MECHATRONICS ADVANCED COURSE

Ericsson A.R.M.S.

Author:
Arian Hosseini,
Axel Karlsson,
Christoffer Olsson,
Johan Danmo,
Erik Svensson,
Felix Büttner,
Pontus Mjöberg,
Raphael Hasenson

Supervisor:
Andrii Berezovskyi

December 16, 2018

ROYAL INSTITUTE OF TECHNOLOGY

Abstract

School of Industrial Engineering and Management
Department of Mechatronics

Mechatronics Advanced Course

Ericsson A.R.M.S.

by A.R.M.S. group

The Autonomous Robotic Maintenance System (ARMS) project was founded by a collaboration between KTH Royal Institute of Technology and Ericsson AB with the goal of designing and implementing an autonomous repair system for the Ericsson base stations which provide cellular data across wide regions. For this purpose, the ABB IRB120 robotic arm was utilized and combined with an actuated end-effector to achieve precise handling of delicate optical cables. The end-effector was designed and created using 3D printing technology and combined with an actuated solenoid mechanism for opening the mechanical locks on the cables. The connections and mechanical structure of the gripper was produced from aluminium. The robotic system was then interfaced with a Raspberry Pi and an Arduino Uno to implement autonomus control and movement algorithms. Results demonstrated the feasibility of employing such an autonomous mechanism for cable replacement via remote access or fully automated operation.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	**A**lternating **C**urrent
API	**A**pplication **P**rogramming **I**nterface
CAD	**C**omputer **A**ided **D**esign
CV	**C**omputer **V**ision
DAQ	**D**ata **A**c**Q**uisition
DC	**D**irect **C**urrent
DoF	**D**egrees **o**f **F**reedom
DU	**D**igital **U**nit
GUI	**G**raphical **U**ser **I**nterface
SFP	**S**mall **F**orm-factor **P**luggable
SOTA	**S**tate **O**f **T**he **A**rt
PC	**P**ersonal **C**omputer
PCB	**P**rinted **C**ircuit **B**oard
RPi	**R**aspberry **Pi**
VNC	**V**irtual **N**etwork **C**ommunication

Chapter 1

Introduction

Ericsson base stations are installed all over the globe with the objective of providing cellular data across regions. As these cabinets are installed in areas with a high variation in operation condition or extreme climates; regular maintenance procedure and repair is required. Automatic repair via a remote connection or an autonomous robot would drastically reduce the time and effort required for these maintenance procedures and create a robust, adaptive solution towards addressing these issues regardless of the operating conditions.

1.1 Project Description

The project is defined to address the issues related to the maintenance of Ericsson base stations as mentioned in the previous section. The objective of this study was to design and implement an autonomous robotic solution to perform automatic repair and maintenance and reduce the required time and effort. The project was set as a collaboration between KTH Royal Institute of Technology and Ericsson AB to achieve a proof of concept and assess the feasibility of the implementation of such an autonomous solution. The results of this study would assess whether full autonomous repair would be a possibility by implementing a Mechatronics product on an industrial scale.

1.2 Requirements

The main goal of the project is to automate a process where a common component is replaced to solve a common fault in a base station. The project is focused around the replacement of a broken Small Form-factor Pluggable (SFP) with a new functioning SFP. The stakeholder requirements were listed as:

- The repair cycle shall be fully autonomous
- The robot shall be able to replace optical adapters of different types at certain locations
- The robot shall be carriable/mountable by a single operator
- The robot should not be in the way while in standby mode
- The robot shall report what is being done
- The robot shall not damage any parts of the station (including cables and connectors) while working
- The robot may connect to an operator for monitoring purposes

- The robot shall have access to spare parts
- The total cost of ownership shall be less than that of a human worker

1.3 Delimitations

Several delimitations were chosen to make for a realistic project time plan.

1. The repair system will disregard different types of DU racks. The robot will focus on a specific rack/DU design with set parameters and dimensions.
2. The environmental factors affecting the robustness of the solution such as temperatures, humidity and workplace contamination will not be taken into account.
3. The repair system will be focusing on a single DU rack.
4. The execution time of the replacement process was regarded irrelevant for the proof of concept.
5. Cleaning of cable and adapter will not be part of the scope.

1.4 Readers Guide/Report Disposition

The report follows the pattern below:

Chapter 2 unveils a short literature study and state of the art on selected subjects found relevant for the project.

Chapter 3 presents the methodology of the project and discusses why key decisions were made and how the project was organized in terms of team structure and communication.Project budget and costs are also presented in this chapter along with an itemized list of components that were used in the project.

Chapter 4 describes how the project was implemented. The system architecture, programming, manufacturing and integration is demonstrated here

Chapter 5 displays tests that were performed during the project in order to verify and validate the desired functionality.

Chapter 6 examines the final results of the project and how well the requirements were met.
Chapter 7 contains a discussion on the projects results. The achievements are highlighted and critically reviewed as to how well they match the requirements and what we should have done differently in hindsight.

Chapter 8 presents possible future improvements on the project.
Last appear appendices with extra information on selected topics, components and schematics used in the project.

Chapter 2

Literature Review and State of the Art

There is rich literature on automation of repetitive tasks involving a pick and grab mechanism. The team reviewed the state of the art for the current literature in detail and presented earlier during the project. The previous work and literature which were useful for this work and its prospects are covered in this section.

2.1 Manipulator

The purpose of the manipulator is to be able to move our required tool to specific locations in a controlled manner. This section explores the different manipulator options deemed relevant to the scope of the project.

Automated inspection and maintenance of oil platforms

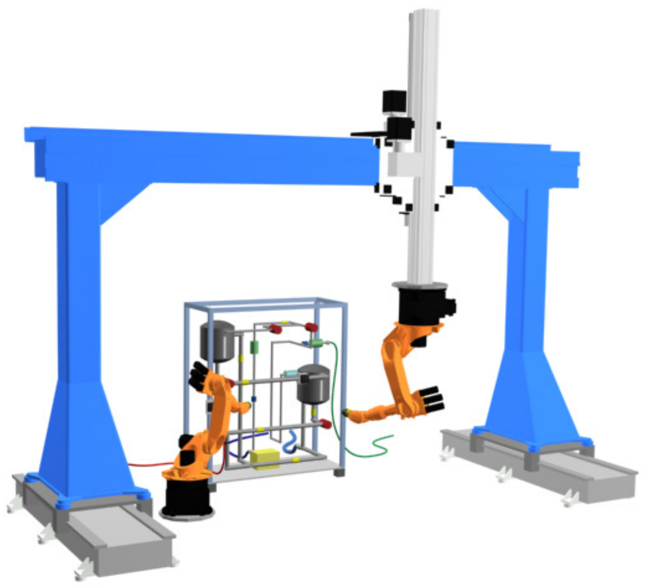
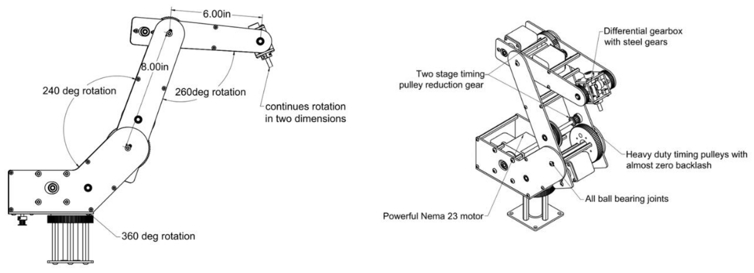


FIGURE 2.1: The facility set up by SINTEF [1].

Researchers at SINTEF have created a concept for automating inspection and maintenance on an oil platform. The concepts viability is demonstrated by simulating a production process in a lab. The purpose of the project is to automate the most common and important maintenance and inspection operations.

The maintenance operations are performed by two six DoF robots of type KUKA KR-16 working in concert with one mounted at a permanent position on the floor and the other mounted on a gantry frame for increased movement as seen in figure 2.1. An overview camera is placed at the right leg of the gantry, one at the base of the gantry robot and an additional at the end-effector of the robot to provide different views to the operator/spectator. The floor-based robot does not have a permanent camera but can attach a stereoscopic camera providing three-dimensional information of the vicinity. Additional sensors used that are of interest to this project are force/torque sensors measuring the contact forces of the manipulators. The concept is using a wide range of tools including a gripper to pick up items and assist in operations i.e. replacing batteries of a wireless sensor. The replacement of the batteries is a joint operation between the two robotic arms. The gantry mounted robot unscrews the lid of the sensor using a custom made battery tool and the floor mounted robot replaces the battery using a gripper. The process is then finished by the gantry robot reattaching the lid to the sensor. The system is controlled remotely by an operator. The operator has the option of controlling the operations directly in a virtual environment linked to the robots. The operator also has the option to use an automatic mode where operations only have to be ordered and the robots perform them automatically [1].

Dorna


6.00in
8.00in
240 deg rotation
260deg rotation
continues rotation in two dimensions
360 deg rotation
Differential gearbox with steel gears
Two stage timing pulley reduction gear
Heavy duty timing pulleys with almost zero backlash
All ball bearing joints
Powerful Nema 23 motor

(A) Dimensions of Dorna. (B) Joints of Dorna.

FIGURE 2.2: The Dorna Robotic arm [2].

Dorna is a cheap industrial grade robotic arm with open source software and firmware which provides a good estimate of quantities such as weight, accuracy and payload.

Dorna features five degrees of freedom with an accuracy of 0.02 mm repeatability. However, the arm is missing one of the rotational axises that are required for keeping the edges of the head perpendicular to floor and cabinet, see Figure 2.2. It has a reach of 863 mm in diameter horizontally and 584 mm vertically. Dorna is a lightweight robot weighing 5.4 kg and is able to handle a payload of 1.1 kg. The Controller board is based on Arduino Due and communicates with USB-device. It is powered by either 110V AC to 18V DC and has a maximal power outlet of 60W [2]. The current price of this device is £890 but it is worth mentioning that the Dorna enterprise is a startup endeavor with an estimated first delivery date of June 2018 [2], which might not be reliable enough. The price of comparable robotic arms starts at around £5000 and upwards [3]. This type of option could offer a stable price for a reliable and time saving solution for transport of the tool head.

For user interaction, Dorna is operated through Dorna Lab which is a Python based program which can be used as a standalone application or be integrated as a library with other applications. One of it's features is a graphical simulator which helps the user visualize and evaluate one's scripts before running them on Dorna.

Canadarm

The Shuttle Remote Manipulator System (SRMS) or Canadarm was a joint venture between the governments of the United States and Canada to supply the NASA Space Shuttle program with a robotic arm for the deployment/retrieval of space hardware from the payload bay of the International Space Station (ISS) [4]. The robot arm, which was the first robot used in space, can be seen in Figure 2.3 and had a length of 25 m. The robot was retired in 2011 and a newer version, Canadarm2, was developed to take its place and is currently used on the ISS.

The Canadarm is remote-controlled rather than autonomous, yet it's use includes repairs done in hard to reach areas. Canadarm2 is also placed on a moving platform to extend its reach across the entire space station as opposed to being fixed in one location.

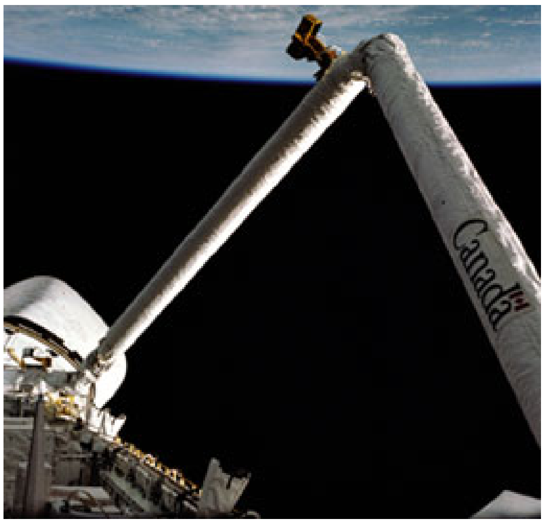


FIGURE 2.3: The Canadarm on the International Space Station, ISS [4].

IRB 120

IRB 120 is an industrial robot from ABB and comes in two versions, the IRB 120 and the IRB 120T which is a high speed version of the robot. It is the corporations smallest industrial robot and weights 25 kg and can handle a payload of 3 kg. It has a horizontal reach of 580 mm and can apply a force up $\pm$ 735 N depending on mounting. It can repeat positioning with an accuracy of 0.01 mm and its mean deviation from programmed position is $\pm$ 0.02 mm. Another option with similar specifications can be found in UR5 [5].

RobotStudio is a software suite from ABB for programming the robot. The software creates a 3D-model of the robot and the environment. The robot is then programmed by simulating the desired movements in the virtual model. RobotStudio have support for offline programming of the robot. Offline programming is when the robot is programmed without the robots involvement and then uploaded to the robot when the program is done. Offline programming allows the robot to remain in production while the robots new movements are programmed and only pause production while the new program is uploaded to the robot, which minimizes downtime.

Cartesian Robot

A cartesian coordinate robot works by moving its tool head in straight linear motion across the x,y and z-directions [6] independent of each other. This would make calculating the system's path very simple, since movement alongside a specific axis would only require actuation from a single actuator. The powertrain of the system can consist of a pneumatic piston system, a band/chain or a ball screw that would be driven by a DC-motor.

A proposed solution for the task at hand was to construct a framework facing the RBS cabinet that would wield this linear robot technology. This would allow the end effector to position itself at the coordinates of the faulty SFP, move in and extract it, as well as transporting and inserting a new one in its place.

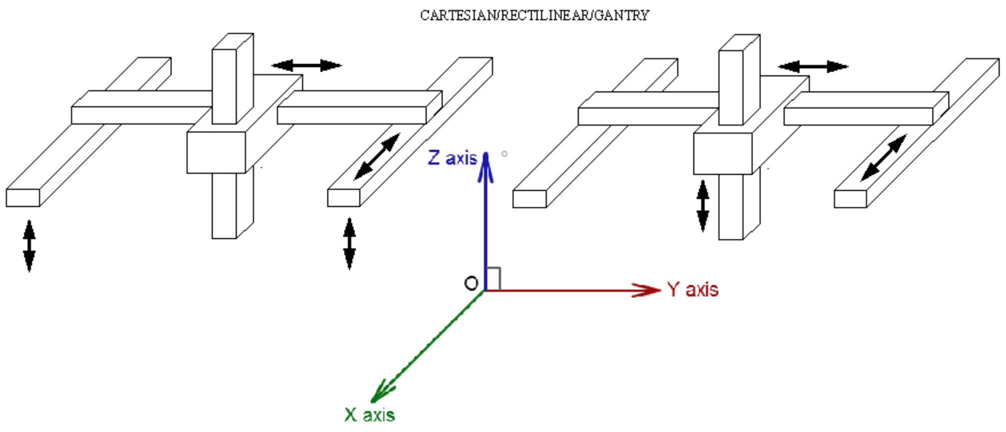


FIGURE 2.4: Alternate ways of moving the tool head in the Z-direction towards the target location. Either the entire z-plane moves towards the target location, or the end effector is extended.

Alternative designs were conceptualized considering whether the end effector movement in z-direction, towards the cabinet, was to be done with an extending arm, or if the entire z-plane was to be moved while having a stationary tool head.

As can be seen in figure 2.4, the moving frame would require more total space in the z-direction; This would in turn increase the weight of the system's frame module, but also allow it to handle larger forces because of the extra support [6]. The extending arm would offer the attributes inversed. However, it also allows the tool to reach a desired location without risk of having its frame being obstructed by the target DU:s surroundings.

Conclusion

Since there are already several high precision robotic arms on the market, one could argue that manufacturing a robotic arm might be best left for the front edge companies. That way a stable price and a trustworthy functionality will be guaranteed. Furthermore, since access to an IRB120 ABB robotic arm has also been granted for the oncoming term this becomes the natural choice.

2.2 Gripper

To be able to extract and transport a cable and/or a SFP, some sort of gripping device is needed on the tool head. One alternative would be simply pressing and pulling the SFP/cable and transport it to it's key location. This could be done with a simple claw solution. Another solution one would be to handle more of the SFP transfer with the tool itself. This would require a higher level of tool versatility which demands a more complex structure. One concept idea for a more complex structure were multiple devices on the same head. If the tool head was allowed to rotate, several tools with different tasks could be active at the same time Several concepts with varying functions and limitations similar to this were studied, some more interesting than others.

GRIPKIT by Weiss Robotics

The gripping solution GRIPKIT by Weiss Robotics is a complete gripping solution with a user friendly API called URCaps plug-in. This allows the user to configure the claw for any pick and place application within the force range of 7.5 to 550 N [7]. The functionality of the gripper is interesting in regards of the scope of this project since it can perform all the tasks without adjustment, except for unlocking the SFP from the DU. The GRIPKIT is a single module which means that it can be mounted upon any moving or stationary application and still be functioning according to the movements created in the API. This means that the gripping part of the robot would be a singular node, independent of other nodes in the system architecture, which would reduce the complexity of the robot.

