BACHELOR THESIS

ESTABLISHING A TCP CONNECTION BETWEEN A GRAPHICAL USER INTERFACE AND THE CONTROLLER OF A ROBOT

Emma Sombroek

SURGICAL ROBOTICS LAB
PROF. DR. S. MIRSA

EXAMINATION COMMITTEE
C.M. Heunis
Prof. Dr. S. Misra
I. Tamadon

DOCUMENT NUMBER
BE - 933

UNIVERSITY OF TWENTE.

MAY 2023

UNIVERSITY OF TWENTE.

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF BIOMECHANICAL ENGINEERING

Declaration of consent

By means of this declaration of consent I, Emma Sombroek S2246023 (name student and student number), give permission to the University of Twente, located at Drienerlolaan 5, 7522 NB in Enschede, to use work from Biorobotics (course name), including my name (from here on together referred to as 'work'), under the conditions as described below:

1. I give permission to use my work *for educational purposes during the* course or module: Biomedische Technologie
   for study program(s): ______
   I give permission to use:
   ◯ All my work from the specified course (pictures, drawings, prototypes, report, code, etc.)
   ◯ All of my work, except the following: ______
   [✓ None of my work]

2. I give permission to use my work for future analysis, modification and (scientific) publication
   ◯ All my work from the specified course (pictures, drawings, prototypes, report, code, etc.)
   ◯ All of my work, except the following: ______
   [✓ None of my work]

For reuse of my work I will always receive proper attribution according to academic and/or licensing standards and in compliance with the Copyright Act and/or will receive (co)authorship of manuscripts according to academic standards.

My permission only applies to the specific data, institution and study program(s) described above. The university needs to request my permission again to reuse my work for any purpose not covered in the agreement stated above. I can withdraw my consent at any time. I am aware that some copies of my work may be made and used exclusively for own exercise, study or use of the person who makes the copies or gives the assignment to make the copies exclusively for his/her own use (article 16 b (1) Copyright Act).

Name student + student number	Name of supervisor + employee number
Emma Sombroek S2246023	Christof Heunis m76636125
Date	Date
18/5/2023	23-05-2023
Student's signature	Supervisor's signature
[illegible]	[illegible]

Abstract

Flux Robotics is a company performing breakthrough research, within the aim to revolutionise the field of minimally-invasive surgery. This is done by developing innovative magnetic robotic technologies that allow more precise and less invasive procedures. This is a very complex research with a lot of steps with respect to algorithms and programming. This thesis focuses on the development of a reliable and stable connection between the graphical user interface (GUI) and the KUKA Sunrise Cabinet, the controller of the Flux One System. This research project aims to replace MATLAB with the KUKA Sunrise Workbench as a suitable platform. Ending with a TCP connection between Qt creator, were the GUI is developed on, and the KUKA Sunrise Cabinet.

The research begins by establishing the necessary requirements set up by the Food and Drug Administration regulations for software development in medical devices, and the requirements set up by the company, Flux Robotics. A process view chart is then developed, outlining the components of the system, interactions and data flow. Through out this process, using KUKA Sunrise Workbench as a replacement for MATLAB, is confirmed.

After this process is developed and the inputs and outputs are understood, the project proceeds with the development of an internal TCP connection. This internal connection is formed on one laptop with Ubuntu as its operating system. The connection is made between Qt Creator (C++) and Visual Studio Code (VSC) that uses Java as programming language. This requires an understanding and adaption of TCP communication protocols in both programming languages.

After the TCP connection is established, the focus shifts to establish a connection between the GUI code and the KUKA Sunrise Cabinet. This involves physically connecting the device, with the GUI code, to the KUKA Sunrise Cabinet and configuring the networks settings. The resulting connection enables robotic control and operation from a GUI on a tablet, within the Flux One System.

Throughout the research, compliance with FDA and company requirements, the development of a process chart, the implementation of an internal TCP connection, and the establishment of the GUI-to-KUKA Sunrise Cabinet connection are accomplished. The successful outcome demonstrates the capability of the Flux One System to now be operated via a connected device with the GUI programmed on it.

This research contributes to the advancement of medical device control, improving patient outcomes. The findings highlight the potential for further development and implementation of the Flux One System in hospitals.

Contents

1 Introduction

1.1 Background and motivation

Flux Robotics [1] is developing a surgical robotics system called the Flux One System. Flux Robotics will be referred to as 'the company' thoughout this report. This system is build and designed to help surgeons perform a fenestrated endovascular aneurysm repair (fEVAR) [2] , via the help of a magnetic guide wire that is guided by a robotic arm with the end effector giving direction also as a magnet. The surgeon will control and command the robot via a tablet. In the operation room, both the company, Flux Robotics, and the Food and Drug Administration (FDA) [3] have set strict requirements. The Flux One System is a medical device, therefore the regulations are set up.

Previously, MATLAB [4] was being used as an application to program the robot. This application does not meet the requirements of the company, because it is not an open source application. Matlab should be replaced by an alternative programming application or method that does meet the requirements.

Before a patient is treated in the operation room a complex process has passed. In this process the robot is prepared to perform the surgery as efficient as possible. Currently, the inputs and outputs from each step of the procedure, from diagnosis to surgery, are unclear. To find the most suitable application that should be installed on the controller to combine all the inputs, the entire process must be viewed.

A Graphical User Interface (GUI) [5] is programmed on a tablet to make it accessible to people who do not have any background knowledge of the Flux One System's hardware and software. This user interface should be connected to the robot's controller so that it can send commands. However, this connection has various requirements, including safety and stability. In this research, these requirements are explained, and the best way to establish the connection is argued and developed.

Because this connection is happening in real time during a surgery, it should be safe and reliable. To develop this connection a transmission control protocol (TCP) [6] should be followed and established. However a TCP connection in a surgical robot presents various challenges that need to be adressed for an efficient operation. Without the connection of the GUI and the controller the Flux One System cannot be used in the operation room.

A TCP is a communication protocol that ensures reliable and ordered data transmission over networks [7]. It facilitates the exchange of data between different devices. In the FLux One System this would be the controller of the robot, with the tablet. However setting up such a communication comes with challenges and complexities that should be taken into consideration to ensure optimal performance and safety for the patient.

Finding a solution to remove MATLAB from the process and cover the requirements by mapping the function of each step taken in the process, will improve the efficiency and accuracy of the company. Developing a TCP connection for the Flux One System enhances the accuracy and precision of the fEVAR surgery and will improve patient outcomes. Using a tablet in the operation room gives the surgeon more space to work, since they do not have to work directly with the controller but use the developed GUI for this.

1.2 Previous work and limitations

In the development of the Flux One System, some advancements have been made in previous work. To start with the hand-guiding code [8] that is already developed using Matlab . This hand-guiding code enables the surgeon to move the robot manually without the use of a joystick or tablet. However, a limitation with the Matlab implementation is its cost, as it requires a license that can be provided by a high expense. To address the requirement of cost-free and open source, an alternative approach is needed.

