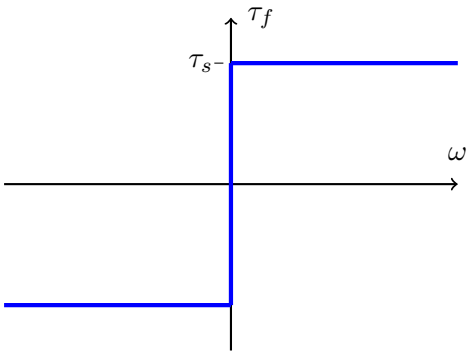



Figure 5.6: final friction model

### 5.2.3 Applying the model to data

The model derived was not applied to the input of the system but used to detect external forces. If external forces are applied to the end effector, the output of the force estimation is fed back, if not, no feedback is sent. The estimated force is fed back as long as the value of the current is sufficient to overcome friction.

## 5.3 Simulation

Both the yaw and pitch model serve as a good approximation of force, but they do not adequately capture the full dynamics of the EndoWrist. As mentioned in section 4.4, having a better description of the general dynamics would allow for state estimates usable for state feedback control [35].

State feedback control would allow us to control the force output by the EndoWrist without having the ability to measure it during operation. Estimated states can be used to calculate the force being applied. Even though the current models do not allow for useable state estimates of the EndoWrist, in this section the opposite is assumed and simulated on the yaw force model.

As seen in *Figure 5.7* our simulation model assumes that the EndoWrist yaw force dynamics consists of the linear system and input nonlinearities identified in chapter 4. Our hypothesis is that full reference following capability can be added to the nonlinear system using only position error measurements and the linear model as part of a Kalman filter.

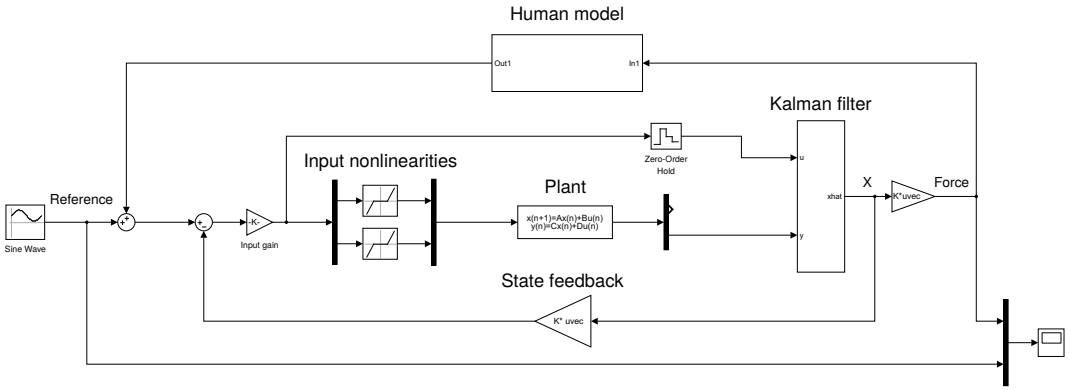



Figure 5.7: Simulation setup.

We can see that the outer part of the system consists of a sinusoidal force reference, which is affected by oscillations from the human model. This represents the operator's reaction to force feedback while giving a force reference.

The input nonlinearities in the system represent the effects of friction on effort and velocity, as they are the inputs to the plant (EndoWrist). Only the position error measurement is used as an input to the kalman filter, along with the inputs.

As the Kalman filter outputs the state estimates, they are used for estimating force and providing state feedback to the input. The estimated force is then sent to the geomagic touch, which transfers it back to the operator.

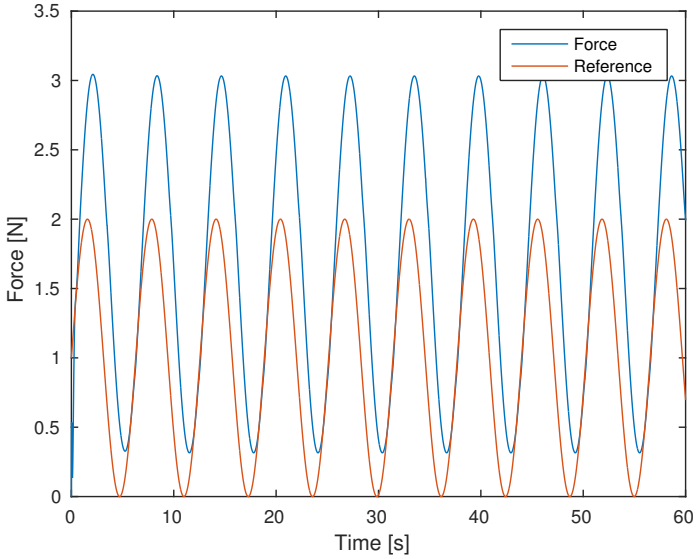



Figure 5.8: Force reference following.

As seen in *Figure 5.8* the transient behavior of the force matches that of the reference signal. The offset and amplitude can be corrected by implementing a gain and offset to the reference.

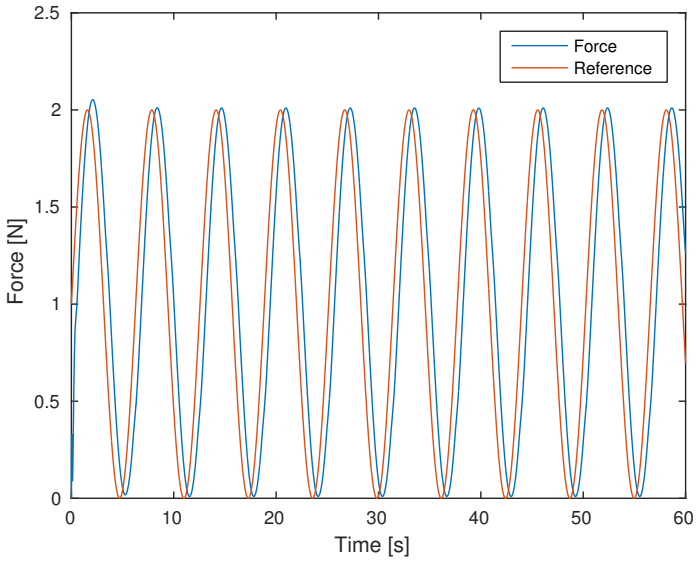



Figure 5.9: Force reference following with modified reference.

In *Figure 5.9* we can see that the modified reference indeed only leaves a delay in response of the system of about 0.3 seconds. While this delay is significant in a surgery scenario, we believe it can be further reduced using various state-space methods.

## 5.4 Embedded control system

It is imperative to create a controller which minimizes the effects of communication delay. Therefore the aim was to implement as much of the motion control on the embedded system

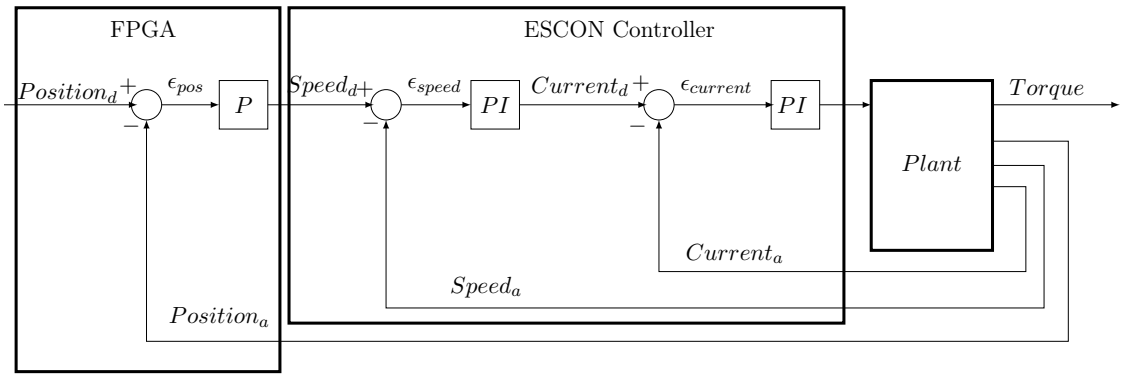



Figure 5.10: Embedded cascade control structure. d index means the desired control value, a index means actual value, $\epsilon$ is the error

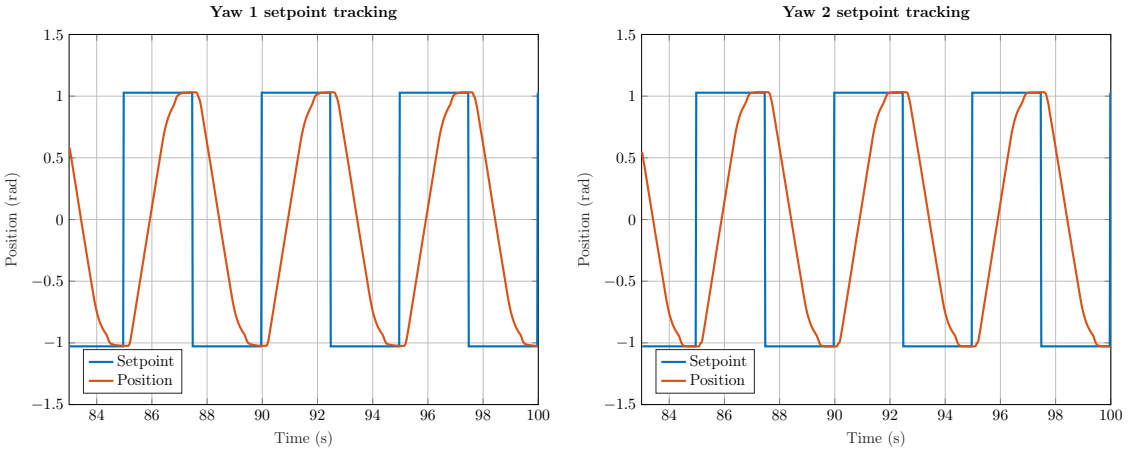
as possible. The human operator inputs the required position by moving the GT. These positions are sent as position reference to the sbRIO. The sbRIO has an implemented cascade position-speed-current control, see *Figure 5.10*, which takes care of the EndoWrist positioning tasks. The P position control that tracks the setpoint is running on the sbRIO FPGA. The FPGA sends the speed reference to the Escon controller in the form of PWM signals. The Escon controller is running the PI speed control with inner PI current control loop. The inner loops are faster than the outer loops.

The chosen controller parameters are listed in *Table: 5.3*:

| Controller type | P gain | I time constant | Sampling rate |
|---|---|---|---|
| Position controller | 10 | $\emptyset$ | 2 kHz |
| Speed controller | 426 | 28 ms | 5.36 kHz |
| Position controller | 1121 | $38\mu s$ | 53.6 kHz |

Table 5.3: Cascade position control parameters

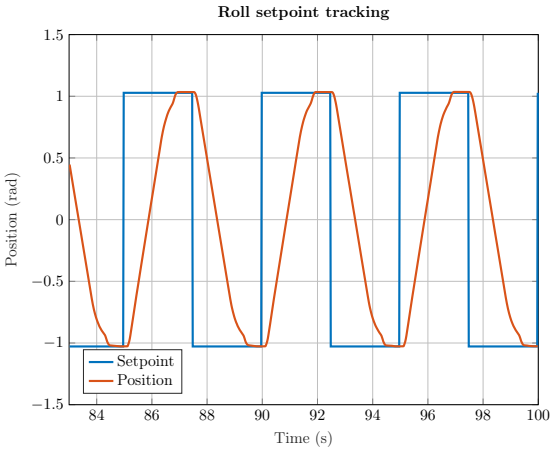
The speed and current controller parameters were calibrated automatically by ESCON Studio. The program queries the motor parameters, desired current limits and maximum acceleration values and calculates the controller parameters based on those. The EndoWrist can not handle high speeds and high acceleration, thus speed was not the main priority in tuning the position PID control. We focused on precision and avoiding overshoot. By running experiments, we decided upon a P controller of 10 for each motor. A lower gain would result in slower motion, a higher gain would result in overshoots. The EndoWrist-motor system has inherent damping in the form of friction, inertia and elasticity, thus the system is stable even without integral and differential controller. For the experiments with free-running EndoWrist, see *Figure 5.11*. The setpoints square signals moving jumping between the limits the EndoWrist can move between. This is a movement that would never happen in normal operation. The delay defined in these graphs is the time it takes for the positions to reach the setpoints with an error of 5%.