Linear Motion Robotic Claw with Interchangeable Grippers

Erlingsson et al. [8] designed a linear claw with interchangeable grippers. The design utilizes servo motors to achieve a more robust and precise design. By using the servo motors instead of pneumatic actuators, the final design also became lighter than the alternatives. Interchangeable grippers for this mechanism need the be designed with careful orientation towards the desired application.

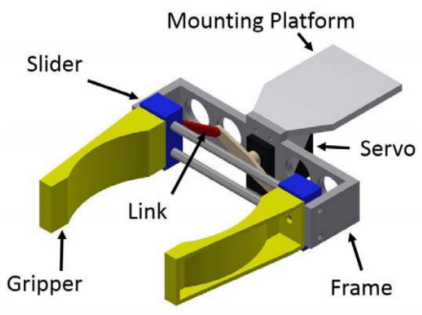


FIGURE 2.5: Linear robot with interchangeable grippers. [8]

Servo-Controlled Robotic Gripper

Utilizing servo motors for control of robotic grippers is a classic approach for designing robust mechanisms. In a recent study, Shaw et al. [9] designed a robotic gripper for a varying set of objects using a rack and pinion mechanism with force control. Furthermore, they implemented a PID controller with anti-slip criterion for controlling the gripper. The figure below presents the overview of the rack and pinion mechanism used for the study.

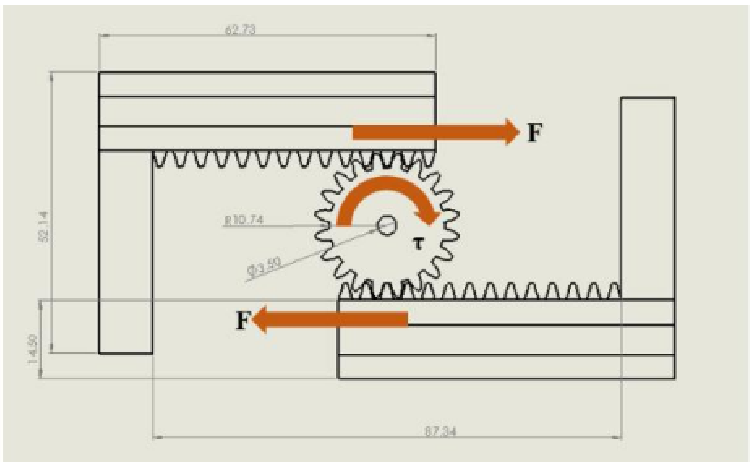


FIGURE 2.6: Rack and pinion mechanism designed by Shaw et al. [9].

The gripper was able to exert a maximum force of 32 N on the object and efficiently grabbed heavy objects which was not possible using earlier gripper designs. The results demonstrate that heavier objects increase the number of slips and required grabbing time.

Conclusion

Since the gripped object is well defined and a reliable unlocking function of the cable while having a firm grip of it is needed, the suitable choice for the task seemed to leave a custom made claw as the optimal alternative. This because no claw offered the full functionality that was sought after.

2.3 Control Unit

To control the entire system and interpret signals, a control unit is required. The most common kinds of controllers are microcontrollers and microcomputers. Both have different areas of expertise and different advantages, which will be explored in this section.

Microcontroller

The microcontroller is an integrated circuit consisting of a microprocessor, flash memory, RAM and peripheral features. The microcontroller only runs the code that is uploaded to it, making it ideally run in real-time, which makes it perfect for executing tasks that require a fast response, such as controlling actuation and other movements and also responding to inputs from the real world. The processing power is limited, however, making it a bad choice for things like computer vision, machine learning and more advanced control algorithms.
There is a wide range of microcontrollers available. The most common one is Arduino (20-35 EUR 210-370 SEK), which is an open source hardware with a large internet community behind it [10]. It is available in many different configurations and sizes based on your needs. The Arduino boards generally come with a 16 MHz processor clock speed, operating voltage of 5 V and DC current per pin is limited to 20 mA. Differences between the builds concern the amount of pins as well as the built-in flash memory, SRAM and EEPROM. Other alternatives include ST32 Discovery, the Nanode and the MSP430 Launchpad, which are all very similar to the Arduino.

Single Board Computer

In comparison to a microcontroller, the single board computer has a lot more processing power and usually runs an entire operating system. Due to many tasks needing to run in the background, the computer cannot be used effectively in real time. The increased processing power, however, enables computing of more complex tasks such as machine learning and computer vision.
The most common single board computer is the Raspberry Pi (444 SEK), produced by the Raspberry Pi Foundation. It comes with an integrated OS called Raspbian which is based off the Debian OS. It has 26 GPIO ports, of which all can output software PWM signals and 4 can output hardware PWM signals [11]. Just like the Arduino, it has a large community behind it and is available in both larger and smaller configurations. Another alternative is the Beagleboard, which is much more expensive (49 dollars 435 SEK) but comes with a 1 GHz processor, 512 MB RAM and up to 65 digital pins, of which 7 are capable of hardware PWM output.

Conclusion

The most critical part of the whole process is the detachment and attachment of the of the cable and SFP, and thus this will dictate what control setup is used. As previously mentioned, the main difference between microcontrollers and single board computers is real time control versus heavy calculations. This project will however require a bit of both, since both positioning and gripping requires controlling

with regards to feedback from the computer vision sensing. This boils the decision down to whether to let the single board computer handle the controls as well using scheduling, or using nodes of microcontrollers to distribute the workload.
The other alternative is to have a microcontroller and a single board computer working in parallel. This allows for a faster system since the computer only needs to send signals to the controller and not switch tasks. There are two easy ways to do this. The first way is to use a raspberry pi compatible Arduino board, such as the Gertduino [12] which directly connects to the raspberry pi. This allows the raspberry pi itself to program the microcontroller and use various functionalities on the Gertduino. The second way is to connect the Arduino and the raspberry pi using USB or a serial setup.
Since the detachment and attachment of the of the cable and SFP require great precision and computer vision will be implemented, it is vital that the controlling is as smooth and as close to real time as possible. This cannot be done only with a single board computer since it would need to alternate between the computer vision sensing to get the errors and the control algorithms to compensate, which would result not only in a slow system but a hard to control system as well since the latency would be very high.
A feasible solution is to let the microcontrollers handle the control algorithms while the single board computer supplies them with the errors that need to be compensated. This will reduce latency and allow the robot to operate as close to real-time as possible. The computer will also handle the diagnostics and logging of movements.

2.4 Helper Station

It is common to add another module for supporting functions to enable automation of the task at hand. For example Sneyders WISLA bottling machine uses a starwheel to properly position the empty bottles for filling. Other modules for support function can range from a conventional conveyor belt to an entire robotic arm such as the setup used by SINTEF described previously.

In order to replace the SFP modules satisfactory, the robot requires a Helper Station that needs to fulfill two functions; managing the cable currently being operated on and providing spare parts.

When providing spare parts, it is imperative that the location and rotation of the spare part is determined in order to allow the robot to pick it up. For this purpose, four different concepts were evaluated:

Vending machine

A vending machine type of spare part dispenser would be able to house multiple spare parts of varying sorts and dispense the required one at any given time, in a set order should multiple spare parts be needed for a job.
It would be able to guarantee a limited final location of the spare part, but would not guarantee a set rotation for the parts. The tray in which the spare parts land would need to be designed keeping the dimensions of all the various spare parts in mind, where the largest spare part would determine its size. This in effect becomes the uncertainty for the smallest spare part dispensed. Refilling a vending machine type of spare part dispenser comes with ease as the spare parts are kept in internal compartments, but the fact that they are located internally does restrict the robots access to them.

This solution requires communication and actuators to be implemented, and needs to be designed with all possible spare parts in mind.

Filing Cabinet

Placing the spare parts in individual compartments that slide out when called upon ensures a set location and rotation of each part. It would be easy to load the compartments that are empty when replenishing spare parts. The location of each spare part would change depending on which one is dispensed, therefore making additional communication between dispenser and robot necessary as well as additional actuators for the compartments. Due to the spare parts being housed within the structure, limited access is given to the robot but one cabinet could house multiples of the same spare part as well as parts of varying kind.

Clip

Arranging spare parts in clips allows for the rotation and location of the parts to be deterministic. The clip in turn would be mounted at a defined location, and keeping count of how many spare parts has been used, the location of where the next one is available can be deduced. The fact that a Clip type of dispenser holds the spare parts in a hollow structure allows the robot access. However, replacing a clip may induce waste as the clip may contain one or multiple spare parts left on it at time of replacement. This can be circumvented by manually loading the clip as opposed to replacing the clip on the spot, which in turn may be time consuming.
The clip would only be a mechanical structure, thus not needing communication nor actuation. However, one clip would need to be designed for each type of spare part to be held.

PEZ Dispenser

Stacking the spare parts vertically within an outer housing allows for the location and rotation of each part to be fully determined. The next spare part to be used would be placed on top of the stack, and when required would be pushed horizontally outwards a set distance to make it easier to grab. Adding a compressed spring at the bottom of the stack would make sure that once a spare part is used, the next one will arrive at the same location as the previous one.
This would require actuators and communication between multiple systems but makes it easy to replenish spare parts as needed. The fact that the next-in-line spare part is located at the top of the stack gives the robot access to them. A PEZ type of dispenser may only hold one type of spare part.

Cable Management

When deciding about the cable management, it is important to discern between two different concepts: whether the Helper Station should only hold on to the cable, thus freeing up the robotic hand to do one operation at a time or to assist in inserting the cable into the new SFP adapter.

Looking into the first option which alleviates the end effector, it is of importance to dedicate an area to hold the cables that are currently being operated on.

The emphasis of this location would be to preserve the rotation and location of the cable end which will later be inserted into the new adapter. One solution to this is a purely mechanical structure designed in such a way that it either holds the cable end in the place where it was dropped off, or that the cable end ends up in a set location with a determined rotation.
An example for the first design is to have flexible circles that connect between which the cable can be squeezed. The flexibility of the rings would be so that when the cable is released, the ensuing friction is enough to hold the cable in place.
The second design would encompass two parallel bars that are spaced in such a way that the cable can be inserted between them, yet narrow enough that the cable head does not fit through, thereby holding the cable head at a certain location. Adding support beams would ensure that the desired rotation can be achieved.
Issues that could arise with the first design is that torsional forces on the cable when it is moved rotates the cable end, rendering the rotation of the cable end unknown.
Examining the second option of assisting in the insertion of the cable into the new SFP adapter, one may differentiate between doing so passively or actively.
Assisting passively would entail a purely mechanical structure such as a cone or funnel which would guide the cable towards the SFP adapter. Actively aiding the insertion involves the robot handing over the cable to the Helper Station which assembles the cable and SFP and subsequently returns the assembly to the robot for further manipulation. This can be achieved by having the robot to place the cable on a receiving platform which is aligned to the SFP. At the end of the receiving platform is a customized grip, which will ensure that the rotation and position of the cable remains determined. By moving the platform with a linear motion towards the SFP, the cable will be inserted and promptly handed over to the robot.

Location

One aspect of the Helper Station is its location. This can either be on the robot, mounted on the side of or on top of cabinet or mounted inside the cabinet, similar to a DU. Should the Helper Station be mounted on the robot, then the advantage lies in that its position relative to the robot will always be the same. However, this causes for further requirements when installing the robot.
If the Helper Station is placed next to the cabinet, then issues regarding available space could arise depending on the surroundings of the cabinet, which can not be guaranteed to be the same for all. Similarly to placing the Helper Station on the robot, this solution will also require more space around the cabinet.
Mounting the helper station within the cabinet in the same fashion as the DU allows the robot to only operate in one plane throughout all tasks. This also will not interfere with any space outside of the cabinet and also lowers the complexity of the robot movement.

2.5 Computer Vision

Since blindly navigating to the given port isn't a feasible option, control through feedback is a requirement. Humans use their eyes and their brain to gather visual information about the world, analyzing it and make a decision depending on the deducted information. Computer vision is a branch of computer science with the aim of emulating the human vision [13]. The problem is computationally addressed through three processing components:

1. **Image acquisition** is the process of converting light reflected of an object into a digital quantity. Tools for converting visual information into matrices of digits are digital cameras.

2. **Image processing** is where algorithms are applied to the numerical matrices obtained from the first step, this to deduce basic features of the image. The basic features are segments,image edges or point features, all which are geometric element that builds up the image.

3. **Image analysis** is the final step where visual information from step one is combined with step two to obtain high-level data such as object recognition or classification [14].

By using computer vision it is possible to provide the robotic assistant a visual perception which may be used for calibration and error adjustments of the system.

Automatic Fruit Harvesting Robot

The innovation of this solution is the use of stereoscopic vision in the grabber to gather positional data regarding the heads position, which can provide additional feedback of the vicinity. The robot proposed is combining a low cost stereo-vision camera and a gripper attached to be able to estimate size, distance and positions of fruits. The stereo-vision camera system is used to find correlated information in two images from different angles using the centroid of the object [15].

Stereo vision

Stereo vision is the ability to derive information about how far away objects are, based solely on the relative position of two cameras as seen in figure 2.7. This is done by feature matching corresponding points in the images of the respective camera to one another, since the relative position between the two cameras is known triangulation can be used to calculate the distance to the object [13].This allows for measurements in both width, height and depth, which is the feedback necessary to precisely navigate and handle the equipment.

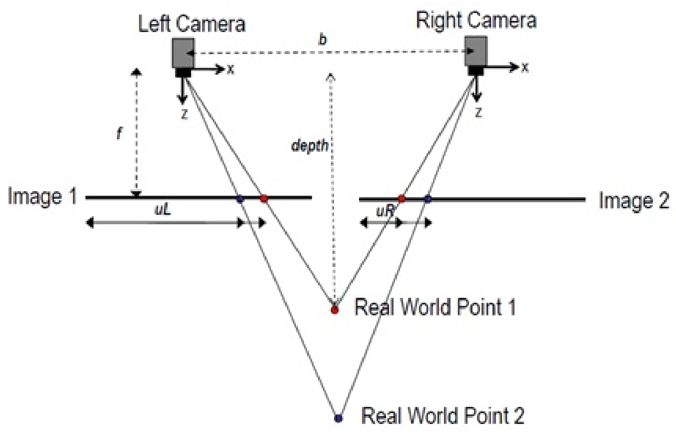


FIGURE 2.7: Basic setup of Stereoscopic Vision [16].

Hough Line Transform

Hough Line Transform is a method for detecting lines and edges in an image [17]. This is done by scanning the entire image for lines. If a line is confirmed multiple times in the image, it is assumed to be an actual line. The amount of times a line has to be confirmed is called the threshold, and can be altered to reduce the noise of small insignificant lines. If the observed object has known dimensions and the lines that define the object are found, both angles and relatives position in all three dimensions can be derived with high accuracy.

OpenCV

There is a lot of software available for Computer vision. One of the most reputable is OpenCV, an open source, completely free library of programming functions aimed at real-time Computer Vision. The main programming language used is C++, but there are bindings in both Python and Java. Due to being both free and both established and extensive, OpenCV is chosen as the main Computer Vision software [18].

Conclusion

Using Computer Vision would allow extra redundancy for the robot to handle the extraction and insertion of the SFPs and cables with enough precision. Another application is to use the Computer Vision for positional calibration after installation. Since there is no guarantee of the operator installing the robot at an exact predetermined position, the robot could instead calibrate itself, thereby making the installation process easier and guaranteeing positional accuracy.
With the many benefits that Computer Vision brings, it is chosen as the method of acquiring Control Feedback.

2.6 Design Proposal

The final design was proposed to be based on purchasing or borrowing an ABB IRB120 robotic arm as the manipulator. The gripper for this concept has to be manufactured and oriented towards the task of grabbing the SFP and flipping the latch mechanism. The Arduino is considered as the main microcontroller for the control application and the Raspberry Pi is proposed as the OS platform to handle the Computer Vision and the communication with the base station. The BeagleBone Black is also considered as an alternative all-in-one package for the control and OS operation applications. An algorithm based on Hough Line Transform is chosen to correct errors in the robot movements for certain tasks, using OpenCV for CV related methods. By combining these systems with ABB's programming interface, one can achieve a robust real-time control structure while operating the peripherals on the memory from the single-board computer such as Raspberry Pi. The helper station concept design is to be implemented similarly to a DU within the rack to save space whilst assisting the robotic arm.

Chapter 3

Methodology

This section focuses on describing the project planning, team structure and communication and design approach towards the implementation phase. The team utilized previously gained knowledge through project experience and courses to propose and utilize a structured, time-efficient plan for achieving its objectives.

3.1 Team Communication

Team communication and synchronization was performed on a weekly basis during general meetings. Issues that concerned the whole group were discussed and addressed during these meetings; leading to a team decision regarding the priorities of the respective issues and plan of action during the next iteration. To provide an easier mean of communication outside general meetings, the team decided to use *Slack* as the main communication platform. The platform allowed easier communication and synchronization with team members and was present until the end of the project.

3.2 Project Management

The team organized itself internally to achieve the set objectives and goals throughout the project. The following sections present a general overview of the team structure and organization, and design approach for the realizing the final product.