While significant progress has been made in the preparation steps of the Flux One System, establishing a connection between the designed GUI and the controller of the robot has not been made. The steps taken before the procedure starts are documented and used for this research.

There is a large volume of published studies [9] describing the importance of developing a standard, open and expandable communication protocol. These protocols are aiming for a real-time transmission without latency. The composition of image data packets [10] is for example analysed and used to research the TCP connection. Specifically network communication protocols are developed to integrate on image- guided procedures. DICOM [11] is a known standard for transferring image data through TCP/IP networks.

1.3 Contributions of this study

This paper researched the alternative options for MATLAB, and how this should be implemented in the Flux One System. Regarding the requirements set up by the company and the FDA. Also the object of this study is to develop a process view chart to understand function in each phase and how they should come together. This will lead to a connection that will be formed between the tablet and the controller of the robot.

All of these points described above are beneficial to the company since it will give it a better understanding, an improvement on the requirement and a finished connection.

1.4 Research question

In the context of the Flux One System, which specific subsystems and processes interact with each other and how is a TCP communication established between the designed graphical user interface in the program Qt on Ubuntu, with the KUKA Sunrise Cabinet on Windows which is the controller of the robot?

This research question will be answered by different deliverable described in chapter 1.5.

1.5 Main deliverable

This research holds several key deliverable aimed at establishing a connection between the different programs used in the Flux One System. These deliverable are outlined below:

1. *Set up the requirements for the application replacing Matlab on the controller.* Defining the necessary specification for the new application that will replace Matlab on the controller. This step requires considering both the requirements established by law, as well as the requirements set up by the company itself. The aim is to get a clear framework of the requirements so that the application can be implemented.

2. *Process description including a process view chart.* This step aims to gain a clear understanding of the connections, programs involved, and the different connections being made during the entire process. By mapping the process and identifying the inputs and outputs with each step, this step will provide an overview of the workflow.

3. *Understanding Java and C++.* The programming language of the KUKA Sunrise Cabinet, the controller of the robot, is Java [12]. The language of Qt [13], the program used to design the GUI is C++ [14]. Both these languages are no prior knowledge from the study this research is done from, Biomedical Science. Therefor these languages need to

be learned and understood in order to form the connection. This is done by analysing TCP connection examples implemented in both languages. To gain knowledge in these languages, the adaptions can be made to integrate them into the Flux One System.

4. *Develop an internal TCP connection one the Flux Laptop.* The Flux Laptop has the Qt applications installed on it, and runs on the operating system Ubuntu. The Java language is implemented in Visual Studio Code (VSC) [15] that also works on Ubuntu.

5. *Develop a connection from the Flux Laptop to the KUKA Sunrise Cabinet.* The laptop can be connected to the controller, and after adjusting the Java code first used on VSC to implement it on the KUKA Sunrise Workbench [16] this connection should also be made. At this step the SmartPad should be used as interface to visualise the connection made.

6. *Develop the final connection between the Flux PC that has the designed GUI on it, to the KUKA Sunrise Cabinet.* The final step is to obtain the connection between the Qt code that contains the GUI with the robot controller. This is how the situation in real life will happen and thus the final deliverable.

1.6 Thesis layout

This paper is organized as follows: Section 2 describes the literature review written on this research. This section covers the theoretical approach on answerring the research question. This is followed up by a system analysis and design, in section 3. Here the entire system is described and the working of both the controller of the robot, the GUI and the TCP connection are explained. In section 4 the requirements for the implementation are described. Section 5 shows the results of the succesful TCP connection and the developed process chart. Finishing with in section 6 the conclusion and future work.

2 Literature review

2.1 Requirements for the application on the controller

When designing a medical device, for every step there are different requirements set up. These requirements differ per country / government. In the United States of America they hold account of the Food and Drugs Administration regulations [3]. The FDA classifies medical devices in 3 different classes.

In the European Union they have the IEC regulations. When looking at the software from a medical device, the requirements are set up based on the IEC 62304 [17]. This regulation is for medical device software, the software life-cycle processes. The IEC 62304:2005+A1:2015 gives direction on the general requirements. Within these requirements the focus lays on; the quality management system, risk management, software safety classification and the legacy of the software. In this classification the software is classified in one of the three safety classes. For each different class count different rules to the risk control. The IEC 62304 is not mandatory in the United States nor in the European Union, they have adopted it as part of their regulatory framework.

The company (Flux Robotics) also set up a requirement. The application should be free, thus open source. The reason for this is because the product they want to deliver should not include an extra cost post for the software. Before starting this research the used application was MATLAB. Considering that MATLAB is not free and open source, this does not meet the requirements. Therefor the first question to answer is: What is a suitable application to replace Matlab, that does meets all the requirements.

2.1.1 FDA

In the United States the mandatory regulation is the Food and Drug Administration. In these regulations a difference is made between the software IN a medical device, and the software AS a medical device [18]. The Flux One System is a SiMD because the software used on the robot is not a software on its own. The software is considered a component of the medical device. The software used on the system is used on hardware so therefore it is not a SaMD.

Since the aim of Flux Robotics is to market in the United States, in this research there will be elaborated on the FDA regulations.

The FDA provides different steps on how to develop, validate and test the software. Because in this research there will be no focus on developing a software, but considering already available options, these steps are not taken into account.

Before the Flux One System may be marketed it needs a 510(k) submission [19] . Within the different classifications of the FDA, the Flux One System is in Class II. In this class the devices with a moderate risk are classified. This is because when only looking at the robotics of the Flux One System, it is not classified in the highest risk class. All the medical devices in the second class need a 510(k) submission before going to the market.

Throughout all the different requirements in the Class II of the FDA there are no hard demands on what programming language nor program to use. Despite the advice of using a static language, and the avoidance of SOUP [20]. A static language is a programming language where the type checking is performed at compile time, in stead of at runtime. So the variables and functions are checked by the compiler before the code is executed. This is different from dynamic programming languages, in those cases the checking is performed runtime. The static language is necessary because they catch errors and bugs in the code before the code starts running.

Software Of Unknow Provenance is a software component that is integrated into a larger software system, but of whose origin, development etc are unknown. The use of SOUP in medical devices makes it hard to test the safety and reliability of the software. The FDA requires to avoid the use of SOUP or minimize it as much as possible. These two requirements will be used further in this research.

Comming from these requirements the programming language Java and C++ will be the most suitable to work with. The application used should not make use of SOUP, thus the KUKA Sunrise Workbench is a suitalble option.

2.2 Importance of developing a reliable and stable connection

Whithin the Flux One System the development of systems and applications require the integration of multiple programming languages. This integration requires a reliable and stable connection between these languages. The purpose of this section is to describe the significance of developing a reliable and stable connection between different programming languages, in the context of the Flux One System.