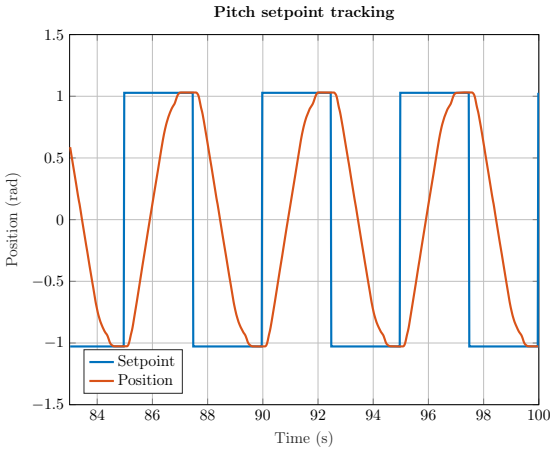

(a) Yaw 1 delay is 1.7 s




(b) Yaw 2 delay is 1.7 s



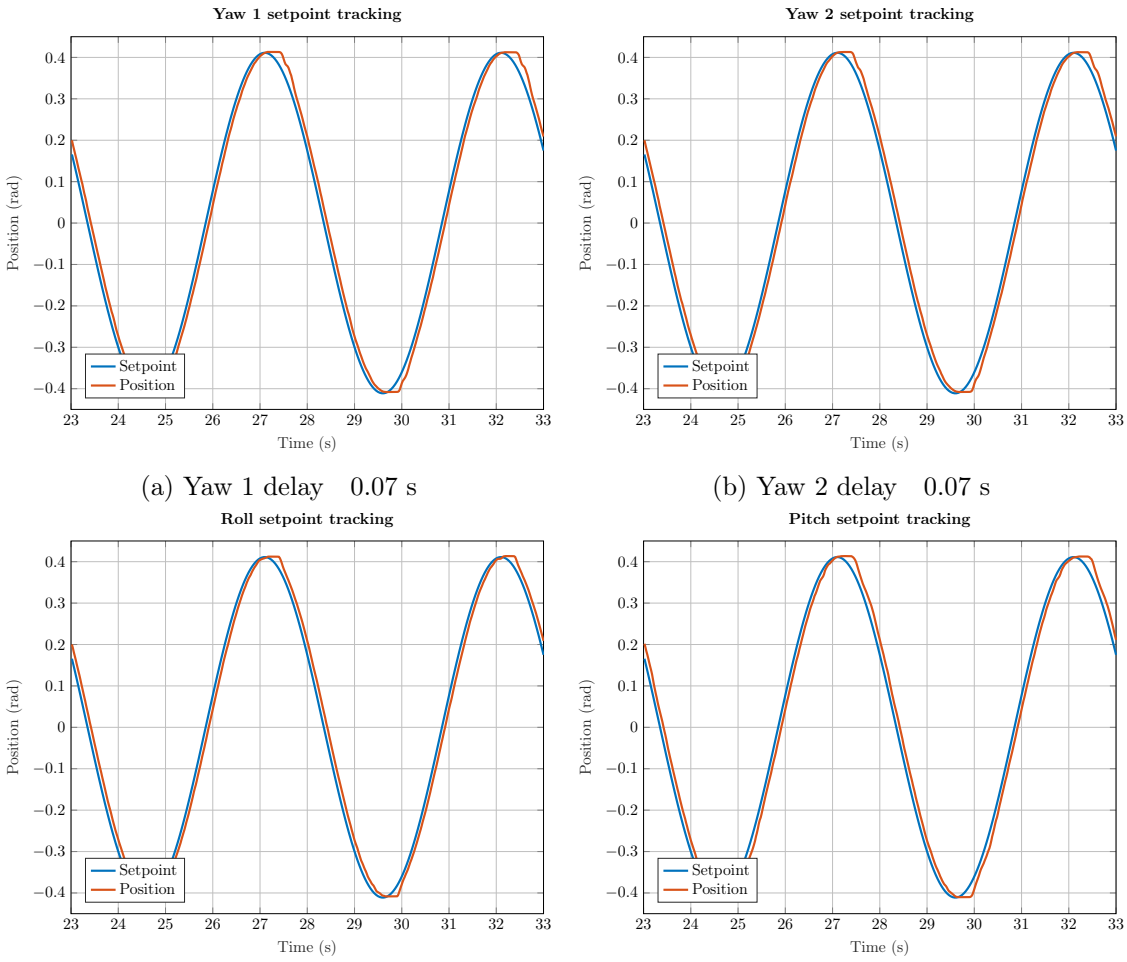

(c) Roll delay is 1.7 s




(d) Pitch delay is 1.7 s

Figure 5.11: Tracking of setpoint jumps between limits

For the measurements with sinusoidal setpoint signals, see *Figure 5.12*. This is much more similar to normal operations, where instead of jumps, the operator makes continuous moves. The delay defined in these graphs is the time between the setpoints and positions being at the same value.




(a) Yaw 1 delay 0.07 s

(b) Yaw 2 delay 0.07 s

(c) Roll delay 0.06 s

(d) Pitch delay 0.08 s

Figure 5.12: Tracking of sinusoidal setpoint signals

### Encoder tick counting

The FPGA has implemented encoder signal counter, that directly receives the signals from the ESCON controller's two encoder sensors. One sensor is shifted by 90 deg. The FPGA steps one unit each time the state of any of the sensor changes and FPGA compares the state of encoder A to the state of encoder B using a XOR function. This way the direction of the movement can be determined. The resulting position resolution is four times as high as the resolution of encoder marks.

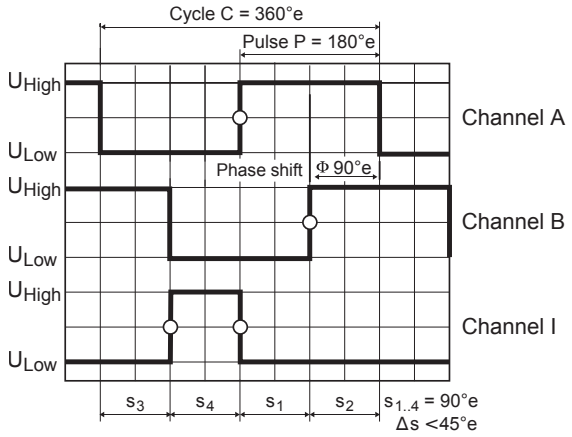



Figure 5.13: Encoder signal sensor states depending on position [12].

### FPGA

The FPGA built into the sbRIO is taking care of the interfacing between the controllers and the microprocessor. The code running on the FPGA is much faster than the one running on the microprocessor. To reach maximum possible speed, we implemented whatever we could on the FPGA, however the FPGA is incapable of handling higher level functions such as string handling and UDP networking.

The FPGA's main functions are the following:

- Count the encoder ticks coming from the controller
- Read the controller's analog and digital inputs
- Calculate position PID control signal and send the corresponding PWM signal to the controller
- Enable the motors

Since the current value coming from the ESCON is noisy, the FPGA also needs a built in low pass filter.

### ESCON motor controllers

The ESCON controllers are programmed to output a speed corresponding to the FPGA PWM signal. The speed controller has an inner current controller, which also keeps the motors from taking overcurrent. The current ramps can be adjusted to fit the user's needs. The speed control gain can be adjusted by the onboard potmeter. The outer speed loop provides the control signal for the internal current controller. The inner control loop must have a higher frequency than the speed control loop. The PI current controller is running at 53.6 kHz, while the PI speed controller is running at 5.36 kHz. The inner loop ensures fast response, the outer higher precision [36]. The advantages of the cascade structure:

- Better setpoint tracking
- Better disturbance rejection
- Less delay and phase lag

The position is controlled by setting the duty ratio of the incoming PWM signal. The speed reference is 0 at 50% duty ratio and grows linearily at higher values. The ESCON controller has 2 programmable analog outputs, thus we are unable to read speed, actual current and

demand current simultaneously. The controllers have autocalibration functionalities, but this ability is obstructed by the gearing limits.

ESCON Studio provides the tool for logging data from the ESCON controller by using only a USB cable.

## 5.5 Mapping

The setup studied in this project does not allow the translations of the EndoWrist. Although the models derived can be used on the full da Vinci robot, the mapping of the movements of the Geomagic Touch to the tool cannot be the same. As described in Section *2.6: Geomagic touch*, the GT can only actuates three joints, these joints are the one that needs to be used for the human to easily estimates the quality of the force feedback. It was decided to control the EndoWrist by controlling the roll, pitch and clamping independently. To do so, each of these three joints are assigned one vector. All three vectors are orthogonal to be able to sense the feedback of each joint individually. The roll of the EndoWrist was associated to the roll of the GT. The clamp of the EndoWrist is associated to the radial vector going from the base of the GT to the end of the stylus. The pitch is controlled by the normal vector to the plan defined by the two previous vectors which is a vertical vector. The vectors in the horizontal plan are represented in *Figure 5.14*. The Cartesian z axis and the vector for the pitch of the EndoWrist are collinear and are not represented on the figure.

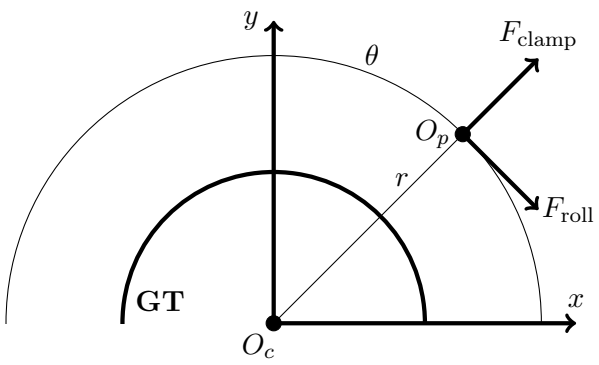



Figure 5.14: Vectors in the Cartesian horizontal plan

With:

| | |
|---|---|
| $\theta$ | the angle between $\overrightarrow{y}$ and $\overrightarrow{O_cO_p}$ |
| $r$ | the distance $\overline{O_cO_p}$ |
| $O_c$ | Origin of the Cartesian space |
| $O_p$ | Position of the end-effector |

The phantom_omni node[37] used to communicate can only handle Cartesian positions and forces. Thus, conversion from cylindrical to Cartesian force vectors, see *Equation: (5.8)*, and from Cartesian coordinates to cylindrical coordinates, see *Equation: (5.9)*, are required.

$$\begin{aligned} F_x &= F_{\text{clamp}} \cdot sin(\theta) + F_{\text{roll}} \cdot cos(\theta) \\ F_y &= F_{\text{clamp}} \cdot cos(\theta) - F_{\text{roll}} \cdot sin(\theta) \\ F_z &= F_{\text{pitch}} \end{aligned} \tag{5.8}$$

With:

| | |
|---|---|
| $(x, y, z)$ | the Cartesian coordinates of the GT |
| $F_{\text{clamp}}$ | the force feedback from the clamp |
| $F_{\text{pitch}}$ | the force feedback from the pitch |
| $F_{\text{roll}}$ | the force feedback from the roll |

$$\begin{aligned} P_{\text{roll}} &= a_{\text{roll}} \cdot \theta + b_{\text{roll}} \\ P_{\text{pitch}} &= a_{\text{pitch}} \cdot z + b_{\text{pitch}} \\ P_{\text{clamp1}} &= -a_{\text{clamp}} \cdot r + b_{\text{clamp}} \\ P_{\text{clamp2}} &= a_{\text{clamp}} \cdot r - b_{\text{clamp}} \end{aligned} \tag{5.9}$$

With:

| | |
|---|---|
| $P_i$ | Position command sent to the joint $i$ |
| $a_i$ | constant scaling coefficient for joint $i$ |
| $b_i$ | constant scaling offset for joint $i$ |

# Results and Discussion 6

## 6.1 Results

An operator clamps a finger by controlling the Geomagic Touch. As no equipment is available for measuring the force applied by the end-effector, the estimated force is compared with the position error and the current applied to the actuator. The experiment is made for the maximum frequency reached of 638 Hz in *Figure 6.1*, which shows the operator slowly clamping the finger. For comparison, measurements for the original frequency of 100 Hz are displayed in *Figure 6.2*, where the operator clamps and releases the finger at a fast pace.

