The design and evaluation of a human in the loop control system for the McRobot 3.1

R. (Robert) van der Wal

BSc Report

Committee:
Ir. F.S. Farimani
Dr.ir. J.F. Broenink
Dr.ir. J.R. Buitenweg
V. Groenhuis, MSc

August 2016

027RAM2016
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

This research report presents how a human in the loop control system can be created for the McRobot using the Omega 6 haptic joystick. In the medical field there is a need of robotic systems that are able to perform operations real-time in the MRI. The McRobot is an attempt to make a MRI-compatible robot, controlled by the Omega 6 joystick. A literature research was performed to learn the state of the art and gather ideas for possible control loop concepts. Some concepts have been chosen and tested with several experiments to build the control loop system. Eventually concept 2 (Equal load of duty system) is chosen to design the final control loop system for the McRobot. From the gathered results can be concluded that the designed system is working. At last, there are some recommendations for the McRobot that will improve the total system.

Keywords

McRobot, Omega 6 joystick, Human in the loop control system

Contents

List of Figures

List of Tables

Abbreviations

DoF	Degrees of Freedom
McRobot	MRI Compatible Robot
MRI	Magnetic Resonance Imaging
MII	Minimally Invasive Intervention
MIS	Minimally Invasive Surgery
NMR	Nuclear Magnetic Resonance
OS	Open Surgery
RaM	Robotics and Mechatronics
RF	Radio Frequency
ROS	Robotic Operating System
RSS	Robotic Surgical System
SRS	Surgical Robotic System

Chapter **1**

Introduction

In recent years robotics have found a fast increase in replacement of modern techniques [1]. The rise of robotics has a huge impact on the surgical operation procedure. A surgeon with a robotic surgical system (RSS) can make a more precise incision compared to the manual hand [2]. The RSS can consist of force, haptic and imaging feedback systems that help the surgeon to make a precise incision. The recovery time and complications after the surgical operation has been greatly reduced with the use of robotic (assisting) systems [3].

One of the biggest challenges for robotic systems is the distrust that humans have in robotic systems [2]. Most of the operations are still done with the surgical hand instead of using a surgical robotic system. These operations are likely to be open surgery operations. The patient takes a lot of damage to the body when performing an open surgery.

In this research, there will be made an control loop system to include a human operator in the magnetic resonance imaging compatible robot (McRobot). Literature research has been conducted in order to design the McRobot control loop system. The system consists of several different subdevices. A human interface is one of the required subdevices for designing a human operator control system. To solve this requirment the Omega 6 can be used as a high advanced joystick manipulator. The coding of the control systems will be done using the platforms ROS (C++) and Arduino (modified C/C++). A large amount of coding work with applying filters and kinematics will be done in ROS. The Arduino microcontroller will drive the actuators to the designated orientation coordinates of the McRobot. The requirments of the control system are based on literature research and the design of the McRobot. Consequently the design of the global and local subsystems will be made for assembly of the prototype. Further, the system will be tested on functionality and design. Any other improvements that are not done in this research report will be referred in recommendations.

1.1 Problem statement

The main question during the bachelor assignment is the following one:
Is it possible to assemble/create a McRobot human in the loop control system using Omega 6 joystick, ROS and an Arduino Mega microcontroller?

Minimally invasive systems are used to inflict minimal damage to the patients body. Robotic controlled surgery is used for higher precision. Combined, we have minimally invasive robotic systems (MIRS). The MIRS is controlled by a supervisor that uses the information from the imaging systems for navigating the robotic tools. Magnetic resonance imaging has a very high resolution. However, the problem with using the MRI system for surgical robotic systems (SRS) is that it cannot be performed real-time at the moment. The real-time connection with imaging systems is important for compensation movements of the SRS of moving organs. The McRobot is one of the first compatible SRS for real-time MRI monitoring control system. The McRobot is designed for performing minimally invasive intervention (MII) to inflict almost no damage to the body of the patient.

In this research the control system of the McRobot 3.1 will be made to make a human in the loop control system. The connection between human interface and McRobot needs to be as real-time as possible.

Chapter 2

Literature research

In this part of the research report some information is given of the used operating systems and devices. The literature is used to design a human in the loop control system for the McRobot. Other basic information is expected to be known by the reader for understanding the principles of a loop control system.

2.1 Omega 6 haptic joystick

One of the most accurate surgical joystick on the market is the Omega 6 from the Swiss brand "Force Dimension" (see figure 2.1) [4]. This joystick has a high-precision for 6 degrees of freedom kinematics interface what gives a high complexity of different movements. The control system consists of a counterbalanced encoding - and motor kinematic system, what gives the user more stability in placing the pen-shaped end-effector in the designated place. However, the mobility resolution of the omega 6 system consists of a translation- of 0.01 mm and a rotation resolution of 0.09 degrees. The electronic part of the controller consists of a multithreaded system with a refresh rate of 8 kHz [5].

Furthermore, the Omega 6 uses a library and example package. This package is coded in the language C++ and includes demo's for testing the Omega 6 joystick. These demo's got some important code information that will be used for further kinematic calculations of the Omega 6 [5].

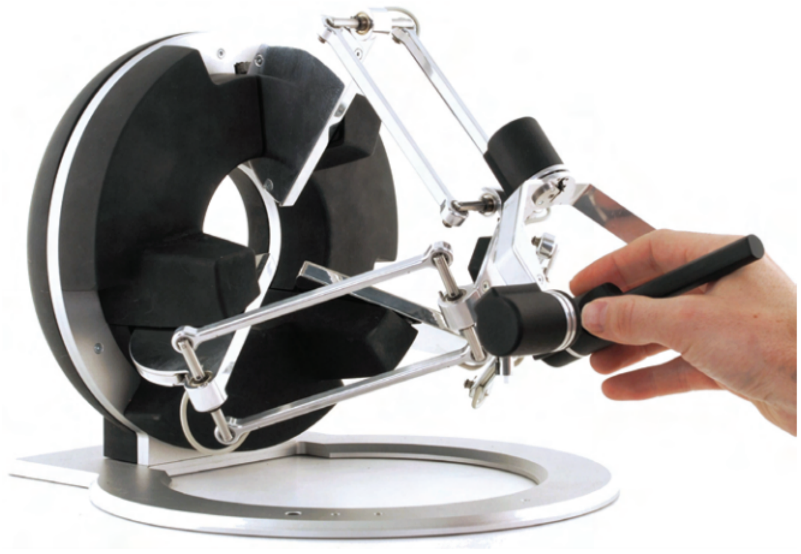


Figure 2.1: *The Omega 6 haptic joystick with 6 degrees of freedom [4].*

The haptic joysticks of the brand "Force Dimensions" are widely used for different actions. For example the Omega 6 can be used for space shuttle driving to apply gravity and acceleration compensation force. This keeps the joystick in the middle of the console interface except when the human operator applies extra force on the joystick [6]. In the medical field there is a large use of the Omega 6 joystick for research or surgical operations. [7].

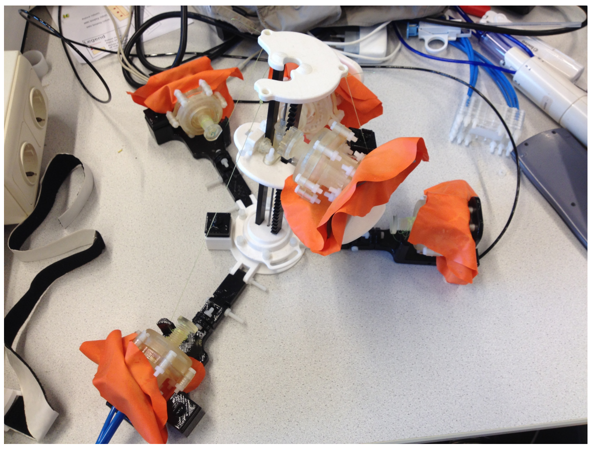
There are a lot of other haptic human surgery interfaces that are being used for MIS. Including one of these devices is the phantom desktop interface and is made by brand "Geomagic" that is used for surgical operations or simulations [8]. However the Omega 6 has a more precise encoder system than other surgical human interface systems [4].

2.2 McRobot 3.1

One of the biggest challenges of the medical robotic branch is to design new systems for minimally invasive interventions of cancer. At the group Robotics and Mechatronics (RaM) on the University of Twente research has been done to make an "image guided tele-manipulated system for minimally invasive interventions" what is called the McRobot (MRI compatible tele-manipulated system) see figure 2.2a [9]. This operating system uses imaging systems to construct a 3D picture for locating the designated target. The purpose of the robotic system is to locate with the imaging subsystems the cancer location and to guide the needle by a surgeon. From the MRI images information is gathered for locating the position of the organs and how the McRobot needs to position its needle for punctuation of the skin. This research will adress the possibility to use the Omega 6 haptic joystick as a human interface for controlling the McRobot.

The McRobot consist of components that are MRI compatible (see figure 2.2b). Most normal used actuators have dia/ferro/para-magnetic parts that are attracted by the MRI. However, this problem is solved by making all of the components of the McRobot from polymers. The actuator is the most difficult component to build from polymers. This will be further discussed in section 2.4.

The design of the McRobot control system is not very complex. In fact the controlling system consist of a forward kinematic scheme with calculation of the desired wire length. These coordinates are extracted from the Omega 6 pen-shaped endeffector orientation.


(a) *The made McRobot 3.1*

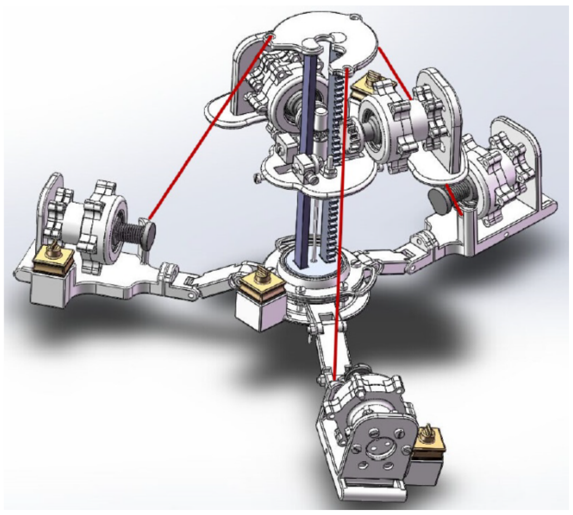
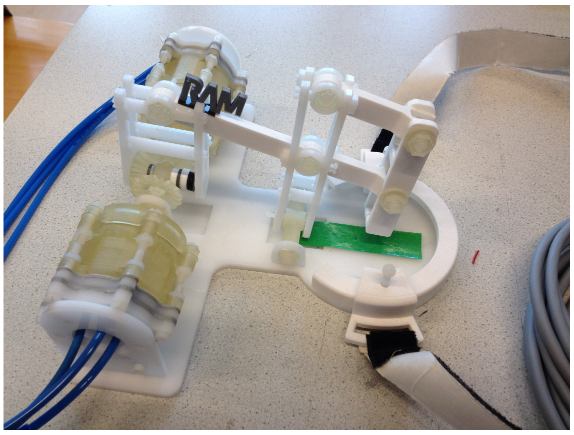

(b) *The Solidworks model of the McRobot 3.1 [9].*

Figure 2.2: *The McRobot 3.1 mechanical system*

2.3 McRobot 3.2

McRobot 3.2 is the second made MRI compatible SRS at the University of Twente and is designed by Dinah Kleiboer [10]. The parallel manipulator structure of McRobot 3.2 allows orientation movements of the needle around the needle insertion opening (see figure 2.3a). The system uses the newest actuators for turning the system [10]. These actuators can resist more pressure on the pistons and are better air sealed than the ones made before. Two air pressure actuators are used for turning the system around the x or y axis. The actuator can turn the system forward or backward around the x axis and with the y axis for turning the system left or right. Further the actuator movement resolution in the McRobot system is 11.02 degrees. However, the used McRobot 3.2 has a problem in the kinematic configuration what leads to impaired movements diagnolly in contrast with vertical and horizontal movements. To solve this issue, one of the two actuators needs to be placed behind the gear wheel for having a more solid structure (see figure 2.3b). The MRI compatible robot will be used and tested for designing a second controlling system.



(a) *The made McRobot 3.2 with the wrong actuator positions*

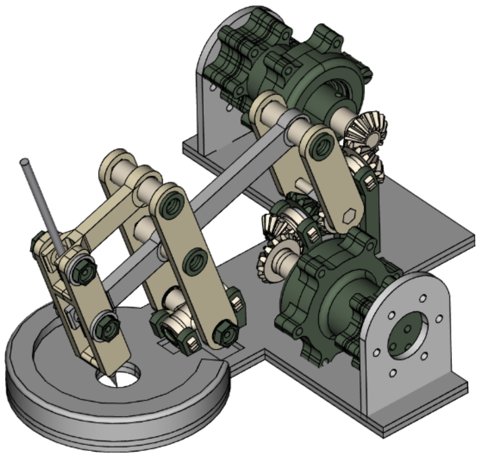

(b) *The Solidworks drawing of the McRobot 3.2 with the correct position of the actuators [10].*

Figure 2.3: *The McRobot 3.2 mechanical system*

2.4 Actuator

Actuators are precise motor systems that can change the direction according to the piston that is pressed. The design of the actuator of the McRobot is done in the article of Xiaolong (see figure 2.4)[9]. One of the biggest challenges of the McRobot actuator was to create an actuator that is compatible with an operating MRI. The MRI has a strong magnetic field that attracts all dia-, ferro- and paramagnetic materials in the surrounding area. Almost every actuator works with an electrical system to steer and adjust the motor direction. This electrical system consists of multiple metals that are attracted and can be influenced by the magnetic field of the MRI. For the McRobot an actuator needs to be designed with non-magnetic materials. The material of the actuators exist of a polymer that is non-magnetic and can be used in the MRI. Further the actuator mechanics works on an air pressure mechanism that can move the teeth inside the actuator to cause a rotation. In the group RaM multiple members are doing research on finding a better actuator system to replace the existing actuator. The first actuator system has some problems with precision after some turns and has no encoder which is a must for making a robotic control loop system. Other system components of the actuator will be discussed in required subcomponents.

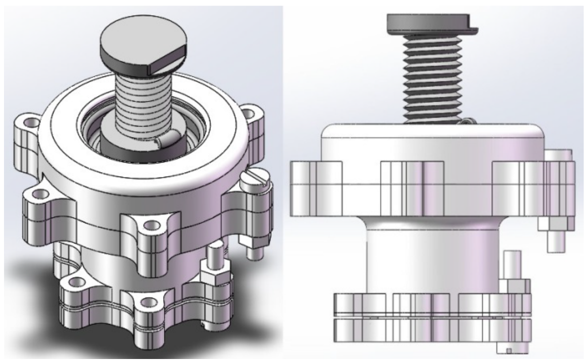


Figure 2.4: *The design of the first made MRI compatible actuator for the McRobot 3.1 [9].*

2.5 Kinematics

Complex robotic joint systems make use of mathematical equation to calculate the required joint angles for positioning the end-effector. For positioning the McRobot in the correct orientation that is given by the Omega 6 joystick a kinematic library is used. For understanding the principles of kinematics a short summary is given.