1. Enhanced interoperability [21] Using different programming languages allows the strengths and capabilities of each language. Every language has their own advantage and by letting these communicate via a stable connection the exchange of data and information will be efficient.
2. Special tools and libraries Different programming languages include different libraries, tools, templates etcetera. By developing a reliable connection between different programming languages the power of those special tools and libraries are maintained. These libraries can then be used to accelerate the development process. It makes the process of developing the Flux One System more efficient.
3. Scalability The Flux One System is a complex system that possibly meet evolving needs and will change future specifications. By establishing a stable and reliable connection, the system can be designed such that is stays adaptive. This allows new features and added external systems without disrupting the already working system. The languases do not need to adapt to each other so staying innovative is possible.
4. Stability and reliability. A stable and reliable connection between different languages ensures the overal stability and reliability of the Flux One System. Unreliable connections can lead up to errors, data inconsistencies and communication failures. These mistakes can lead up to the system malfunctioning and brining risk to the patient.

This study set out to determine that it is necessary to obtain a stable and reliable connection between different programming languages used in the Flux One System.

2.3 Different communication protocols

2.3.1 User datagram protocol

User datagram protocol (UDP) [22] is a communication protocol used across the internet for time-sensitive transmission. It can transfer data very quick, but that also causes packets to become lost in the transmission. Therefor it is not guaranteed to be a reliably delivery. UDP does not make a connection before transmitting the data. This makes it faster, but includes more risks.

2.3.2 Hypertext transfer protocol

Hypertext transfer protocol (HTTP) [23] is most of the times used for communication between web browsers and servers to transfer text or other content. It is a client-server protocol, which means requests are initiated by the recipient. The messages sent from the server to the client are called requests and responses. This protocol is based on TCP, it uses it as an underlaying transport protocol for a stable data delivery.

2.3.3 Message queuing telemetry transport

Message queuing telemetry transport (MQTT) [24] is a lightweight messaging protocol, designed for unreliable networks. It minimises bandwidth usage and provides such that it prioritizes message delivery. It allow asynchronous communication, thus enabling real-time data updates.

Considering that a TCP connection is reliable and detects errors to retransmit lost or corrupt packets so the data is transmitted fully, it is the best option to use when letting the GUI communicate with the KUKA Sunrise Cabinet. A TCP is a widely implemented protocol, supported by the operating systems Linux and Windows. So concluding, it makes a well-suited connection protocol to be used in surgical robotics.

2.4 Socket connection

Before a TCP connection can be made a Java and C++ socket [25] should be made. This is done by developing a client and server setup. The client connects, sends message to the server and the server shows them via the socket connection. A socket connection is used to connect two different devices, this is the tablet with the GUI on it, and the controller of the robot. Within a socket connection two devices have information about each the netwerk locations and the TCP port.

```
Socket socket = new Socket("192.170.10.2",23456)
```

The first argument is the IP adress of the KUKA Sunrise Cabinet. This IP adress will be used in the 2 last deliverables. When developling the connection on the Flux laptop internally, the IP adress of the localhost should be used. This is 127.0 0.1 and runs the socket on a single device. The second argument is tge port number. The port numbers should be in sync when developing the codes on the Java part as on the C++ part.

3 System analysis and design

The Flux One System [26] is currently working with the KUKA LBR iiwa 14 R820 arm. This is a lightweight robot specialized in the make way for human-robot-collabaration in workspace. The end effector can hold up to 14 kg, and the reach of the arm length is 820 mm. It has seven axes of motion and therefor is usable for a wide range of tasks. Within each joints there are torque sensors allowing precise force control. The LBR iiwa 14 R820 is designed to collaborate with humans in safe and responsive ways.

The program that is used to program the robotic system of the KUKA LBR iiwa 14 R820 arm is called the KUKA Sunrise Workbench [16]. It is an integrated development environment (IDE) [27], an application that helps programmers develop software code. The applicaion comes with a set of tools and libraries for developing and simulation robot applications. IT is a user-friendly programming environment that uses the programming language Java. In the KUKA Sunrise Workbench code can be written to develop motions and modes for the arm. What also comes in with the KUKA Sunrise Workbench is the simulation option. With this robot programs can be validated and tested. This is used for example when planning the trajectories, joint limints and collision detection.

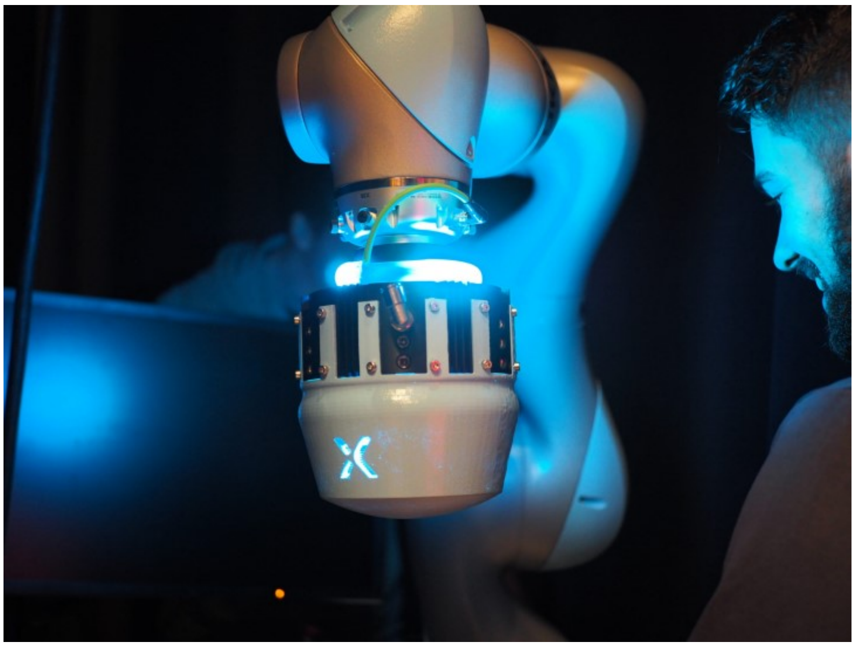


Figure 1: The robotic arm of the Flux One System

3.1 KUKA Sunrise Cabinet

The KUKA Sunrise Cabinet [28] is the controller used in conjunction with the KUKA Sunrise Workbench. It executes the programmed robot applications. The hardware part of the Sunrise Cabinet houses the power supplies and communication interfacecs to connect with the robot arm and other external devices, such as the GUI in the Flux One System. The cabinet ensures the reliable control of the movements of the robot. The Sunrise Cabinet communicates with the Sunrise Workbench vai a TCP/IP communication.

By understanding the working principles of the KUKA Sunrise Workbench and the KUKA Sunrise Cabinet a TCP connection can be programmed and established.

3.2 Graphical user interface

The GUI that is designed for the Flux One System is developed on Qt Creator on Ubuntu. The version of this application is Qt Creator 5.15 [29]. This application is an IDE for developing user interfaces like these ones. The programming language used on Qt Creator is C++. Qt offers libraries and tools that help the development of applications.

To develop a TCP communication using Qt the module QtNetwork is used. This module provides a QTCPSocket class which represents a TCP socket and creates an application programming interface to connect to the host. In the connection between the KUKA Sunrise Cabinet and the Qt program the C++ code makes the QTCPSocket [30] class to create a socket and form a connection with the IP adress and the port of the robot.