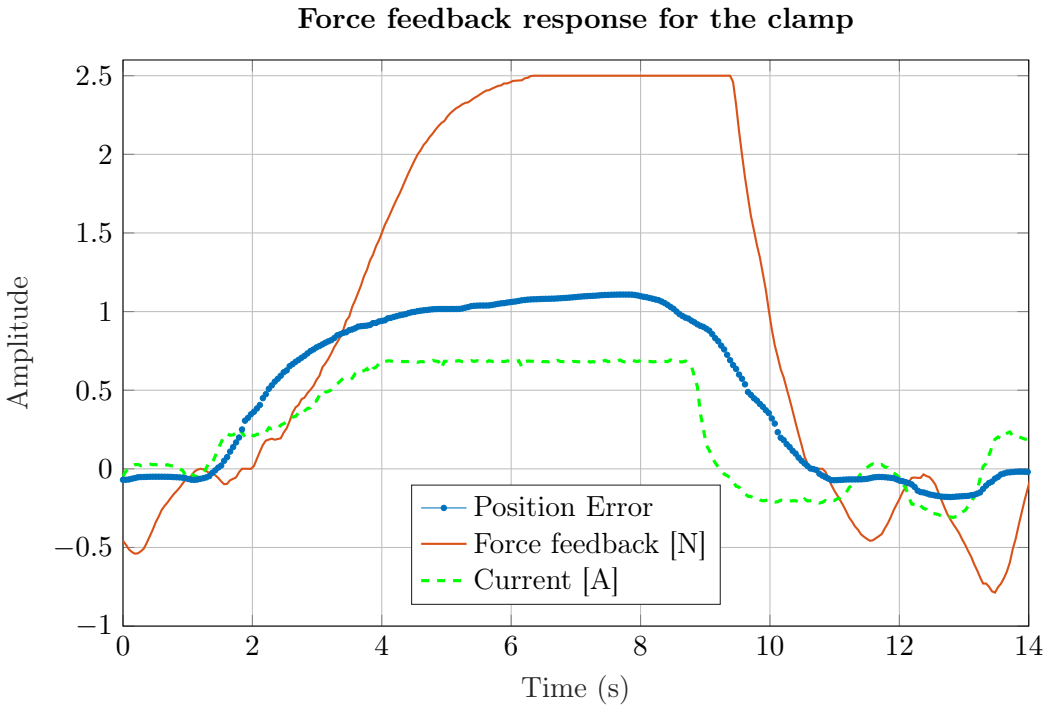



Figure 6.1: Clamping a finger using a refresh rate of 638 Hz for the communication

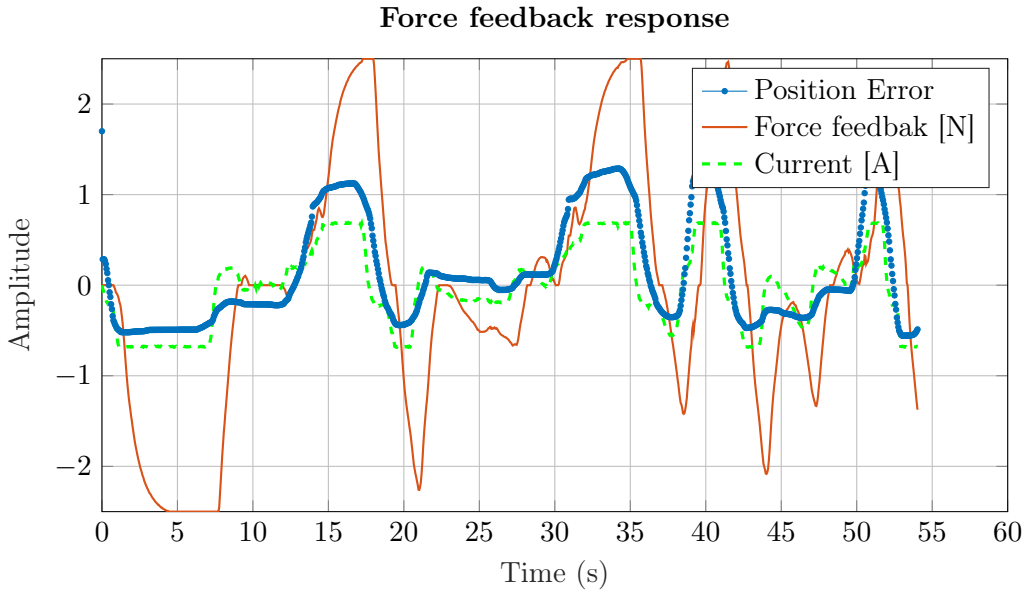



Figure 6.2: Clamping a finger using a refresh rate of 100 Hz for the communication and a varying command

Similarly to the experiment described for the gripping force, the operator uses the Geomagic Touch to control the roll and the movement of the end-effector are hindered by synthetic skin, the resulting measurements are represented in *Figure 6.3*.

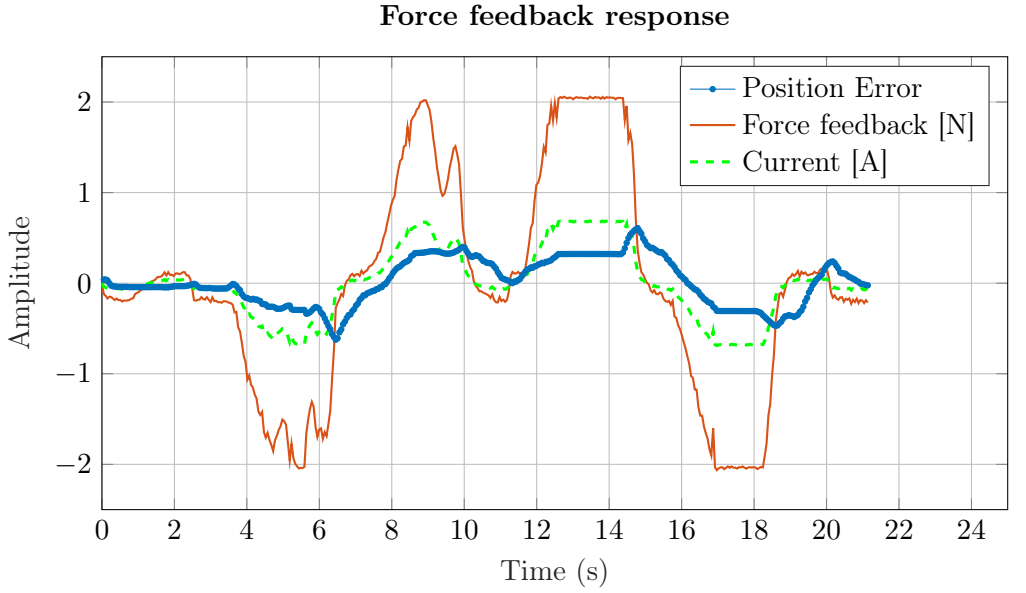



Figure 6.3: Response of the force feedback for the roll while varying command with a refresh rate of 100 Hz for the communication

The pitch force model currently results in much higher estimates (around 10 N) on a real time system. The cause of this is not clear at the time of writing. As it is dependent on velocity, it is possible that velocities calculated from positions in the data sets, are differently scaled to the ones measured by the sbRIO.

The friction model derived in Section *5.2: Friction model* was applied to the data of *Figure*

*6.1*, the obtained feedback is represented in *Figure 6.4*.

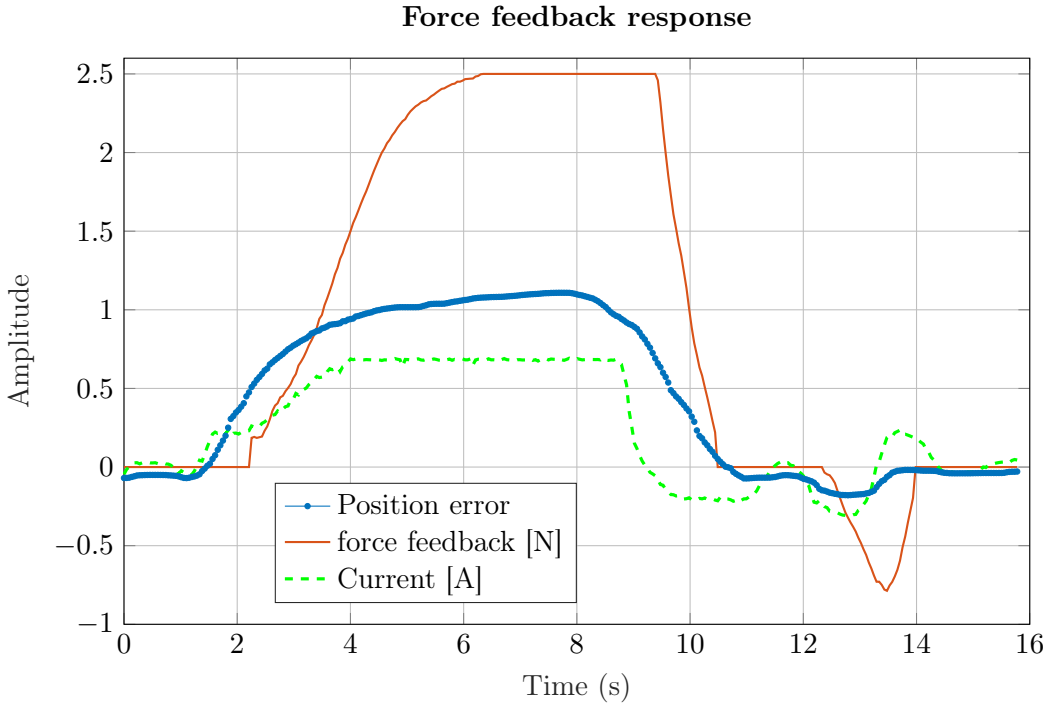



Figure 6.4: Friction model applied to the clamping response

## 6.2 Discussion

From *Figure 6.1*, the yaw force fed back to the user by the Geomagic Touch dynamically corresponds to both the current increase and the position error. From Section *5.2: Friction model*, the friction value computed for the gripping force are 0.20 and 0.21 A. This value is overcome at 2 s and the force fed back starts to increase at 2.2 seconds. Thus a delay of 0.2 seconds between applying force to the external environment and receiving feedback is present. Considering that the communication loops have a refresh rate of 638 Hz and 1000 Hz, amounting to a delay of 2.6 ms, the delay resulting from the communication can be neglected in front of the one resulting from the model. In addition, delay of 0.7 seconds is present when releasing the finger. From *Figure 6.2*, it can be seen that with a refresh rate of a 100 Hz the model can still estimate the dynamics of the force applied to the end-effector even when varying the command sent.