First of all there are two types of important dimensions that are used for kinematics: cartesian space and joint space. Cartesian space (desired end-effector position) can be transformed to joint space and vice- versa. This gives the information that Cartesian space uses 3D positioning coordinates and the Joint space uses joint angles. The mathematical method to translate Cartesian space to the joint space is called "Inverse kinematics" (see figure 2.5)[11]. This process is very computational expensive and needs a lot of calculations to find the right position angles of all the body frames of the interface manipulator. Some interface controllers have a positioning 3D part that only needs to calculate the position of a single joint instead of multiple body frames. This technique of 3D control has a huge benefit in time reduction by less amount of difficult mathematical equations. This method is also used with the Omega 6 coordinates. To reverse the created space into movements of the end-effector, there must be made a kinematic model

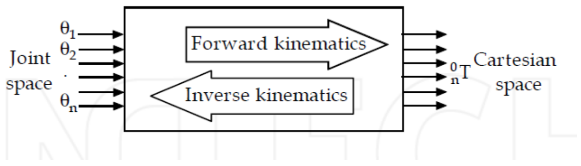


Figure 2.5: *A global kinematic transformation between cartesian - and joint space [12].*

of the McRobot. This process to calculate the positioning of the body frames with respect to a point in the 3D space is called "Forward Kinematics" [12]. In contrast to inverse kinematics it uses a less complex mathematical calculation for the controlling system to translate the point in the joint space.

2.6 Ubuntu (Linux)

Ubuntu is a modified platform of the operating system distribution Linux. This operating system is commonly used, because it is free for users. The system contains some free software applications that are easily in use. It also can cooperate more easily with other operating systems like ROS. The operating system can be changed using the terminal. It is a coder friendly operating system that gives full access to all directories [13].

2.7 Robotic Operating System (ROS)

In the past couple of years a rise of codes for robotic systems is found. A big problem in the robotic development is the sharing system of existing codes for robotics. Many codes are written for the same purpose and application what is a waste of effort and time to create the same code over and over. The "Robotic Operating System" (ROS) changes the way of code sharing by using packages that can be downloaded and directly be used [14]. Furthermore the software package works on multiple operating platforms, but prefers the operating system Ubuntu (Linux). Another feature of ROS is that the codes can be either written in C++ or in Python with given commands in the ROS environment. A recommendation for beginners is to read the ROS manual. This help to create programs and learn to run them. There have been made several documents to help setup the ROS environment and how to work with the operating system. These documents can be found in the report files directory of the zip file.

The ROS system operates on the computer and can be controlled with terminal commands. ROS has many advantages over several other robotic software distributions. One of the biggest advantages that has already been mentioned is the sharing system with other ROS users. The platform has some standard libraries that helps to setup connection between a joystick and an output motor with using several nodes. Nodes are processing units for computation of several tasks. Two important node tasks are publishers and subscribers. These nodes can share information between different running systems. For example from the Omega 6 haptic joystick is information received by making a subscriber to the messages that the Omega 6 is sending. The message system of ROS can contain different sort of information. ROS has included message packages that are needed to write multiple data strings on one message.

ROS is very nice system for connecting subdevices for assembly of a robotic system. The system is also used at the moment for different RSS [15]. Many surgical systems are already implented in the ROS environment what makes the assembly of a RSS easier.

2.8 Magnetic Resonance Imaging (MRI)

Recently surgical robotic systems has found an increase in using medical imaging systems for supporting MIS [16]. One of the most promising imaging technique for linking to robotic systems is MRI. It is a safe and precise way to image the body of a patient. The MRI produces detailed computer generated pictures of tissue and organs by receiving radio waves [17].

The MRI works with creating a magnetic static field that uses the physical phenomenon called nuclear magnetic resonance(NMR). By stimulating the spins of the electrons that float around the nucleus of different atoms there can be found a little magnetisation vector [17]. In the human body this magnetisation vector is caused by hydrogen atoms that have a high spin rate in comparison to other molecules. The amount of magnetisation in a body region can be registered by the radio frequency (RF) coils of the MRI (see figure 2.5). These coils can locate the density of hydrogen atoms in a body part slice.

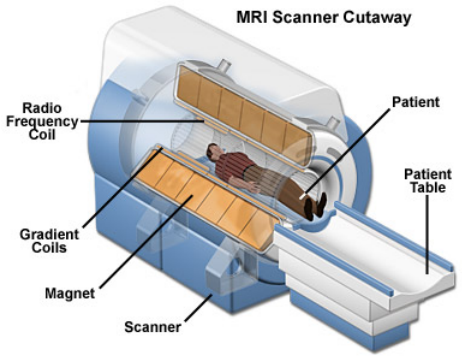


Figure 2.6: *A global structural view of the MRI [18].*

Cancer can be found at locations where the density of hydrogen molecules is denser than the original density of the tissue. Also the structure of the body tissue is important to analyse for diagnosing cancer. MRI has the advantage of having a high resolution in comparison to other imaging techniques. This gives the benefit to analyse tissue structures very well. At the moment MRI is one of the most precise imaging methods to diagnose a patient for cancer at a location [19]. Most imaging systems have already an recognition program for detecting tissue that is identified as cancer tissue. In the future this program could be used for creating a full robotic guided surgical system.

2.9 Surgical robotic control systems

The medical field is searching for new possible solutions for performing surgical operations minimal invasive. Surgical robotic systems are one of the greatest opportunities for performing a surgical operation minimal invasive with use of modern technologies. Some surgical robotic systems(SRS) will be introduced and explained in this section. Also the control system of the surgical robotic systems will be analyzed.

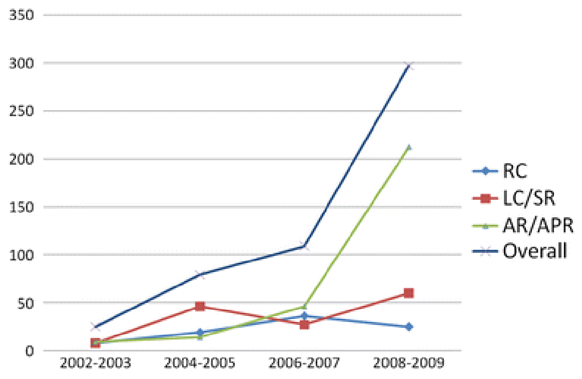


Figure 2.7: *The increase of using surgical robotic systems for surgery over 8 years. [20].*

Furthermore the rise of SRS over the last decade shows the urgency of designing new SRS [20]. The medical field has acknowledged the benefits of using medical assisting robots during a surgical operation. This increase can be found in figure 2.7 A couple of the most innovative or new surgical robotic systems are discussed below.

The —Da Vinci— surgical robotic system

The best known available surgical robotic system that is used in the medical field today is the "Da Vinci" robot. This is an approved surgical robotic operating system that is allowed for practical use in medical surgical operations. Furthermore the "Da Vinci" robot has some important control systems that are needed to take into account of the modern upcoming used surgical robotic systems [21].

The most important part of designing a MIS robot and is used by the "Da Vinci" robot is that the medical expert does not have to adapt to the new working situation. What this means is that the control of the imaging system, the movement of the end-effector and the force applied on the interface system needs to give the medical expert a natural feeling of the system. Therefore it would feel like working with the hand, but when using a robotic system the operation will be more precise.

Providing imaging information of the imaging subsystems is a real challenge. The information needs to show where the used instruments are and to give the needed information of the human body part. This can be viewed from a ocular rift system (see figure 2.8). The benefit of using the oculus rift is that the medical expert can look inside the body and also see depth.

Medical experts gain operation knowledge with age, but also the trembling of the hands get worse. Compensation control systems can be used to remove any tremble noise from the wanted information with use of complex filter systems that work with real-time control. One of the most common used filters what is used by the "Da Vinci" robot for real-time control is the Kalman filter [22].

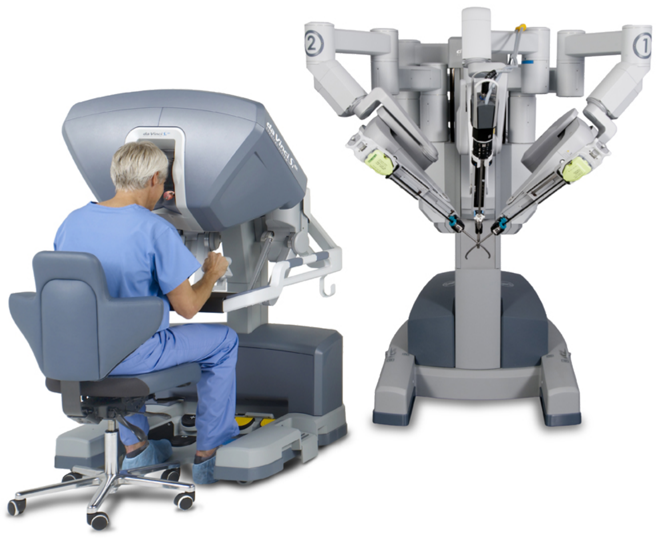


Figure 2.8: *The Da Vinci surgical robotic system with human interface and robotic system. [23].*

Research groups around the world want to test and improve the system of the "Da Vinci" robot. However, this is only against the medical regulations for adjusting surgical robotic systems. Such a new robotic control system design needs to be tested and evaluated. The company that has made the Da Vinci robot wants the surgical robotic system unchangeable in software [20]. As a result this gives the opportunity to find new improvements and adjustments a stop. On the other hand the Da Vinci components are not protected by the company and can be used to assemble new designs of surgical robotic systems that could be tested.

The Da Vinci robot is the only available surgical robotic system what is approved and used for medical operations. Other robotic surgical system are researched and investigated on safety regulations rules for medical approval. The Zeus system is another surgical robotic system that uses almost the same components and joint space as the Da Vinci robot, but is not approved because of a not approved control system.

—Zeus— surgical robotic System

Another surgical robotic system that is experimental used in hospitals is the robot called Zeus. This robot uses almost the same robotic joint connections and movements as the Da Vinci robot (see figure 2.9). In comparison with the Da Vinci robot, the Zeus robot differs only with making use of a voice control system in the robot. This can be used for moving the joints or arms on voice command. Europe counts 15 operating Zeus surgical robotic systems and America counts 30 units.

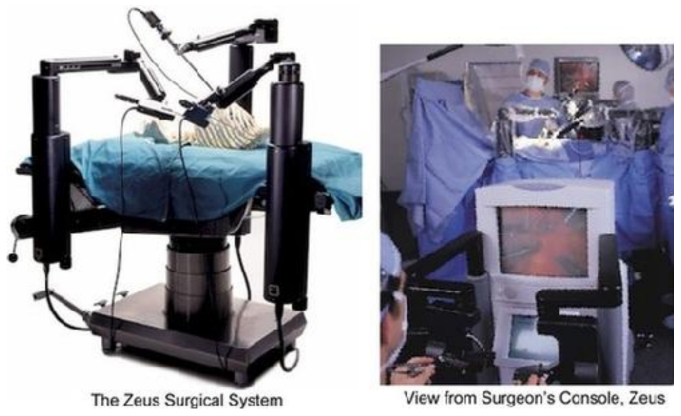


Figure 2.9: *The Zeus surgical robotic system with interface and endeffector system. [24].*

The surgical robotic system Zeus is used and experimented during laparoscopic radical prostatectomy operations. The control system consists of a master-slave connection between the interface system and the Zeus system [25]. There is also the possibility to setup an assisting system for Zeus using the automated endoscope system for optimal positioning (AESOP) [26]. This system is one of the main reasons the Zeus is used for several prostatectomy operations. Not only the operation mode, but also the assisting mode uses high image quality and tremble compensation mechanisms for precise control of the endoscope.

The —DLR MIRO— surgical robotic system

A very prospective surgical robotic system with a great future ahead is the DLR MIRO (see figure 2.10). This is the second generation system of the Versatile arm. The surgical robotic system can be switched between full control system or assisting control system. For doing different operations where the robot needs to be made for assisting or performing the operation can differ a lot.

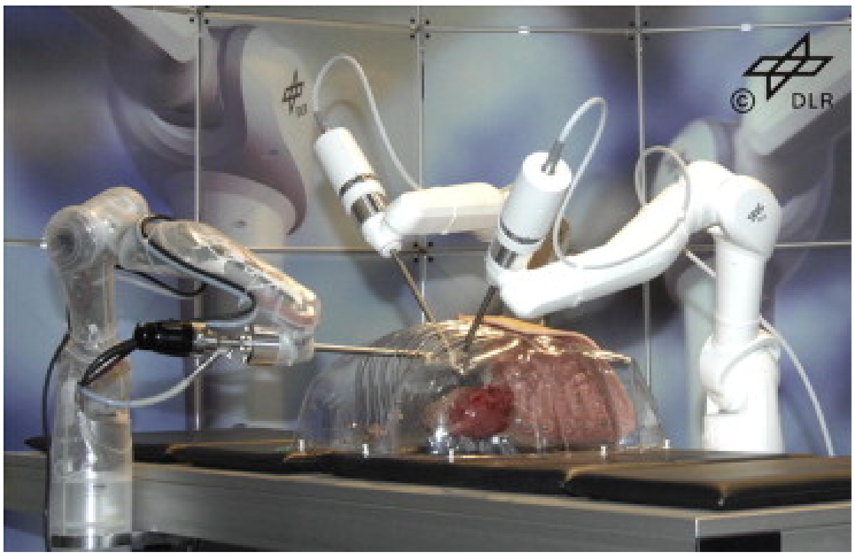


Figure 2.10: *The DLR MIRO system attached to the operation table [27].*

The DLR MIRO system is designed at the institute for Robotics and Mechatronics in Germany. In comparison with the Da Vinci or Zeus surgical robotic systems the new DLR MIRO system has some advantages over the other systems. The DLR MIRO is very small and got the newest technology of servo controlling systems. This gives the benefits that the DLR MIRO can perform operations in small workspaces, the system is very light in weight and got force sensors to transmit haptic feedback to the surgeon [27]. The last benefit is not integrated in the Da Vinci surgical robotic system and is an overall improvement for surgical robotic systems. Also is mentioned that the robot can work with small workspaces what is achieved by attaching the robotic arms of the DLR MIRO system on the operation table [27].

The system is tested at the moment and has not been taken into practical use for safety measures. Overall the DLR MIRO system will give a huge impact on the surgical robotic systems that are now used in the hospitals [28].