3.3 Connection protocols

Trough the TCP connection, data can be exchanged between the C++ and the Java application. In the Flux One System the KUKA Sunrise Workbench and the Qt Creator environment. The C++ code can send commands, data or instructions to the KUKA Sunrise Workbench Java application, to get among other things the robot moving. The Java part can provide feedback and responses to the C++ client. In this way there is an effective coordination and synchronisation between the two different devices.

The TCP connection follows a client-server [31] model. The C++ code on Qt acts as the client, it initiates the connection. The KUKA Sunrise Workbench acts as the server, it waits for the incoming connection. The connection protocol has a bi-directional communication between the Qt client and the KUKA Sunrise Workspace server. This type of communication works with a data flow into both directions so it can effectively control the robots actions.

4 Implementation

The integration of the C++ client with the Java server requires a TCP connection to enhold a communication between the two different applications on different devices.

The hardware requirements are including the following:

- C++ client (the GUI on the tablet)
  - Computer system capable of running the Qt framekwork. So an operating system on Linux (Ubuntu).
  - Network connectivity to establish a TCP/IP connection
- Java server (the KUKA Sunrise Cabinet)
  - Industrial robot system with the KUKA Sunrise Cabinet Controller
  - Network connectivity to establish a TCP/IP connection

The software requirements are including the following:

- C++ client (the GUI on the tablet)
  - Qt version 5.15
  - Qt Creator version 3.0 or higher
  - QtNetwork module for the TCP connection
- Java server (the KUKA Sunrise Cabinet)
  - KUKA Sunrise Workbench software version 1.11

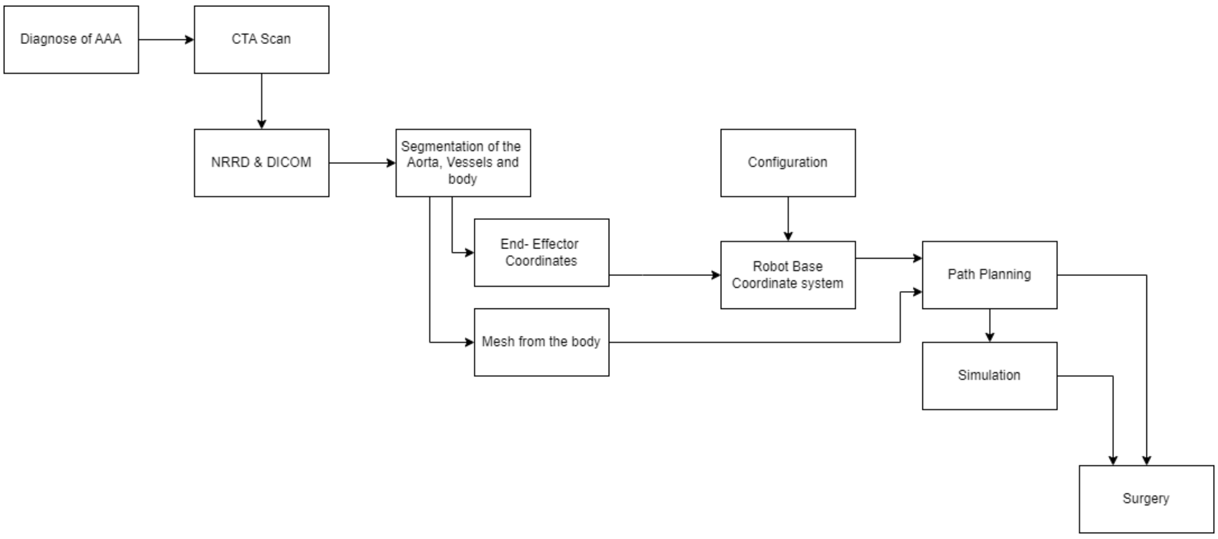
By meeting both the software and hardware requirements, the TCP connection should be able to be developed. The details of this development will be further discussed in chapter 5.

5 Results

5.1 Process description

The first step before starting to build the TCP connection was to develop a better understanding of the actual proces that is happening around the Flux One System to set up and to carry out a surgery.

The first global outcome of the development of the process description is the flowchart seen in image 2. Seen in this diagram is that there is a long way before the procedure starts that is being transferred and prepaired to make the surgery happen.



Diagnose of AAA
CTA Scan
NRRD & DICOM
Segmentation of the Aorta, Vessels and body
Configuration
End- Effector Coordinates
Robot Base Coordinate system
Path Planning
Mesh from the body
Simulation
Surgery

Figure 2: Global flowchart of the process of a patient being treated

When looking at the task devision between steps the hospitle performs, and steps the company Flux Robotics performs, a unified modeling language chart with the use of swimlanes is developed. From this diagram the taskdevision becomes clear. It is also shows the data that is transferred in different steps and the inputs and outputs with every task. This is visuable in image 3.

5.1.1 Consult and diagnose of an abdominal aortic aneurysm

The first step that is taken in the whole process of the FLUX One System is the patient that comes in with an abdominal aortic aneurysm (AAA) [32]. This is a swelling of the aorta. The aorta is the main blood vessel that leads down from the heart to the rest of the body. When the AAA has a size bigger than 5 centimeter across, surgery is recommended. Screening is done with ultrasound, the measure of the aneurysm is done with a CTA Scan.

5.1.2 CTA Scan

After the patient is diagnosed and there is decided that the an EVAR should take place, a CTA scan is made. Cardiac computed tomography angiography (CTA) [33] scan is a noninvisive test that uses X-rays to focus on the coronary arteries. From this scan a 3D image is developed and this will lead to a decision that a FEVAR makes use of the suitable graft method. From this image the exact diameter from the aorta is obtained, both the aneurysm as the normal parts of the aorta. The stent is produced from this scan.