Team Structure & Organization

The project followed a modified SCRUM approach coupled with Kanban boards for installing tasks. The team structure was divided into the following subgroups:

- Team coordinator (*Axel*): The team coordinator was responsible for improving communication between subgroups and allow easier synchronization of each subgroup with respect to the project goals during each SCRUM iteration. The coordinator was also allowed to take decisions in the approach that concerned the whole group.
- CAD & Structures Team: This team was responsible for the design and creation of the required structures for the final package. Solid Edge was used as the main CAD software to visualize and create the respective designs.
- Python Programming Team: This team was responsible for creating the code on the Raspberry Pi for communication with the subsystems and creating the top-level system architecture.

- RAPID Programming Team: This team was responsible for developing the RAPID program used in RobotStudio to initiate the autonomous robot movements as well as network communication over Ethernet.
- C Programming Team: The objectives and tasks of this subgroup were mainly focused on developing Arduino functions for for interfacing the respective component with the motor driver, Raspberry Pi and the pressure sensor.

In order to allow all members to learn and contribute, a rotation cycle was implemented to switch the team members in every SCRUM iteration. The following sections describe the duties of every subgroup and the approach of the whole team as a unit towards the project:

Modified SCRUM & Kanban Approach

The objective of implementing the SCRUM and Kanban approaches was to have a weekly synchronization between all the team members and to discuss the present tasks and assign respective priorities to them. For this purpose, the team used the following softwares to implement and combine the SCRUM and Kanban approaches over the course of the project:

- *Trello*: This software was used to set up the subgroup work areas and provide the means to tabulating and keeping a record on all present issues and on-going work. However, due to its limitations in describing a task in detail; the team decided to move to GitHub later on.
- *Google Sheets*: The Google Sheets was set up to provide a general overview of the whole project breakdown and each member's assignments and on-going work. The sheet keeps track of the state of the work and counts the performed number of tasks and number of remaining tasks.
- *GitHub*: A GitHub branch was created in the Ericsson database to act as a structured method of keeping all the codes updated and performing push/pull requests. This platform was mainly used for keeping track of the codes and their revisions at the start.
- *Planning Poker*: The team decided to tabulate non-programming tasks in GitHub as well due to its abilities in assigning priorities to tasks and ease of implementation. This priority was assigned based on a Planning Poker scenario where the priority of each task was debated and decided upon before the creation of an issue based on the respective task.

3.3 Design Approach

The process of replacing a SFP was devised to be consisting of six consecutive actions, as seen in Figure 3.1.

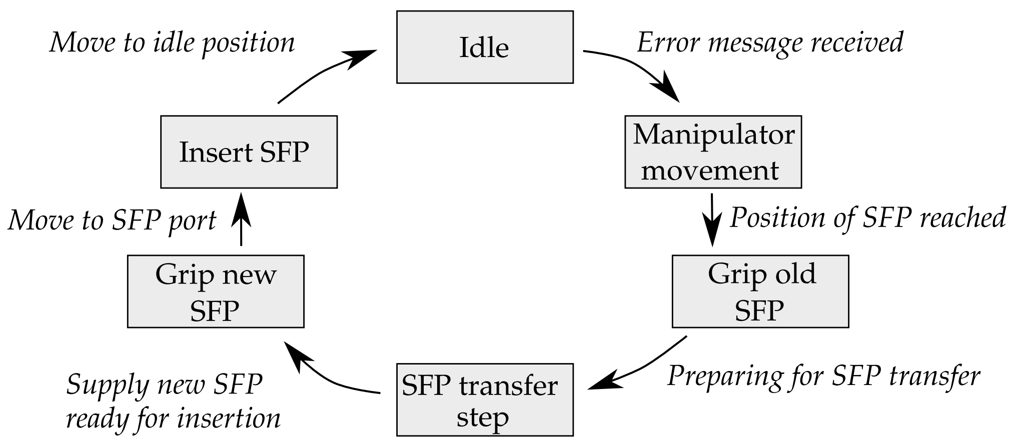


FIGURE 3.1: A complete work process cycle of the remote maintenance system.

Once receiving an error message of a faulty SFP, the end effector of the maintenance system has to be moved to a specific port location in 3D-space with high precision to be ready for the next step.
When the location has been reached, the end effector has to be able to grip the cable, followed by the SFP, in such a way that these can be moved and placed in a specific position with regards to location and angle; after all, the SFP and its cable can only be inserted back into the socket from one specific direction in 3D-space.
When the cable and the old SFP have been extracted, the SFP has to be swapped for a new one that is ready for insertion. Consecutively the cable has to be fitted into the newly inserted SFP. Alternatively the maintenance system could use a helping module to piece the cable and the SFP together and hand it back to the end effector, thus allowing it to insert the whole payload in one go.
The closing steps of the cycle are similar to the starting ones; when the payload is ready for insertion it has to be moved and inserted in the port location of the old SFP. This is followed by going back to idle mode.

In order to create a fully autonomous repair system, two major design approaches were considered at the start of the project:

- Design and implementation of a Cartesian Robot (also known as linear robot)
- Implementation of an existing robotic arm for autonomous repair

The team considered the advantages and disadvantages of utilizing each of the respective systems. The table below presents the pros and cons associated with each of these designs:

TABLE 3.1: Comparison of Design Concepts

Linear Robot	**Robotic Arm**
+ Easy to control	- More difficult to control
- Costly to build	+ Already available (ABB IRB120)
- Blocks operator access	+ Allows operator access
- Less degrees of freedom	+ Higher degrees of freedom

For these reasons, the team chose to proceed with the implementation of a robotic arm. An IRB120 ABB Robotic Arm was provided by the department for implementation during the project. Afterwards, A modular systematic approach was utilized to develop and integrate all of the components into the system according to the V-model as seen in figure 3.2.

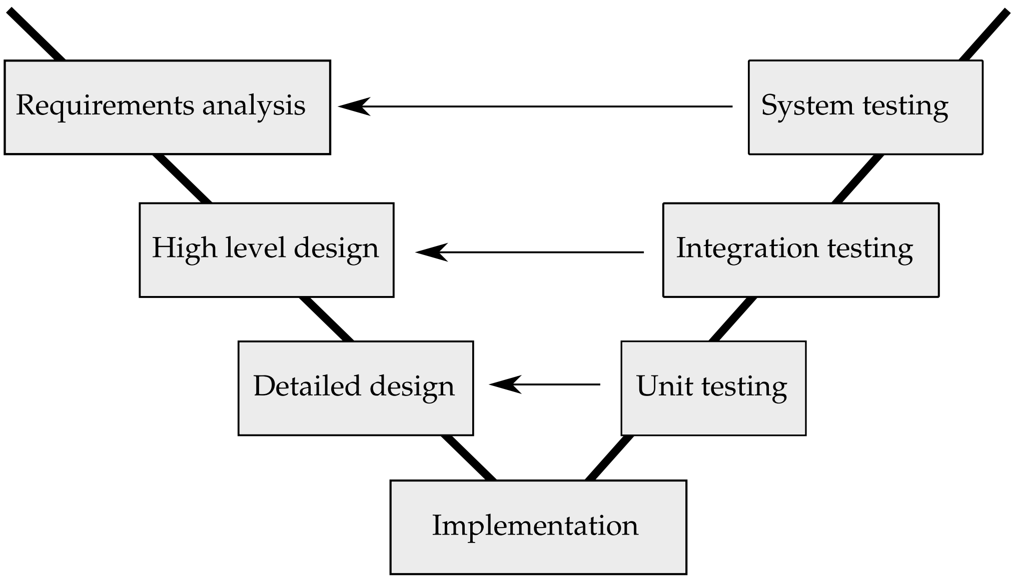


FIGURE 3.2: The V-model.

3.4 Budgeting & Costs

The supplies for this project was provided as a joint collaboration between Ericsson and KTH. The table below presents the items and their respective acquirement costs for the project.

List of purchases				
Category 1: Not included in the final product				
Name	**Category**	**Quantity**	**Cost/Piece**	**Total Cost**
DC-Motor	Gripper	1	245.99	245.99
Cogwheel	Gripper	3	39.2	117.6
DC-Driver	Gripper	1	210.96	210.96
ABB Cable	Arm	1	3651.25	3651.25
Current Sensor	Gripper	1	59	59
Category 2: Services used				
Water Cutting	Service	10	450	4500
Workshop Processing	Service	10	450	4500
Category 3: Included in the final product				
Stepper Motor	Gripper	1	150	150
Motor Driver	Gripper	2	177.5	355
3D Printer PLA Roll	Gripper	1	199.9	199.9
SKF Bearings	Gripper	25	36.34	908.5
Aluminum (kg)	Gripper	0.5	80	40
Screw/nut-kit	Gripper	1	59.9	59.9
Pi Camera	Vision	1	225	225
Pressure sensor	Gripper	1	153.34	153.34
Micro SD Card (32 GB)	Vision	1	312.1	312.1
Bag of Starlocks	Gripper	1	40.36	40.36
Arduino	Gripper	2	249.9	499.8
Raspberry Pi	Gripper	1	349.9	349.9
Network Switch	Gripper	1	192	192
Acrylic Plastic	Gripper	1	300	300
Electronics				
Multistranded Cables (m)	Gripper	5	0.5	2.5
Arduino Cable	Gripper	1	29.9	29.9
rPi Cable	Gripper	1	99.9	99.9
Solenoid	Gripper	1	179	179
Transistor (kg)	Gripper	1	50	50
Resistors	Gripper	10	0	0
Female Headers	Gripper	18	1	18
Male Headers	Gripper	40	0.225	9
Molex Contacts	Gripper	120	0.026	3.12
LEDs	Vision	4	1	4
PCB material (mm2)	Gripper	100	1	100
Total Cost				**17500.72SEK**

Chapter 4

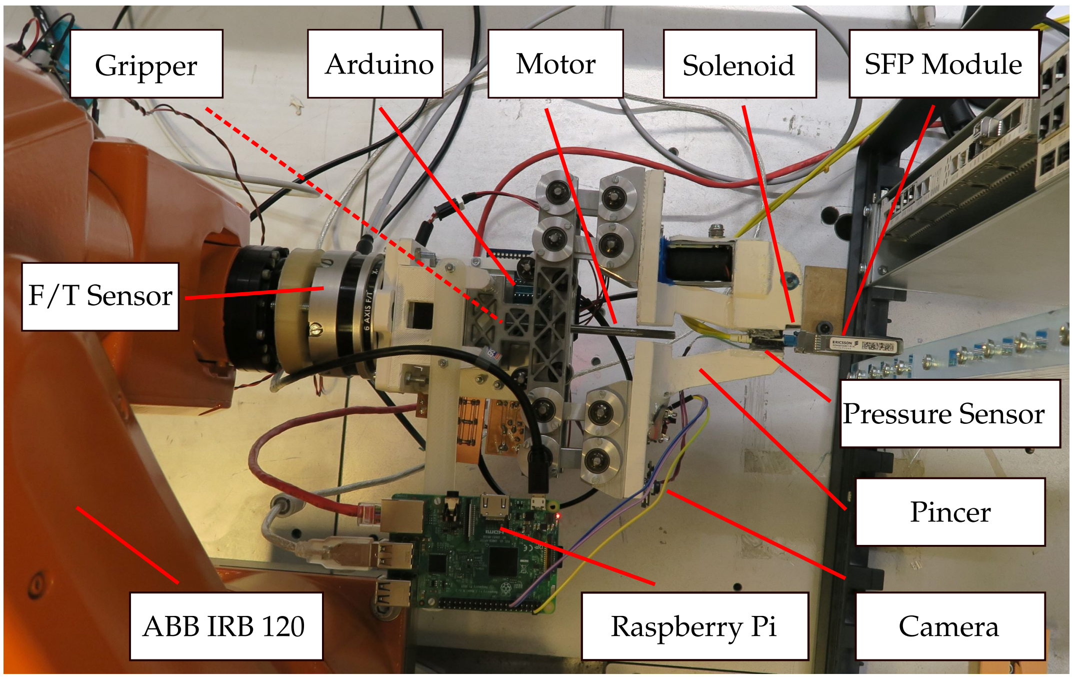
Implementation

This chapter focuses on describing the means of implementation based on the design approach. The system architecture will be demonstrated in a deep level and will be followed by the programming, manufacturing and integration of various subsystems.

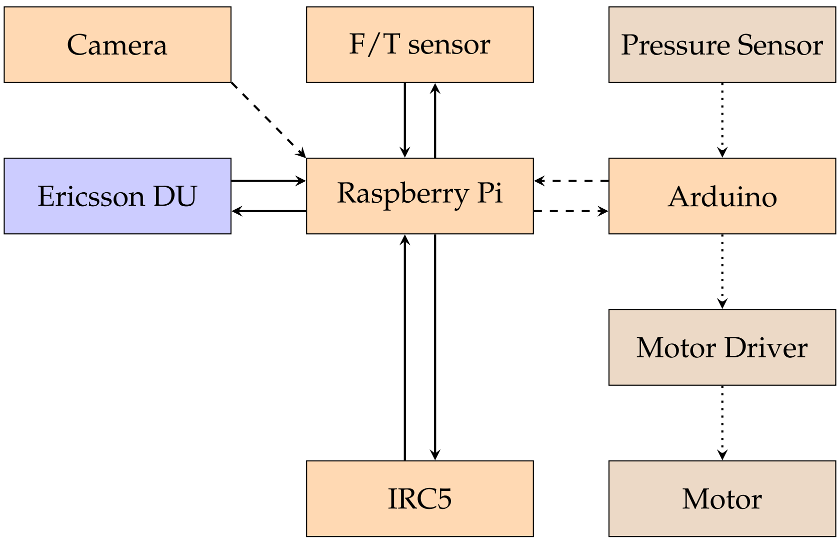
4.1 System Architecture

The overall goal is to replace a malfunctioning SFP module with a new one. For this purpose a gripper arm was developed which consists of several components, see Fig. 4.1a. The communication between these components is shown in Fig. 4.1b. Furthermore, the cable replacement procedure follows the steps below:

1. Incoming message from the DU triggers the start of the operation. All needed information for the replacement of the faulty SFP are provided on a web server.
2. The Raspberry Pi fetches these information and sends a command to the IRC5 to move in front of the faulty port.
3. The Pi Cam provides fine-tuning for positioning the arm directly in front of the port.
4. The Raspberry Pi commands the Arduino to control the motor for closing the claw and gripping the faulty SFP until a needed pressure threshold is reached.
5. The Raspberry Pi controls the IRC5 to remove the faulty SFP, pick a new one from the helper station and place it in the DU cabinet.
6. A message is sent to the Ericsson base to confirm that the operation was successful.



(A) Fully equipped gripper arm with ABB IRB 120.



(B) Flow chart of the system. The arrows represent a specific type of communication: (→): Socket Communication, (- - →): Serial Communication, (····→): Digital Communication.

FIGURE 4.1: The developed gripper arm with all its components and the respective flow chart.

Network

In order to facilitate numerous communication lines via Ethernet, a network was set up and configured using a switch. As the IRC5 was configured to be on a specific network, the remaining nodes were configured to match this. Nodes on the network include the IRC5, Raspberry Pi, computer with RobotStudio, the 6-axis force-torque sensor DAQ interface.

4.2 IRC5 Operation and Programming

The ABB IRB 120 robot comes with an in-built controller module which can either be controlled manually with the FlexPendant (operator) or automatically with the ABB RobotStudio program. Figure 4.2 provides an overview of the IRB120 operating zone.

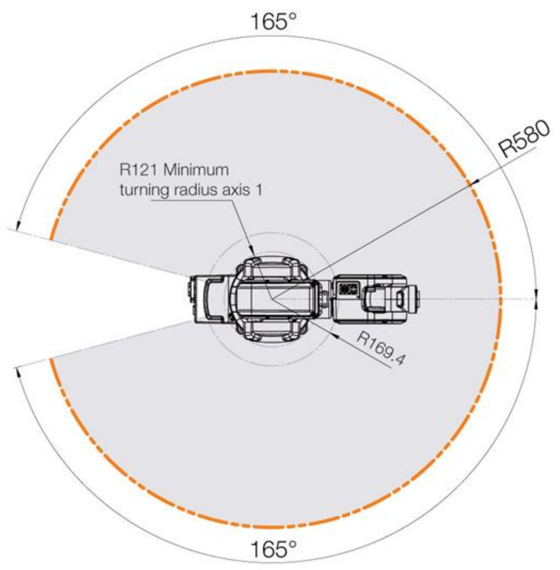


FIGURE 4.2: Operating zone of the IRB120 [19].

The IRB120 was utilized in the automatic mode for this application. The RobotStudio environment provides a live monitor of the real-time system. The RAPID programming language was used to produce the following functions in the RobotStudio program:

- Move tool to absolute coordinates.
- Move or rotate the tool relative to the main axes.
- Record work object coordinates.
- Broadcast current state of operations.
- Start a host server for remote connections.
- Send and receive messages over an ethernet connection.

These commands often operate based on the origin of the used coordinate system. In the case of using the tool coordinate system, this origin is set every time the robot is calibrated and therefore may change with each iteration. To address this issue with the coordinate systems, a deeper study is required to understand the IRB mechanisms.