—NeuroArm— surgical robotic system

One of the most precise robotic systems that has been ever created for the medical field is the NeuroArm (see figure 2.11)[29]. The NeuroArm is one of the most innovative surgical robotic systems that has been made at the moment. This robot can also be used with the MRI systems and can be compared with the McRobot. In addition to the McRobot, the NeuroArm is not MRI compatible and uses the MRI images that are made before the surgery and is not working real-time. Furthermore, the NeuroArm is restricted to do only neurosurgical operations in comparison with the McRobot. The reason why is that the tissue in the brain doesn't move a lot. So the anatomical structure does not change that much and from this can be concluded that the cancer spot stays at the same spot. With this information the medical supervisior only needs to guide the needle to the cancer spot founded from the MRI images. So this surgical robotic system is only used for neurosurgical operations and is not usable for lower limb surgical operations. This is why the McRobot is a further improvement of the NeuroArm system. The NeuroArm is still in development, but is already used in several hospitals in America. Another reason

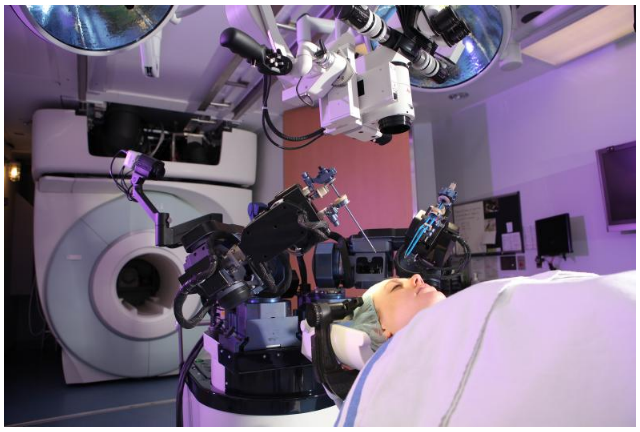


Figure 2.11: *The NeuroArm is a neurosurgical robotic system what makes use of images from the MRI for robotic surgery. [30].*

why the NeuroArm is extraordinary is because the robot can use many imaging systems for guidance of the image surgical robotic control system of the NeuroArm [31]. Other features of the NeuroArm are integrated haptic force feedback, guidance systems and precise kinematic motor control systems. At the moment the robot is only used for many neurosurgical operations, but can be changed for other surgical operations. It is compatible with different operating systems and is adjustable for the required surgical operation procedure [32].

Haptic Feedback Systems

Many new made surgical robotic systems have intergrated haptic feedback systems for force feedback to the surgeon. These systems were not used in the old surgical robotic systems like the Da Vinci robot system, but at the moment several experiments are done to create a haptic feedback system to old approved surgical robotic systems [33].

Haptic feedback systems are important new features for surgical robotic systems to give the surgeon feedback when force is applied on a robotic joint. External forces on the jonts can be caused by tissue pressure and indicates what for sort tissue is touched [34]. Haptic feedback is at the moment the limiting factor for surgical robotic systems. Combining haptic feedback to SRS is very difficult to give the surgeon the same tactile information when using the hand for doing surgery. Research has shown in experiments and simulations that the trauma that is caused to the tissue is less great when using haptic feedback systems [34].

Real-time monitoring

One of the most important things of making a surgical robotic system is making a system that has no delay in joystick to end-effector movements [3]. This can be done by making an embedded system that can perform heavy calculations for the kinematics of the Omega 6 and the McRobot. There are some software packages available for real-time monitoring. The software system "Real-time control system" does use a generic Hierarchical control system that is real monitoring.

ROS uses the languages C++ and Python. For real-time control the option to choose C++ is better than using Python. Furthermore the language C++ uses less functions than Python what gives less loading time on extra processes.

Evaluation surgical robotic systems

All investigated surgical robotic systems have shown there potential for future systems. From the literature research there can be concluded that the best surgical robotic control system for comparing with the McRobot is the NeuroArm. In comparison with the McRobot, the NeuroArm uses also MRI images for guiding the robotic arm to the right position. However, the NeuroArm uses images that have been taken before the surgical procedure with the SRS. The McRobot will do the imaging and the surgical procedure at the same time for real time monitoring of moving organs. Also the control system of the DLR MIRO has an interesting switching mode system for using in the McRobot control.

In the future perspective SRS will get more applications and fine tuning of the system to gain the speed that is needed for real time monitoring. Some new features will be better haptic feedback systems that are integrated in all dimensions of the joystick and SRS. Also SRS will be more complex in system, but user friendly for surgeons when using a SRS.

Chapter **3**

Requirements

3.1 System requirements

In this research there are several requirements setup for using the omega 6 haptic joystick for the surgical MRI compatible operation robot. The main objective is to program a human operator control loop system that can encode the movements of the joystick into movements of the surgical robots with several requirements of the system.

Below is a list made with all the must -, should -, could -, would not have criteria for the research objective. These criteria are made with the information gathered from the supervisor, medical expert and references.

Must

1 **ROS control implentation**
Problems with the time delay that is occurred using python programming language. ROS can use C++ what is more preferred for real-time control systems. Another advantage of the robotic operating system is that it includes libraries for kinematic calculations.

2 **Key features of control system.**
It must contain all needed subcontrol systems to make a full functional interface system. This includes filters and libraries.

3 **Kinematica description and formulas of joystick and robot movements**
Kinematic description and formulas of joystick and robot movements.

4 **Literature information about the control systems.**
There needs to be enough information to support decisions and validity.

5 **Safety**
If something happens that the robot makes the wrong movements a security stop can be pressed.

Should

1 **C++ Implementation**
Set the controlling system on a real time based control system instead of python.

2 **Calibration**
The system can check itself with actuator position, joystick position and sensitivity. Plug in and Done

3 **Without using a pc**
Just use a microcontroller that can calculate the dimensions and translate it in coordinate system of the used actuator..

4 **ROS 2.0 or IROS using for control.**
There are multiple different versions of ROS that can be used and maybe some of the other versions is better in use.

Would

1 Controlling the device with haptic feedback or force

2 Using other microcontrollers than "Arduino"
Better performance maybe when using mbed controllers (C++)

3 Simulation of McRobot 3.1
For testing adjustments to the code for the McRobot 3.1 a simulation can be made for fast experimenting

4 Electronic circuit board
For making the controller system better a electronic circuit needs to be soldered instead of using breadboards.

Would not

1 Changes in the Omega 6 joystick.

2 Changes in the surgical robot.
Updates to concept parts can be used, but it takes effort for making the new kinematic calculations.

3 New difficult procedure for medical experts.
A procedure with a lot of calibrations and adjustments for making the system work.

4 Other programming software
This includes the programming language Python for example.

3.2 Required subcomponents

One of the requirements that is not named, but needs to be taken into account is to find all the materials for making the assembly of the McRobot 3.1 and the electronic circuit board.

3D printed parts

The actuator parts are 3D printed with the 3D printers at the group RaM. There are some parts for the actuator and some for the total McRobot 3.1 construction. In the group RaM some members are working on new versions of the actuator. These parts can be tested during the control system development. The actuator consist of many subparts what needs to be 3D printed all separate from each other. The Solidwork files will be made into STL files that can be 3D printed. When the parts are 3D printed the assembly will begin of the actuator. The other subparts are then required for steering the actuator system.

Arduino Mega

To process the movements of the omega 6 haptic joystick into movements of the McRobot, a small processing microcontroller must be used for the controlling system. The used microcontroller is an Arduino Mega microcontroller that is one of the most common used microcontrollers for experimental robotics with multiple ports [35]. Furthermore this microcontroller uses a modified programming language C with some changes from the developers of Arduino. The programming language can be used to implent mathematical equations for controlling and positioning the actuators of the Mcrobot. The microcontroller is a very basic microcontroller with some processing speed, but there are more efficient microcontrollers that can calculate multiple equations faster. In the experimental setup a small delay can be expected with movements of the Omega 6 to the McRobot when loading all the kinematics to the Arduino Mega. When it is required to do all the kinematic equations on the Arduino, a different Arduino microcontroller is suggested. This microcontroller is the Arduino Galileo [35]. This microcontroller uses a special processor from Intel that has a higher calculation speed than any other Arduino microcontroller available on the market.

Important Electronic Components

A possible connection setup between the Omega 6 and the Arduino Mega was directly by using the Arduino from an USB port to receive data from the joystick. Unfortunately the refresh rate of the Arduino Mega is not high enough and does not contain the required driver settings for the USB port connection. So the option to buy an USB shield is not yet important.

Furthermore, the air pressure system needs some electric components for making the system work. The required voltage for activating the air valve system is 24 V what is more than the Arduino Mega can deliver with its 5 V output. An external voltage resource is needed to provide the 24 V to the air pressure system. However, the only problem is that the Arduino Mega still needs to send the right signals to activate the air pressure system, but cannot handle when connected directly to the 24 V output. This can be solved by using a transistor that sends the 24 V when a signal on the other port is received. A NPN transistor can be used for changing the inflow and outflow of the 24 V.

Chapter 4

Concepts

In the section concepts there will be looked at which possibilities exist to design the control system. Each approach is rated with the quality of a certain control system. This rating can be done with looking at the design or comparing with literature research. The best concept of the human operator loop control system will be used to make a final model for the McRobot control loop system. At the end of each concept an evaluation of the criteria and the model is done. The criteria of rating the concepts are the following:

- Real-Time
- Multiple platforms
- Easy setup system
- Future compatibility
- Debugging

The rating score 1 stands for a bad quality of the concept and the rating score 5 is the best possible score for a criteria.

4.1 First concept: Arduino heavy duty system

The first designed concept of the Omega 6 control system to McRobot 3.1 is coded the most part in Arduino (see figure 4.1). This concept uses ROS as minimal as possible. ROS will only send the raw Omega 6 data to the Arduino. The Arduino microcontroller will do all the heavy duties like the Kalman filter implementation. There is a high possibility with this concept that the Arduino will have a delay in actions. To enchant the performance of the processing speed of the Arduino another sort then the Arduino Mega can be used. The Arduino Galileo is a high processing microcontroller that uses a good processor from Intel.

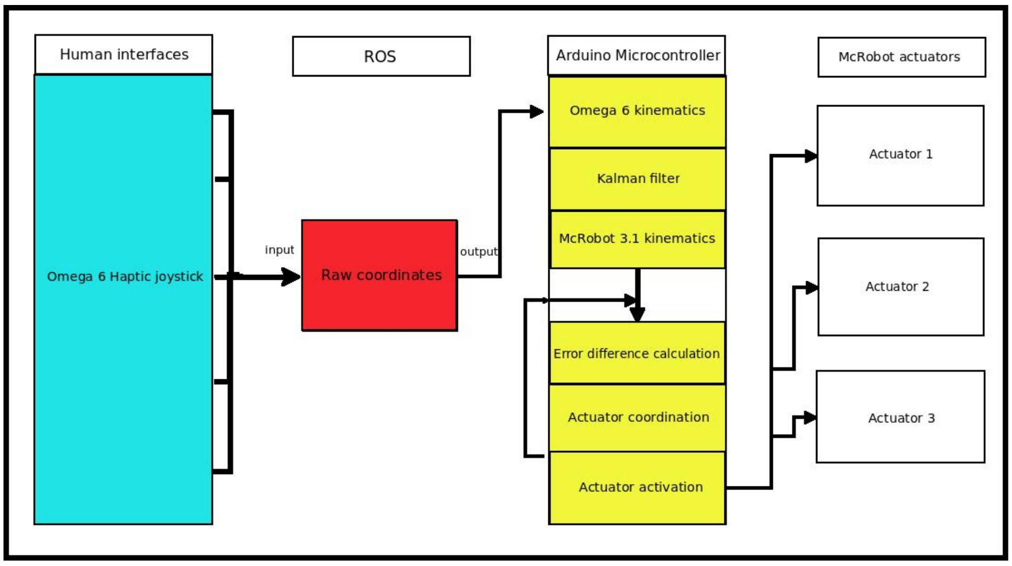


Figure 4.1: *The human operator control loop system of concept 1.*

Concept 1 criteria evaluation	
Criteria	Rating score (1/5)
Real-time	2
Multiple platforms	1
Easy setup system	4
Future compatibility	2
Debugging	4
Total	13/25

Table 4.1: *Concept 1 criteria with rating of overall system specification.*

Concept 1 evaluation

The human operator control system of the first concept is not good for integrating into surgical operating systems . The Arduino Mega has almost no processor speed and the language is not the best one for use. Also when someone wants to use another human interface platform, the code will not adapt to the new platform. The system is very easy to setup. The ROS environment only needs be configured and be turned on to connect with the Omega 6 and the Arduino. The further processing is done by the Arduino. For future perspective the Arduino will not be used and be replaced by a better compatible microcontroller that can directly use the coordinates from the Omega 6. The debugging in the Arduino environment is very easy so that is also a good point of this concept. Overall the concept is not very bad, but lacks at some of the important control system parts (see table 4.1). The Omega 6 coordinates library is also already in the ROS environment so the first concept is not chosen for multiple reasons.

4.2 Second concept: Equal load of duty system

This concept is based on dividing the heavy processing parts between the Arduino Mega and ROS (see figure 4.2). The computer processor will be more used by ROS for doing heavy calculations of the Omega 6 kinematics and applying the Kalman filter. The coordinates can be send to the Arduino Mega by publishing the coordinates on a message board. The Arduino Mega only needs to process the information to check and do the rotations of the actuators when the wire length is longer or smaller. The check system (encoder) is still a heavy duty process for the Arduino microcontroller, because it uses a lot of variables to keep track of the error. The error is the difference between the Omega 6 orientation and the current position of the McRobot 3.1. This code has already been written and can be found in appendix A.1 The concept control system of equal heavy load duty can be seen in figure 4.2. The Arduino and ROS have both some complicated equations and functions.