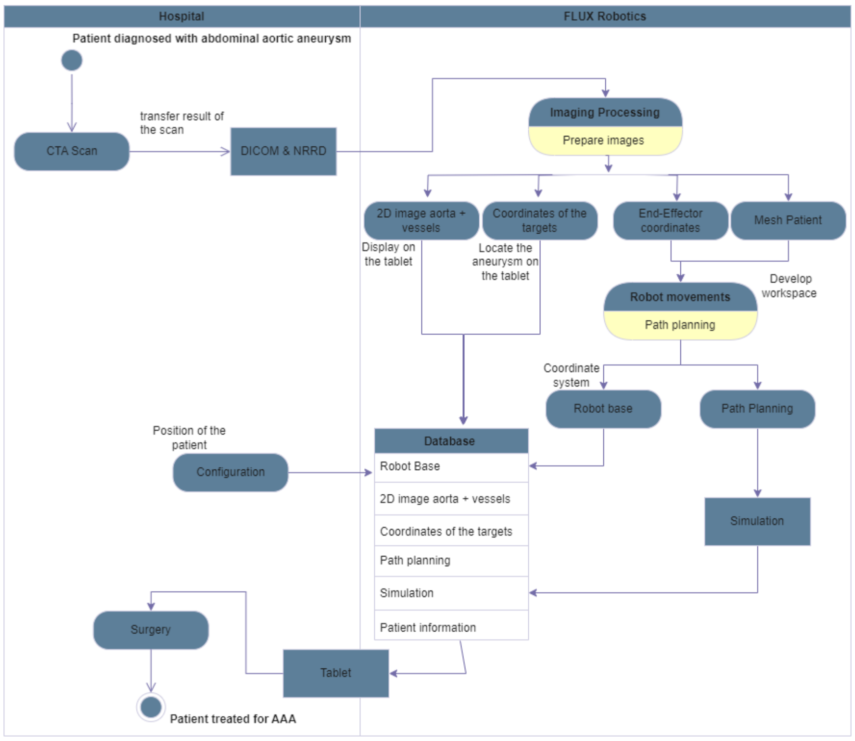


Figure 3: UML Proces view description

5.1.3 DICOM and NRRD

The output of this scan is are DICOM and NRRD files. DICOM [11] is the international standard to transmit medicial imaging information. This protocol is followed and an Nearly Raw Raster Data (NRRD) [34] is transferred to the next step. An NRRD is used for the representation of processing n-dimensional raster data. These files are used in the next step, an algorithm that develops an image of the aneurysm and the patient their body.

5.1.4 Imaging processing

The files from the CTA Scan are processed, compiled and checked so that the full scan can be used. This is important for the development of the actual graft. In this step the NRRD and DICOM files are transferred into the algorithm so the next steps can be done.

5.1.5 Create a 2D image

A 2D view of the scene is created to be displayed on the GUI. The segmentation of the aorta and the branches is used for this process. It is important to show this image on the GUI so the surgeon is able to see the organ before the operation. This is a .png file that is sent to the GUI.

5.1.6 Coordinates of the targets

The last output of the segmentation are the coordinates for the markers. The markers are circles seen on the GUI image of the aorta, at the exact position where an aneurysm is found. The surgeon has to click on the different markers to start the procedure. To give the surgeon the best impression on where the aneurysm is the specific coordinates are sent to the GUI to show it. This is a .txt file.

5.1.7 End-effector coordinates

Then an array with the positions of the end-effector is developed. This is divided in multiple different files. The first one is the .obj file with the model of the patient with the corresponding positions of the end-effector. The second .obj file is the one from the aorta.

The arrays for the data in 3D are written to a single .txt file.

This information is then send to the next (still pre-operative) step. In this step the base of the robot and their joint positions are calculated.

5.1.8 Mesh of the body

From the processed files the first step is that a mesh from the segmentation of the aorta is made. This is done so both the aneurysm is visible and the graft can be made and the path planning can be accurately calculated.

The second step done in this process is a mesh of the segmentation of the outline of the patient. This is done so the robot does have direct contact with the patient, and therefore will cause no harm.

Also in this step the position and direction of the branches are determined. This results in a .png file with a 2D file of the 3D model in which the branches of interest are visible.

5.1.9 Robot movements

The first thing done in this planning is the development of the robot base. This is a coordinate system that is calculated by the configuration. Both of the room and the patient. The values form the .txt file are automatically loaded in and processed into this robot base coordinate system. The position of the patient with respect to the robot is in a different step calculated.

The next part of the procedure is the actual robot planning. The positions of the end-effector are known, and now it is crucial to calculate the joint angles to go to the different position. Some factors that are important to take into consideration are; the boundaries of the patient (the skin), the operation table, the CT scan and the joint limits. These are incorporated in this phase. The goal is to move the robot safely from point a to point b.

To reach the different positions for the end effector joint angles have to be calculated. This is done in the path planning step. Every planned movement the robot makes is calculated in this code. The output of this step is a huge .txt file with all the joint angle for every path.

The desired positions of the end-effector together with the obstacles from the body (such as the branches and other boundaries such as the skin) are known. These in combination are used to develop a planning that manages the graft to travel safe through the vessels to the correct position. This is done by moving the magnetic end-effector in different directions to pull the guide wire.

This path is calculated before the operation is performed.

5.1.10 Simulation

During the pre operative planning a simulation is also developed. This will be shown to the surgeon so they can check the movement the robot will make. This is also a safety prevention because it gets the surgeon aware of the distance the arm of the robot will reach. This simulation will be displayed on the GUI.

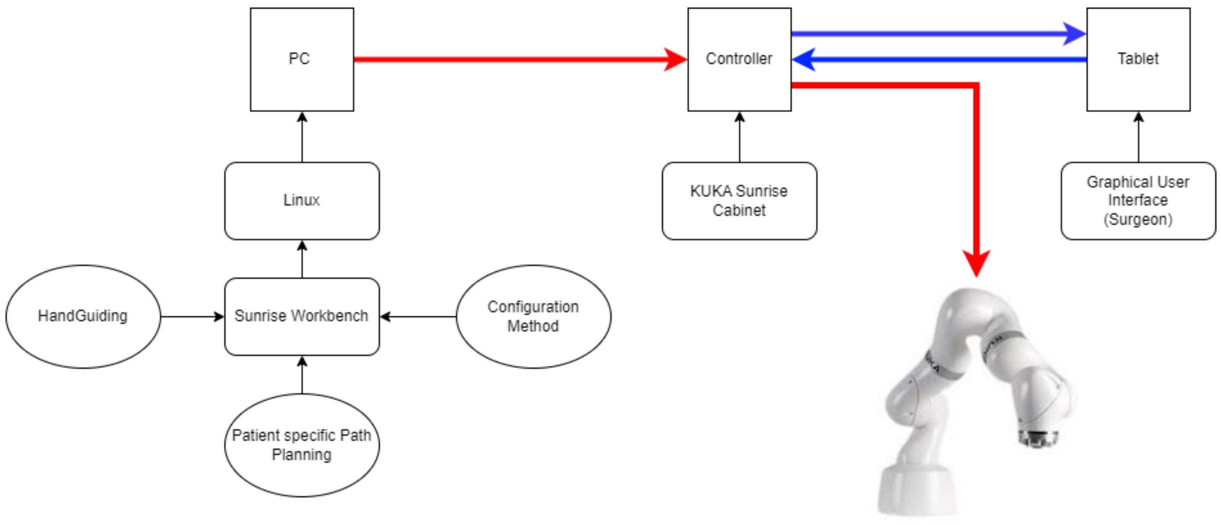


Figure 4: The KUKA robot arm connected to its controller and the rest of the software. The blue arrows indicating the TCP/IP connection.

5.1.11 Database

The database acts as the centre for all the information. In figure 3 the database box contains the robot base, 2D images of both the aorta as the vessels, the coordinates of the targets, the path planning, the simulation and the patient information. The GUI will load all this information so that the surgeon has access to the files.

5.1.12 Configuration

Within the robot base coordinate system the position of the patient with respect to the robot is the most important combination. The desired result for this situation is that the patient is located on the exact same position every surgery, but that is in real life not going to happen. This is desired because than this step can be done pre-operative as well.