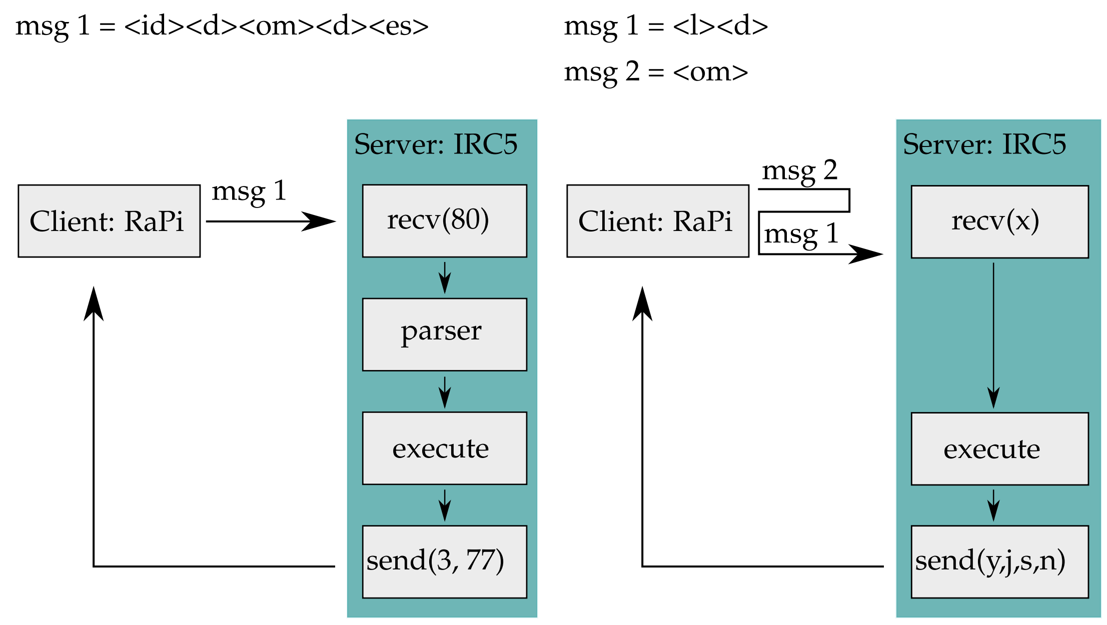
The IRB120 uses two main coordinate systems for operation:

- The base coordinate system: This coordinate system is fixed and does not move or rotate with the robot. This allows all points in space to have a static map and allow easier understanding of the robot positioning.

- The work object coordinate system: The work object coordinate system is connected to a user defined or default work object. This means that this coordinate system is independent of the robot but moves with the work object.
- The tool coordinate system: This coordinate system is fixed to the tool and rotates with the robots movements. The z+ axis of this coordinate system is always in the direction of the linear movement of the tools tip. This coordinate system can be used to move the robot in a linear fashion but requires more careful modeling and analysis of inverse kinematics.

Since the rack was installed perpendicular to the floor surface, a custom-defined coordinate system based on the base coordinates was used to provide a simpler approach and more understandable robot behavior for autonomous movement. The program was implemented such that the robot can move to any point in space using the base coordinates or move any relative value on the three main axes. This setup allows the IRB120 to be controlled automatically by a Python code from the Raspberry Pi after initialization by a PC and calibration.

Communication with Raspberry Pi



(A) Project *open_abb* [20].   (B) Code refactoring of *open_abb*.

FIGURE 4.3: Comparison of different solutions to access the ABB RAPID API via socket communication.

The communication between the RPi and the IRC5 is established via socket communication (see appendix D.2). Thereby the IRC5 is set up as a server and waits for incoming requests. The main problem to discuss here is how the ABB RAPID API can be accessed with the RPi by sending a message to the IRC5.

One popular work was published by Michael Dawson-Haggerty under the name *open_abb* [20] and works as follows,(comparable with Fig 4.3a):

- The RPi sends a message (command request) of a specific format that is expected by the IRC5. This message contains the following information: <*id*><*d*><*om*><*d*><*es*>, where

<*id*>: Used to address a specific case in the TEST-CASE instruction in RAPID. The id number has to consist of two digits, e.g 01, or 10.

<*d*>: The delimiter, which equals a space (' ').

<*om*>: The original message to send to the IRC5, which only consists of data up to 75 bytes, e.g. coordinates. This is an optional parameter and the values have to be separated with a space (' '). A maximum of 8 values are allowed. The given information will be casted into an array of size 10 by the IRC5.

<*es*>: The end sign which indicates the ending of the whole message and equals the sign #.

- The IRC5 always tries to receive the maximum amount of 80 bytes.
- A parser stores the received message into several internal variables which can be further used.
- Eventually the command request is executed.
- An answer containing the addressed case (id), a boolean value which indicates if an error occurred or not and data (optional) is sent back to the Rapi. The id number and the boolean value take 3 bytes and the data up to 77 bytes of space.

Despite the fact that "only" 75 bytes of data can be sent to and only 77 bytes of data can be received by the IRC5, the most important reasons for refactoring the code are the use of a parser and the need of trying to receive the full possible 80 bytes. The latter results in the need of putting the RPi to sleep. Otherwise if the RPi sends two command requests (messages) quickly after another it could happen that the IRC5 ends up fetching both of them. Since this message would be no longer of the correct format, the parser gives out an error and the command can not be executed. Moreover, the job of the parser can be crossed out by introducing a better message protocol, which has potential to increase the overall performance of the IRC5.

The proposed solution is sketched in Fig. 4.3b and tries to eliminate the disadvantages of *open_abb*. The core idea is a different message protocol, which contains the information <*l*><*d*><*om*>, where

<*l*>: Length of the original message in bytes.

<*d*>: The delimiter \n.

<*om*>: The original message to send to the IRC5, which equals an array of size 12 of max. 80 bytes. The first element of the array represents the id number and therefore a specific case in RAPID. The remaining nine elements can be used for data, e.g. coordinates. The array will be casted into an array of size 12 by the IRC5.

The biggest consequence is that the IRC5 now reads a dynamic amount of bytes x from the socket which enables it to handle a sequence of command requests. Moreover, the full 80 bytes are available for data to send/receive and the message to send back is divided into four chunks (y, j, s, n). The first chunk y equals the length of the original message (answer, chunks j, s, n), which is in JSON format and contains the id number, a boolean value indicating if an error occurred or not and data (optional).

Furthermore, a parser is no longer needed because the IRC5 directly translates the original message into an internal array.

These changes have made the code more readable and increased the overall performance. Despite all that, several error codes were introduced to give better feedback to the client:

id = -1:	There is currently not a case specified in the RAPID code belonging to the given id number.
id = -2:	The given length l of the original message exceeds the bounds $0 < l \leq 80$. OR: The length could not be read/ casted into a number.
id = -3:	The given original message could not be casted into an array.
id = -4:	The given length is greater than the actual length of the original message.
id = -5:	The given id number is below zero, which is not allowed since the id numbers < 0 are reserved for error codes.
id = -10:	The data supposed to send back is too long ($>$ 80 bytes) and has to be reset.

4.3 Gripper

The gripper is the subsystem used to facilitate the functions intended to extract the cable from the faulty SFP, retain the orientation of the cable during translocation between the faulty and functional device, ensure steady grip when inserting the cable into the new SFP and the assembly in the DU rack. Due to the characteristics of the lock mechanism of the cable and the SFP, the gripper needs to accommodate a mechanical apparatus to enable unlocking them.

Due to the complexity of the task it was deemed unfeasible to purchase a gripper of the shelf. Unknown parameters such as the shape and stability requirements of the cable unlocking mechanism and the strict width limitation of the gripper to fit between multiple SFP:s led to the choice of constructing a highly customized and modular gripper, which could be used as an experimental platform to host a wide range of pincer constructions.

Thus, the following requirements were produced:

- The housing shall be able to withstand the working forces of the motor both before and after force conversion.
- The mechanism shall be able to produce a pinching force described by $F_{pinch} = \frac{F_{insert}}{\mu}$, where F_{insert} is the required pushing force for insterting an SFP (or cable) and μ is the friction number between the cable and the grip surface.
- The mechanism shall be self-locking, so that if there is a power-out the claw will not let go of it's pressed object.
- The mechanism shall be able to close around objects with varying size.
- The total weight of the gripper shall not exceed 3kg [19].

Gripping Mechanism

The purpose of the gripper mechanism is to translate the rotational forces from the motor into as high a lateral force as possible at a predefined point. The gripper mechanism module begins at the electrical motor and ends at the platforms onto which the pincers will be mounted. Specific care has been taken to design this mechanism in order to not constrict the design of the pincers more than necessary, i.e. allow for greater freedom in designing the pincers. Combined with the overarching requirements for the gripper, these extended requirements are applicable to the mechanism:

- The mechanism shall be self-locking, allowing for sequential programming of the motor driver as well as safety against over-constraining the motor.
- The mechanism shall be able to withstand the minimum force of 24 N from gripping the cable.
- The mechanism shall be able to withstand the force of 24 N from inserting an SFP into the DU.
- The mechanism shall be able to withstand the force of 16 N from inserting a cable into the SFP.

A sketch pertaining to where and how these different forces affect the system can be found in figure 4.4, where F_I is the insertion force, F_P is the pinching force, and F_M is the force experienced by the nut from the motor.

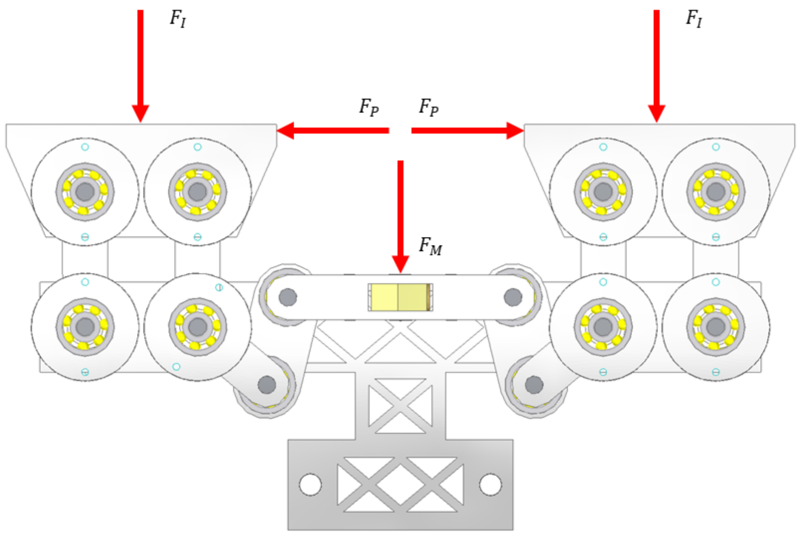


FIGURE 4.4: Forces affecting the Gripper Mechanism

Generation of Motor Force

The forces with which the gripping strength is controlled is generated by a rotational motor. From the start of the SOTA process up until construction of the gripper, a rack and pinion design, see part 2.2, was thought to be the optimal choice. This however turned out to not be the case since the solution does not provide a self locking mechanism. When power goes out or if one simply wishes to not constantly have to strain the gripper motor for a constant pressing force, this is a problem.
The new design became a leadscrew with a V-thread solution, visualized in figure 4.5. This solution also produces high precision and high force transmission by sacrificing opening/closing speed which is of little importance, as explained in chapter 1.3. Thus, the generation of the motor force follows the equation set out in equation 4.1. This transforms the rotational force from the motor into linear force.

$$F_M = \frac{2T_\mathrm{M}(1 + \mu \sec(\alpha) \tan(\lambda))}{d_\mathrm{m}(\mu \sec(\alpha) + \tan(\lambda))} \tag{4.1}$$

d_m	Average leadscrew diameter
λ	Lead angle
α	Half thread angle
μ	Friction coefficient between rod and nut
T_M	Holding torque from motor

TABLE 4.1: Calculation of motor force [21]

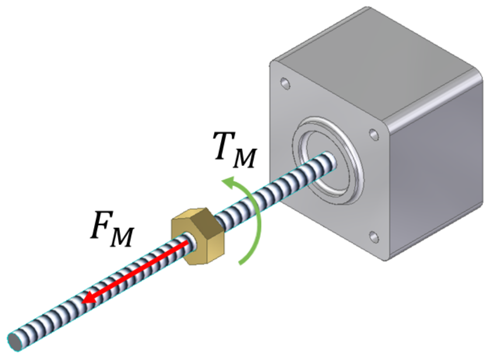


FIGURE 4.5: Forces affecting the Gripper Mechanism

Transforming linear force in one direction to a perpendicular direction

In order to achieve an opening and closing force, the linear force from the screw needs to be transformed into forces perpendicular to its motion. This was accomplished by introducing a linkage system visualized in figure 4.6.

Perpendicular grip

To provide the function of being able to close around objects with varying size the mechanism had to be designed so that the pincers are always perpendicular to the gripper base. This is solved by implementing a four bar mechanism which allows for a controlled movement through the entire gripping process [22].

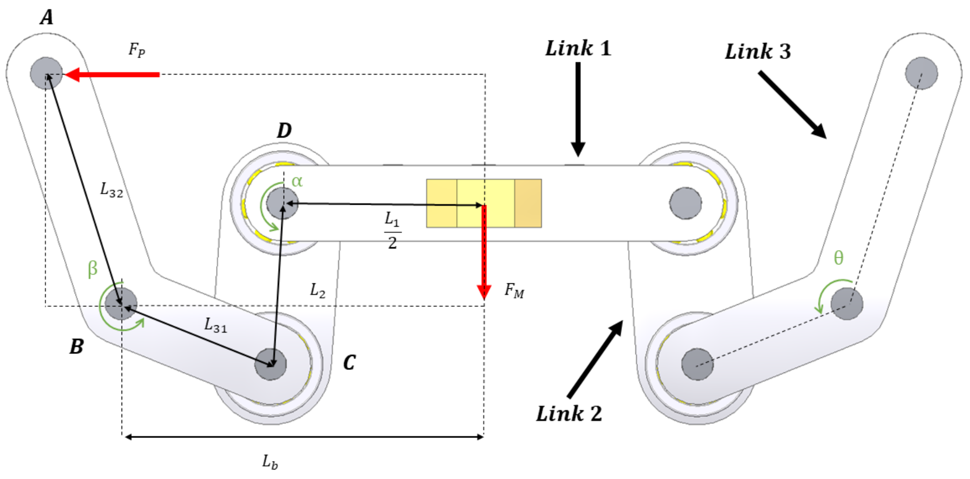


FIGURE 4.6: External forces affecting the linkage in the mechanism Mechanism

Material of the Mechanism

Various materials were tested as material for the linkage. The first material considered was the PLA plastic. The plastic is used in 3D printers which made manufacturing the models from CAD easy. Due to increasing complexity of shape and low tolerances of the linkage the 3D method had to be abandoned. The second and final choice of material was Aluminum, it was more durable than the plastic but involved a longer process to manufacture the modeled parts but provided the necessary tolerances to make a play-free construction.

Bearings

To ensure that the force generated from the motor gets leveraged efficiently when applied to the grippers, the linkage mechanism must provide smooth and low-friction rotation even when the forces F_P,F_I and F_M acts upon the linkage. This is achieved by fitting all rotating joints with ball-bearings to minimize the wear and friction. The main reason for choosing ball-bearings over other kinds such as plain friction bearings is because of the superior friction coefficient $\mu = 0.0015$, making it possible to neglect the friction forces in analysis of the linkage. Also and the availability of calculations tools to choose the correct bearing for the given application. Another aspects that weighed in to choosing ballbearings over other types of bearings is the insights provided by the available equations needed in the process of deciding upon a suitable bearing.

As seen in figure 4.7 there are in general two components of the resulting force acting upon the ballbearing which are of interest, F_R and F_A which are the radial and axial force. Since all the external forces acting upon the gripper is in the radial plane of the ball-bearing, the force F_A is estimated to be zero in this application. F_R is however the resulting force of the motor force F_M and the insertion force F_I. The pincer force, F_P, is excluded in this case as it is an reaction force of F_M. In the gripper application the bearings will never make full rotations, hence the dimension of the

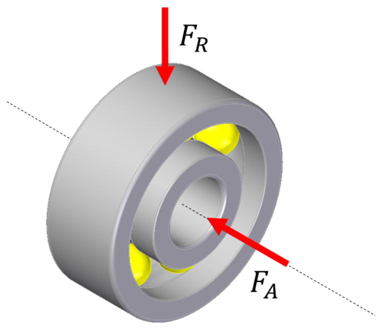


FIGURE 4.7: Forces acting upon a bearing in the mechanism

bearings will be done according to the static bearing load. The resulting static load

$$P_0 = X_0 F_R + Y_0 F_A \tag{4.2}$$

is given by the radial force F_R based on the conclusions of forces only acting the the radial plane. X_0 and Y_0 are weight factors depending on the load case of the bearing. The static bearing capacity

$$C_0 = s_0 P_0 \tag{4.3}$$

is the determining parameter from which the bearings can be chosen accordingly. The value s_0 is the static safety factor and varies with application. Recommended values are accessed from the SKF catalogue[23].