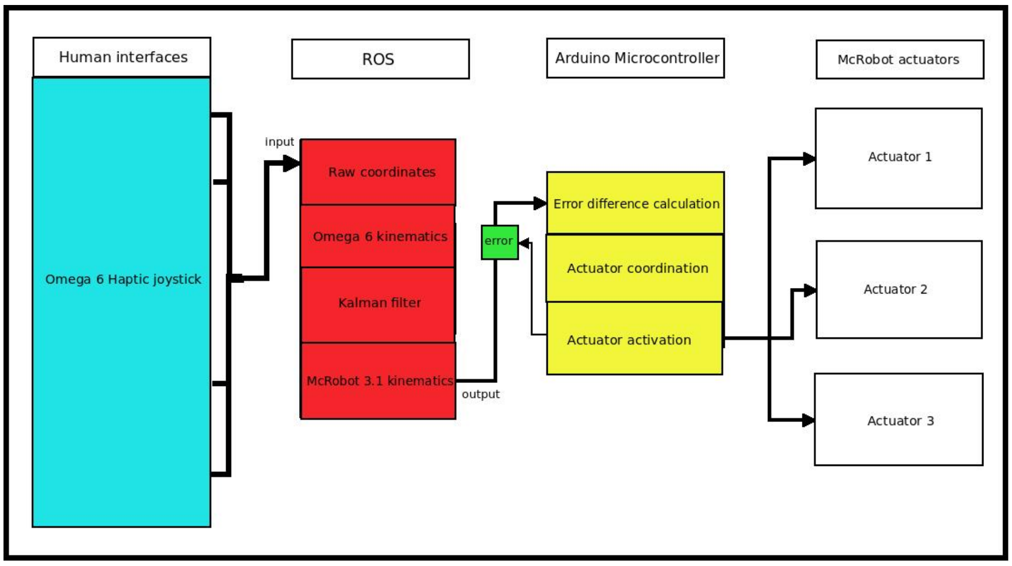


Figure 4.2: *The human operator control loop system of concept 2.*

Concept 2 evaluation

The system is more stable in comparison with concept 1, because the large processing parts are done in the ROS environment. This gives a better real-time control system. A disadvantage is that the system can only be used by the omega 6 and is limited in changes to the script. This gives the low rating in multiple platforms (see table 4.2). The system is still easy to setup and for debugging the ROS environment. The only problem is what to send for information to the Arduino. This system can be used in the future, but there are still better options to choose. The debugging is average in comparison to concept 1, but still is possible.

Concept 2 criteria evaluation	
Criteria	Rating score (1/5)
Real-time	4
Multiple platforms	3
Easy setup system	3
Future compatibility	4
Debugging	4
Total	18/25

Table 4.2: *Concept 2 criteria with rating of overall system specification.*

4.3 Third concept: ROS heavy duty system

ROS is a fast operating system that needs a few microseconds to do a calculation. This is because of the use of the C++ coding system. There can also be chosen to use Python, but to create a better real-time control system the language C++ is mostly preferred. With the high processing speed of the processor in the computer, large and difficult equations can easily be solved by ROS. Concept 3 uses ROS with every possible calculation for steering the actuators (see figure 4.3). The Kalman filter and the kinematics of the Omega 6 were also used in concept 2, but a new part for ROS is the tracking system of the actuators. The Arduino in conept 2 only listens to ROS, but concept 3 can also send information back. This system is the most complex of all the three concepts, because of the subscriber and publisher node.

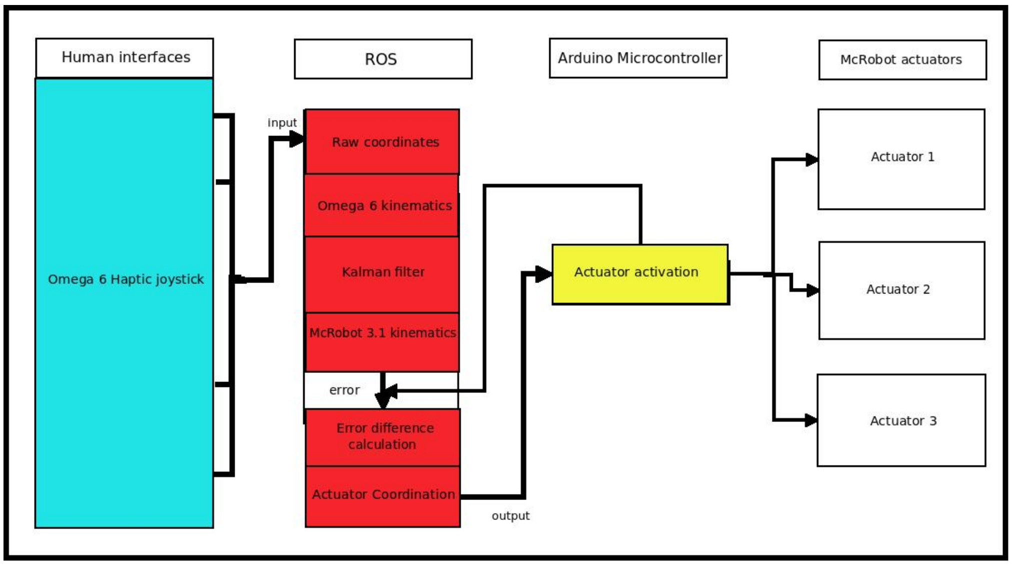


Figure 4.3: *The human operator control loop system of concept 3.*

Concept 3 evaluation

On multiple criteria is concept three better than the other two concepts (see table 4.3). When using only ROS, the system works on software written using a low level programming language (with minimal overhead) with a fast processor unit of the PC. This combines high processing power with soft-realtime control for surgical operating systems. When applying the coordination system of the McRobot in the ROS environment, the script can easily be switched to other McRobots or human interfaces. The ROS environment has a more complex debugging system to change settings with messages, but when it is working the system is very easy in use. Future perspectives is to use the system for microcontrollers that can use operating systems and directly ordering the pins to steer the actuators. This can be done with

Concept 3 criteria evaluation	
Criteria	Rating score (1/5)
Real-time	4
Multiple platforms	4
Easy setup system	3
Future compatibility	5
Debugging	3
Total	19/25

Table 4.3: *Concept 3 criteria with rating of overall system specification.*

a Raspberry Pi microcontroller for example. The debugging system needs to be learned for the ROS system, but once learned it is normal in debugging.

4.4 Chosen Concept

Concept 2 and 3 are both chosen for building the control system of the McRobot. From the ratings can be found that concept 3 is the best concept for modelling. The only problem is that the debugging in the ROS environment is not the easiest one. The overall code will be first tested with concept 2. The modelling of the system is done in the next chapter.
The kinematics of the Omega 6 and the Kalman filter are already in use of the ROS environment so this will be always done in ROS. This is the reason for not making a code for concept 1. The Arduino will debug the problems in the coding part to set and drive the actuators to the right position. When the debugging is completed and operates well it will be written in C++ language in the ROS script.

Chapter 5

Design and Models

5.1 Total control system design

The global design of the human operator control loop system can be seen in figure 5.1. The programming software ROS firstly needs to be designed in order to make a connection between: (1) the Omega 6 haptic joystick and the computer (ROS) and (2) the computer (ROS) and Arduino. The total control system design between human operator and the end-effctor of the McRobot 3.1 includes several steps. First, the human operator (medical expert) uses the human interface (Omega 6 haptic joystick) by moving the joystick and thereby producing 3D coordinates. Second, the produced coordinates are send to the computer (with ROS). Third, the computer collects the incoming data, processes it and sends it to the Arduino Mega (microcontroller). Fourth, the Arduino Mega uses the incoming data for selecting the required digital output pins and sends it to the McRobot 3.1. Fifth, the digital output pins rotate the actuators of the McRobot 3.1. Sixth, the McRobot is attached to the patient and positioned in the MRI bore. Final, the needle of the McRobot punctures the skin of the patient to reach the cancer spot and burns the spot.

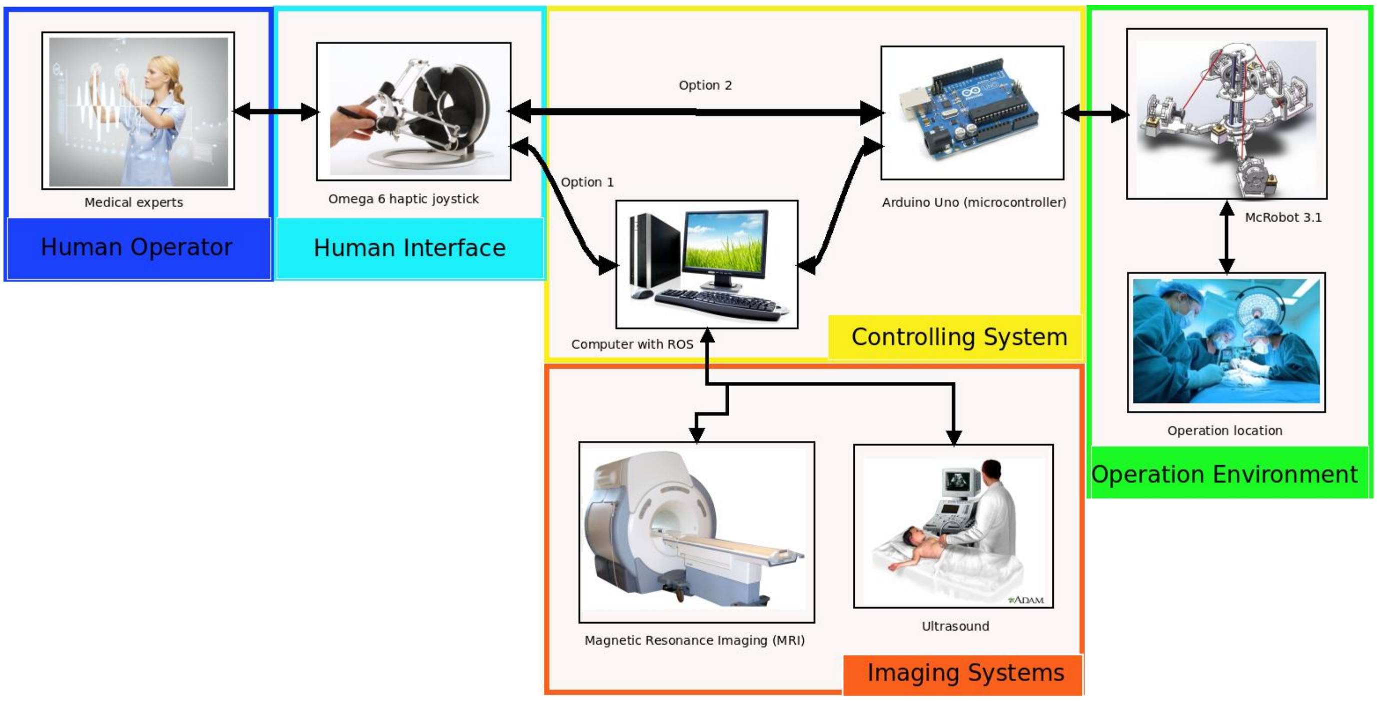


Figure 5.1: *The total control system of the McRobot with including interface*

Kalman filter

To create smooth movements for the SRS from the trembling hand of a medical expert there are several options to reduce/remove the trembling movements. One of the first possible options is to use a filter. However, most filters use time delay to investigate the signals over a period of time. This gives rise to the problem that there will be a delay in signal to the McRobot. To solve the problem a Kalman filter will be used for the McRobot control loop system. This filter uses forward calculations to predict the movements in the next couple of microseconds and uses the real founded movements to compare it with the prediction value. Furthermore, this prediction calculation can be made to act on specific frequencies. For example, lower frequency patterns than 9 Hz can pass the filter what helps to filter the tremble

movements of a surgeon. The tremble movements are around 9 Hz. So a Kalman filter is implented with a prediction of a couple of microseconds and let's frequencies pass that are lower than 9 Hz.
The Kalman filter is a very common used filter for robotic systems. Particularly the filter uses a prediction algorithm to predict the movement of the robot. For making the Kalman filter, there are multiple matrixes needed and calculations for finding the predicted value and to change the gain of the system. First of all, the Kalman filter begins with initializing the matrixes that are needed for computation of the filter. After initialization the filter will calculate directly the predicted position of the robot. The method for predicting the position is used to look at the difference of the prediction- and the real position of the robot(see equation 5.1) [36].

$$\begin{aligned} \hat{\mathbf{x}}_k &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}_k} \\ \mathbf{P}_k &= \mathbf{F_k} \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{5.1}$$

In the prediction formula for the Kalman filter there are multiple important variables to look at. The first equation calculates the new predicted position($\hat{x}_k$) with using the old predicted position ($\hat{x}_{k-1}$) and using the known external influence (u_k). As a result the information of $\hat{x}_k$ can be used for steering the McRobot. The other equation is needed for calculating the uncertainty of the predicted movement. Due to comparing the new calculated uncertainty (P_k) with the old uncertainty (P_{k-1}) and the uncertainty from the environment (Q_k). Next can be found the update equations of a kalman filter.

$$\hat{\mathbf{x}}'_k = \hat{\mathbf{x}}_k + \mathbf{K}'(\mathbf{z}_k \check{} \mathbf{H}_k \hat{\mathbf{x}}_k) \tag{5.2}$$

$$\mathbf{P}'_k = \mathbf{P}_k \check{} \mathbf{K}' \mathbf{H}_k \mathbf{P}_k \tag{5.3}$$

$$\mathbf{K}' = \mathbf{P}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{5.4}$$

The update equations (equations 5.2 and 5.3) are important for comparing the predicted values with the real found values. If the predicted position ($\hat{x}_k$) and real position ($\hat{z}_k$)have a large difference between the two values, the system will set a high gain ($\mathbf{K}'$) to minimize the difference between the two values. This adaption of the gain is also affected by the previous position of the robot. If the delta of the position between the past and present is very different, the adaption of the robot position needs to track the position with greater distance (see equation 5.4). This calculation of prediction and updating of the gain is done when the Omega 6 receives the coordinates. A simple flow chart of the filter can be seen in figure 5.2.