The roll model of a lower order can accurately track the dynamics of the force without showing the same delay, however it is more sensitive to noise as it can be seen at 6 seconds in *Figure 6.3*.

Unlike the yaw model, data acquisition for the pitch model was more difficult, as the EndoWrists structure affected measurements. This results in additional nonlinearities in the measurements, since force was not always applied to an angle perpendicular to the load cell. As a consequence, the model somewhat underestimates the applied force.

An attempt was made to implement state estimation the correct the force estimate. The steady-state Kalman filter was implemented, with position error and velocity measurements used for state estimation. Simulation results have shown that such a system would not improve the systems, as the current models do not capture the dynamics adequately.

In the future, an improved model could be utilized with state estimation to provide state feedback control of the outputed force. The force reference could be directly mapped to the Geomagic Touch movement axes, providing a greater degree of control to the system.

In *Figure 6.1*, the operator attempted to reach a point with no feedback outside of the main clamping. However it can be seen that before 2 seconds and after 11 seconds, that the operator moved trying to find that point and received feedback at that time. The first deduction is that the mapping and scaling of the command should not allow to attempt to open the clamp more that it is possible as the current increases when trying to exceed the range of movement possible. This conclusion was supported by the negative feedback visible in *Figure 6.2*. The second deduction was that an accurate model of the nonlinearities was required. A model of friction was derived in Section *5.2: Friction model* and promising results were obtained in *Figure 6.4*. The oscillations are removed from the feedback when no strong current is applied. The negative force fed back from 12 to 14 seconds is due to the operator trying to open the clamp more than it is possible. It is believed that implementing this model would reduced the oscillations even further than it does in the simulation as the operator would have less difficulty in finding the point were there is no feedback.

«««< HEAD +Although the refresh rate of the communication does not allow to reach the goal of 550 Hz for the feedback loop, a significant improvement can be noted. To further increase the refresh rate, three methods are considered: compressing data, optimizing the programs and implementing on a real-time system.
+Compression of exchanged data was evoked in Section *3.2.2: Minimizing the size of the transmitted messages*. It is believed that the implementation of a fast compression algorithm would improve the refresh rate in a communication between two computers. However additional precautions must be taken when implementing it on the embedded system, as the computation power is not as high as it is for a computer. The computation time required to compress the small amount of data sent should be compared to the transmission time of said data.
+According to [38] it is possible to receive a million packets per second on a Linux system. Although such a high frequency cannot be reached when other tasks have to be performed on the computer. It should be possible to increase the frequency by optimizing the programs used for the project. Using low level libraries for communication would be the first step to improve the communication. In addition, the program currently runs many thread to communicate with external devices and between internal processes. These threads are not synchronized or prioritized, thus a new software architecture could reduce the load on the processing unit and ensure all threads can be properly executed.
+The last method considered is an extension of the previous suggestion to create a new software architecture to increase control over the different thread. By using a real-time system, better control over the priority of each process can be achieved and thus less computation power can be allowed to processes non-essential for the force feedback, such as the graphical interface.
+Another aspect that should be considered for future works is the safety in the communication. Currently only a simple detection of connection timeout has been implemented. In future application, security against cyber-attacks should be considered as the system could be extended to remote teleoperation. ======= Although the refresh rate of the communication does not allow to reach the goal of 550 Hz for the feedback loop, a significant improvement can be noted. To further increase the refresh rate three axis are considered: compressing data, optimizing the programs and implementing on a real-time system.
Compression of exchanged data was evoked in Section *3.2.2: Minimizing the size of the transmitted messages*. It is believed that the implementation of a fast compression algo-

rithm would improve the refresh rate in a communication between two computers. However additional precautions must be taken when implementing one the embedded system as the computation power is not as high as it is for a computer. The computation time required to compress the small amount of data sent should be compared to the transmission time of said data.
From [38] it is possible to receive a million packets per second on a Linux system. Although such a high frequency cannot be reached when other tasks have to be performed on the computer, it should be possible to increase the frequency by optimizing the programs used for the project. Using low level libraries for communication would be the first step to improve the communication. In addition, the program currently runs many thread to communicate with external devices and between internal processes. These threads are not synchronized or prioritized, thus a new software architecture could reduce the load on the processing unit and ensure all threads can be properly executed.
The last axis considered is an extension of the previous suggestion to create a new software architecture to increase control over the different thread. It is believed that implementing the force feedback on a real-time system such as Real Time Application Interface (RTAI). By using a real-time system, better control over the priority of each process can be achieved and thus less computation power can be allowed to processes non-essential for the force feedback such as the graphical interface.
Another aspect that should be considered for future works is the safety in the communication. Currently only a simple detection of connection timeout has been implemented. In future application, security against cyber-attacks should be considered as the system could be extended to remote teleoperation. »»»> 89a7ccb61601cc4df197703cab17ed11992421b8

# Conclusion 7

A new communication protocol between the embedded system and the processing unit have been implemented. Furthermore a new communication protocol have been implemented. THe old JSON format have been discarded and replaced by a binary representation of the data. This have made a decrease in the data size send between the embedded system and the processing unit with 85 %.
The change from TCP to UDP and the change from JSON to a binary representation of the data send, have increased the communication speed from 100 Hz to 638 Hz. This was however not enough to fulfill the goal of 1000 Hz between the embedded system and the processing unit.
The theoretical communication speed from the embedded system to the Geomagic Touch have increased from 91 Hz to 390 Hz, which is 160 Hz lower than the goal of this project.

A Hammer Stein Weiner model has been implemented for estimating the force applied to the End-effector. This model is capable of estimating the trend of the force applied to the end-effector, by utilizing the velocity and effort available, which also is available on the da Vinci robot. Even though the model can estimate the trend of the force, a short delay appear between the end-effector and the force fed back. This delay was between 0.2 to 0.7 seconds, depending on force applied of released.

It was observed that that force was fed back under movement, even when no external force was applied to the End-effector. This was due to the nonlinear friction in the EndoWrist that had to be overcome before any movement. A simple friction model has been derived to compensate for this feedback. The friction model has only been simulated on existing data, but have given promising results.

An attempt has been made on implementing a Kalman filter for improving the estimation of the force. However no significant improvement was observed.

# Bibliography


[1] K. Schwab, "The fourth industrial revolution: what it means, how to respond." https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond. Accessed: 15-12-2016.

[2] Lenox Hill Hospital, "Bladder cancer surgery - robotic cystectomy." http://www.roboticoncology.com/laparoscopic-cystectomy/. Accessed: 12-12-2016.

[3] B. T. Bethea, A. M. Okamura, M. Kitagawa, T. P. Fitton, S. M. Cattaneo, V. L. Gott, W. A. Baumgartner, and D. D. Yuh, "Application of haptic feedback to robotic surgery," *Journal of Laparoendoscopic & Advanced Surgical Techniques*, vol. 14, no. 3, pp. 191–195, 2004.

[4] Intuitive Surgery, *EndoWrist/Single-Site Instrument & Accessory Catalog*, 2014.

[5] T. R. Coles, D. Meglan, and N. W. John, "The role of haptics in medical training simulators: a survey of the state of the art," *IEEE Transactions on Haptics*, vol. 4, no. 1, pp. 51–66, 2011.

[6] G. C. Burdea and F. P. Brooks, *Force and touch feedback for virtual reality*. Wiley New York, 1996.

[7] S. Booth, F. Angelis, and T. Schmidt-Tjarksen, "The influence of changing haptic refresh-rate on subjective user experiences-lessons for effective touch-based applications," in *Proceedings of eurohaptics*, pp. 374–383, 2003.

[8] K. D. Hansen, S. Jensen, C. Sloth, and R. Wisniewski, "Instrumentation of the da vinci robotic surgical system," in *3rd Aau Workshop on Robotics*, pp. 15–18, Aalborg Universitetsforlag, 2015.

[9] da Vinci surgery, *The da Vinci surgical system*, November 2015. Downloadet: 05-12-2016.

[10] Maxon Motor, *A-max 22 Ø22 mm, Graphite Brushes, 6 Watt, with terminals*, April 2016. Downloadet: 04-11-2016.

[11] Maxon Motor, *Planetary Gearhead GP 22 B Ø22 mm, 0.1 - 0.3 Nm, Sleeve Bearing*, April 2016. Downloadet: 04-11-2016.

[12] Maxon Motor, *Encoder MR, Typ M, 512 CPT, 2 Channels, with Line Driver*, May 2016. Downloadet: 04-11-2016.

[13] "Escon module 50/5, 4-q servocontroller for dc/ec motors, 5/15 a, 10 - 50 vdc." http://www.maxonmotor.com/maxon/view/product/control/4-Q-Servokontroller/438725. Accessed: 09-11-2016.

[14] Measurementest, "National instruments introduces ni single-board rio embedded devices with multifunction i/o for custom application development." http://www.measurementest.com/2012/03/national-instruments-introduces-ni.html, 2012. Accessed: 16-11-2016.

[15] E. T. Bray, "The javascript object notation (json) data interchange format," *RFC 7159, Google, Inc.*, 2014.

[16] K. D. Hansen, "davinci driver." https://github.com/AalborgUniversity-RoboticSurgeryGroup/davinci_driver, May 2014. Accessed: 9-10-2016.

[17] J. Postel, "Transmission control protocol - darpa internet program," *RFC 791, USC/Information Sciences Institute*, 1981.

[18] M. V. F. Marcelloni, "A simple algorithm for data compression in wireless sensor networks," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 411–413, 2008.

[19] W. N. Ross, "An extremely fast ziv-lempel data compression algorithm," *Proceedings of Data Compression Conference*, pp. 362–371, 1991.

[20] lilyco, "A udp-based reliable data transfer library." http://www.codeproject.com/Articles/11046/A-UDP-based-Reliable-Data-Transfer-Library, 2006. Accessed: 09-11-2016.

[21] L. Salzman, "Reliable udp networking library." https://github.com/lsalzman/enet, 2012. Accessed: 09-11-2016.

[22] Cisco, "Understanding jitter in packet voice networks (cisco ios platforms)." http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/18902-jitter-packet-voice.pdf. Document ID: 18902, Accessed: 16-12-2016.

[23] C. Y. Kim, M. C. Lee, R. B. Wicker, and S.-M. Yoon, "Dynamic modeling of coupled tendon-driven system for surgical robot instrument," *International Journal of Precision Engineering and Manufacturing*, vol. 15, no. 10, pp. 2077–2084, 2014.

[24] Y. Zhu, "Estimation of an n–l–n hammerstein–wiener model," *Automatica*, vol. 38, no. 9, pp. 1607–1614, 2002.

[25] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

[26] W. Gawronski and J.-N. Juang, "Model reduction in limited time and frequency intervals," *International Journal of Systems Science*, vol. 21, no. 2, pp. 349–376, 1990.

[27] C. Van Loan, *Computational frameworks for the fast Fourier transform*, vol. 10. Siam, 1992.

[28] P. Van Overschee and B. De Moor, "N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, 1994.

[29] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: with matlab exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, by Brown, Robert Grover.; Hwang, Patrick YC New York: Wiley, c1997.*, vol. 1, 1997.

[30] B. Friedland, *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.

[31] D. A. Lawrence, "Stability and transparency in bilateral teleoperation," in *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, pp. 2649–2655, IEEE, 1992.

[32] T. Tsuji, K. Goto, M. Moritani, M. Kaneko, and P. Morasso, "Spatial characteristics of human hand impedance in multi-joint arm movements," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 1, pp. 423–430, IEEE, 1994.