Pincers

This section will describe the functionality and purpose of the pincers in the claw assembly. The different functionalities will be explained in terms of why and how they are implemented.

Overview

The pincers of the robotic end effector are used to grip the optic cable and the SFP module throughout the steps of the replacement procedure. An overview of the pincer assembly can be seen in Figure 4.8.

The following requirements were listed for the pincers:

- The pincers shall be able to grip the cable, with a force less than what breaks it, so that it cannot move while being inserted or ejected.
- The pincers and it's components shall not break under forces and circumstances required to fulfill the replacement process.
- The pincers shall be able press the unlocking pin of the cable with a force greater than 1.5 N, see chapter 5.

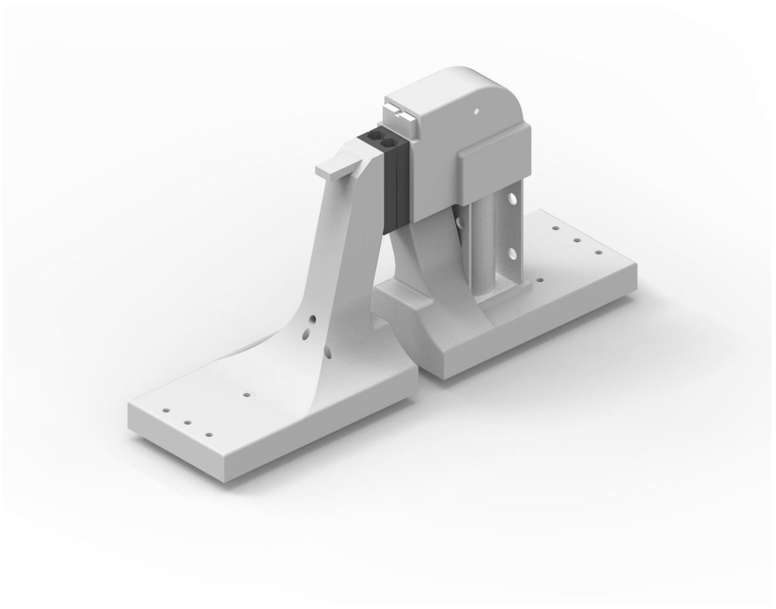


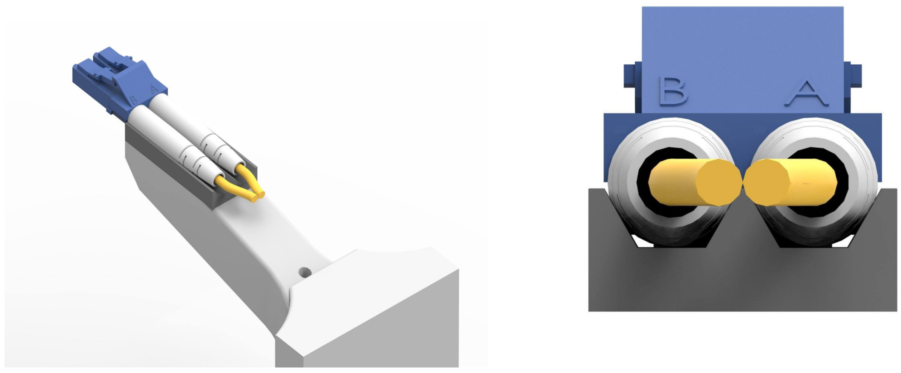
FIGURE 4.8: Pincer structure together with solenoid and tooth solution.

Bulk Parts

In order to enable quicker prototyping steps the material chosen for the bulk structure of the pincers was chosen to be of printable PLA plastic, so that requirements regarding strength and form could be satisfied through testing through iterations of the model.

Touching Surface

A custom fitting surface was designed to be able to grip the cable in such a way that the cable does not under any circumstances move between the pincers. Ninjaflex is chosen as contact surface material due to the material properties such as flexibility and that it is able to manufacture by 3D printing [24]. This means that the gripping surface, and therefore gripping friction, can be increased smoothly by simply applying more pressing power. For calculations regarding how much pulling force the gripper can use before there is a risk of it losing it's grip on the cable, the friction number $\mu = 0.67$ was assumed. This is the friction number of a rubber belt in contact with the plastic type urethane [25]; a combination which was deemed close enough to the relation between the rubbery Ninjaflex and the hard plastic cable casing.



(A) (B)

FIGURE 4.9: The Ninjaflex contact pad. (A) The length is enough to grip the entire rigid part of the cable. (B) The profile of the Ninjaflex pad centres the cable in a firm grip.

The profile has been purposely made to have a rough fit with sharp edges, see 4.9b. This provides reliable centring even though the Ninjaflex material has a way of shrinking in the printer.

Pressure Sensor

To be able to accurately measure the force with which the touching pads are subject to, a force sensor pad was stationed under one of them, see C. This combined with a suitable control algorithm was to decide when the pincers pressing force was at an acceptable level. For details of the circuit running the sensor, see B.

Solenoid

To be able to repeatedly toggle the locking pin of the cable before pulling it out, a pin-pushing functionality was to be implemented. This functionality is provided by a solenoid-driven pushing pin mechanism applied to the top pincer, see Figure 4.10.

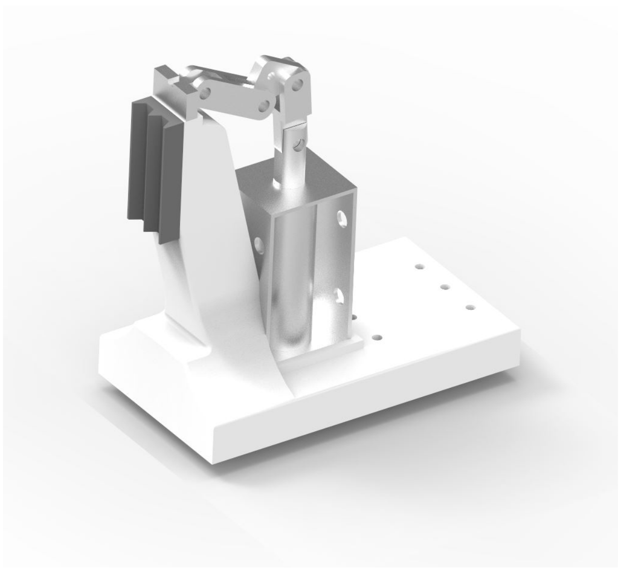


FIGURE 4.10: Solenoid link mechanism.

The purpose of the link mechanism is to translate the solenoid pulling force to a pushing force directed 90 degrees from it's original direction. This is done with an L-shaped link which has it's center axially aligned to a pin connected to the solenoid housing. When applying it to the model developed in CAD, the solenoid pin never has to extend further than 2 mm. According to the solenoid datasheet, one could expect a continuous force of at least 2 N within that range. Since the pushing device is not expected to act continuously but during short bursts, the solenoid duty cycle can be assumed to be lower, which means even more force could be derived [26].
The link mechanism's force exchange could be simplified by disregarding friction losses and then calculated by assuming that the work done for the solenoid rod and the pushing pin is the same. Since the CAD-model provides the distances d_{solenoid}, $d_{\text{pushingpin}}$ and the datasheet provides the applied force [26] an equation can be formed for the force of the pushing pin. To simplify, the applied work is assumed to be fully preserved through the solenoid mechanism, thus forming the following equation.

$$F_{\text{pushingblock}} = \frac{F_{\text{solenoid}} \cdot d_{\text{solenoid}}}{d_{\text{pushingblock}}} \tag{4.4}$$

To control the solenoid a PCB-driver was created. The circuit design can be seen in B. The signals controlling the circuit comes from the Arduino module while the power is taken from the power distribution module.

Hooking Mechanism

The hook is used to unlock the latch of the SFP-module, which prevent unintended removal. The claw is intended to tilt and drag the hook along the surface of the SFP-module until the latch is open. After the procedure is done, the gripper will remove the SFP-module and continue with the operation.

Physical connection with ABB arm

In order to interface between the gripper and the robotic arm a fastening was designed. Since a multi-axis force sensor was also to be placed between the gripper and the arm, the interfacing had to be done in two parts; one interfacing between the gripper and the sensor, and the other between the sensor and the arm.
The requirements were:

- The interface shall not break under the working forces of the systems work cycle.
- The interface shall guarantee that the grippers position will not be able to deviate from it's initial fastening over multiple consecutive work cycles.
- The interface shall not affect the functionality of the multi-axis force sensor.
- The interface should preferably be of low weight and size.

The multi-axis force sensor from OptoForce was designed to be mounted on a Universal Robot (UR), thus making the proposed attachment in the data sheet insufficient for the ABB robotic arm. A custom made fastening was therefore designed to attach the sensor to the arm. An exploded view of the proposed sensor fastening can be seen in Figure 4.11 and the custom made fastening interface in Figure 4.12.

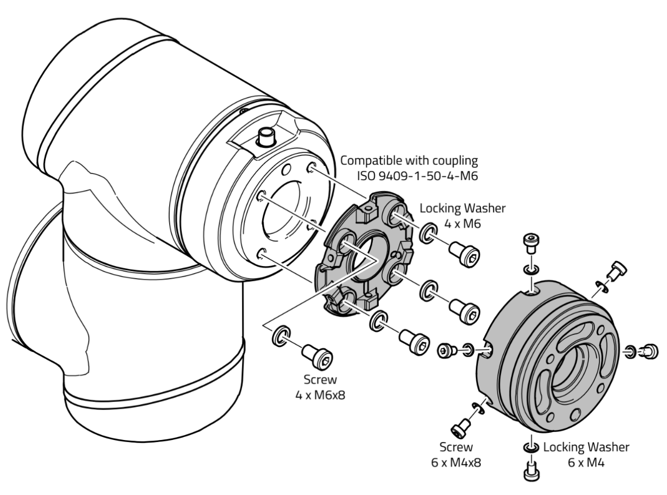


FIGURE 4.11: Multi-axis sensor attachment proposal from OptoForce to a UR. [27].

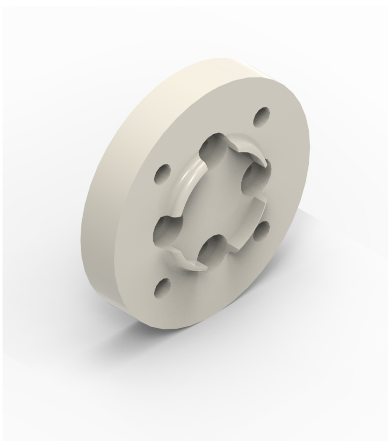


FIGURE 4.12: Fastening interface between the ABB robotic arm and multi-axis sensor.

The expected working forces on the gripper module were thought to be relatively low (just above 24 N, see part 5.3), therefore a lighter material such as printable PLA was chosen, both for the material properties but also due to the fast manufacturing since 3D printers were available with the desired material.

Gripper controller

The purpose of the gripper controller is to make the mechanical actuators i.e the pinching mechanism and the cable unlocking mechanism controllable from the main system. To fulfill this requirement the chosen micro-controller (appendix C.6) was installed with the driver (appendix C.10) capable of empowering the electrical motor (appendix C.1).
To realize this function, a controller having two coupled parameters was implemented. The parameters, the amount of steps sent to the driver and the perceived pressure of the sensor, are coupled in the sense of both being able to stop the action of pincer movement under certain conditions. Starting with the problem of using

the pressure as a control parameter. As mentioned in the touch pad section 4.3 the gripper was fit with a pressure sensor to ensure that the cable would not be squeezed too hard as it may sustain damage but hard enough the maintain a sturdy grip. This meaning that when gripping with a small force the precision can be coarse but have to increase with the amount of pressure applied to the object being gripped. Implementing a simple relative error measurement would not be sufficient since the relative value of a small value would be even smaller and thereby unreachable by the motor. This would lead to oscillations around the reference due the step size of the motor. Instead the metric introduced is

$$\text{Variance} \cdot \text{Reference} + \text{Bias} < \text{Accepted Value} < \text{Variance} \cdot \text{Reference} - \text{Bias} \tag{4.5}$$

which allows for a small relative error for high values yet a bias which let you set the minimal value to a level high enough to fit within the precision of the motor. This also serves as a measure to contain the noisy and non-linear data within a linear bound. The motor control was performed by a timer-interrupt which was started and stopped given two control parameters; the number of pulses to be run and the resulting gripping force to be experienced. The timer interrupt for the motor kept track of the number of pulses, whereas a secondary timer interrupt was set up for measuring the pressure sensor. As these interrupts may influence each other and thus compromise the system stability each interrupt was scheduled with a specific frequency not to disturb each other whilst still being fast enough to provide the needed responsiveness to control the gripper. Since the whole process of replacing an SFP had to be fully automatic a number of safety limits has been placed to make sure that the system would not be damaged or harm the environment around it. On a low level two pressure limits were introduced; a soft stop controllable from the main system and a hard stop which is encoded in the micro-controller. Reaching the soft stop is the same as reaching the pressure reference set from the main system, this triggers a flag to be set to zero, all movement of the pincers cease and the amount of pulses actually sent to be returned to the main code. A hard stop stops all commands concerning opening and closing the gripper since the measured pressure is close to yielding damage on the gripper. To reset this hard stop, the pressure sensor must be set to zero. This is forcing the operator to take a look at what actually caused the gripper to reach the critical limit. After establishing the reason the operator can open the gripper with the pressure sensor unplugged until the values return within the allowed bound. To control the micro-controller incoming messages sent from the main system in the form of

$$[\text{ID}, \text{Number Of Data Points}, \text{Data 1}; \ldots; \text{Data N};] \tag{4.6}$$

was interpreted into various commands for controlling the gripper. The message ID states which function to be run with the data points as input. At the end of the function, a return message was created which elaborated on the current state of the gripper to allow the main system to keep track of the overall system state. In the same manner as the micro controller has its main loop, the main system has a thread polling the connection. When a message is received the respective function matching the ID is run. This structure allows for precise monitoring of the pincer positions at all time. The gripper was programmed in the main system to self adjust through recursion if the reference pressure was not reached by either opening or closing a small amount. Identically, errors were mapped to different combinations of the received pressure and pulses for the system to trigger a cease operation case.

4.4 6-axis Force Sensor

To be able to insert the new SFP into the DU and feel forces in the Z-axis, a 6-axis Force Sensor is used C.9. It is mounted between arm and the claw, as can be seen in figure 4.13. The sensor is connected via a serial interface to a DAQ device that interprets the data and transmits it in USB or ethernet form.

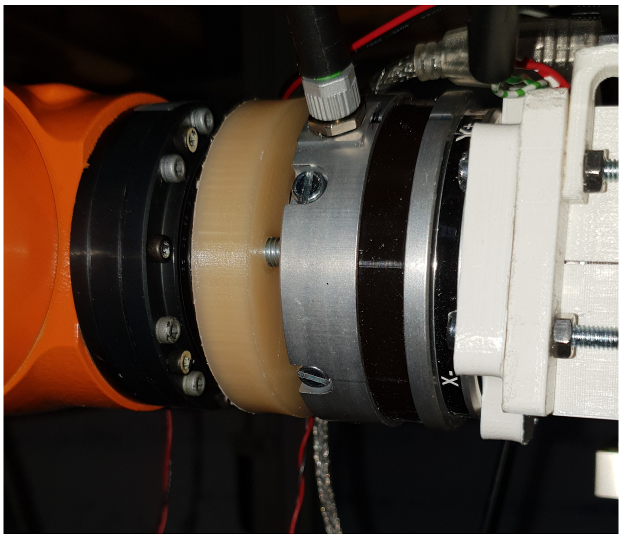


FIGURE 4.13: Sensor mounting with the arm to the left and the claw to the right.

Sensor Interface

The DAQ device provided from OptoForce had a wide array of working interfaces, but due to the fact that a microcomputer running Linux was at the other end of the communication the choices narrowed down to USB or Ethernet.

The USB interface was thoroughly explored and due to incompatibilities found with the Raspberry Pi it was determined to be a dead end. Ethernet however had a pre-written C script that was proven to work on the Pi, hence it was chosen as the sensor interface.

UDP versus TCP

There were two Ethernet protocols provided: UDP and TCP [28]. These two protocols differ in terms of speed and reliability, with UDP sacrifing reliability by not having any error-checking after a message is sent to reaffirm that the message was sent correctly. This allows for increased speed of communication, which is vital in some applications. This application however had no time requirement critical enough to warrant this lack of reliability, thus TCP was chosen.

Interfacing with the Python code

In order to use the provided C code it required interfacing with the python code. This was done using SWIG [29], which allows for creation of a usable library of functions from one programming language to another. Four functions were designed on the C side to be used in Python:

- **Connect** - In order to set up the sensor, a connection had to be established with the sensor. This was done by the main C code and simply established a TCP connection. But in order to use the sensor it needed to be calibrated, which gave a conversion framework to be able to return the forces in N and torques in Nm. After these steps were taken the sensor was ready to be used.
- **Read** - Once a connection is established, the sensor values can be read by sending a request to the DAQ. This would return all the forces and torques as efficiently as possible.
- **Ping** - Due to the fact that the DAQ would shut down the connection if no requests were made within one second of the last request, a ping function had to be implemented to send a request without doing anything else.
- **Disconnect** - Before shutting down the program the sensor needed to be disconnected, which simply meant sending a disconnection message to the DAQ before exiting the program.