Arduino Microcontroller

In the Arduino design there are two options available in the programming part. There could be an indirect connection between the omega 6 haptic joystick to the Arduino Mega or a direct approach. The indirect approach is the first "must" criteria of the bachelor assignment and is using a computer with ROS to connect the Omega 6 and the Arduino indirect. The difficult calculations of the kinematic calculations can be done in this approach by ROS. The second approach is to make a direct connection between the Omega 6 and Arduino. This design is more preferred than the other. The requirement to have a computer that needs to boot up the system of ROS takes a lot of time and effort to start the demonstration/operation. So it is preferred that the system can be plugged in and is almost directly

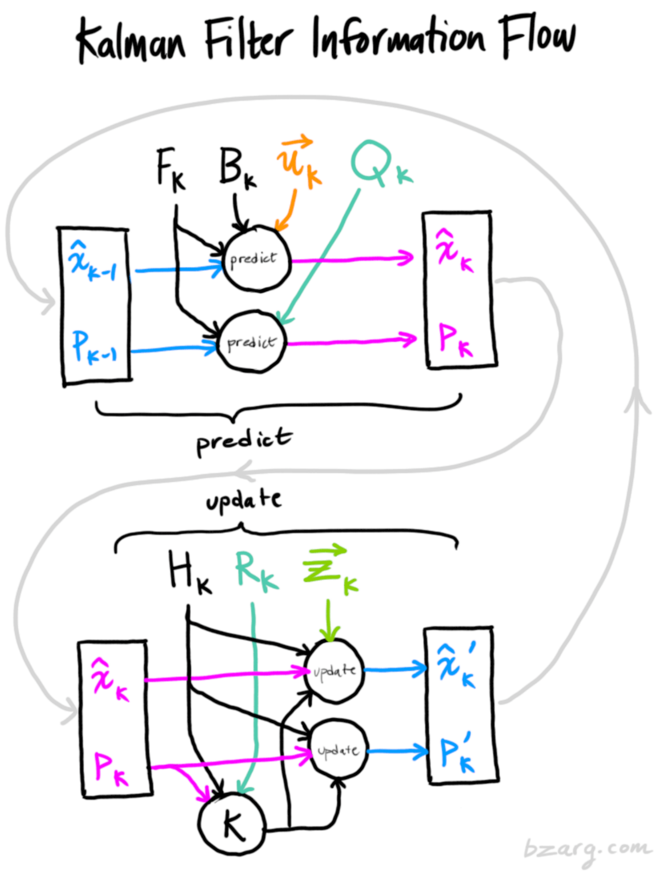


Figure 5.2: *A simple flow chart of the Kalman filter[36]*

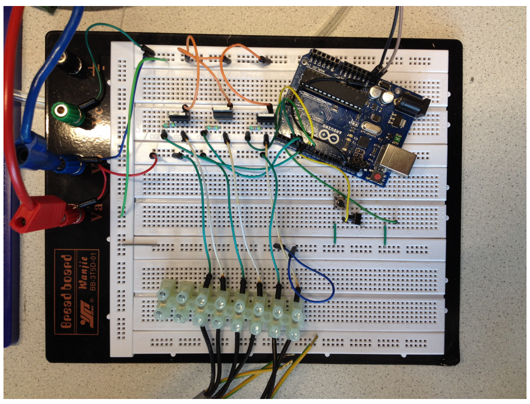
ready for use. A calibration is always required for starting up the connection between interface and controller.
However the Arduino needs a special modification to get more processor speed for heavy calculations. When using direct connection of the Omega 6 the Arduino needs to process all the kinematics and the filter systems that otherwise were calculated by the computer. When a standard Arduino Uno is used there could be found a delay in the signal. To meet the "must" criteria that there is no delay an Arduino Galileo could be used [35]. This Arduino microcontroller consists of an intel processor that can do the heavy calculations of the used formulas. The Arduino Galileo is a software compatible board with the Arduino software packages.

Electronic connection system

The electronic connection for the movements of one actuator can be seen in figure 5.3a. As we can see the power source is set to the + and - pole of the breadboard. In fact the power source is in real a 24 V source what cannot be found in the used design program. So therefore a battery source with 6 V will be used to replace the 24 V source. Additionally, the Arduino Mega uses the gnd and the following digital pins:

- **Digital pin 5**
- **Digital pin 6**
- **Digital pin 7**

These pins are used to initiate the rotation of the actuator. All of the digital pins are each connected to the first pin of the NPN transistor with a resistor between the digital pin and the transistor. Concerning the third pin of the transistor is connected to the ground poles of the breadboard. As final, the signal is channeled through the second pin what turns the valve system on. The + pole of the power source is connected to the terminal stone and when activated it lets 24 V through what is required for using the air system. Further, the air system consists of 3 seperate air valves. This air valve system including the actuator is replaced in figure 5.3b with a stepper motor.


(a) *The electronic connection for steering one actuator*

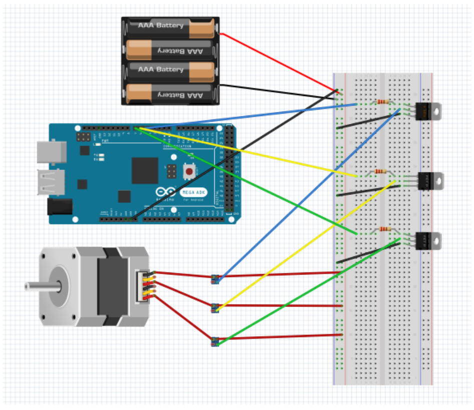

(b) *The electronic connection scheme for rotating one actuator in a direction*

Figure 5.3: *Electronic connection scheme and example for McRobot actuators*

The total electrical circuit design is more efficient and more clear in use of the system. An improvement of the board is the plug in system. This solves the problem of finding the right pins of the Arduino microcontroller in the right headers. The total electrical circuit needs to turn 5 actuators. There are 3 transistors needed for one actuator so 15 transistors are needed that are connected with the collectors to the Arduino digital pins. Eventually, the assembly has been made in an electrical circuit maker and can be seen in figure 5.4.

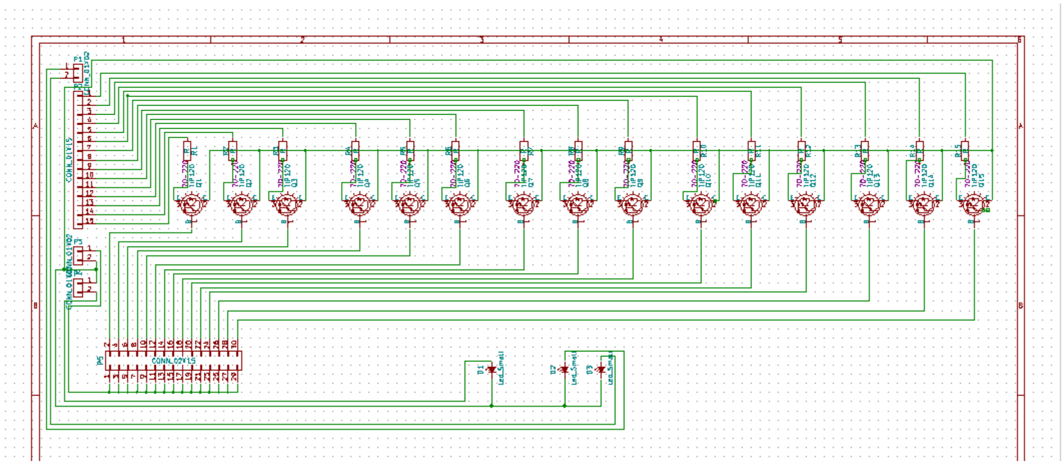


Figure 5.4: *The electronic circuit scheme of using a board for connecting the electrical components.*

5.2 Kinematic design

Omega 6 kinematics and calibration

The Omega 6 got an example code where the translation and angle are calculated of the end-effector for the McRobot 3.1. The important part of the example is the rotational kinematics calculation code. This code part consists of orientation of the pen and can be translated to the McRobot 3.1 in needle angle orientation. This is all done in the founded library of the Omega 6.

However the angle coordinates that are found are not world coordinates. There are some simulations done for setting the local coordinates into world coordinates. These simulations were done in MATLAB and can be viewed from the zip file. Furthermore the simulation has the rotational matrixes of all the rotation points of the Omega 6. This includes the rotation location of each joint. With knowing the rotational matrixes of the rotating joints, world coordinates can be calculated to a fixed point (see figure 5.5).

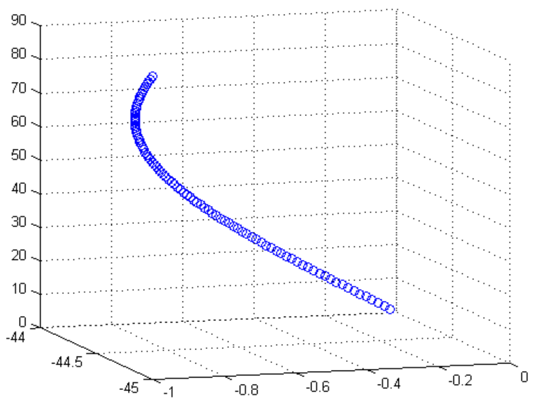


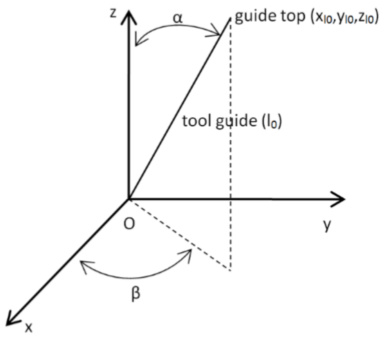
Figure 5.5: *The Omega 6 coordinates are transferred into world coordinates with using Matlab.*

The kinematic scheme of the rotational kinematics can be made when looking at the rotation parts. The pen can rotate on its own axis (yaw), on the first rotating frame (roll) and the second rotating frame (pitch).

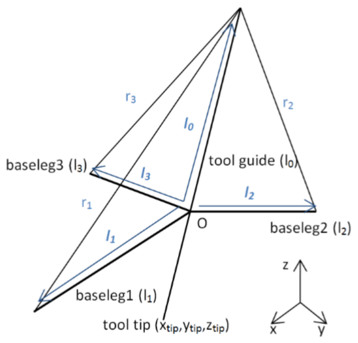
The kinematics of the Omega 6 can be directly used for kinematic calculation of the McRobot. Thus the Matlab file is only to support the idea, when the Omega 6 coordinates needs to be placed into world coordinates. The Omega 6 joystick has two important encoders that can measure the rotation position very precise. It is however important to calibrate the joystick at every surgical operation. How to calibrate the Omega 6 joystick can be watched in movie "Calibration Omega 6". When the first LED blinks of the Omega 6 joystick it gives an indication it needs to be calibrated. What needs to be done for calibration is to rotate the pencil endeffector to every maximum of its rotation axis. When the maximums have been found and the insertion of the joystick in the point has been done the system will turn the blinking LED on. This gives the signal that the Omega 6 has been calibrated.

McRobot 3.1 kinematics

The kinematics of the McRobot 3.1 consists of a forward kinematica formula. The formulas are not very complex, but will be explained. The Coordinates of the Omega 6 are first translated to coordinates of α and β what is shown in figure 5.6a. The α coordinate is used for giving information about the distance between the z axis and the guide top. The β is used for expressing the location in the xy plane. With these two coordinates the calculation for finding the wire lengths of the actuators can be calculated. Each actuator is positioned somewhere in the xy plane with this information we can look how much wire length is needed to reach the guide top (see figure 5.6a, 5.6b and 5.6). This can be done with the next three formulas (see equation 5.5). Each formula is used for calculating the wire length of one actuator.



(a) *The coordination system of the McRobot 3.1.*



(b) *The kinematic modelling of the McRobot 3.1.*

Figure 5.6: *The McRobot 3.1 coordination and kinematic system [9].*

$$\begin{aligned}
\mathbf{r}_1 = \mathbf{k}_1\theta_1 &= \sqrt{(\mathbf{x_1} - \mathbf{l}_0 \sin\alpha\cos\beta)^2 + (\mathbf{y_1} - \mathbf{l}_0 \sin\alpha\sin\beta)^2 + (\mathbf{z_1} - \mathbf{l}_0\cos\alpha)^2} \\
\mathbf{r}_2 = \mathbf{k}_2\theta_2 &= \sqrt{(\mathbf{x_2} - \mathbf{l}_0 \sin\alpha\cos\beta)^2 + (\mathbf{y_2} - \mathbf{l}_0 \sin\alpha\sin\beta)^2 + (\mathbf{z_2} - \mathbf{l}_0\cos\alpha)^2} \\
\mathbf{r}_3 = \mathbf{k}_3\theta_3 &= \sqrt{(\mathbf{x_3} - \mathbf{l}_0 \sin\alpha\cos\beta)^2 + (\mathbf{y_3} - \mathbf{l}_0 \sin\alpha\sin\beta)^2 + (\mathbf{z_3} - \mathbf{l}_0\cos\alpha)^2}
\end{aligned} \tag{5.5}$$

With finding the wire lengths the actuator control system can be setup in the Arduino code.

McRobot 3.2 kinematics

The kinematics of the McRobot 3.2 are very easy, but can be tricky when using the actuator system [10]. The McRobot 3.2 consists of a parallel manipulator system what can turn the system with two types of the α coordinate in comparison with the McRobot 3.1 (see figure 5.7). The alpha coordinates are split into two angles. The first angle is for rotations around the y axis and the other around the x axis. This results in only a forward or backward movement with one angle and the other to turn right or left. This coordination system is simple for finding in which direction the actuators need to turn, but the control system of the actuators is more complex. When the system wants to turn both forward and right the actuator needs to do one of the commands and later execute the other one.

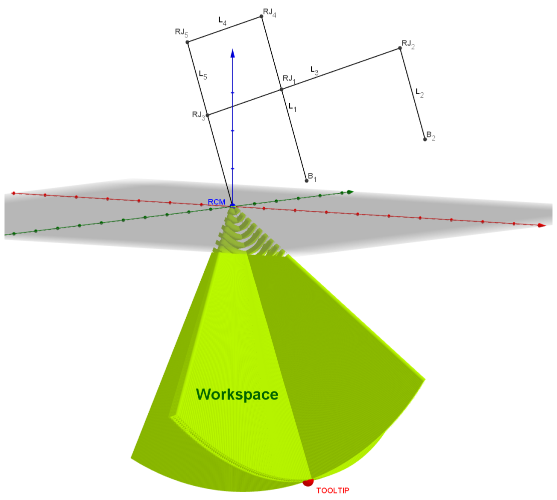


Figure 5.7: *The possible movements of the McRobot 3.2 with the founded kinematics [10].*

Actuator control system

To provide the surgeon or medical expert a well good feeling of the interface there needs to be a force applied on the interface for speed compensation of the McRobot. The used actuators have a low turning speed what can give rise to a delay. When the joystick uses force feedback, the maximum allowed speed can be compensated to that of the McRobot turning speed [9]. The step motion of the actuators is founded and is given in degrees.

5.3 Model assembly

All submodel parts are collected and set into the control system. The total control system can be found in the preferances. The ROS control code can be found in A.1 and the Arduino code in A.2 and A.3.

ROS code assembly

The ROS code has the most heavy computation parts of the control system. The code will be explained and divided in different parts.

The code begins with including all the libraries that are required for the code. These libraries are very important for understanding how information is gathered from the Omega 6 and how the message system works in ROS.