To manage the calibration there should be at least one known reference frame. In this situation this is the robot and the operation table. The specific methodology for performing this calibration calculation is undisclosed at present.

5.1.13 Tablet with GUI

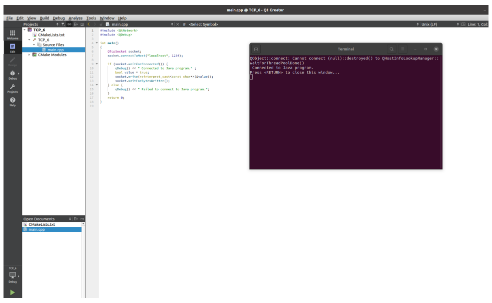
This tablet is the device that wil be used in the operation room as a commanding device for the robot arm. This device is the one that will have the developed TCP connection to the controller of the robot on it. In figure 4 the blue arrows show the TCP connection that is formed to connect the tablet to the KUKA Sunrise Cabinet.

5.2 Internal TCP connection

In this section the implementation and evaluation of the internal TCP connection that is established between Qt Creator and VSC on the same laptop is explained and evaluated.

Through developing and testing the codes on the application Qt and the application VSCO the connection was developed.

In A the code used for the Qt Creator application was used. This code consists of two parts. The first one being the part for the main file, and the second part being the one for the CMake file [35]. This CMake file is a tool to build projects across different platforms, exactly what



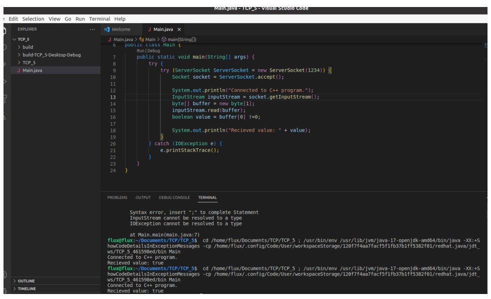
```cpp
#include <QtNetwork>
#include <QDebug>

int main()
{
    QTcpSocket socket;
    socket.connectToHost("localhost", 1234);

    if (socket.waitForConnected()) {
        qDebug() << " Connected to Java program." ;
        bool value = true;
        socket.write(reinterpret_cast<const char*>(&value));
        socket.waitForBytesWritten();
    } else {
        qDebug() << " Failed to connect to Java program.";
    }
    return 0;
}
```

```
QObject::connect: Cannot connect (null)::destroyed() to QHostInfoLookupManager::
waitForThreadPoolDone()
 Connected to Java program.
Press <RETURN> to close this window...
```

(a) C++ on Qt Creator.



```java
public class Main {
    public static void main(String[] args) {
        try {
            try (ServerSocket ServerSocket = new ServerSocket(1234)) {
                Socket socket = ServerSocket.accept();

                System.out.println("Connected to C++ program.");
                InputStream inputStream = socket.getInputStream();
                byte[] buffer = new byte[1];
                inputStream.read(buffer);
                boolean value = buffer[0] !=0;

                System.out.println("Recieved value: " + value);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
        Syntax error, insert ";" to complete Statement
        InputStream cannot be resolved to a type
        IOException cannot be resolved to a type

        at Main.main(main.java:7)
flux@flux:~/Documents/TCP/TCP_5$  cd /home/flux/Documents/TCP/TCP_5 ; /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java -XX:+S
howCodeDetailsInExceptionMessages -cp /home/flux/.config/Code/User/workspaceStorage/120f7f4aa7facf5f1fb37b1ff5382f01/redhat.java/jdt_
ws/TCP_5_461598ed/bin Main
Connected to C++ program.
Recieved value: true
flux@flux:~/Documents/TCP/TCP_5$  cd /home/flux/Documents/TCP/TCP_5 ; /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java -XX:+S
howCodeDetailsInExceptionMessages -cp /home/flux/.config/Code/User/workspaceStorage/120f7f4aa7facf5f1fb37b1ff5382f01/redhat.java/jdt_
ws/TCP_5_461598ed/bin Main
Connected to C++ program.
Recieved value: true
```

(b) Java code on VSC.

Figure 5: Screenshots of the working TCP connection developed internally on one device.

we need in this case. CMake has a different documentation method, therefor this is a different code.

In figure 5b the code for the Java part in VSC is visuable. On the bottem of this image in the terminal the output says 'Connection made to C++ program'. This proves that the TCP connection is deveped on the Java side.

In figure 5a the code for the Qt Creator part is visable. In the terminal, that popped up when building the code, the message says 'Connection made to Java program'. This also proves that the TCP connection is formed on the C++ side.

In summery, the results of the internal TCP connection between Qt en VSC demonstrate the successful establishment of the TCP connection on the laptop. With these results the next step is build to connect to the KUKA Sunrise Cabinet, in stead of the VSC application.

5.3 TCP connection from laptop to KUKA Sunrise Cabinet

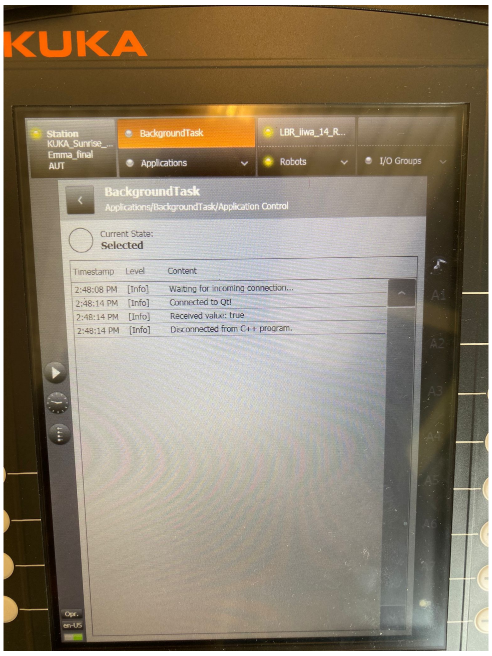
The next step was to establish a TCP connection between the laptop, that contained the C++ code, and the controller of the robot. In the Flux One System the controller is the KUKA Sunrise Cabinet [28]. The connection was successfully achieved by adapting the Java code used in VSC to a Java code compatible with the KUKA Sunrise Workbench [16]. This is the application on the KUKA Sunrise Cabinet that works as an IDE [27]. This adapting involved making modifications to the required packages, their installation and how they were referenced within the code. This code is visible in C.

To establish a physical connection between the laptop and the KUKA Sunrise Cabinet, a standard Ethernet [36] cable was used. The network settings on the laptop were adjusted to match the IP address [37] of the KUKA Sunrise Cabinet. This is done within the Internet protocol version 4 (IPv4) [37]. The IP address of the KUKA Sunrise Cabinet is 192.170.10.2. To connect the laptop with this address, it should be changed on the laptop to 192.170.10.1. Because the last number cannot be the same or it will not form a connection. The netmask number of the KUKA Sunrise Workbench is 255.255.255.0. This one should be the same on the laptop, this is added in the IPv4 options when connecting to the controller.