[33] J. E. Speich, L. Shao, and M. Goldfarb, "Modeling the human hand as it interacts with a telemanipulation system," *Mechatronics*, vol. 15, no. 9, pp. 1127–1142, 2005.

[34] M. Lazeroms, *Force Reflection for Telemanipulation - applied to Minimally Invasive Surgery*. 1999. PhD Thesis.

[35] D. Yue, Q.-L. Han, and C. Peng, "State feedback controller design of networked control systems," in *Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on*, vol. 1, pp. 242–247, IEEE, 2004.

[36] A. C. R. D. A. A. Anagha, Ranjith K., "Cascade speed control of dc motor," *International Journal of Electrical, Electronics and Data Communication*, 2014.

[37] D. Powell, "Ros node for sensable phantom omni devices." https://github.com/danepowell/phantom_omni, 2013. Accessed: 20-09-2016.

[38] M. Majkowski, "How to receive a million packets per second." https://blog.cloudflare.com/how-to-receive-a-million-packets/, 2015.

[39] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. John Wiley & Sons, 2008.

[40] Bogde, "Hx711." https://github.com/bogde/HX711, 2016. Accessed: 18-12-2016.

[41] Arduinotech, "Weight measure 1 kg." https://arduinotech.dk/shop/vaegt-foeler-1kg/. Downloadet: 29-11-2016.

[42] Avia Semiconductor, *HX711, 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales*. Downloadet: 29-11-2016.

[43] Wikibooks, "Communication networks/tcp and udp protocols." https://en.wikibooks.org/wiki/Communication_Networks/TCP_and_UDP_Protocols. Accessed: 22-11-2016.

[44] R. operating system, "Ros/ introduction." http://wiki.ros.org/ROS/Introduction. Accessed: 13-11-2016.

[45] R. operating system, "Ros/ concepts." http://wiki.ros.org/ROS/Concepts. Accessed: 13-11-2016.

# Github repository A

This chapter contains the link to the data collected for this project.

https://github.com/bomlee/16gr735.git

# Installation Guide B

## B.1 Installation of the Geomagic Touch

The Phantom omni can be installed on both Windows, and Linux, embedded systems. Use the provided installation guides on
http://dsc.sensable.com/viewforum.php?f=15

**Note to the linux system:** If this installation is running on a non-US version of Linux, a readout of the different joint positions can be corrupted. This is because of the difference in the EU and US format of numbers, where the ',' and '.' is read differently.
The solution to this problem was found on:
http://dsc.sensable.com/viewtopic.php?t=5644#top

To solve the problem add the following to /.bashrc

```
1 export LC_NUMERIC=en_US.UTF-8
```

and use the following command to reload the file

```
1 source ~/.bashrc
```

## B.2 Connecting to the Geomagic Touch and the sbRIO

When connecting both devices to the same computer one or both of the devices are disconnected. The problem was identified using the *route -n* command. Two default routes to the local network (169.254.0.0/16) were present in the table, each of them leading to a different physical interface.

To solve this issue, the ip address of the interface connected to the sbRIO was fixed to have always the same value: 169.254.4.10. The default route to the interface connected to the sbRIO was removed. And a new route was added to send every packets that should be sent to the sbRIO to the right physical interface. The ip address of the sbRIO is always 169.254.4.42.

This manipulation has to be done every time the two devices are connected to the computer. For effiency, a bash script was created:

```
1 #!/bin/bash
2 ifconfig eth0 169.254.4.10 netmask 255.255.0.0
3 route del -net 169.254.0.0 gw 0.0.0.0 netmask 255.255.0.0
4 route add -net 169.254.4.42 netmask 255.255.255.255 gw 169.254.4.10
```

# Models C

Below are included models for yaw and pitch force, none of the models have the feedthrough component ($\mathbf{D} = 0$) .

**Current and velocity to yaw force model**

$$\mathbf{A} = \begin{bmatrix} 0.9745 & -0.3699 & -0.0620 & 0.2917 & 0.5007 & -5.5391 \\ -0.0002 & 1.0135 & 0.8285 & -0.3743 & -1.4557 & -2.0967 \\ -0.0002 & -0.0123 & 0.9149 & -1.1440 & -0.9316 & -2.7853 \\ -0.0001 & -0.0002 & 0.0275 & 0.7842 & 1.6365 & 1.8560 \\ 0.0000 & -0.0001 & -0.0019 & -0.0809 & 0.8869 & 2.4092 \\ 0.0000 & -0.0000 & 0.0001 & 0.0080 & -0.0583 & 0.8887 \end{bmatrix} \tag{C.1}$$

$$\mathbf{B} = \begin{bmatrix} 1.9790 & 0.0335 \\ -2.4107 & 0.9244 \\ 12.9892 & -0.8068 \\ 0.0178 & 0.3667 \\ 1.8671 & -0.1057 \\ -0.0295 & -0.0104 \end{bmatrix} \tag{C.2}$$

$$\mathbf{C} = \begin{bmatrix} -0.3687 & -0.4929 & 0.5447 & 0.4463 & -0.2808 & 0.1893 \end{bmatrix} \tag{C.3}$$

**Current and velocity to pitch force model**

$$\mathbf{A} = \begin{bmatrix} 0.9546 & -0.3739 & -0.1507 & -0.7514 & 0.4866 & -4.3793 \\ 0.0144 & 1.0017 & -0.4851 & 1.5654 & -0.6023 & 6.5551 \\ 0.0015 & 0.0096 & 0.8208 & -0.2425 & -0.4349 & -4.8771 \\ 0.0001 & -0.0026 & -0.0708 & 0.7185 & -0.8808 & -0.0027 \\ 0.0000 & 0.0006 & 0.0861 & 0.1456 & -0.4513 & 0.7099 \\ 0.0000 & -0.0004 & 0.0179 & -0.0745 & -0.0216 & 0.7915 \end{bmatrix} \tag{C.4}$$

$$\mathbf{B} = \begin{bmatrix} 2.7203 & -0.0934 \\ -5.1162 & 0.1742 \\ 5.0941 & -0.1705 \\ 3.5189 & -0.1129 \\ 1.7066 & -0.0474 \\ 0.3376 & -0.0111 \end{bmatrix} \cdot 10^4 \tag{C.5}$$

$$\mathbf{C} = \begin{bmatrix} -0.3860 & -0.4915 & -0.4790 & 0.3761 & -0.1281 & -0.3845 \end{bmatrix} \tag{C.6}$$

# Kinematics D

In Chapter *2: System overview*, an overview of the system available on Aalborg university was introduced. In this chapter we derive the kinematics. This is done to describe each joint and end effector positions for e.g the Geomagic Touch or the hand of the da Vinci robot.

In the following the Denavit Hartenberg (DH) notation is going to be used[39]. This eases calculation of the different positions and gives a simple , unambiguous description of the kinematics using the least amount of parameters. The DH notation only uses four parameters to describe the transformation between the different frames. Each joint is given its own coordinate frame do describe the link between the different joints and are denoted as $O_i$ from 0 to $n$. To utilize the DH notation some rules got to be fulfilled and are as followed[39]:

- DH(1): The axis $x_n$ is perpendicular to axis $z_{n-1}$
- DH(2): The axis $x_n$ intersects the axis $z_{n-1}$

furthermore the joints rotates around the z axis where the right hand rule is used.

The DH convention is described through the transformation matrix shown in *Equation: (D.1)*[39]. This matrix is calculated for each joint and gives the transformation and rotation. Note that sin() is denoted as $s$ and cos() is denoted as $c$.

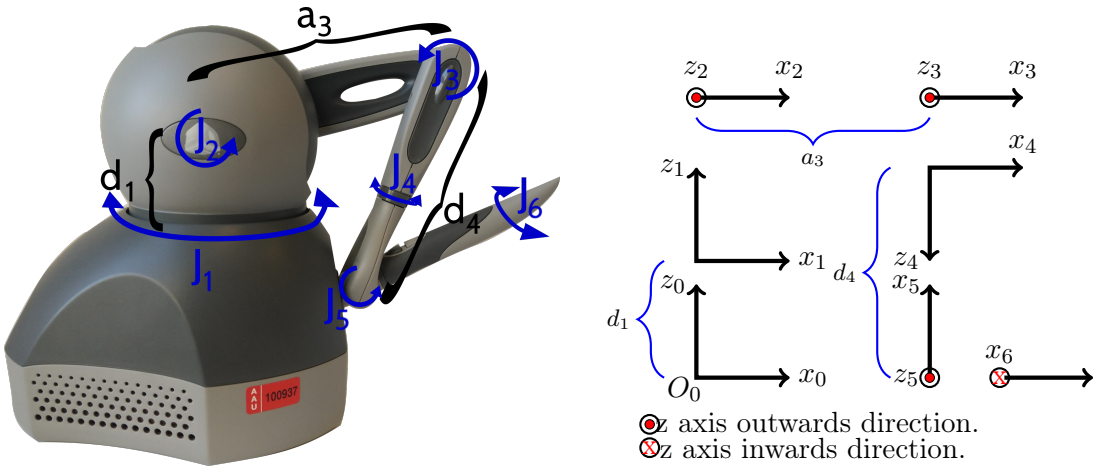
$$
\begin{aligned}
A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{D.1}
$$

Where

| | | |
|---|---|---|
| | $a_i$ | is the distance between the axes $z_i$ and $z_{i+1}$ measured along the $x_{i+1}$ axis, |
| | $\alpha_i$ | is the angle between the axis $z_i$ and $z_{i+1}$ measured in a plane normal to $x_{i+1}$, |
| | $d_i$ | is the distance between the origin $o_i$ and the intersection of $x_{i+1}$ axis measured along the $z_i$ axis, |
| and | $\theta_i$ | is the angle between $x_i$ and $x_{i+1}$ measured in a plane normal to $z_i$. |

## D.1 Kinematics for the Geomagic Touch

As shown on *Figure D.1*, the Geomagic Touch has 6 joints. The coordinate frame for these joints can be derived as shown on *Figure D.1b*. The 6 frames on the DH scheme correspond to the links on the Geomagic Touch and define their orientation and position.




(a) The Geomagic touch with defined joints numbers and the translations parameters.

(b) The different frames for the Geomagic touch.

Figure D.1: Shows joints positions, the translations DH parameters and the coordinate frame for each joint. The base frame is difined as $O_0$.

The DH notations for the Geomagic touch, can be derived as shown in *Table: D.1*.

| $j_i$ | $a_i$ | $d_i$ | $\alpha_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | $d_1$ | 0 | $q_1$ |
| 2 | 0 | 0 | $\frac{\pi}{2}$ | $q_2 + \theta_1$ |
| 3 | $a_3$ | 0 | 0 | $q_3$ |
| 4 | 0 | $d_4$ | $\frac{\pi}{2}$ | $q_4$ |
| 5 | 0 | 0 | $-\frac{\pi}{2}$ | $-\frac{\pi}{2} + q_5$ |
| 6 | 0 | 0 | $-\frac{\pi}{2}$ | $\frac{\pi}{2} + q_6$ |

Table D.1: The DH notations for the Geomagic Touch

## D.2 Kinematics for the Da vinci robot

This section is covering the derivation of one hand of the da Vinci robot.

It can be seen on *Figure D.2*, at the end of the hand there is marked a stationary point. This point is seen as the reference point for the hand. The hand has three DOF where two of them is rotations and the last one is the translation. Both rotations can be put at the stationary point due to the fact that each rotation will not translate the stationary point of the hand. The translation is handling the position of the end-effector in respect to the reference point, where the distance is marked as $d_3$ on *Figure D.2*.