Then the message declaration is done with the next example code.

```
// ******************************** LIBRARIES *******************************
#include "ros/ros.h"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <string>
#include "kalman/ekfilter.hpp"
#include <cmath>
#include "plane.h"

#include <geometry_msgs/Twist.h>
#include <geometry_msgs/Vector3.h>
#include <omega6/Int.h>
#include <limits>
#include "drdc.h"
#include "dhdc.h"

#define REFRESH_INTERVAL  0.05   // sec
#define NUM_LINES 12
```

First the library is chosen for general sort message (geometry-msgs) then the specific sort message is chosen(Twist). The Twist message can contain 6 variables in comparison with the Vector3 what can contain 3 variables. Then the name of the message is declared. This name is used to specify your message name and declaration in the code.

```
// ****************************** MESSAGE SYSTEM ********************************

geometry_msgs::Twist actualPose;      //pose of the Omega6
geometry_msgs::Twist actualVelocity;    //pose of the Omega6
geometry_msgs::Vector3 actualForce;     //actual forces
geometry_msgs::Vector3 Pos_act;     //setted forces
omega6::Int buttonState;
```

After the message declaration the variables are set with values or none.

```
// ***************************** GLOBAL VARIABLES ***********************************

```

```
double t1,t0;           // Time variables
int c;

double xl1 = 157.25;
double yl1 = 0;         // Coordinaten van locatie actuator 1
double zl1 = 0;

double xl2 = −78.625;
double yl2 = 136.182;       // Coordinaten van locatie actuator 2
double zl2 = 0;

double xl3 = −78.625;
double yl3 = −136.182;          // Coordinaten van locatie actuator 3
double zl3 = 0;

double k0_degr = 0.08726;               // mm/graden
double k0_rad  = 5.002;                 // mm/radialen
double O_insertion = 0;       // Insertion needle begin location
double l0 = (185 − (k0_degr * O_insertion));  // Insertion needle location

double a;
double a_pub;
double B;
double B_pub;

double pi = 3.14159265359;      // pi

double r1_real = 242.80;
double r2_real = 242.80;                          // Calibration script needs to be used
    for right wire lengths.
double r3_real = 242.80;

double r1_d = 0;
double r2_d = 0;        // Length error difference
double r3_d = 0;

bool flag_omega6 = true;
```

The next part consists of multiple functions what can be called during the main script. When variables are made in the function, the variables will be forgotten and not saved. This is why the variables that needs to be saved are declared in the beginning. These variables are world variables and can be changed and saved from every part of the code. In function there are local variables that can only be changed or saved during the function. Functions help to order different processing parts. The print function will print information to the terminal for examining the results. If the results from the Omega 6 are strange, the system needs to be reset for accurate measurments.

The main function begins with "int main". After setting up the main function there is a device configuration setup made. This setup looks if there are any devices available and specifically the Omega 6. The Omega 6 needs to be plugged in and have acess to the operating system. This acess setting system can be found in the given ROS guide for McControl with Omega 6.

There is a publisher and a subscriber needed for gathering and sending the information between ROS and the Arduino system. This is published and gathered in the main while loop.

The position calculation for the McRobot is done in the " *McRobot 3 1 Calculation*" or "*McRobot 3 2 Calculation*" function of the script. The position is compared with the real position of the McRobot what is taken from the Arduino publisher. With this information it is possible to calculate the difference length between the wanted position and real position of the McRobot.

Arduino code assembly

The Arduino code is very simple for the control system of the McRobot 3.1 actuators (see figure 5.8)

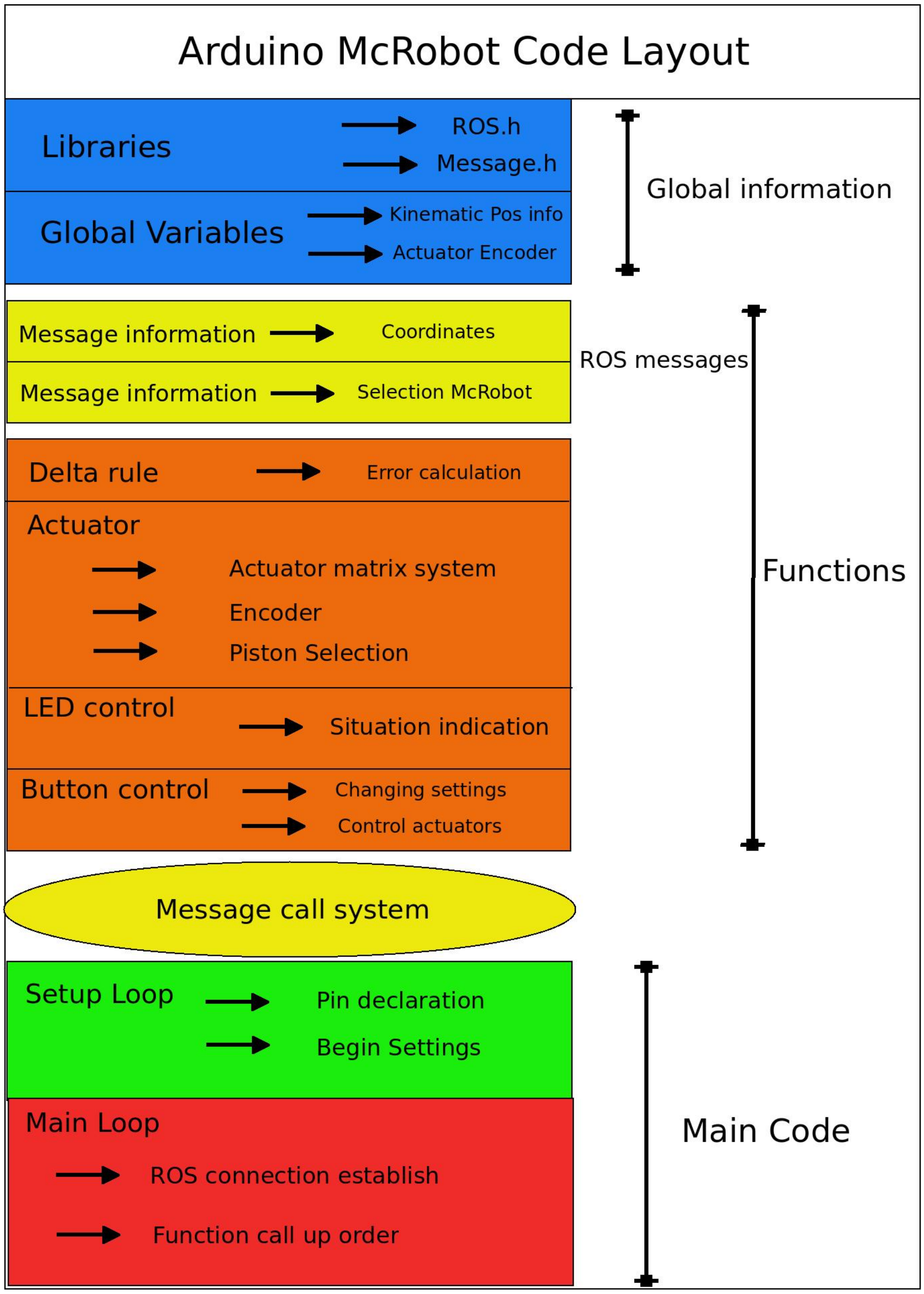


Figure 5.8: *The Arduino code layout of the control system for the McRobot with using ROS.*

The code receives the flags to turn the actuator forward or backward. Each actuator has its own forward and backward flag to drive between them. When the information is received it is being processed into the code part to activate the actuators. First the retrieved information is processed in the delta system to calculate the difference between real and desired position (see code part below).

```
void Pos_McRobot_3_1(){

for (int i = 0; i < 3; i++){
 r_delta[i] = r_d[i] - r_real[i];
}
}
```

When the encoder can be used, the $rd[i]$ needs to be replaced by the encoder system value in wire length. The actuators are activated in the script separate from each other with using matrixes. Actuator 1 is first processed with $i = 1$. The actuator is turned on if one of the flags is true. If none of the flags for forward or backward is true it will stop the actuator from setting pressure on the selected piston. The total turns are counted to go to the wanted position. With this counting information there can be made an estimation for when the actuator is standing in the right position. For concept 3 this process is done in the ROS code of the McRobot. The information of total activated counts for turning the actuator are send to the ROS msgs system. This is done by making a publisher in the Arduino McRobot code and a subscriber in the ROS code. This information is used for calculating the error between the angle of the Omega 6 and the McRobot 3.1.

Also the Arduino code for concept 2 has been made. This code contains no publisher for sending information back to the ROS environment. In exchange the Arduino code needs to have a track system for the actuators. This is implented and through the bachelor assignment improved a lot. The Arduino code was at the beginning very long and had many repetitions in the code. After a while matrixes were introduced for information storage in less lines of code. The code shrunk with 3 times it size.

Furthermore, a code is written for the McRobot 3.2 and for the made electronic board during the research. The actuator steering code of the McRobot 3.2 is compared with the McRobot 3.1 a little different. In the McRobot 3.2 script the robot can only move forward, backward, left or right to change the orientation of the robot. Also the movement resolution is much higher in comparison with the McRobot 3.1 and is around 11 $degrees/step$. The electronic board is an overall improvement for controlling the system and indicates what happens in the control system.

Another improvement of the Arduino code is the option to choose between two available robotic systems. In ROS there can be made a decision for using the McRobot 3.1 or 3.2 kinematics. This helps users to choose fast between certain robotic systems.

Chapter 6

Methods & Experiments

During the prototyping chapter multiple codes will be presented to understand the progress and development of the code. Also important decisions are made when certain problems are encountered. The two best concepts (concept 2 and 3) will be worked out and tested on functionality and progress.

6.1 Control loop system setup testing

For making the connection between multiple platforms, the testing of the system is done in several steps. First the Omega 6 haptic joystick is connected to the ROS environment. The received coordinates from the Omega 6 haptic joystick are processed and sended to the Arduino microcontroller. The Arduino microcontroller processes the information for controlling the air valve system what directly is connected to the McRobot. At the end of the setup there will be done some experiments to update and improve coding scripts.

First experiment: Omega 6 to ROS

The first experiment was to try the connection between the Omega 6 and the ROS system (see figure 6.1). There were encountered some problems with receiving the coordinates from the Omega 6. The commands that are used for extracting information from the Omega 6 joystick are needed for receiving the orientation coordinates. When coordinate information of the Omega 6 is received, the sharing system between the Omega 6 joystick and ROS has been completed.

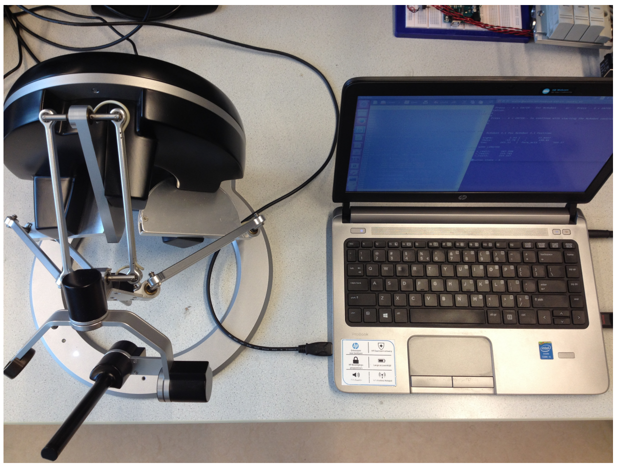


Figure 6.1: *Omega 6 joystick connection with ROS.*

Second experiment: ROS to Arduino

For the second experiment the connection to publish something to the Arduino took some time for setting up the system. The Arduino microcontroller needs to be updated and needs to be serial connected (see figure 6.2) with a node that can be made with the used terminal commands in section System setup instructions . The Arduino code can extract messages from the ROS environment with a callback function. This callback function was also first used for setting the actuators to their designated position. What happened to the code when using the Callback function for steering the actuators is that the calculation and turning of the actuators on or off is sometimes interrupted for getting new information. For the real-time control loop system this gave a large problem. To fix the problem there has been made a separate function and time step maker in the void loop for receiving coordinates and controlling the actuators in a certain time. When the Arduino communication has been setup the Arduino code needs to have the kinematic calculation system.

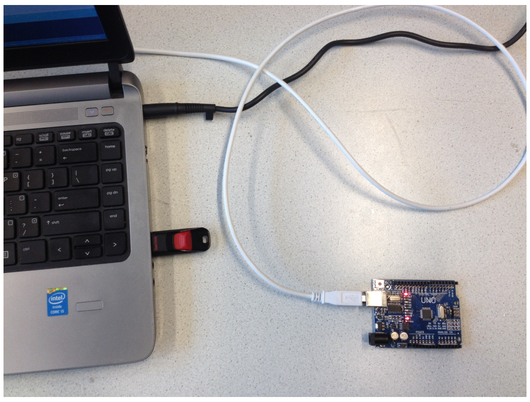


Figure 6.2: *ROS connection with the Arduino microcontroller.*

Third experiment: Arduino to McRobot

After making the connection with the ROS environment to the Arduino microcontroller, the kinematics of the McRobot can be setup in the Arduino. The kinematics changed during the experimental phase a lot. Concept 2 and 3 were tested for finding the best possible concept for calculating the kinematics. Concept 3 was made with making a publisher that sends data to ROS where the kinematics for the McRobot were calculated. With concept 2 the computational power of the Arduino microcontroller is used for doing the calculations.

The Arduino receives first information about the orientation of the Omega 6 joystick. Eventually the ROS code of the McRobot calculates the kinematics of the McRobot to the wanted lengths of the actuator wires for McRobot 3.1. The difference of the wanted lengths and the real lengths of the wires is used for setting the McRobot 3.1 to the position where the difference (delta) between the values is almost zero. This tracking system has been made and used for keeping track of the actuator position.

Fourth experiment: Air system

The air system is tested with several codes from the Arduino microcontroller and seems to work after some struggles with the callback function. The air system consists of 15 air valves that can be activated by the Arduino microcontroller and electronic circuit connection. The air valves can be tested without air pressure on the valves. When a valve is activated, the valve turns a red led on to show the valve is opened. The air system is tested with the made codes for the Arduino microcontroller and the results can be viewed in the made "controlled air system" video.

For steering the old McRobot actuators there is a lot of pressure needed. During the testing of the total connection of the McRobot there has been found a problem with steering multiple actuators at the same time. This problem is told in the next section with possible solutions to solve the problem.

Problem encounter McRobot 3.1

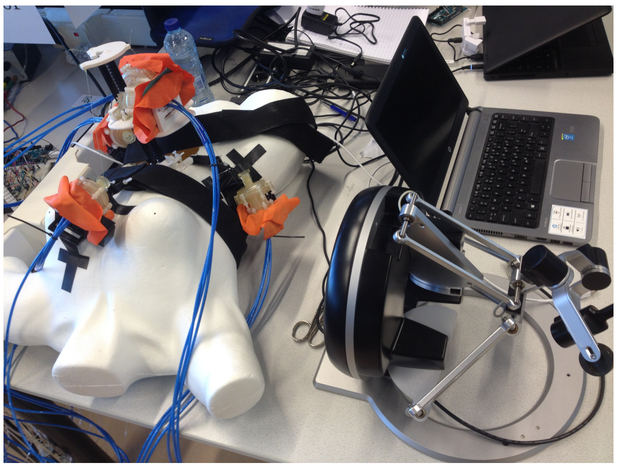
When testing the robot with the made human operator control loop systems there was found a problem. The actuators of the McRobot 3.1 are not well air sealed what gives the problem that the pressure of the system can drop down to one bar when using a system pressure of 4.5 bar. When one actuator is used the pressure of the system drops to 2.3 bar what is the minimum pressure for steering an actuator. When using all three actuators at the same time, there can be found a rapid decrease of air pressure of the system. The pressure can drop down to 1.2 bar what is not acceptable for working with a surgical robot. The robot system is very unstable with such a low system pressure. The problem can be solved with several solutions:

- Increasing the system pressure from 4 bar to 6 or 7 bar.
- Using improved actuators that keep the pressure high.
- Using a different robot with less actuators.