The Python side was constructed as a Sensor class which has corresponding functions for all the previously mentioned C functions:

- **Connect** - Simply uses the interfaced Connect function.
- **Zero** - Reads the current values of the sensor in order to set them to zero, since the original values are taking the claw and mounted electronics into account which was unnecessary.
- **Read** - Uses the read function to read the raw values from the sensor and then subtracts the zeroed values to get the relative forces and torques. Due to the fact that the sensor is rotated in reference to the robot's coordinate system, the forces were rotated in order to be able to be easily used with the rest of the code. This is explained in the section below. The function then saves those values to class attributes that can be accessed from outside the class.
- **StartPing/StopPing** - To avoid having to manually ping the DAQ to keep the connection alive, a thread is created that continuously runs the Ping function and then waits for approximately 0.9 seconds. This ensures that the connection is alive because the only scenario where the time between pings would be longer than the wait time would be when the sensor is being read by another function. This thread needs to be killed when sensor usage is complete, which is done with StopPing.
- **Disconnect** - Uses the interfaced Disconnect function to end the connection.

Axis Rotation

Due to the fact that the sensor coordinates are rotated in relation to the robot coordinates around the z-axis, a transform had to be made to keep all the coordinates of the main code the same. This was done by multiplying the coordinates with a rotational matrix according to this formula:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.7}$$

This calculation is performed after the values have been read and zeroed, allowing for the angle to be changed if necessary.

4.5 Computer Vision

As the process enters the stage of inserting the cable into the new SFP, a Computer Vision algorithm is used beforehand. The robot moves to a position where the camera, mounted on the lower end of the claw, is directly in front of the SFP. A picture is then taken and processed by the algorithm which outputs the movement required to reach the optimal position for insertion.

The Camera Module

The camera used for this project is a PiCamera Module 2 C.8. It was the optimal choice due to having extensive compatibility with the Raspberry Pi C.7, being very light-weight and also being able to capture pictures with a very high resolution (3280 x 2464), giving high accuracy for the algorithm. This is especially important since the algorithm focuses only on a small part of the picture due to being so far away.
The camera is mounted on the lower end of the claw, as seen in figure 4.14. In order to take a picture, the claw needs to move a fixed distance along the x-axis to reach the position required for taking a picture. After the algorithm is complete and the adjustments have been made, the robot simply moves back the offset distance to being in front of the SFP and begins insertion.

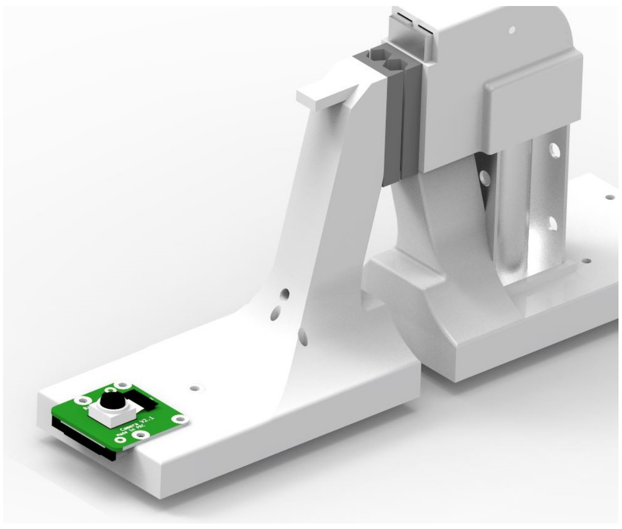


FIGURE 4.14: Camera Module mounting.

The LED Flash

To help the algorithm detect lines, a LED flash circuit was designed B.2. Because the algorithm relies on differences is light levels the LED flash is mounted next to the camera. Shining from the side it helps further exaggerate the edges of the SFP. It is being fed 5V from the Raspberry Pi's GPIO pins and is controlled with a transistor from the Raspberry.

The Algorithm

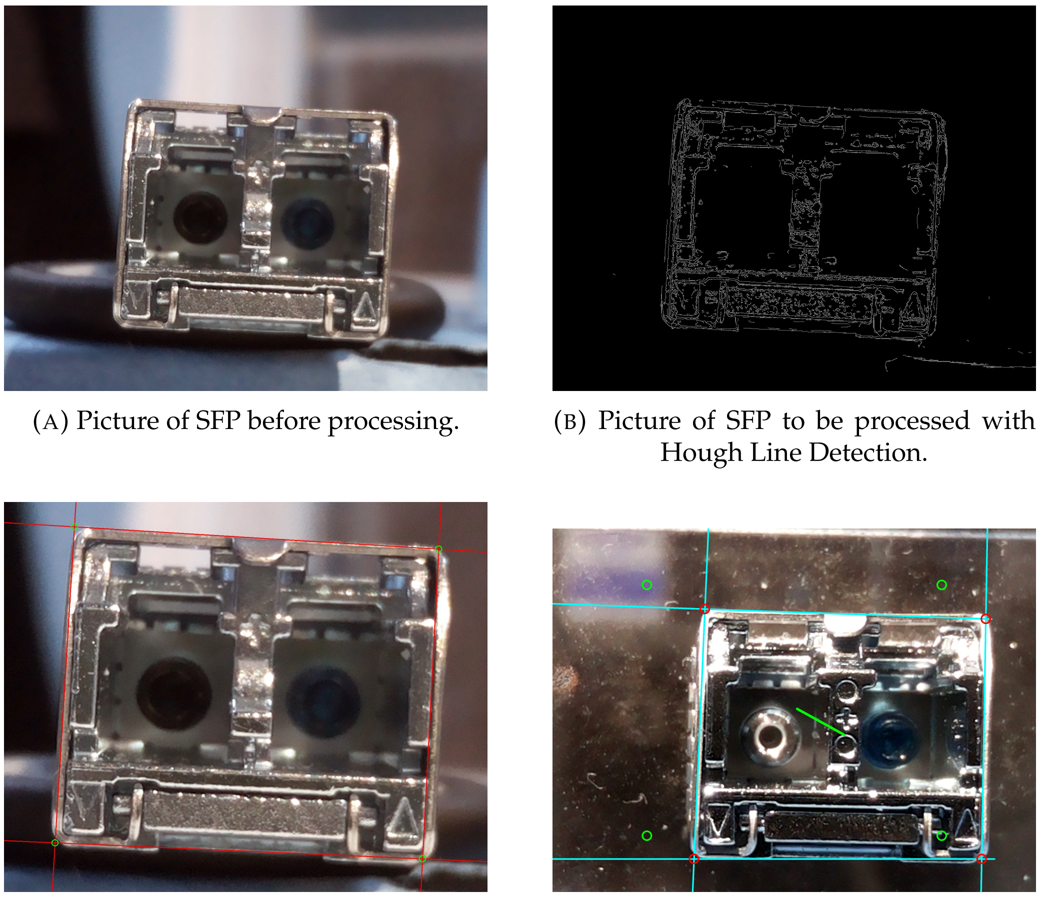
For the computer vision calculations OpenCV [18] is used. It is an open source library for everything Computer Vision-related.

Picture Capture

The PiCamera has a pre-installed library installed in Raspbian which can be easily used to capture a picture. The PiCamera is initialized and configured to maximal resolution and cropped to fit only the important area of the picture. This area is in the middle of the picture with 30% of the width and height of the original picture. This area is guaranteed to contain the SFP due to the known accuracy of the automatic movements.
Before capture the LED flash is illuminated through the Raspberry Pi's GPIO port, resulting in a picture like the one seen in figure 4.15a. The picture is then stored in a local array object for ease of use.

Line Detection

The main method used to distinguish the lines of the picture is Hough Line Transform [17]. In order to use this method, distinguished lines are required.
To achieve this, the picture is first converted to grayscale which is required for Canny Edge detection. The picture is then blurred [30] in order to erase smaller, less significant lines, since the edges of the SFP are very distinct shapes and therefore still distinguishable.
After blurring the picture is processed using Canny Edge Detection [31], returning a picture like the one seen in figure 4.15b which has very distinct lines. This picture is then processed using Hough Line Transform to detect all potential lines in the picture. In order to further filter out smaller lines, the threshold for the Transform function was put at a level where most false positives could be ignored.


(A) Picture of SFP before processing.

(B) Picture of SFP to be processed with Hough Line Detection.

(C) Picture of SFP with the final lines and points detected with the algorithm.

(D) Picture of SFP with ideal positions and the required movement, shown as the green line.

FIGURE 4.15: The entire Computer Vision process.

Line Filtration

The lines returned from the Hough Line Transform are in the form of ρ and θ, where ρ is the pixel at the end of the picture at which the line starts and θ is the angle of the line in radians. These lines are iterated over to divide them into vertical and horizontal groups, and also to filter out the line which do not align with the horizontal or vertical axis with a 0.1 radian margin. This allows for small angles in the lines in case the robot or SFP is slightly tilted but still requires the lines to be more or less horizontal or vertical.
The lines are then iterated over once again to find the outermost lines. This is done comparing the ρ of every line to find the ones with the lowest and highest values for both the horizontal and vertical lines. The points at which these lines intersect are then derived, and can be seen in figure 4.15c.

The desired position of the SFP is in the middle of the screen with a width and height that is known due to the known distance to the SFP. The required movement to reach this position is then calculated by using the mean distance between each of the corresponding points according to equation 4.8.

$$d = \frac{1}{4}\sum\sqrt{x^2 + y^2} \tag{4.8}$$

This results in a movement which will be mostly accurate and smooth out errors in the prior processes, seen in figure 4.15d. The robot will move according to this desired movement and then do the whole process again to check if the points are within a tolerated threshold of about 0.3 mm and then proceed with the insertion.

4.6 Power Distribution and Cable Management

The gripper had several components that required different amount of voltage. A power distribution from 1 power input where the different voltages split up to the components was therefore necessary to avoid multi-socket power supply. A block diagram of the power distribution can be seen in Figure 4.16.

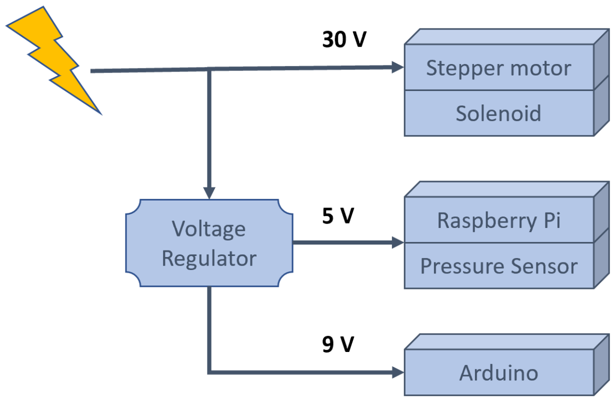


FIGURE 4.16: Block diagram of power supply to the gripper.

This solution allowed 1 power connection to the claw where the input voltage, which can be seen in the figure above, was 30 V. The power input was ought to be placed close to the power distribution unit, consisting of 2 voltage regulators and 1 PCB that had parallel coupling for 30 V and 5 V. The ABB robot end effector shall not be exposed to a weight of more than 3 kg as mentioned earlier in this chapter under requirements. It was therefore crucial to minimize the weight of the gripper. A solution for this was to have some components attached to the ABB arm, rather than on the gripper. The arm provided designated areas for where extra load or components could be mounted, as seen in Figure 4.17.

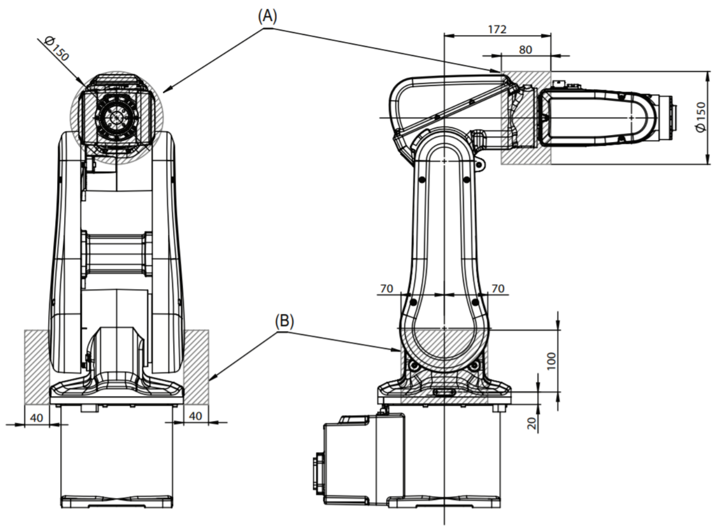


FIGURE 4.17: Load areas of the ABB IRB 120 robot [32].

The power distribution unit was attached on point A, which also had the main power supply for the tool. A mounting plate for the power distribution unit was designed to be attached to the arm. The mounting place was designed in Solid Edge with marked areas, identical to those in Figure 4.17 and was thereafter 3D printed. The final design of the power distribution unit can be seen in Figure 4.18.

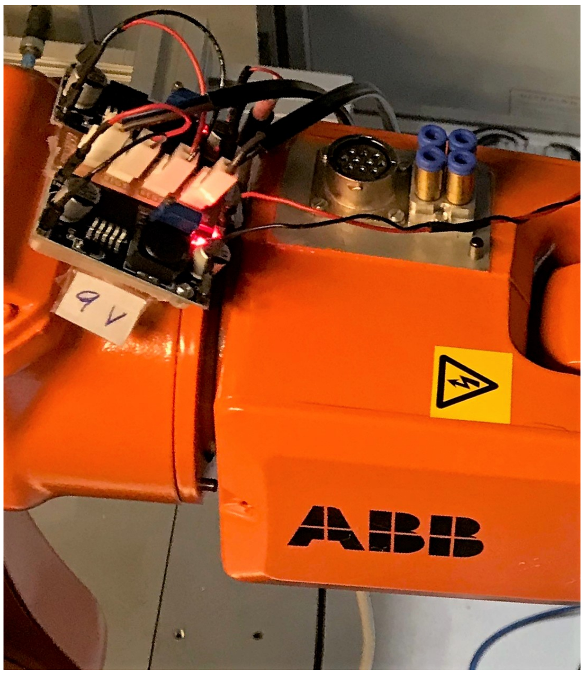


FIGURE 4.18: Power distribution unit on robotic arm.

The voltage input to the power distribution unit was connected with an R&D power supply where the cabling was connected directly from the power unit along the outside of the arm. The power was at an early stage thought to be distributed from within the robotic arm with the internal cabling. This solution was not the most optimal due to the risk of over-exceeding the allowed amount of current going through the robot. The cabling was therefore placed outside of the robot. The cables between the power distribution unit and the gripper was twisted to reduce the electromagnetic interference emitting from the wires. The cables were not lead from the power distribution unit to the gripper with a specific method, but rather connected directly to their end point.

4.7 Control Strategy

The problem of inserting a SFP module in a port is related to the *Peg in Hole* problem, which has been extensively researched over the years. Here, the peg is the SFP and the port is the hole.

In general, there are two common methods to find the hole location:

- Using visual information such as from a camera [33, 34].
- Using a force/torque sensor to estimate the position of the hole by considering the reaction moments when the peg approaches the hole [35, 36].

While the former often suffers from limited camera resolution and calibrations error [37], the latter highly depends on excellent resolution of a force/torque sensor, a precise model of a dynamic system, and a very fast control sampling time [38].

However, when a person inserts a plug into a hole, he or she usually only needs the approximate location of the hole rather than the exact position. This person would then rub the plug against the surface of the object with the hole until it finds it. Thus, the idea was developed to combine these two methods to a single controller design to be able to compensate for each others drawbacks.

System description

The setup can be seen in Fig. 4.1a and consists among others of the robotic arm, the camera module, the six-axis force and torque sensor (F/T sensor) and the gripper arm. To ease further discussions, the gripper arm and the F/T sensor are seen as one element, because the former is expected to be stiff. Moreover, since the gripper is holding the cable in which the SFP is plugged it and therefore not the SFP itself, it is assumed that the SFP module and the gripper are not (always) perfectly aligned as can be seen in Fig. 4.19. These assumptions result in the subsystem shown in Fig. 4.20, which introduces the coordinate systems:

K_r : Coordinate system x_r, y_r, z_r of the rack, which is equipped with the DU and the helper station.

K_s : Coordinate system x_s, y_s, z_s of the F/T sensor.

K_m : Coordinate system x_m, y_m, z_m of the SFP module.