The DH parameters for the da Vinci hand can be seen on *Table: D.2* as two rotations and one translation.

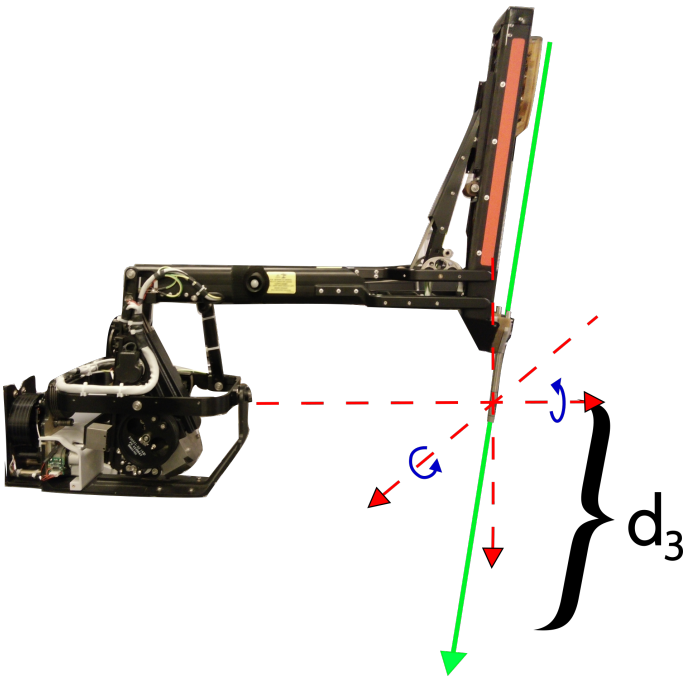



Figure D.2: The da Vinci hand. The cross section shows the stationary reference for the hand. The blue arrows illustrate the two rotations in in the reference point. The green arrow is the movement of the Endo-Wrist and $d_3$ is the distance between the reference and end-effector.

| $j_i$ | $a_i$ | $d_i$ | $\alpha_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | $-\frac{\pi}{2}$ | $q_1$ |
| 2 | 0 | 0 | $\frac{\pi}{2}$ | $\frac{\pi}{2} + q_2$ |
| 3 | 0 | $d_3$ | 0 | 0 |

Table D.2: The DH notations for the da Vinci hand.

# Measurement E

In this chapter a description of the data collected for the model estimation is made. Each section contains a description of how each measurement was made, a list of the hardware that was used and a graph of some of the data collected. Each graph does not contain all the measured data but enough to see a pattern.

## E.1 Force measurements for the end-effector

As the Endowrist tool is highly nonlinear, measurement of how the end-effector responds to external force is made. These measurements are made for the model estimation of the EndoWrist.

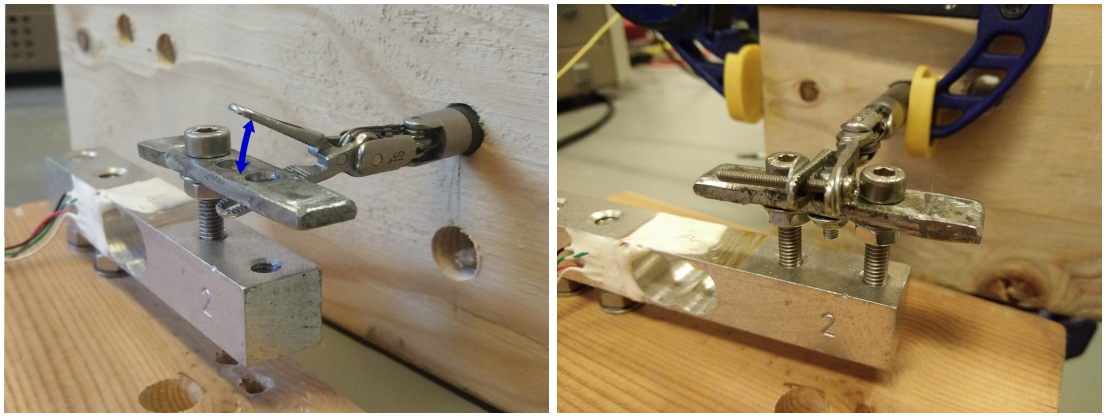
### Purpose:

The purpose is to define a dynamic model for the EndoWrist.

The measurements are done on three different setups:

- Grip force of the clamps (yaw),
- Down/upwards force of the pitch on the end-effector (pitch),
- and the rotation of the end-effector (roll).

In the following subsections the different tests defined in the list above are made. For making the force measurements, item one to three, different attachment to a load-cell are made. These can be seen on *Figure E.1*.



(a) Upper clamp which is allowed to add or release force applied to the load-cell.

(b) The end-effectors two clamps are attached to the load-cell, such that the pitch force can be measured.

Figure E.1: Attachments between the end-effector and the load-cell used for force measurements

### Electrical circuit

For making the force measurement for the EndoWrist a simple hardware construction was build. A simple block diagram of setup can be seen on *Figure E.2*

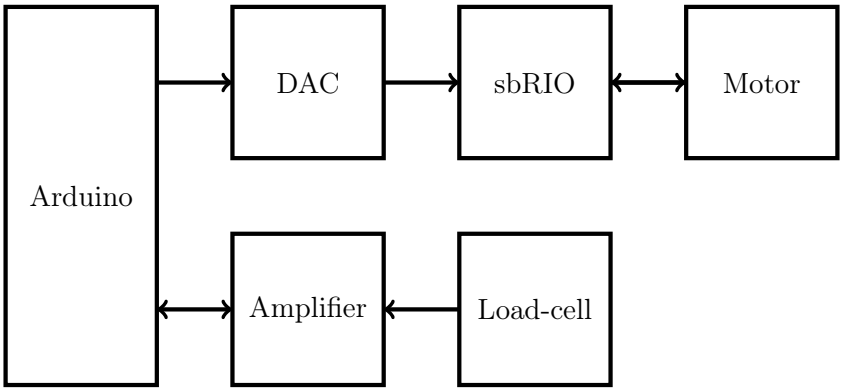



Figure E.2: Diagram of the test setup for the force measurements.

The load-cell's output is amplified and send to an Arduino which calculate the force applied to the load-cell. The force is then translated to a corresponding voltage output, $mV \cdot \frac{g}{4 \cdot mV}$. Since the Arduino does not have a digital to analog converter (DAC), the voltage is translated to a binary code and send to a "Max531 DAC" through a serial peripheral interface (SPI). The voltage output from the DAC is read at the sbRIO board which also samples the current send to the motor, motor position and time stamps.

As explained, an Arduino is used for the translation of force applied to the load-cell. This was done since a library for the HX711 amplifier was made for the Arduino IDE[40]. This library included features such as calibration, reset and averaging of the load-cell.

### E.1.1 Grip force

This test uses the test setup seen on *Figure E.1a*. The positive force is defined as downwards direction on the load-cell, see *Figure E.3*.

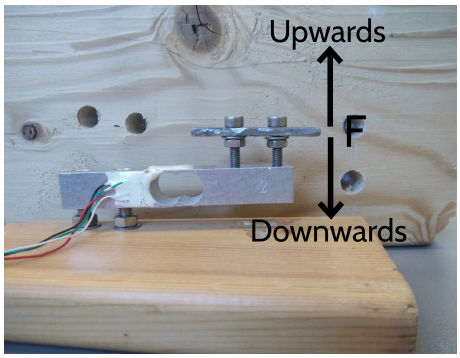



Figure E.3: Load-cell with defined upwards- downwars force.

### Test equipment:

- Endowrist model 420093 (AAU number: #4).
- Maxon 110160 motor with attached Maxon gearhead 110356 and Maxon encoder 201937.

- Load cell rate for 1 kg of force [41].
- HX711 - Load cell amplifier [42].
- Arduino uno with Max351 DAC.
- sbRIO board.

### Procedure:

The following procedures was made for the dynamic measurements of one clamp on the EndoWrist.

1. One clamp is enabled and put above the load-cell as seen on *Figure E.1a*.
2. The load cell is set to zero.
3. A triangular wave of setpoints are send to the motor controlling the upper clamp, which make the clamp moves up and down.
4. Current, force and velocity are sampled.

Each test is running for a short period of time, since the input setpoints are periodic. This test was done 11 times.

### Measuring data:

The data form one measurement is shown on *Figure E.4*.

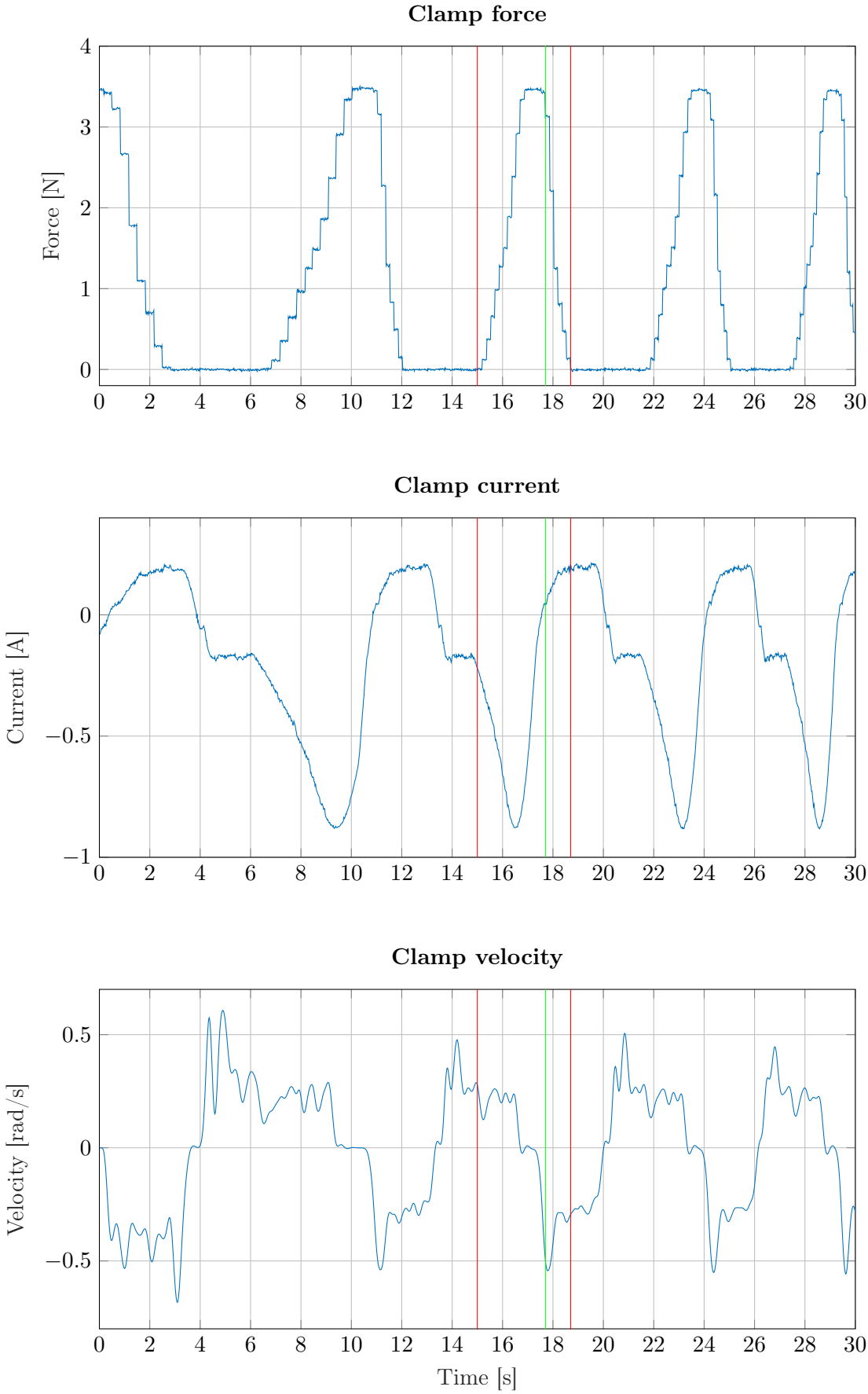



Figure E.4: One set of data that shows the force, current and velocity. The interval between the red lines shows one period of contact with the load cell and the green is the release of the force applied.