The first option of increasing the system pressure is simple to apply. The actuators can work with an air pressure of 6 or 7 bar and the system pressure can easily be increased in the biomedical RaM lab. The only disadvantage is that the danger increases with using more force to set the actuators to their position. When one of the devices cannot hold the pressure, the damage is much greater when using more pressure for the system.

There are new improved actuators designed that are better air sealed, more precise and stronger than the previous actuators. The newest actuator is very large and is changed in lengths. This gives the problem that the McRobot design needs to be changed for placing the new actuator in place. This is done by Xiaolong and will be 3D printed in the lab.

The last option is to use another McRobot design that requires less actuators for steering the needle to the right orientation. In one of the other bachelor assignments there is made another McRobot (3.2) that uses two of the newest actuators. It is a total different approach of kinematics in comparison with the McRobot 3.1, but the kinematics are not very difficult. For using this different McRobot there needs to be made another script for switching between McRobot systems. The final result of the McRobot 3.1 can be viewed in figure 6.3.


(a) *McRobot setup with the Omega 6 joystick, ROS and the McRobot 3.1.*

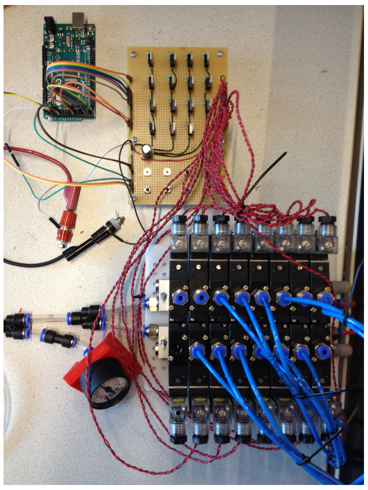

(b) *The electronic board system and the air valve system.*

Figure 6.3: *The McRobot 3.2 mechanical system*

6.2 Board testing

The design of the electrical circuit of the McRobot 3.1 was very unordered and needed some structure. During the bachelor assignment there has been made an electronical circuit that is soldered to a solder board (see figure 6.4). There are multiple improvements made to the electronical board of the McRobot 3.1. The electronical circuit board contains multiple buttons. There are buttons made for steering the actuators or to change the system. The system buttons are the most important feature of the board, because it has an emergency stop for safety that is indicated when the red LED is burning. The other setting button is to switch the emergency stop off or to switch between ROS coordinates or button control for controling the McRobot 3.1. The green LED is burning when the right script is used on the Arduino microcontroller. The yellow LED indicates the system is adjusting the position of the McRobot with blinking. Eventually the blinking of the yellow LED stops when the right orientation is reached. The other part of the electronic circuit board is for selecting the right valves with using the digital pins of the Arduino microcontroller. There are TIP120 transistors used between input and output for transferring an activation signal of 24V. The transistor lets the external voltage supply only go to the output. It protects the Arduino microcontroller for receiving a voltage of 24V.

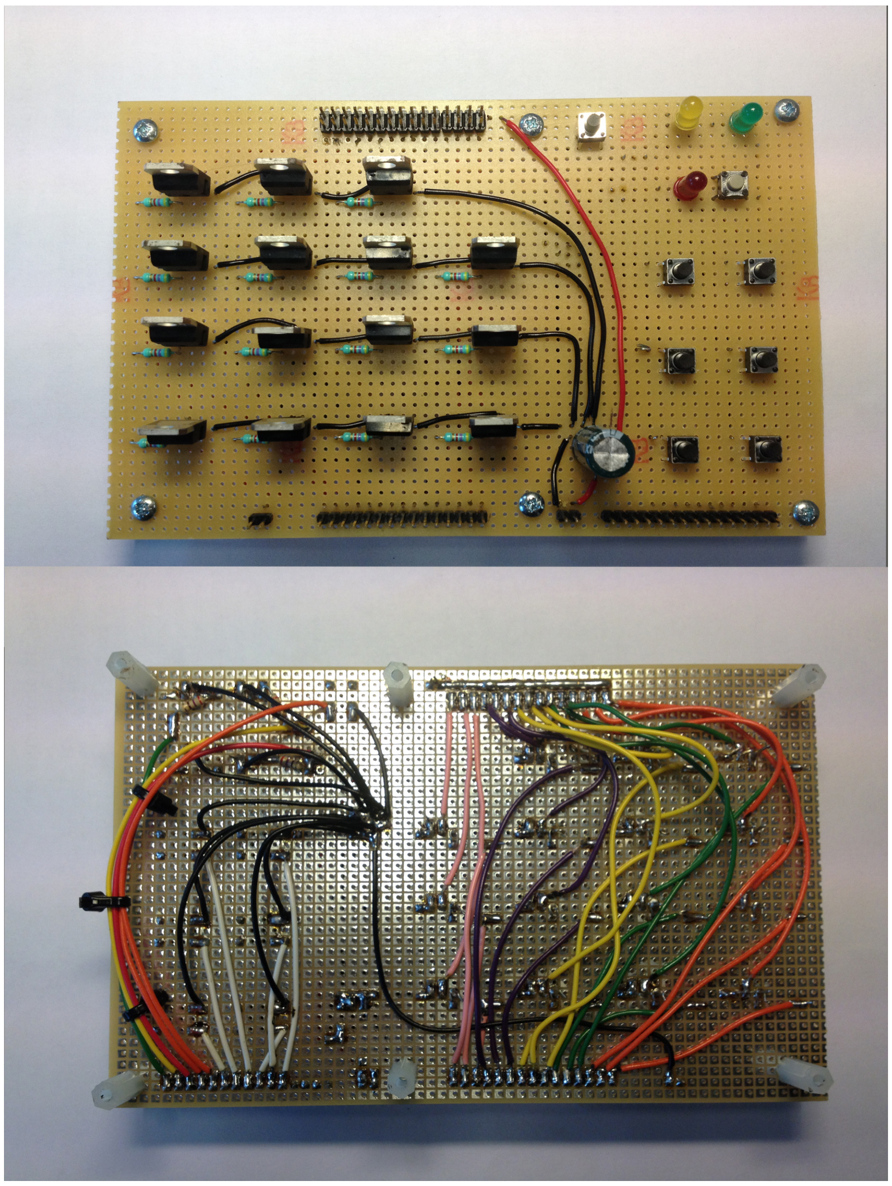


Figure 6.4: *The made electronic board for the McRobot 3.1 with actuator activation, robot control and LED indication.*

The board is soldered and pinned for making a tidy overview of the connections between pins and electronic components. With marking three actuators pins it gives a clear view for finding which transistor is used for which digital port.

At the moment the board has some interconnections what needs to be erased. The board should function when the board is repaired.

6.3 Concept Results

The concepts are tested with connection of all the subsystems that are used for the human operator control loop system of the McRobot 3.2. The McRobot 3.1 was not operating so another McRobot is used for testing the control loop system.

Concept 3

The best founded concept was concept 3. It exist of multiple underlying subsystems in ROS and a simple model in Arduino. The most workload is experienced in the ROS environment.
Concept 3 has been assembled to try operating on the ROS system and the Arduino Microcontroller. The electronic scheme of one actuator has been made for multiple actuators and can work with the kinematic scheme for the McRobot 3.1. This script will be tested for validity and problem shooting.
There have been multiple codes written to test how the publisher and subscribe system of ROS to Arduino functions when using a serial connection.

The Arduino Code
The Arduino code contains a publisher and a subscriber node that will be active at the same time. The code will first extract the information from the ROS message environment and will use this to activate or deactivate the actuators. The information of counting turns is sended to the ROS environment back. The information contains the total counts of how much the system turned the actuator. This is in normal situation zero and can go to positive or negative values of turns.

Results
The testing of the system showed that there are some flaws in the system. The Arduino disconnects at a certain time from the ROS environment with publishing data. This causes that the system cannot track the delta error compensation. The communication between the ROS environment and the Arduino is done with a serial communication what can conflict with each other when sending and receiving messages. But at the beginning when the system runs well with having a publisher the system is working very well. The Arduino code has some problems with loop systems and needs to be changed for a working control mechanism. Concept 2 will be used now for experimenting and as main human operator control loop system.

Concept 2

The biggest part of experimenting and coding is done with concept 2. This code works with an infrastructure that is equal loaded with processing of data. This can be more beneficial in use. Concept 2 has a very well basic concept with slave-master connection what is preferred using human operator systems. Only when implenting force feedback into the system a publisher is needed for giving the ROS environment information.

The beginning of the control loop system was with implenting concept 2. The Arduino code uses only a subscriber and the ROS code is only sending coordinate data of the delta length of the wires. First the wire length was also calculated in the Arduino, but this requires some computational power and is preffered to be done in the ROS code. The code was expanded to a larger code where the Kalman filter was introduced and the total kinematic model was implented. When the code was almost finished it got some changes in the Arduino and ROS code to make concept 3. . Concept 2 seemed to have a better system with use of safety. The communication between ROS and the Arduino is mostly stable, but it is preffered to use as less subscribers or publishers.

The Arduino Code
The Arduino code contains only a subscriber node. The Arduino knows at which location the actuator is located. The actuator uses the difference in length of the wire between the given Omega 6 wire length and that the McRobot has. When there is found a difference, the McRobot will compensate the difference by turning the actuator to the designated orientation.
The Arduino code works with a subscriber node that uses a callback function. It is important to notice that only the messsage extraction takes place in this callback function. If the actuators steering code is placed in this callback function it can give an unstable system for the Arduino control part.

First Prototype
The first prototype of the McRobot control loop system is a simple connection between the ROS code to the Arduino Mega microcontroller. The system will be tested and investigated at improvements in the system.

Results
In the RAM lab for the McRobot design several tests were executed. The first test was to look if the code was executed well. The robot showed responses to the ROS system and with changes to the actuator position. After some tests the actuators could move with the made Arduino code and the digital outputs could be set to the wanted air valve order. The order of air valves is very important to steer the actuators into a direction.

Second Prototype
The code has been improved with new features for working with multiple other subsystems. From the ROS terminal can be chosen to use the McRobot 3.1 or 3.2. The Arduino receives information about which McRobot is chosen and uses the function for the specific McRobot. The steering system of the two McRobots work with a different driving system for the actuators.

Results
The code has on the moment some problems with only using the McRobot 3.2 function. All of the selected digital output pins are activated at a moment what is not wanted. The code of the McRobot 3.2 needs to be tested seperate from the overall McRobot script. The problem in the McRobot 3.2 code can than be located very easy if there is any problem in the code.

Final model
The electronic circuit board has been finished and tested on functionality. The Arduino code of the McRobot needs to be adjusted for using the board. There are multiple new features on the board. System control buttons, LED indication signals and actuator control buttons for three actuators.

Results
During the testing of the board there have been multiple errors that needed to be remade from the beginning. Eventually the board was working and the code worked directly. The board with code is tested on the McRobot 3.2 what showed some interesting results. The McRobot 3.2 was able to turn the actuators for getting almost the same orientation as the Omega 6 haptic joystick (see figure 6.5). This result can be watched in the movie "McRobot 3 2 control movie". The only problem of the McRobot 3.1 and 3.2 is the actuator system. The actuator uses rubber strings for pushing the pistons back, but the rubber strings are not reliable. Sometimes the actuator is stuck and is not able to move, because of the rubber string what is not strong enough to push the piston back. This results in unwanted movements what can be viewed in the movie "McRobot 3 2 control movie".
The parts of the electronic board are operating well with the code. The emergency stop can be executed with a red LED for indication of the stop. Also the yellow blinking LED is working for indication that the actuators are turning. The actuator control buttons can be coded better, but there is no time and also no need for use at the moment. These actuator control buttons are an extra feature for direct control of the McRobot systems.

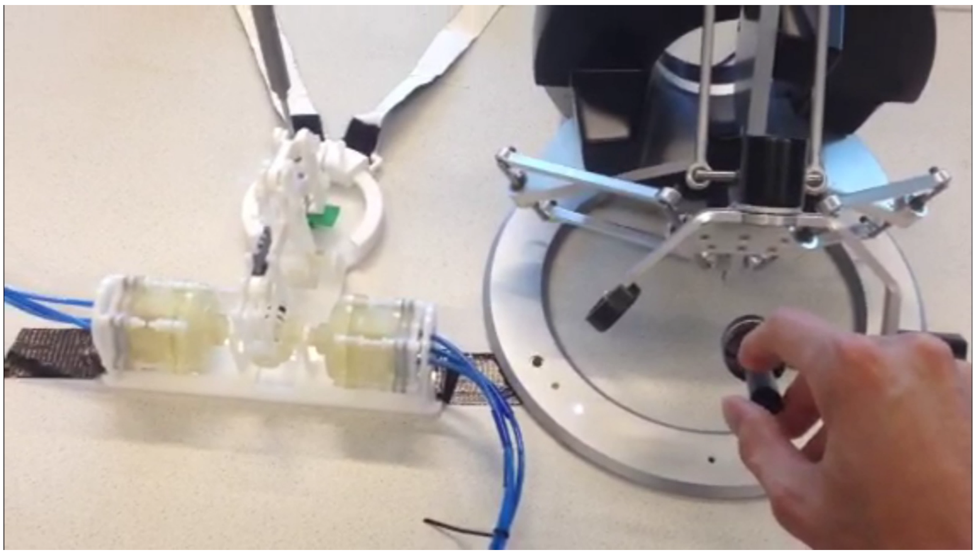


Figure 6.5: *Experimenting with the control system of McRobot 3.2 with using the Omega 6 joystick.*

Chapter 7

Conclusion and Discussion

7.1 Conclusion

Given the succesfully tested control system of the McRobot, it can be concluded that a connection can be made between the human interface and the McRobot controlled by the Omega 6 joystick. The goal of this study was to create a McRobot human in the loop control system using an Omega 6 joystick, ROS and an Arduino Mega microcontroller. This research presented the design of a total control system, from human operator to the real surgical operation. It is found that a link between human interface and Arduino Mega microcontroller can be made using a computer that runs ROS. Finally, an existing breadboard has been improved by soldering and adding new components, resulting in more efficient usage and having a clearer overview when demonstrating it.

7.2 Discussion

This research used C++ as programming language in ROS, including codes for designing the human in the loop control system. However, the codes might be written clearer or shorter in order to be more efficient. Someone with more expertise in the coding languae C++ could improve the code.

Concept 3 could not be used for this control system, but further research needs to be done for investigating if it is possible to use a Raspberry Pi for replacing the PC and the Arduino microcontroller. This could result in a better real-time system in comparison with the made control loop system.

In addition, the encoder system was not yet available for the McRobot actuator system during this research. Despite the absence of the encoder, a system has been developed using a step counter. The step counter did not recognize mechanical system errors. An encoder system would have led to a system that tracks the delta errors of the system.