Thereby the additional assumption is made that the distance between K_s and K_m, denoted by $\boldsymbol{d_{ms}}$, does not change during the operation (no slip), so that the coordinate transformation

$$\begin{bmatrix} x_\mathrm{m} \\ y_\mathrm{m} \\ z_\mathrm{m} \end{bmatrix} = \boldsymbol{R} \cdot \begin{bmatrix} x_\mathrm{s} \\ y_\mathrm{s} \\ z_\mathrm{s} \end{bmatrix} + \boldsymbol{d_{ms}} \quad ,\text{or} \quad \begin{bmatrix} x_\mathrm{s} \\ y_\mathrm{s} \\ z_\mathrm{s} \end{bmatrix} = \boldsymbol{R}^\mathrm{T} \cdot \begin{bmatrix} x_\mathrm{m} \\ y_\mathrm{m} \\ z_\mathrm{m} \end{bmatrix} - \boldsymbol{R}^\mathrm{T} \cdot \boldsymbol{d_{ms}} \qquad (4.9)$$

holds at any time. The rotation matrix $\boldsymbol{R}$ (orthogonal) is given by [39]

$$\boldsymbol{R} = \boldsymbol{R}_{z_s,\gamma_{\text{zms}}} \cdot \boldsymbol{R}_{y_s,\gamma_{\text{yms}}} \cdot \boldsymbol{R}_{x_s,\gamma_{\text{xms}}} \tag{4.10}$$

and can be expressed with the angles γ_{xms}, γ_{yms}, γ_{zms} as

$$\boldsymbol{R}_{x_s,\gamma_{\text{xms}}} = \begin{bmatrix} cos(\gamma_{\text{xms}}) & 0 & sin(\gamma_{\text{xms}}) \\ 0 & 1 & 0 \\ -sin(\gamma_{\text{xms}}) & 0 & cos(\gamma_{\text{xms}}) \end{bmatrix}, \tag{4.11}$$

$$\boldsymbol{R}_{y_s,\gamma_{\text{yms}}} = \begin{bmatrix} cos(\gamma_{\text{yms}}) & -sin(\gamma_{\text{yms}}) & 0 \\ sin(\gamma_{\text{yms}}) & cos(\gamma_{\text{yms}}) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.12}$$

$$\boldsymbol{R}_{z_s,\gamma_{\text{zms}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma_{\text{zms}}) & -sin(\gamma_{\text{zms}}) \\ 0 & sin(\gamma_{\text{zms}}) & cos(\gamma_{\text{zms}}) \end{bmatrix}. \tag{4.13}$$

Note that the indices follow the logic:

d_{ms} : Distance between K_m and K_s expressed in K_s coordinates.

γ_{xms} : Angle between x_m and x_s expressed in K_s coordinates.

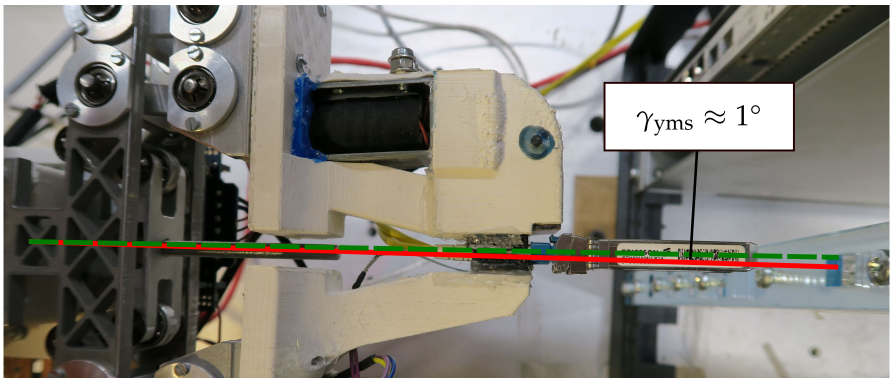


FIGURE 4.19: Usually the gripper and the SFP module are not perfectly aligned.

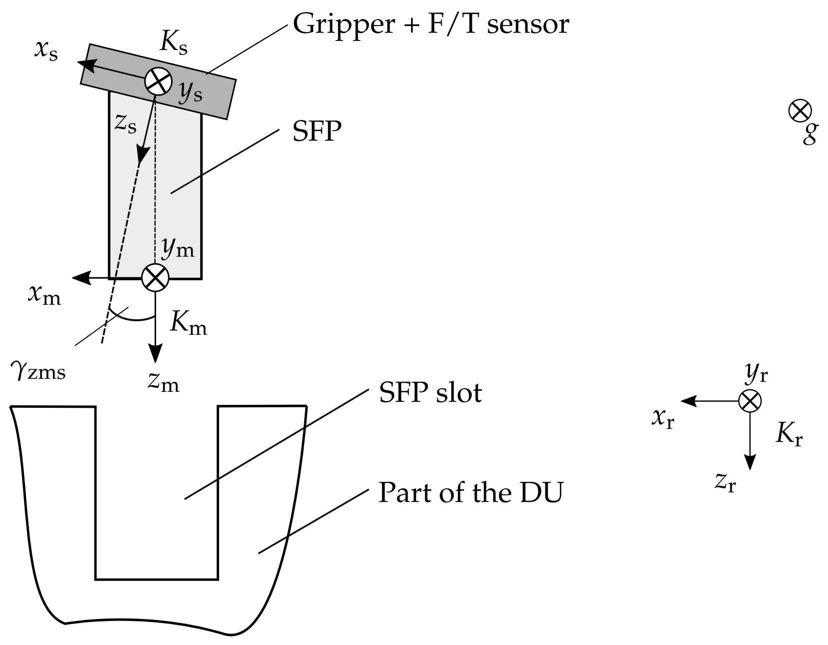


FIGURE 4.20: A part of the system with its respective coordinate systems.

Every possible case of the contact between the SFP and the DU are shown in Fig. 5.2 and 5.3, compare with [38]. It shows contact happens at four points, which includes the left and right lower-corners of the SFP, and the left and right upper-corners of the SFP slot.

Algorithm

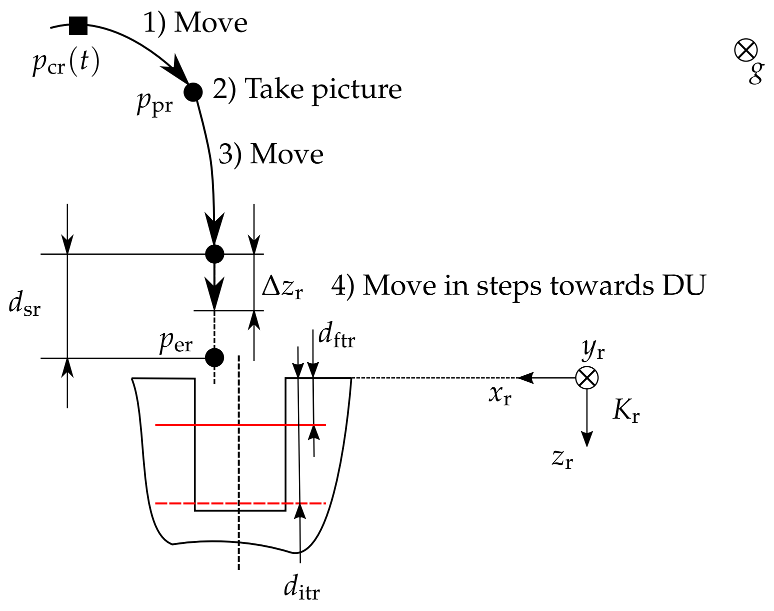


FIGURE 4.21: Definition of important points in space and illustration of the first four steps of the Algorithm 1.

The problem of finding the slot and inserting the SFP module in it can be divided into two parts respectively, since different information are needed for each individual solution.

The proposed solution for *finding the hole* assumes that the robotic arm just grabbed the new SFP module and hence is in a position away from the DU, compare with Fig. 4.21. Since the exact location of the slot is unknown, the first step is to move to a predefined position where an optimal photo can be taken to estimate its position p_{er} (step 2). It is expected that p_{er} is off from the center of the slot and that γ_{xms}, γ_{yms}, $\gamma_{\mathrm{zms}} \neq 0$. For these reasons a safety distance d_{sr} has to be chosen to ensure that the robotic arm does not bump into the DU when moving closer to it in step 3.

The next subtask is to find the hole with the help of the F/T sensor. All possible contacts and corresponding force vectors are sketched in Fig. 5.2. The main idea is to make a correction in position in the direction of the resulting force $\boldsymbol{F}$ or torque $\boldsymbol{T}$. It can be seen that the only consideration of the force or the torque results in unhandleable cases. For instance, the only information of the force would lead to a wrong correction in the position for all cases on the left side in Fig. 5.2a and Fig. 5.2b. The same applies to the torque, for which the unhandleable cases are drawn in Fig. 5.2b. Moreover, it can be seen that an optimal solution only based on the force and torque values can not be found. However, considering the forces and torques would solve most cases. The problem here is the implementation. Due to inaccuracies in production of all parts and the limited resolution of the F/T sensor, the task of finding optimal threshold values is a difficult and time consuming process. Since falsely actions can not be excluded completely, the easiest approach of only considering the torque information seems reasonable. The proposed algorithm works as in Alg. 1 and the idea is to move in small steps towards the due and take an action when the force in z_{s} direction, F_{zs} reaches the threshold F_{tzs}. The threshold ensures that the contact between the SFP module and the DU is strong enough to get a clear measurement and also that no damage to the SFP module is done.

Once the SFP module passes the threshold, marked as a straight red line in Fig. 5.2 and Fig. 5.3 (distance d_{ftr}), the problem of *inserting the SFP into the slot* starts. As can be seen in Fig. 5.3, all relevant cases can be solved by moving in the direction of $\boldsymbol{F}$, when the assumption of $\alpha \approx 90°$ is made. This seems valid, since the precision of the robotic arm is really high and tests showed that γ_{xms}, γ_{yms}, $\gamma_{\mathrm{zms}} \approx 1°$, compare with Fig. 4.19. The proposed algorithm works as in Alg. 2 and corrects the position when the contact force between the SFP module and the slot reaches a threshold, which ensures that the module is not damaged during the insertion process. Moreover, when the SFP passes the positioning threshold, marked as a dashed red line in Fig. 5.2 and Fig. 5.3 (distance d_{itr}), the task of insertion is completed.

4.8 Remote access to Raspberry Pi

We used Virtual Network Communication (VNC) to connect to the Raspberry Pi. VNC works by setting up the Pi as a server and the remote computer as a client. This allows the Raspberry's desktop environment to be shared with a remote computer over internet, thereby allowing remote access to the Raspberry Pi for code updates and monitoring purposes.

Algorithm 1 Finding the SFP slot (hole). Compare with Fig. 5.2.

```
Require: Predefined position in front of the SFP port of interest, $p_{\mathrm{pr}} = [x_{\mathrm{pr}}, y_{\mathrm{pr}}, z_{\mathrm{pr}}]^{\mathrm{T}}$
    in $K_{\mathrm{r}}$ coordinates.
 1: Move to the predefined position $p_{\mathrm{pr}}$. {Step 1.}
 2: Take a photo of the SFP port and estimate its position where $z_r \approx 0$. {Step 2.}
    The estimated position is in the following denoted by $p_{\mathrm{er}} = [x_{\mathrm{er}}, y_{\mathrm{er}}, z_{\mathrm{er}}]^{\mathrm{T}}$ in $K_{\mathrm{r}}$
    coordinates.
 3: Move to $p_{\mathrm{er}} - [0, 0, d_{\mathrm{sr}}]^{\mathrm{T}}$, where $0 < d_{\mathrm{sr}} < |z_{\mathrm{er}} - z_{\mathrm{pr}}|$. {Step 3.}
 4: while $z_{\mathrm{cr}}(t) < d_{\mathrm{ftr}}$ do
 5:   if $|F_{\mathrm{zs}}| \geq F_{\mathrm{tzs}}$ then
 6:     Move back to $p_{\mathrm{cr}}(t) - [0, 0, d_{\mathrm{sr}}]^{\mathrm{T}}$. {Step 5.}
 7:     if $|T_{\mathrm{ys}}| \geq T_{\mathrm{tys}}$ then
 8:       if $T_{\mathrm{ys}} > 0$ then
 9:         Move left to $p_{\mathrm{cr}}(t) + [\Delta x_{\mathrm{r}}, 0, 0]^{\mathrm{T}}$. {Step 6.1}
10:       else {$T_{\mathrm{ys}} \leq 0$}
11:         Move right to $p_{\mathrm{cr}}(t) - [\Delta x_{\mathrm{r}}, 0, 0]^{\mathrm{T}}$. {Step 6.2}
12:       end if
13:     else if $|T_{\mathrm{xs}}| \geq T_{\mathrm{txs}}$ then
14:       if $T_{\mathrm{xs}} > 0$ then
15:         Move out of plane to $p_{\mathrm{cr}}(t) - [0, \Delta y_{\mathrm{r}}, 0]^{\mathrm{T}}$. {Step 7.1}
16:       else {$T_{\mathrm{xs}} \leq 0$}
17:         Move out of plane to $p_{\mathrm{cr}}(t) + [0, \Delta y_{\mathrm{r}}, 0]^{\mathrm{T}}$. {Step 7.2}
18:       end if
19:     end if
20:   end if
21:   Move towards the DU to $p_{\mathrm{cr}}(t) + [0, 0, \Delta z_{\mathrm{r}}]^{\mathrm{T}}$. {Step 4.}
22: end while
23: Stay in current position denoted by $p_{\mathrm{fr}} = [x_{\mathrm{fr}}, y_{\mathrm{fr}}, z_{\mathrm{fr}}]^{\mathrm{T}}$ in $K_{\mathrm{r}}$. {Step 8.}
```

Algorithm 2 Inserting the SFP module.

```
Require: The end position of Algorithm 1: $p_{\mathrm{fr}}$.
1: while $z_{\mathrm{cr}}(t) < d_{\mathrm{itr}}$ do
2:   if $|F_{\mathrm{zs}}| < F_{\mathrm{tzs}}$ then
3:     Move towards the DU to $p_{\mathrm{cr}}(t) + [0, 0, \Delta z_{\mathrm{r}}]^{\mathrm{T}}$. {Step 1.}
4:     if $|F_{\mathrm{ys}}| \geq F_{\mathrm{tys}}$ then
5:       if $F_{\mathrm{ys}} \geq 0$ then
6:         Move out of plane to $p_{\mathrm{cr}}(t) + [0, \Delta y_{\mathrm{r}}, 0]^{\mathrm{T}}$. {Step 2.1}
7:       else {$F_{\mathrm{ys}} < 0$}
8:         Move out of plane to $p_{\mathrm{cr}}(t) - [0, \Delta y_{\mathrm{r}}, 0]^{\mathrm{T}}$. {Step 2.2}
9:       end if
10:    else if $|F_{\mathrm{xs}}| \geq F_{\mathrm{txs}}$ then
11:      if $F_{\mathrm{xs}} \geq 0$ then
12:        Move right to $p_{\mathrm{cr}}(t) + [\Delta x_{\mathrm{r}}, 0, 0]^{\mathrm{T}}$. {Step 3.1}
13:      else {$F_{\mathrm{xs}} < 0$}
14:        Move left to $p_{\mathrm{cr}}(t) - [\Delta x_{\mathrm{r}}, 0, 0]^{\mathrm{T}}$. {Step 3.2}
15:      end if
16:    end if
17:  else {$|F_{\mathrm{zs}}| \leq F_{\mathrm{tzs}}$}
18:    break {Step 4. (Safety stop.)}
19:  end if
20: end while
```

Chapter 5

Verification and Validation

This chapter describes what kind of tests that were performed to verify and validate the desired functionality of the different subsystems.

5.1 Software

ARMS [40] uses the command line tool *tox* to run its tests for the purpose of:

- Making sure that the file *setup.py* is configured properly so that the final distribution can be successfully installed under the environment of Python 3.6.
- Making sure that *arms* runs properly by calling the unit test framework *pytest*, which takes the test cases in the folder *tests/* in the root directory into account.

In addition, the plugin *pytest-cov* is installed to calculate the test coverage, which describe the degree to which the provided tests cover the source code.

In general, tests can be divided into *unit tests* and *functional tests*. The former checks the functionality of an individual unit such as a method, whereas the latter proves the functionality of the system, (e.g. the whole repair process). [41].

In contrast to functional tests, which were conducted by doing experiments (see section 6.3), unit tests were written for almost all introduced classes and functions including the RAPID code. The respective requirements and tests can be found in the github repository.

When writing unit tests, the problem is often how many tests to write and what to test to proof the correct functioning of the test object. The software engineer Sandi Metz recommends to only test incoming queries and commands as well as outgoing commands [42], see Fig. 5.1. This recommendation is based on the observation that all units are properly testes if the test coverage is close to 100%, since an outgoing query of a function is an incoming query for the function it calls, and so on.

Moreover, each unit was tested by mocking its calls with the help of the library *mock*. This ensures that the individual test does not depend on the correct functioning of the functions/classes it might call. Another advantage is that the performance of the test suite is increased dramatically.