The port numbers were both set on 12345 to know at what location the connection should be made. The port number is later modified to 23456 because the initial port number resulted in errors during the initial attempts, and after trouble shooting, this port number worked.

After making these adjustments, the TCP connection between the Qt program and the KUKA Sunrise Cabinet was succesfully established. This is seen in figure 6.

To note that during the initial phase of the connection setup, several challenges were encoun-



```
Timestamp    Level    Content
2:48:08 PM   [Info]   Waiting for incoming connection...
2:48:14 PM   [Info]   Connected to Qt!
2:48:14 PM   [Info]   Received value: true
2:48:14 PM   [Info]   Disconnected from C++ program.
```

Figure 6: The KUKA SmartPad displaying the confirmation of the TCP connection, indicating that the connection between the laptop with Qt and the KUKA Sunrise Cabinet was established.

tered, resulting in errors, adjusting the network settings and the port numbers. However, by troubleshooting these obstacles were overcome to achieve a stable and functioning TCP connection.

These results lead to further development of the integration of the tablet in the Flux One System. The final step will be integrating the C++ code written for the TCP connection in the entire C++ code written for the GUI.

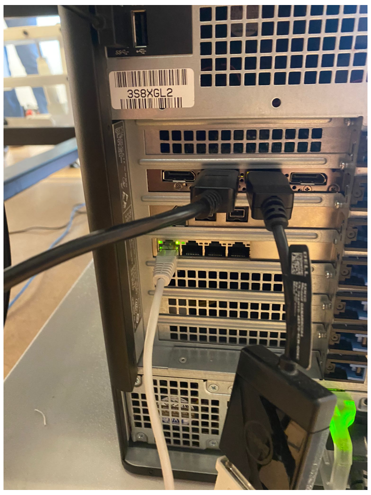


Figure 7: The most left port is used, and the settings are changed in order to this inport. The grey cable is the ethernet cable used.

5.4 TCP connection from the GUI to the KUKA Sunrise Cabinet

After finishing the connection from the laptop to the controller, the last step is to implement the C++ code in the developed GUI code. This step is crucial to validate the overall functionality of the connection.

To enable the TCP connection some adjustments were made in the CMake file. This is done because the CMake file tells the application were certain libraries are installed. For a TCP connection the QtNetwork package is needed. This was done by adding the piece of code:

```
 target_link_libraries(GUI_FluxOne
Qt5::Network)
```

Additionally, a different cable was used to established the physical connection between the computer running the GUI and the KUKA Sunrise Cabinet. This is done because the other cable was too short to reach that connection. In figure 7 is seen in what port the Ethernet cable is placed.

After several iterations and testing, the connection between the GUI code and the KUKA Sunrise Cabinet gained the same successful result as seen in 6.

6 Conclusion and further research

The most obvious finding to emerge from this study is that the TCP connection between the developed GUI and the KUKA Sunrise Cabinet is working.

Some other key findings and achievements that have been made where a compliance with the FDA requirements. The research started there by setting up both the requirements from the company as from the FDA for the development of software for medical devices. These requirements ensured that Matlab could be replaced by the implementing of the hand-guiding code to KUKA Sunrise Workbench. From that point on the process chart development started to understand the components of the system and their interactions.

The process chart in 3 shows a clear task division and structured overview of the entire process behind the Flux One System. This chart highlights the feasibility of replacing Matlab with the KUKA Sunrise Workbench as a suitable platform for controlling the Flux One System.

After understanding this part an internal TCP connection was successfully developed between Qt Creator and VSC. This was done on one device. The implementation involved understanding and adapting TCP communication protocols in both C++ and Java languages.

After applying changes to the Java code developed for VSC, to get a Java code that works on KUKA Sunrise Workbench, a connection was made between the laptop and the controller. This took some network changes, for example the IP address and the port number, but after finding the right configurations this also worked.

The final step was to implement this Qt C++ code onto the actual GUI code developed in Qt, and developing the TCP connection. Again the network settings were adapted and a stable TCP connection was formed. This significant milestone demonstrates the successful integration of the Flux One System.

Overall, this research project has succesfully met its objectives of developing a reliable and stable connection between the GUI and the KUKA Sunrise Cabinet, so that in the future the actual GUI can be used to control the movements and behaviour of the robot. The achieved results contribute to improving efficiency and automation of the surgical procedure executed by the Flux One System, ultimately benefiting the healthcare industry and patient outcomes.

6.1 Limitations and further research

Implementing the code is done based on getting the TCP connection to work. When looking at C and 6 the printed outcome is a boolean with the value true. This is a message that is sended across the TCP connection. When continuing to implement and develop this code, the printed message should be changed to a string so it can actually send messages and numbers. This will result in a connection that can actually send readable messages and makes testing the connection easier for surgeons in the future.

While working on developing the code for the KUKA Sunrise Workbench and adapting the TCP connection, the biggest time consuming element was that the SmartPad had to bee rebooted after every error. When troubleshooting or trying something on a code, this takes up a lot of time. In future research adding a piece of code that makes it possible to excecute the code, will make the development process easier.

A limitation to this research is that the actual connection time is not measured and tested. It took 6 seconds to connect the KukaSmartpad to the GUI seen from figure 6. But this was with the responding time from running the KUKA Sunrise Workbench code, to building and running the C++ code on the qui. So this connection time does not relate to how fast the actual connection was made. This would be profitable for future research.

The last step that is needed before the actual surgery can happen, is the implementation of the hand-guiding code into this TCP connection. Right now the TCP connection code is running in a seperate file from the hand-guiding, and this cannot be changed during running one of the files. These codes should be merged, so that the KUKA Sunrise Cabinet can be connected to the GUI and perform the hand-guiding mode at the same time.

References

[1] C. Heunis. *Flux Robotics B.V.* [accesed : 2022]. 2022. URL: https://www.fluxrobotics.nl/.

[2] Haji Zeinali A. M. Abbasi K. et al. "Fenestrated Endovascular Aortic Aneurysm Repair (FEVAR) for Complex Thoracoabdominal and Abdominal Aortic Aneurysms". In: *The journal of Tehran Heart Center* 13.2 (2018), pp. 88–89.

[3] Center for Devices, Radiological Health & Center for Devices, and Radiological Health. *Overview of Device Regulation. U.S. Food And Drug Administration.* 2020. URL: https://www.fda.gov/medical-devices/device-advice-comprehensive-regulatory-assistance/overview-device-regulation.

[4] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b).* Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[5] S. Levy. *graphical user interface. Encyclopedia Britannica.* 2023. URL: https://www.britannica.com/technology/graphical-user-interface.

[6] Shacklett M. E. Novotny A. & Gerwig K. *TCP/IP. Networking.* 2021. URL: https://www.techtarget.com/searchnetworking/definition/TCP-IP.

[7] W. R. Stevens. *TCP/IP Illustrated.* Vol. 1. The Protocols. Addison-Wesley, 2000.

[8] K. Karytsas. *End-Effector Precise Hand-Guiding For A Medical Robot During fEVAR Interventions.* Internship Report. 2023.