Furthermore, this research included three possible control systems for designing the real-time robotic system and the concept with the ROS heavy duty system was selected to use. This concept has the most future compatibility in comparison to the other concepts. Unfortunately, the McRobot malfunctioned when a serial communication was made between ROS and the Arduino Mega microcontroller. This error might be due to a communication problem between sending and retrieving data from the ROS environment. The cause could not be determined because of time boundaries and therefore the equal load of duty system was chosen to be used further. This system was succesfully tested during the experiments. The last possible control system, the Arduino heavy duty system, was rejected in the very beginning of this research since this system has almost no future compatibility.

Thus, the results show that the McRobot system is not stable enough to be used in surgical operations due to a missing encoder, mechanical instability and communication problems when a communication appeared between sending and retrieving data from the ROS environment. However, new and improved parts and systems could lead to great improvements for the future. The next section will discuss these possible improvements.

Chapter 8

Recommendations

The designed human in the loop control system of the McRobot can be improved with several adjustments. Table 8.1 shows the possible adjustments that can be applied in future research.

First, future research could establish whether a more advanced filter system for steering the actuators of the McRobot improves the control loop system. A more precise and accurate prediction filter might improve the movements of the SRS system.

In addition, new McRobots could be added to the control system. These new McRobots are not created yet, where future research could do so. New McRobot systems might work on a more stable mechanism which would lead to fewer errors.

Furthermore, this research tried to use Gazebo as a simulation software. It would have resulted in a simulation model of the McRobot, being a backup when the McRobot shows errors. However, designing even a simple model took more time than planned and it did not fit in the given research time. Future research could investigate the possibilities of using Gazebo for simulation. Appendix 4 shows the work done on Gazebo and could be used as a base for future research.

Fourth, there was no encoder system available during this research. An encoder system should be designed for the McRobot in order to track the delta errors. The McRobot would be more accurate when mechanical errors would be resolved.

Also, the direct connection between the Omega 6 and the Arduino cannot be made at the moment, because the drivers of the Omega 6 joystick are not compatible with the Arduino system. Future research could make own drivers for the Omega 6 joystick resulting in a proper connection.

Lastly, future research can rewrite the McRobot code and test it for functionality. Improved codes would lead to a more efficient robotic system.

	Future research	Description
1	Filter system	Adding advanced filter systems that are new and can be tested
2	New McRobot	Adding new McRobot systems to the code
3	Simulation	Making a simulation of the McRobot and connecting to ROS with Omega 6 steering
4	Encoder system	The McRobot code needs an encoder system to keep track of the turn counts.
5	Direct connection	Make a direct connection between Omega 6 joystick and the used microcontroller.
6	Testing code	More tests for the McRobot code with improvements and updates

Table 8.1: *Possible adjustments to the McRobot system in the future.*

Bibliography

[1] A. Sander and M. Wolfgang, "The rise of robotics," *The Boston Consulting Group, August*, vol. 27, 2014.

[2] J. Guiochet and A. Vilchis, "Safety analysis of a medical robot for tele-echography," in *Proc. of the 2nd IARP IEEE/RAS joint workshop on Technical Challenge for Dependable Robots in Human Environments, Toulouse, France*, 2002, pp. 217–227.

[3] D. Richert and C. Macnab, "Direct adaptive force feedback for haptic control with time delay," *TIC-STH*, 2009.

[4] F. dimensions, "Omega.6 overview."

[5] ——, "specsheet-omega.6."

[6] H. King, T. Low, K. Hufford, and T. Broderick, "Acceleration compensation for vehicle based telesurgery on earth or in space," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.* IEEE, 2008, pp. 1459–1464.

[7] A. Simorov, R. S. Otte, C. M. Kopietz, and D. Oleynikov, "Review of surgical robotics user interface: what is the best way to control robotic surgery?" *Surgical endoscopy*, vol. 26, no. 8, pp. 2117–2125, 2012.

[8] R. Sofronia, G. Savii, and A. Davidescu, "Haptic devices in engineering and medicine," in *Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on.* IEEE, 2010, pp. 373–378.

[9] X. Zhi, "Design and prototyping of mcrobot version 3.1: Multi-modality compatible robot for image guided minimally invasive intervention and therapy," *Confidential*, 2015, mSc Thesis.

[10] D. Kleiboer, "Design and prototype of mcrobot 3.2," 2016.

[11] P. W. Jadran Lenarčič, *Advances in Robot Kinematics.* Springer, 2008, ch. Methods in Kinematics, pp. 3–65.

[12] S. Kucuk and Z. Bingul, "Comparative study of performance indices for fundamental robot manipulators," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 567–573, 2006.

[13] M. G. Sobell, *A practical guide to Ubuntu Linux.* Pearson Education, 2014.

[14] J. M. O'Kane, *A Gentle Introduction to ROS.* Independently published, Oct. 2013, available at http://www.cse.sc.edu/~jokane/agitr/.

[15] A. Bihlmaier, T. Beyl, P. Nicolai, M. Kunze, J. Mintenbeck, L. Schreiter, T. Brennecke, J. Hutzl, J. Raczkowsky, and H. Wörn, "Ros-based cognitive surgical robotics," in *Robot Operating System (ROS).* Springer, 2016, pp. 317–342.

[16] E. Glassman, W. A. Hanson, P. Kazanzides, B. D. Mittelstadt, B. L. Musits, H. A. Paul, and R. H. Taylor, "Image-directed robotic system for precise robotic surgery including redundant consistency checking," Feb. 4 1992, uS Patent 5,086,401.

[17] J. L. Prince and J. M. Links, *Medical imaging signals and systems.* Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[18] K. Coyne, "Mri: a guided tour." MagLab, 8 Januari 2016.

[19] W. DeMartini, C. Lehman, and S. Partridge, "Breast mri for cancer detection and characterization: a review of evidence-based clinical applications," *Academic radiology*, vol. 15, no. 4, pp. 408–416, 2008.

[20] P. Kazanzidesf, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci® surgical system," in *2014 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2014, pp. 6434–6439.

[21] "conversation radiologist."

[22] M. Haghighipanah, Y. Li, M. Miyasaka, and B. Hannaford, "Improving position precision of a servo-controlled elastic cable driven surgical robot using unscented kalman filter," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE, 2015, pp. 2030–2036.

[23] "Surgical robots, part 1," in *Medical Robotics*, 30-06-2016.

[24] "Zeus robotic surgical system," in *All About Robotic Surgery*, 30-06-2016.

[25] S. A. Antoniou, G. A. Antoniou, O. O. Koch, R. Pointner, and F. A. Granderath, "Robot-assisted laparoscopic surgery of the colon and rectum," *Surgical endoscopy*, vol. 26, no. 1, pp. 1–11, 2012.

[26] M. Eto and S. Naito, "Robotic surgery assisted by the zeus system," in *Endourooncology.* Springer, 2005, pp. 39–48.

[27] G. Taylor, J. Barrie, A. Hood, P. Culmer, A. Neville, and D. Jayne, "Surgical innovations: Addressing the technology gaps in minimally invasive surgery," *Trends in Anaesthesia and Critical Care*, vol. 3, no. 2, pp. 56–61, 2013.

[28] P. Gomes, "Surgical robotics: Reviewing the past, analysing the present, imagining the future," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 2, pp. 261–266, 2011.

[29] M. J. Lang, A. D. Greer, and G. R. Sutherland, "Intra-operative robotics: Neuroarm," in *Intraoperative Imaging.* Springer, 2011, pp. 231–236.

[30] U. of Calgary, "3 t intra-operative mr scanner," 30-06-2016.

[31] G. R. Sutherland, P. B. McBeth, and D. F. Louw, "Neuroarm: an mr compatible robot for microsurgery," in *International congress series*, vol. 1256. Elsevier, 2003, pp. 504–508.

[32] G. R. Sutherland, S. Wolfsberger, S. Lama, and K. Zarei-nia, "The evolution of neuroarm," *Neurosurgery*, vol. 72, pp. A27–A32, 2013.

[33] 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.

[34] A. M. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery," *Current opinion in urology*, vol. 19, no. 1, p. 102, 2009.

[35] S. F. Barrett, "Arduino microcontroller processing for everyone!" *Synthesis Lectures on Digital Circuits and Systems*, vol. 8, no. 4, pp. 1–513, 2013.

[36] T. Babb, "How a kalman filter works, in pictures," 11-August-2015.

[37] Gazebo, "Robot simulation with ros."

[38] M. Bailey, K. Gebis, and M. Žefran, "Simulation of closed kinematic chains in realistic environments using gazebo," in *Robot Operating System (ROS).* Springer, 2016, pp. 567–593.

Appendices

Appendix **A**

Appendix

A.1 Setup system instructions

The McRobot human operator control loop system needs to be setup from the terminal commands. In this subsection there will be told how the connection of the Omega 6 joystick to the McRobot can be made.

1 The control loop system needs three terminal screens for different tasks. Start the terminal screens for further working.

2 The first terminal is used to turn the ROS system on with the command "roscore" that will generate in the terminal information of the distribution and about the system specifications. This command is needed to let the different nodes communicate with each other.

3 When ROS is operating, the system can run the used code for receiving, processing and sending data to the McRobot. To start the code there are multiple steps needed to secure no problems. First go to the made directory where the catkin packages can be made. This can be done by looking at the tutorial of making a catkin package. The directory contains a "/devel" directory where the setup.bash file can be found. The setup.bash file is needed for letting the ROS environment know this package is used by ROS. For setting up this file the following commands are needed:

a. cd /directory/devel
b. source setup.bash
c. cd .. (2x)

Place the Omega6 directory in the /src directory of the made catkin-package. The next thing is to compile the whole code of the catkin package. To start the compiling the command "catkin-make" in the directory where the catkin-package is setup needs to be compiled. When the compiling does not work or cannot find anything there can be a problem in the code or the setup.bash file is not founded. Another thing is to find the red line marking in the terminal. When this marking shows up the compiling is done with changes.

4 The Omega 6 haptic joystick can be plugged into the laptop. The commands "lsusb" shows the available devices that can be used for the system. The Omega 6 is an usb device that has no name. Sometimes there are multiple devices available with no name that can create confusion for which is the omega 6. To know which port is occupied by the omega 6, pull the usb connection of the omega 6 out of the laptop and look which port connection is gone. When the Omega 6 port is known (port 5), the connection can be established by the following command with the number of the port: sudo cmod 666 /dev/bus/usb/001/005.

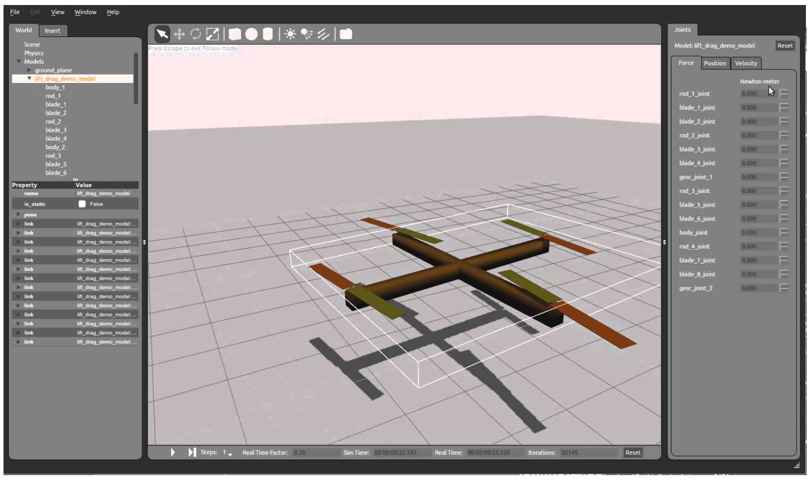
5 The Omega 6 is connected and can the code can be run for receiving coordinates of the Omega 6. The command is "rosrun omega6 omega6".

6 Next is to connect the Arduino Mega microcontroller. Start the Arduino software with the given code. Look in options if the Arduino port settings compile the code to the right board and port. The code can be send to the Arduino and the connection between ROS and the Arduino Mega microcontroller can be established. To use this command the port of the Arduino microcontroller needs to be known (ACM0). The command for starting the connection is: rosrun rosserial-python serial-node.py /dev/ttyACM0.

7 Check if the robot reacts to movements of the end-effector of the Omega 6 pencil.

A.2 Gazebo Information

One of the other packages that is included in the ROS package is the software called Gazebo. Gazebo is a free simulation program for particular the operating system Linux [37]. The software gives the possibility for coders to do simulations with the made code in ROS. The ROS code can communicate with the Gazebo environment to do control functions for inputs or outputs[38].



Property	Value
name	lift_drag_demo_model
is_static	False
pose	
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...
link	lift_drag_demo_model:...

Model: lift_drag_demo_model

	Newton-meter
rod_1_joint	8.000
blade_1_joint	0.000
blade_2_joint	0.000
rod_2_joint	8.000
blade_3_joint	0.000
blade_4_joint	0.000
gear_joint_1	0.000
rod_3_joint	8.000
blade_5_joint	0.000
blade_6_joint	0.000
body_joint	0.000
rod_4_joint	8.000
blade_7_joint	0.000
blade_8_joint	0.000
gear_joint_2	0.000

Steps: 1 Real Time Factor: 0.36 Sim Time: 00 00:00:32.145 Real Time: 00 00:00:25.250 Iterations: 32145

Figure A.1: *The screen of Gazebo with simulation of a quadcopter.*

The McRobot can be made in Gazebo. To setup the right settings of Gazebo it is recommended to reinstall the dependencies. The package that is included and installed with ROS is an old version of Gazebo and needs to be reinstalled for better model making options. The McRobot can be made with editing the sdf files (see figure A.1). This is very hard to do and needs a long time to master the coding of the system. The new made dependencies that are achieved by installing a newer version of Gazebo includes new features and improvements. The model editor is one of the newest functions and is recommended for use with Gazebo. The model editor can be found when clicking the edit function (left upper corner of screen) and finding the model editor option. The model editor gives you the benefit for easy making robot models.
Model editor has two important functions for making a total new robot from scratch. The first function what is needed is the part maker. There are 3 different parts to choose from: Cube, Ball and the cylinder. The parts can be used to make full link models that are complex or separate models for fixation at a later time. With double clicking on the object, the size and orientation of the part can be edited to all six DOF.
When the parts are made and set to the right size and orientation, the joints of the parts can be made. There are several joints available for use in Gazebo and the following joints can be used for the McRobot. The McRobot works with a wiring system what is difficult to make in a simulation program. For making the same system working in a simulation, there are three types of joints made. The first one is fixation and is used to fix the base parts to each other. The second joint is a ball joint. This joint can rotate like the ball in a mouse wheel in every direction. The last joint is the translation joint that is used to shorten or longer the length of a wire.