### Results:

From *Figure E.4*, one of the measurements for the grip force estimation is shown. The first red line marks one of the starting periods, where contact between the clamp and the load-cell is happening. It can be seen that a current increase is happening in when the load-cell is applied force. When the current decreases the force is kept until the sign on the current has changed and force is applied in an upwards direction of the clamp.

### Uncertainties of measurement:

- The force applied is not perfectly orthogonal force to the load cell.
- Input/Output impedance of sensors have a ±10% tolerance.
- Movement of test setup when force is generated.
- The pitch joint of the EndoWrist had a tendency to bend when high force was applied which change the force direction.

### Conclusion:

It can be seen that there is a high friction in mechanical part that handles the clamp. Due to this friction, the tool is able to withhold a force when the current is decreasing.

## E.1.2 Pitch down and upwards force

This test uses the test setup seen on *Figure E.1b*. The positive force is defined as a downwards direction on the load-cell, see *Figure E.5*.

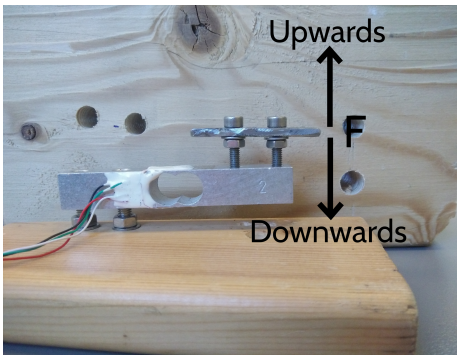



Figure E.5: Load-cell with defined upwards- downwars force.

### Test equipment:

- Endowrist model 420093 (AAU number: #4).
- Maxon 110160 motor with attached Maxon gearhead 110356 and Maxon encoder 201937.
- Load cell rate for 1 kg of force [41].
- HX711 - Load cell amplifier [42].
- Arduino uno with Max351 DAC.
- sbRIO board.

### Procedure:

The following procedure was made:
Downwards force measurements:

1. The end-effector is attached perpendicular to the load cell.
2. The load cell is reset to zero.
3. Current is applied to the motor which control the pitch of the end-effector, at different current levels and the force is measured (downwards direction).
4. Current is increased until 680 mA is applied.

Step two to four is repeated five times, where the current and force is measured in respect to each other.

Upwards force measurements:

1. The end-effector is attached perpendicular to the load cell.
2. The load cell is reset to zero.
3. Current is applied to the motor which control the pitch of the end-effector, at different current levels and the force is measured (upwards direction).
4. Current is increased until 680 mA is applied.

Step two to four is repeated five times, where the current and force is measured in respect to each other.

### Measuring data:

Six of the data measurements can be seen on *Figure E.6* and *Figure E.7*.

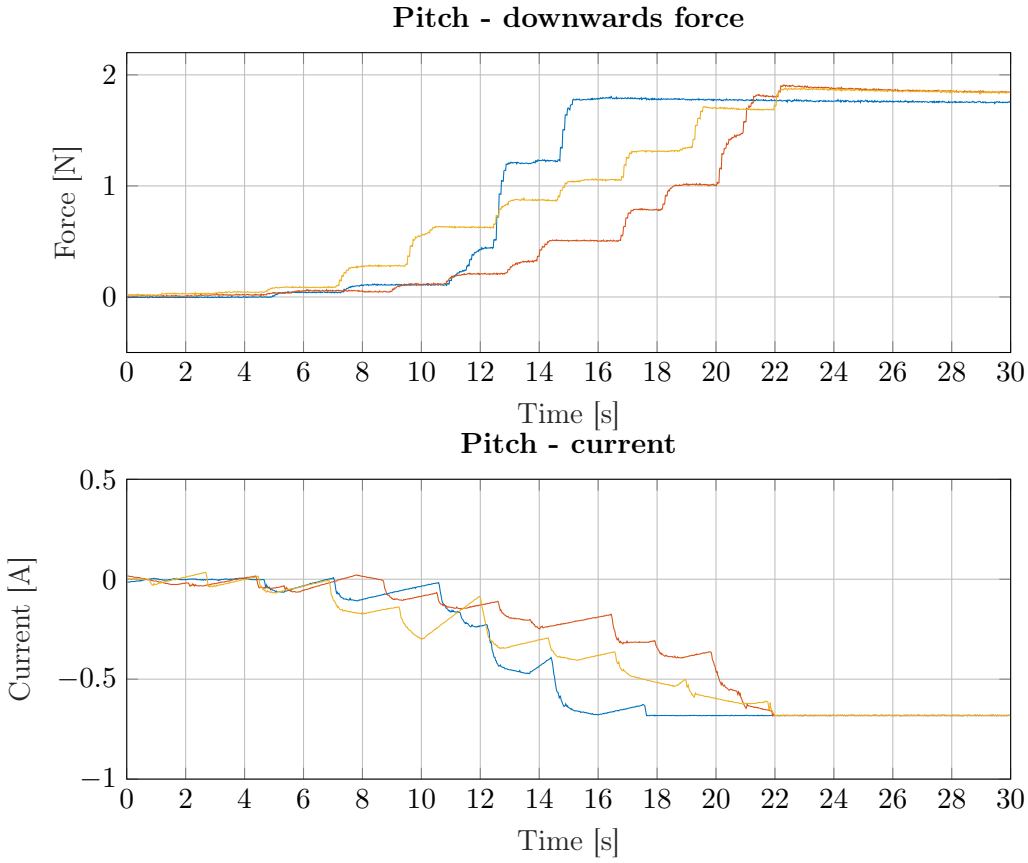



Figure E.6: Force measurements for the pitch in an downwards direction.

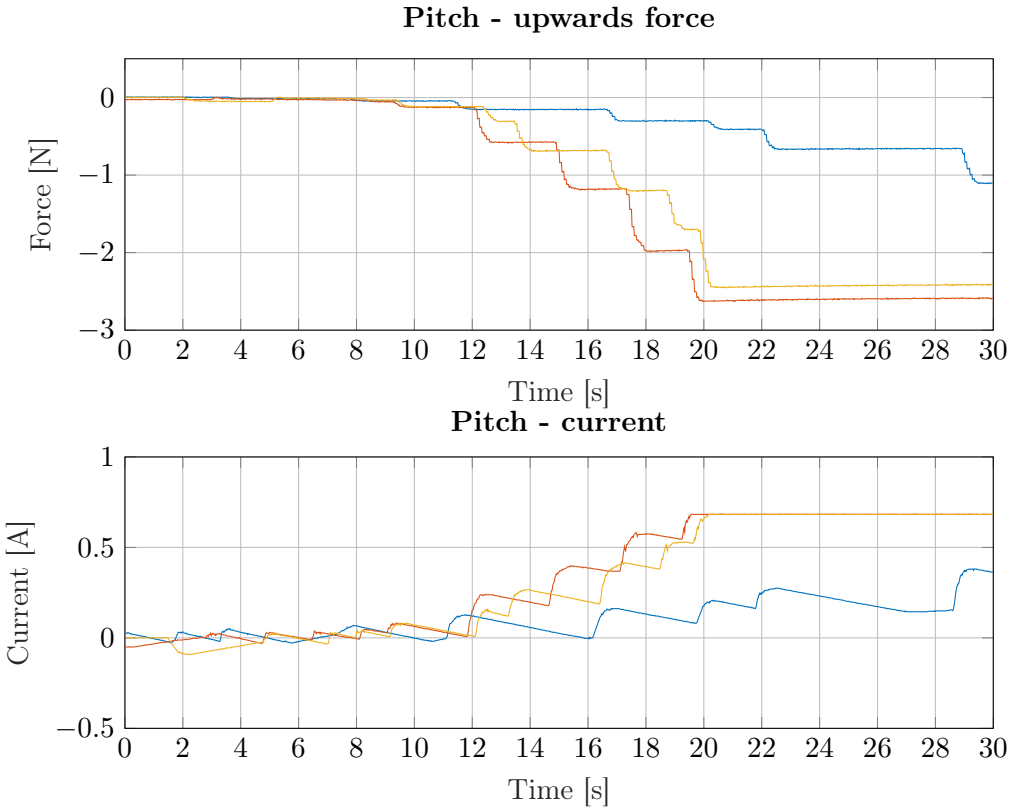



Figure E.7: Force measurements for the pitch in an upwards direction.

### Results:

From *Figure E.6* and *Figure E.7* a similar pattern between the measurements can be seen. From each measurement it can be seen that the force growth for the different measurements are similar. Force is generated on the end-effector from almost the same current start point on both downwards and upwards measurements. For downwards the force is applied between -131 mA to -148 mA and upwards between 151 mA to 191 mA. It can be seen that the current is decreasing for each measurements. This is due to the controller implemented on the motor controller as it gets a setpoint and current is applied until the setpoint is reached and then decreased.

### Uncertainties of measurement:

- The force applied is not perfectly orthogonal force to the load cell.
- Input/Output impedance of load-cell sensors have a ±10% tolerance.
- Movement of test setup when force is generated.

### Conclusion:

Force is generated at the end-effector when a current higher than -131 mA for downwards force and 151 mA for upwards.

## E.1.3 EndoWrist roll measurement

### Test equipment:

- Endowrist model 420093 (AAU number: #4).

- Maxon 110160 motor with attached Maxon gearhead 110356 and Maxon encoder 201937.
- Torque sensor - Holger clasen: MWA-W8-1-P (AAU number: LBNR 08931).
- Active low pass filter set to 100 Hz and gain 1 (AAU number: C2-104-H1).
- Agilent 54621A oscilloscope (AAU number: 56684)

### Procedure:

The following procedure was made for the force measurements:

1. The carbon stick of the EndoWrist is attached to the torque sensor.
2. The torque sensor is connected to the low pass filter and the low pass filter to the oscilloscope.
3. Current is applied or increased to the motor which control the roll of the end-effector, with different current steps and the output from the torque sensor is noted.
4. Current is increased until 1200 mA is applied.

Step three and four are repeated four times, where the current and torque is noted in respect to each other..

### Measuring data:

The data from the measurements can be seen on *Figure E.8*

### Results:

The measurements shows a similar pattern for each measurements and the torque generated starts around 30 mA to 60 mA.