[9] H. & Wang C. Zeng Q. Zhou S. Shen. "A Network Communication Protocols for Robotic-assisted Vascular Intervention Systems." In: *Advances in Biological Sciences Research* (2018).

[10] Tokuda J. Fischer G. et al. "OpenIGTLink: an open network protocol for image-guided therapy environment." In: *The international journal of medical robotics + computer assisted surgery : MRCAS* 5.4 (2009), pp. 423–434.

[11] National Electric Manufacturers Association. "Digital Imiging and Communications in Medicine (DICOM)". In: (2004).

[12] J. & Holmes D. Arnold K. Gosling. *The Java programming language.* Addison Wesley Professional. 2005.

[13] The Qt Company. *Qt Computer Software.* https://www.qt.io/. 2014.

[14] Bjarne. Stroustrup. *The C++ programming language.* Addison-Wesley, 1995.

[15] Microsoft. *Visual Studio Code.* https://code.visualstudio.com/. 2015.

[16] KUKA Roboter GmbH. *Operating and Programming Instructions for System Integrators.* KUKA Sunrise.OS 1.11 KUKA Sunrise.Workbench 1.11. 2016.

[17] *Medical device software - Software life cycle processes.* IEC 62304. International Electrotechnical Commission. 2006.

[18] Arsene. C. "SaMD vs SiMD: What's the Difference?" In: *HealthTECH* (2020).

[19] Center for Devices, Radioloical Health & Center for Devices, and Radiological Health. *Premarket Notification 510(k). U.S. Food And Drug Administration.* 2022. URL: https://www.fda.gov/medical-devices/premarket-submissions-selecting-and-preparing-correct-submission/premarket-notification-510k.

[20] Johner. C. "SOUP – Software of Unknown Provenance". In: *Johner-Institute* (2015).

[21] M. Yilmaz O. et al. Wickel N. Vossel. "Integration of a surgical robotic arm to the connected operating room." In: *Int J CARS* (2023). https://doi.org/10.1007/s11548-023-02926-x.

[22] Cloudflare. *What is UDP?* URL: https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/.

[23] MDN. *An overview of HTTP.* 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

[24] MQTT. *The Standard for IoT Messaging.* URL: mqtt.org/.

[25] GeeksforGeeks. *Socket Programming in Java.* 2023. URL: https://www.geeksforgeeks.org/socket-programming-in-java/.

[26] KUKA AG. *LBR iiwa.* 2021. URL: www.kuka.com/nl-be/producten/robotsystemen/industri%C3%ABle-robot/lbr-iiwa.

[27] Gillis A. S. & Silverthorne V. *integrated development environment (IDE).* Software Quality. 2018. URL: www.techtarget.com/searchsoftwarequality/definition/integrated-development-environment.

[28] KUKA AG. *KUKA Sunrise Cabinet.* 2022. URL: www.kuka.com/en-be/products/robotics-systems/robot-controllers/kuka-sunrise-cabinet.

[29] E. Ziller. *Qt Creator 5.0 released.* www.qt.io, 2021.

[30] *QTcpSocket with Signals and Slots.* 2020. URL: https://www.bogotobogo.com/Qt/Qt5_QTcpSocket_Signals_Slots.php.

[31] S. Gaurav. *TCP Server-Client Implementation - Scaler Topics.* 2022. URL: www.scaler.com/topics/tcp-server-client-implementation/.

[32] NHS. *Abdominal aortic aneurysm.* 2023. URL: www.nhs.uk/conditions/abdominal-aortic-aneurysm/.

[33] Radiological Society of North America. *CT Angiography.* URL: Radiologyinfo.org.

[34] Teem: nrrd. *Definition of NRRD File Format.* URL: https://teem.sourceforge.net/nrrd/format.html.

[35] doc.qt.io. *Build with CMake.* URL: https://doc.qt.io/qt-6/cmake-manual.html#.

[36] Nick Pidgeon. "How ethernet works". In: *HowStuffWorks. com* 1 (2000).

[37] PA Robert and Challinor S. "IP address management". In: *BT Technology Journal* 18.3 (2000), pp. 127–136.

A Internal TCP connection C++ part

The code used in the Qt Creator Main file:

```
    #include <QtNetwork>
#include <QDebug>

int main()
{
    QTcpSocket socket;
    socket.connectToHost("localhost", 1234);

    if (socket.waitForConnected()) {
        qDebug() << " Connected to Java program." ;
        bool value = true;
        socket.write(reinterpret_cast<const char*>(&value));
        socket.waitForBytesWritten();
    } else {
        qDebug() << " Failed to connect to Java program.";
    }
    return 0;
}
```

The CMake file used in the Qt creator:

```
cmake_minimum_required(VERSION 3.5)
project(TCP_6 LANGUAGES CXX)

# tell where the network library is located
set(CMAKE_PREFIX_PATH "/usr/lib/x86_64-linux-gnu/cmake/Qt5Network")

# find the qt5 package
find_package(Qt5 COMPONENTS Network REQUIRED)

# add the source files for the project
set(SOURCES
    main.cpp
)

# tell cmake to create the executable
add_executable(${PROJECT_NAME} ${SOURCES})

# link the executable with the qt5 libraries
target_link_libraries(${PROJECT_NAME} Qt5::Network)
```

B Internal connection Java part

The code used on the VSC application to make the internal TCP connection on one device:

```java
import java.io.IOException;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Main {
    public static void main(String[] args) {
        try {
            try (ServerSocket ServerSocket = new ServerSocket(1234)) {
                Socket socket = ServerSocket.accept();

                System.out.println("Connected to C++ program.");
                InputStream inputStream = socket.getInputStream();
                byte[] buffer = new byte[1];
                inputStream.read(buffer);
                boolean value = buffer[0] !=0;

                System.out.println("Recieved value: " + value);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

C TCP connection from GUI to controller

The code used on the KUKA Sunrise Workbench to establish the TCP connection with the Qt application. This code is both the same for the Qt application on the laptop as the GUI:

```
package BackgroundTask;

import com.kuka.roboticsAPI.applicationModel.RobotcsAPIApplication;

public class Main extends RoboticsAPIApplication (
    // port number used for communication
    private static final int PORT = 23456;

        public void run() (
        try (
            // create a server socket and wait for incomming connections
            ServerSocket serverSocket = new ServerSocket(PORT);
            System.out.println("Waiting for incomming connection...");
            Socket socket = serverSocket.accept();
            System.out.println("Connected to C++ program!");

            // get a message from the C++ program
            byte[] buffer = new  byte[1];
            socket.getInputStream().read(buffer);
            boolean value = buffer[0] != 0;
            System.out.println("Recieved value: "+ value)

            // close the socket
            socket.close();
            serverSocket.close();
            System.out.println("Disconnected from C++ program.");
        ) catch (IOException e) (
            e.printStackTrace();
        )
        )
        public static void main(String[] args) (
            BackgroundTask app = new BackgroundTask;
            app.runApplication();
        )
    )
```