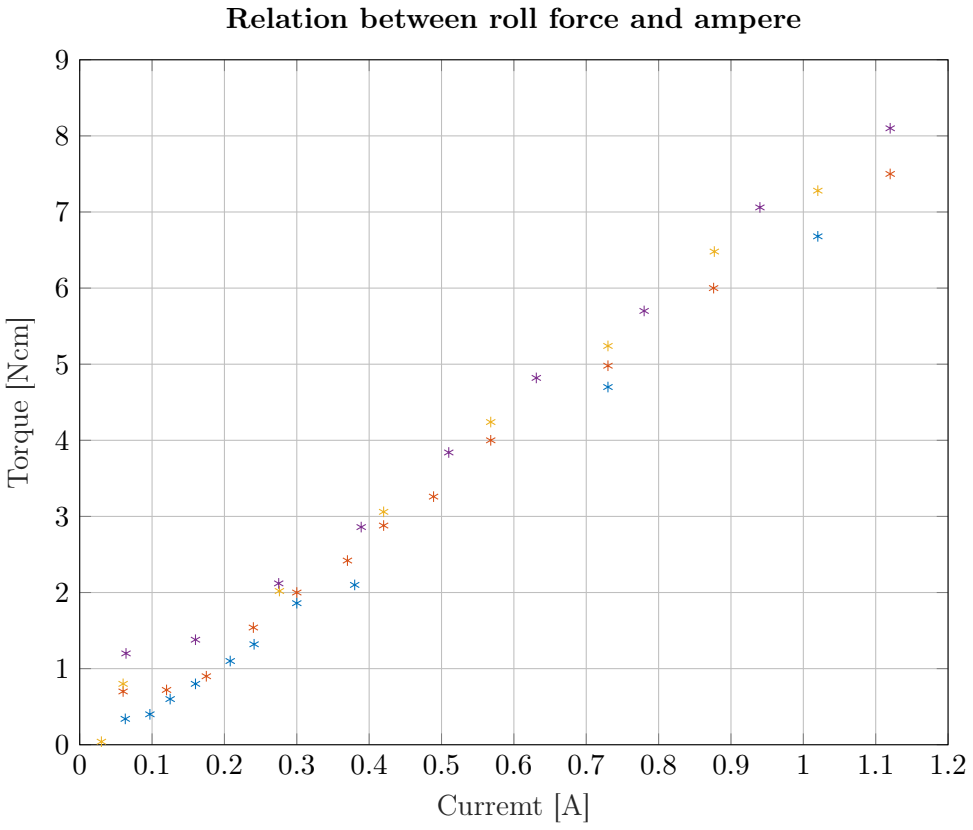



Figure E.8: Torque measurements of the roll on the EndoWrist.

**Uncertainties of measurement:**

- Gain of low pass filter could deviate from 1.
- Noise did still appear after the filter and could affect the output reading.

**Conclusion:**

It can be seen on figure *Figure E.8* that the roll-torque of the EndoWrist has a linear growth from 100 mA to 120 mA and up.

## E.2 Communication

In this section, the measurements made to evaluate the performance of the communication are presented. For each measurements, the procedure is described as well as the equipment used and an interpretation of the results obtained.

### E.2.1 TCP measurements

**Purpose:**

The purpose is to determine the maximum frequency that can be used with the TCP/IP protocol. To do so, Wireshark is used to record information about the traffic. The values that are necessary for analyzing the connection are the delay between two packets received or sent and from that, the actual frequency as it may differ from the value programmed, the jitter and the packet loss.

**Test equipment:**

The sbRIO 9636 is connected to a computer running Ubuntu 14.04 LTS via an Ethernet cable of 1 gpbs. The computer runs the ROS nodes that are used for the project, thus it communicates with both the Geomagic Touch and the sbRIO and it sends random setpoints to the sbRIO. The sbRIO uses the entire code of the project.

**Procedure:**

1. The sbRIO is started and connected to the computer.
2. The ROS node is started.
3. Wireshark is started.
4. Once the recording is done the ROS node is stopped. The recording time here was 20 seconds.
5. The log files are retrieved to be analyzed using Matlab and Wireshark.

Those steps are repeated for the following settings: 100Hz, 500Hz and 1000Hz.

**Measuring data:**

**Results:**

When the frequency is increased above 100Hz, instead of sending more packets it sends bigger packets in which two JSON files or more are grouped. As shown on *Figure E.9*, *Figure E.10*, *Figure E.11*, the more the frequency is increased, the bigger the packets.

| No. | Time | Source | Destination | Protocol | Length |
|---|---|---|---|---|---|
| 1 | 0.000000000 | 169.254.36.85 | 169.254.4.42 | TCP | 127 |
| 2 | 0.000312000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |
| 3 | 0.010882000 | 169.254.36.85 | 169.254.4.42 | TCP | 127 |
| 4 | 0.011192000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |
| 5 | 0.021253000 | 169.254.36.85 | 169.254.4.42 | TCP | 127 |
| 6 | 0.021546000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |
| 7 | 0.031543000 | 169.254.36.85 | 169.254.4.42 | TCP | 127 |
| 8 | 0.031835000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |
| 9 | 0.041917000 | 169.254.36.85 | 169.254.4.42 | TCP | 127 |
| 10 | 0.042206000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |

Figure E.9: Wireshark recordings for 100Hz

| No. | Time | Source | Destination | Protocol | Length |
|---|---|---|---|---|---|
| 1 | 0.000000000 | 169.254.4.42 | 169.254.36.85 | TCP | 324 |
| 2 | 0.000013000 | 169.254.36.85 | 169.254.4.42 | TCP | 371 |
| 3 | 0.004866000 | 169.254.4.42 | 169.254.36.85 | TCP | 66 |
| 4 | 0.004877000 | 169.254.36.85 | 169.254.4.42 | TCP | 188 |
| 5 | 0.009941000 | 169.254.4.42 | 169.254.36.85 | TCP | 323 |
| 6 | 0.009956000 | 169.254.36.85 | 169.254.4.42 | TCP | 249 |
| 7 | 0.020044000 | 169.254.4.42 | 169.254.36.85 | TCP | 324 |
| 8 | 0.020065000 | 169.254.36.85 | 169.254.4.42 | TCP | 310 |
| 9 | 0.029902000 | 169.254.4.42 | 169.254.36.85 | TCP | 321 |
| 10 | 0.029912000 | 169.254.36.85 | 169.254.4.42 | TCP | 371 |

Figure E.10: Wireshark recordings for 500Hz

| No. | Time | Source | Destination | Protocol | Length |
|---|---|---|---|---|---|
| 47 | 0.087359000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |
| 48 | 0.087373000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 49 | 0.087377000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 50 | 0.089953000 | 169.254.4.42 | 169.254.36.85 | TCP | 66 |
| 51 | 0.089957000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 52 | 0.089960000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 53 | 0.093514000 | 169.254.4.42 | 169.254.36.85 | TCP | 66 |
| 54 | 0.093519000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 55 | 0.093522000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 56 | 0.099403000 | 169.254.4.42 | 169.254.36.85 | TCP | 66 |
| 57 | 0.099410000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 58 | 0.099413000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 59 | 0.103133000 | 169.254.4.42 | 169.254.36.85 | TCP | 66 |
| 60 | 0.103140000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 61 | 0.103143000 | 169.254.36.85 | 169.254.4.42 | TCP | 1514 |
| 62 | 0.116679000 | 169.254.4.42 | 169.254.36.85 | TCP | 327 |

Figure E.11: Wireshark recordings for 1000Hz

### Conclusion:

The communication does not have the desired behavior when the frequency is increased. Thus the jitter and packet loss cannot be used to evaluate the performance. This may be due to the default behavior of TCP or to the default behavior of the JSON Stream. It was decided not to look further into this subject as this problem does not appear in the new communication protocol since it does not implement any connection or stream.

### E.2.2 UDP measurements

#### Purpose:

The purpose is to determine the maximum frequency that can be used in our system with the UDP protocol. To do so, we will measure the delay between to received packets and jitter and look at those values as a function of the delay used in the sending loop.

#### Test equipment:

The sbRIO 9636 is connected to a computer running Ubuntu 14.04 LTS via an Ethernet cable of 1 gpbs. The computer runs the ROS nodes that are used for the project, thus it communicates with both the Geomagic Touch and the sbRIO. The sbRIO as well runs the entire code of the project.

#### Procedure:

1. The sbRIO is started and connected to the computer.
2. The ROS node is started.
3. Wireshark is started.
4. Once the Wireshark recording is done the ROS node is stopped.
5. The log files are retrieved to be analyzed using Matlab and Wireshark.

Those steps are repeated for both the serial and parallel case for the following settings: 100 Hz, 500 Hz and 1000 Hz. These settings do not account for computation time and the period of waiting to receive a packet, thus it is expected that the real frequency is lower than the one set.

#### Measuring data:

The data resulting from the measurements of the designed communication protocol are gathered in *Table: E.1.*

| Frequency (Hz) | delay (ms) | Jitter ($\mu$s) | Packet loss (%) |
|---|---|---|---|
| 99 | 10.1 | 4.66E-2 | 0 |
| 474 | 2.1 | 5.51E-2 | 0.2 |
| 638 | 1.6 | 1.16E-2 | 1.2 |

Table E.1: Measurements of the performances for the new communication protocol

#### Results:

From a frequency of 99 Hz to 474 Hz, it can be seen that the jitter as well as the packet loss increase. As both of them are mainly consequences of congestion on the network[22] this phenomenon was expected. However, when the frequency is increased to 638 Hz, the jitter decreases and the packet loss keep increasing. The cause is that the jitter increases but the driver only waits 1 ms to receive a packet from the sbRIO after sending and thus rejects the packets coming after that deadline. The packet loss could be reduced by increasing the waiting time but that would reduce the frequency. A trade-off between frequency and error rate has to be made. The data for the maximum frequency of the driver could not be collected as the congestion on the network becomes too important. When that happens the communication does not behave as intended and each device send more than one packet in a row.

### Conclusion:

The communication cannot reach frequencies higher than 638 Hz due to congestion which does not meet the requirements. However by implementing the new communication protocol the refresh was increased from 100 Hz to 638 Hz. Solutions to further improve the refresh rate are discussed in Chapter *6: Results and Discussion*

# Definitions and explanations F

This chapter is created to give a more clear worksheet for the reader and separate explanations for e.g different protocol.

## F.1 TCP/IP

TCP/IP is a connection orientated protocol, which first establish the connection between two devices[43]. After the connection has been established data can be send in both directions. TCP/IP is a reliable protocol which ensure that the data transmitted, will be received. When the data is send, the transmitter waits for an acknowledgment from the receiver that the data has been received. If the data has been corrupted or the acknowledgment is not send, the transmitter will retransmit the package again. If the case of no acknowledgment is happening, a timer is used to define when to retransmit the data again. The data will then be retransmitted a certain amount of times until an acknowledgment has been received or the connection is defined to be disconnected. This means that a copy of the data is stored at the transmitter until it has been correctly received or the connection is taken down.
This communication protocol induces a delays in the system as the transmitter has to wait on the acknowledgment that the data has been received before loading new data into the sending buffer. If data has been discarded due to error the transmitter has to retransmit the data again if necessary.

## F.2 User datagram protocol

UDP communication protocol is a faster way of communicating compared to TCP/IP[43]. It is connectionless which means that it wont establish a connection before sending data. When data is transmitted, new data is loaded into the transmitter buffer and then send when the connection line is free, thus the data is not stored at the transmitter as in TCP/IP. Furthermore the UDP protocol does not wait for an acknowledgment and thereby just sends the data as fast as possible to the specified address.

## F.3 Robotic operating system

The Robotic operating system (ROS) is an open source software development tool for implementing robotics software. It provides the opportunity of hardware abstraction, low level device control, implementation of commonly used functionalities, messages between different processes and package management[44]. It provide tools and libraries which utilize the the opportunity of communicating between disturbed computers, obtaining, writing and running codes.

ROS has three different levels of concepts[45]

- **The file system level**

Handles the main unit for a ROS system which is packages. A package may include data sets, ROS dependent libraries, configure files etc. to define a ROS process. In ROS a process is denoted as a node.

- **The computation graph level**

  Handles the communication of the peer to peer network of the system in which data is processed. Through the computation graph level, the different nodes can communicate with each other by messages. When a node is sending data it is said to be publishing a topic. The different nodes can then subscribe to this topic to get the information that is published.

- **The Community level**

  ROS has a huge community which contain distribution of software installations, repositories and documentation of ROS. It also has a question and answer section with ROS related topics.
  This community makes the process of learning the system considerably easier.