5.2 IRC5

The verification and validation on the robotic arm was done in stages according to when reliant systems were finished. The first tests performed were basic movements tests with manual input from the flexpendant, an integrated hand controller with a joystick and touchscreen for input. The tests were aimed at getting a feeling for the

H

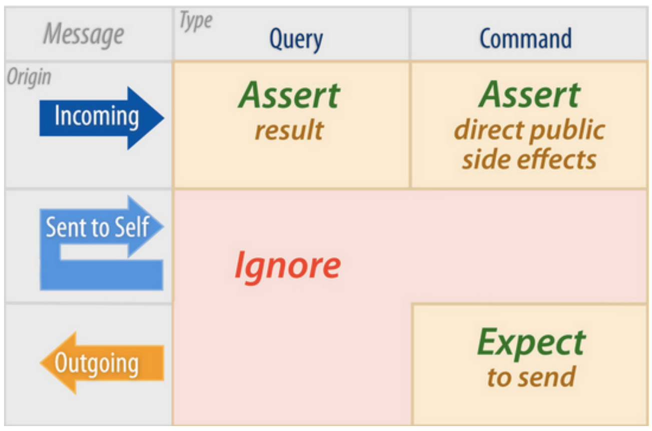


FIGURE 5.1: What should unit tests cover? The picture was taken from [42].

arms capabilities. After manually manipulating the robot with the joystick, a short program was created on the flexpendant where the robot was instructed to move to a recorded positions. Next test was to connect to the robotic arm through socket communication. For this a raspberry pi was connected with an Ethernet cable and when the connect command was sent the IRC5 responded with a connected message.
From this point different coordinate systems and movement commands on the IRC5 were tested and evaluated against our needs. We ended up using the work object coordinate system because it's a fixed system and linear movement on the arm. By using this coordinate system and linear movement on the robot the robot was able to move linearly relative the rack.
Final tests were then made with the gripper attached, where the robot was moved manually to accomplish a repair cycle. Based on the position gained from this process the necessary arm movements was then automated for the repair cycle.

5.3 Gripper

The verification and validation for the gripper was done in multiple steps. At first, the mechanism in itself was tested, after which the various programming solutions were individually tested and successively integrated.

Mechanism

The gripper mechanism was validated using the CAD-program Solid Edge, which identified the dynamics of the structure. It showed that the gripper can be opened and closed by spinning the motor in either clockwise or counterclockwise direction respectively.
Attempts at analytically determining the resulting closing force were thwarted by linearly inseparable angle configurations, and thus a numerical approach using leverage was developed. This model compared the movement between link 1 and the top of link 3, where the difference was interpreted as the leverage for the contraption which in turn gave the resulting force. This model required the input force generated by the motor, and simulations with varying input forces were conducted. The input force from the motor is dependent on the friction coefficient between the motor axle and the nut, and a worst case scenario was assumed in order to properly

verify the gripping force.
The self-locking mechanism proved itself sufficient when conducting empirical tests such as applying a prying force greater than what the pincers are required to experience while inserting a cable or SFP (more than 24 N).
Regarding the weight of the end-effector, this was given by assigning the appropriate materials in the CAD model. As some parts were 3D printed, the weight of the part were assumed to be 100% to ensure that the total weight of the end-effector does not exceed 3 kg.

Integration

Regarding the implemented functionality of the gripper, test were generally performed according to a set methodology. At first, the produced software was tested to run on its own to ensure that there were no compilation errors or logical errors. This was followed up by introducing the software to the designated hardware to ensure that the corresponding actions were indeed executed as desired. For example measuring that a certain pin on the micro-controller would go high when instructed to. Once that step was verified, the respective hardware was connected to ensure that the physical system exhibited the intended behaviour.
The hardware testing followed the trajectory of modelling it in a suitable program, e.g. EAGLE for PCBs and CAD for mechanical parts, followed by testing using a decoupled approach in the sense of applying the desired voltage and current using a powercube.
Once a module had been verified in this way, it was integrated with present functionality to ensure symbiosis, especially with regards to functional degradation when implementing concurrent executables.
This way, the timer driven readings of the pressure sensor was matched to the motor speed to ensure that they together provide a gripping force within a determined percentage of the specified reference value.

Functionality

For the final product, a set of functional tests were devised and run. As the closing and opening algorithms contain two control parameters, namely number of pulses and resulting gripping force, a total of four scenarios were accounted for.
Case 1 involved sending an appropriate amount of pulses with the known resulting gripping force to ensure that the main use case is satisfied.
Case 2 was designed as sending far too many pulses for the given reference pressure to ensure that the gripper stops at the set pressure as opposed to running all the pulses and give the correct error message.
Case 3 included sending a reference pressure that was above the built-in threshold to ensure that the emergency stop kicks in and prevents further movement.
Case 4 involved sending too few pulses for the reference pressure to ensure that the gripper logic can self-adjust slightly to achieve the desired gripping force.

Cable Unlocking Mechanism

To be able to repeatedly toggle the locking pin of the cable before pulling it out, the required force of pushing it down had to be measured. This was done by using a dynamometer connected to the pin in a resting state and then slowly pulling them

apart.
The strength of the unlocking mechanism was initially estimated by describing the force transmission mathematically with parameter values taken from the solenoid's datasheet and the CAD-simulation of the solenoid mechanism. After that it was tested if it was sufficient by having the cable (inserted into a SFP) held by the claw and activating the solenoid. If the SFP was released the requirement was considered satisfied. Since the first test failed, the solution had to be revised. This was either to be done by optimizing the leverage transmission of the system or by increasing the force of the solenoid. The latter was chosen. Since the force output of the solenoid is described by $F = BIL$, where the magnetic field B and the length L are already set, increasing the current to the solenoid was testing by increasing the voltage. After the revision and re-conducting the test, the pushing power of the solenoid mechanism proved sufficient.

5.4 6-axis Force Sensor

As the sensor class only was an extension of already existing functionalities the verification process was quite simple and quick. To determine if the sensor software was functioning properly, the functionalities were tested continuously as they were developed.
Firstly, the C code was compiled and tested by printing the forces and torques to make sure that the connection and program worked. Then the sensor functionality was verified by pushing and pulling the sensor in different ways to make sure that the forces and torques aligned with their respective axes.
To verify the Python side of the sensor code, the same tests were performed continuously during both SWIG interfacing to ensure that it was done correctly and during Python code development to verify all of the interfaced functions. The rotation of the forces and torques were verified by once again pushing an pulling the sensor to induce the different values and ensure their proper alignment. The pinging thread and the zeroing of the values were verified with a simple test program that read the values every three seconds and made sure they were around zero when the sensor was untouched.

5.5 Computer Vision

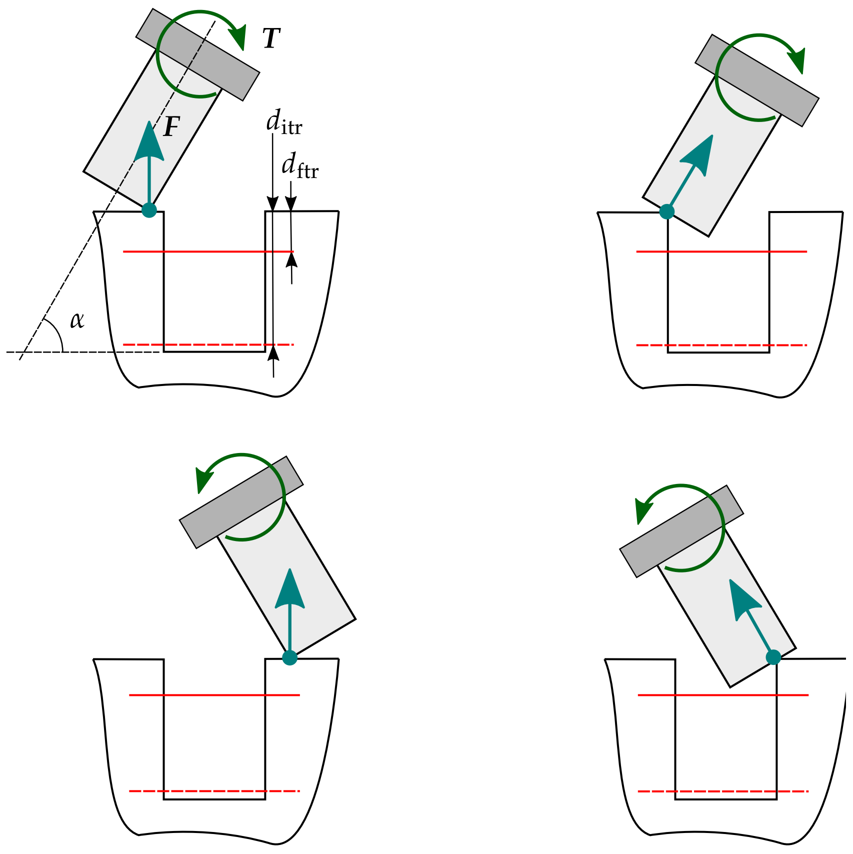
Verification of the Line Detection algorithm was done by taking a picture of the SFP at the same distance as the with a similar light source (same intensity and spread) as the LED flash light circuit. This was tested from multiple light angles and slight variations in camera position and angle to ensure that the algorithm worked for a multitude of conditions. Every picture taken during testing was also saved for later use.
Due to time limitations the full implementation of the Computer Vision was never fully achieved, thus there was never any verification for the actual movement of the robot that was derived from the picture. The derived movements served more as a confirmation that the robot arms was within the safe margins before cable insertion.

5.6 Control Algorithm

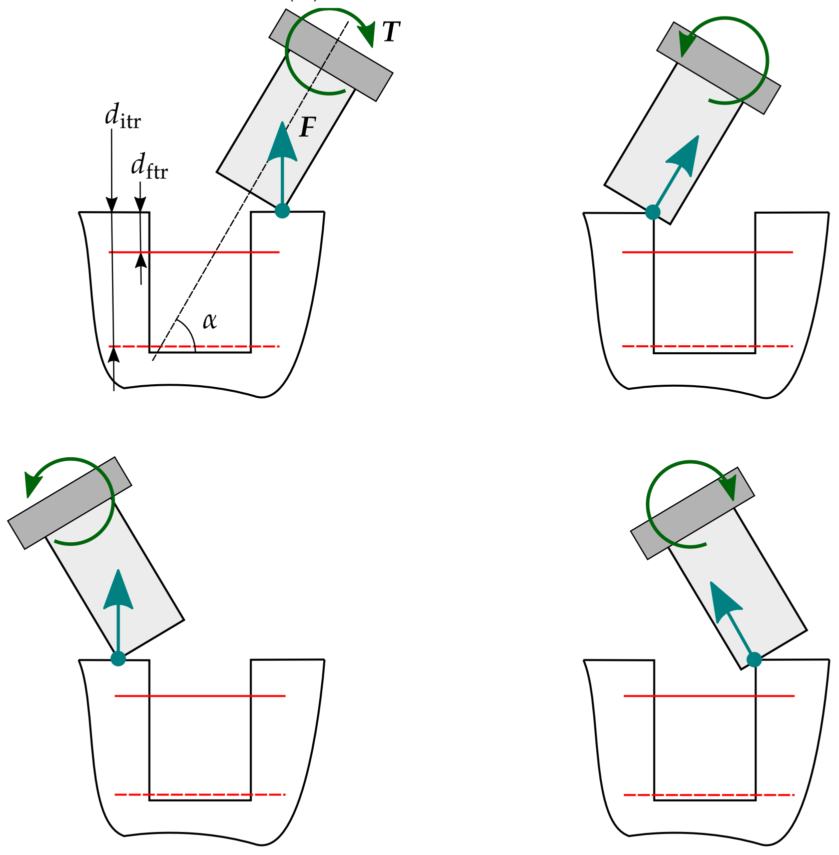
Testing the control algorithm was done in three separate steps, where the problem at hand was divided into various cases as specified in 5.2.
At first, the SFP was positioned in such a way that only one edge would have contact with the edge of the DU to see what, if any, cross-talk was present between the axes on the 6-axis force-torque sensor. The readings were evaluated and formulated into the control algorithm as thresholds to ensure correct movement given a certain case. Regarding the superimposed cases, such as the SFP coming into contact with either corner of the DU, only one such case was evaluated where a positive offset in both x and y was presented and successfully resolved by the algorithm.

The setup for the hardware experiment consists of jogging the arm into the correct position of where an SFP is inserted into the correct port, grab it with the gripper and then save the coordinates. To produce a reproducible error, an offset is introduced to the position in rack coordinates. Forces, torques and positions are recorded at every increment for tracing of the algorithms decisions.

The hardware experiment was conducted using the full setup without the camera module, since it produces a slightly different estimate each time and therefore non-reproducible outcomes. The thresholds were set to $d_{\mathrm{ftr}} = -41.8565\,\mathrm{mm}$, $F_{\mathrm{tzs}} = 10\,\mathrm{N}$, $T_{\mathrm{txs}} = 0.07\,\mathrm{Nm}$, $T_{\mathrm{tys}} = 0.05\,\mathrm{Nm}$, $\Delta x_{\mathrm{r}} = \Delta y_{\mathrm{r}} = 0.5\,\mathrm{mm}$, $\Delta z_{\mathrm{r}} = 0.1\,\mathrm{mm}$, $d_{\mathrm{sr}} = 1.5\,\mathrm{mm}$ for Alg. 1; $d_{\mathrm{itr}} = 5.053\,78\,\mathrm{mm}$, $F_{\mathrm{tzs}} = 30\,\mathrm{N}$, $F_{\mathrm{txs}} = F_{\mathrm{tys}} = 5\,\mathrm{N}$, $\Delta x_{\mathrm{r}} = \Delta y_{\mathrm{r}} = 0.1\,\mathrm{mm}$, $\Delta z_{\mathrm{r}} = 0.5\,\mathrm{mm}$ for Alg. 2. The estimated position $p_{\mathrm{er}} = (72.5, 227.5, -48)$ in [mm] was found by manually trying to insert the SFP module into the slot and then by adding some quite large offset values to the found optimal position. The result for this experiment is shown in Fig. 6.1 and 6.2.



(A) Handleable cases.



(B) Unhandleable cases.

FIGURE 5.2: Possible contacts and corresponding force vectors when trying to find the hole for $\gamma_{\text{xms}} = \gamma_{\text{yms}} = \gamma_{\text{zms}} = 0°$.

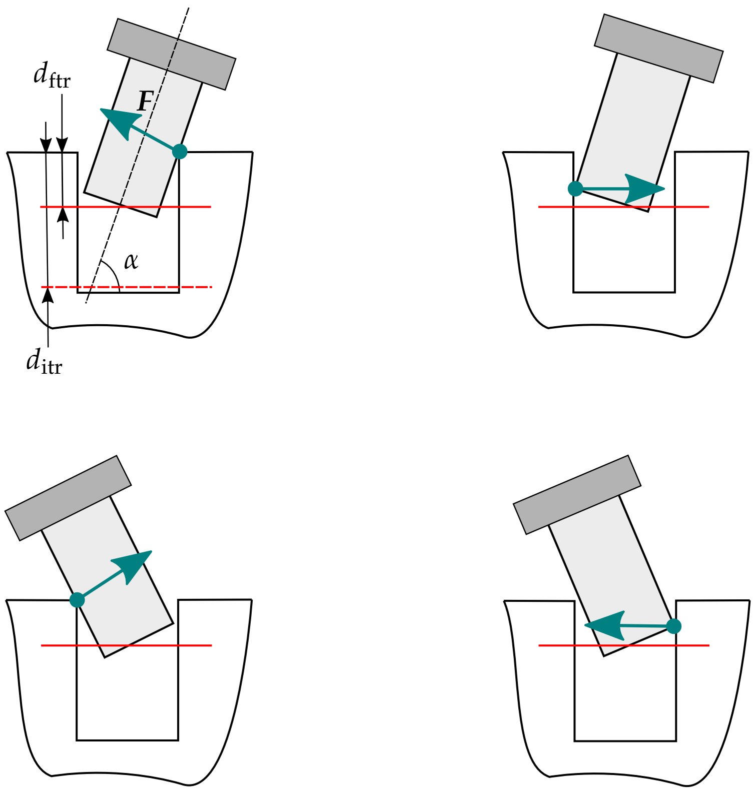


FIGURE 5.3: Possible contacts and corresponding force vectors when inserting for $\gamma_{\mathrm{xms}} = \gamma_{\mathrm{yms}} = \gamma_{\mathrm{zms}} = 0°$.

Chapter 6

Results

The results of the project are discussed in this section. Beginning with stating which of the stakeholder requirements presented in section 1.2 were fulfilled. Continuing with a brief statement on why some of the requirements were not fulfilled.

6.1 Stakeholder requirements

The project were able to fulfill the following stakeholder requirements where ✓ notes fulfilled requirement, x for not fulfilled and – where not verified.

No.	Requirement	Fulfilled
1	The repair cycle shall be fully autonomous	x
2	The robot shall be able to replace optical adapters of different types at certain locations	–
3	The robot shall be carriable/mountable by a single operator	–
4	The robot should not be in the way while in standby mode	✓
5	The robot shall report what is being done	✓
6	The robot shall not damage any parts of the station (including cables and connectors) while working	✓
7	The robot may connect to an operator for monitoring purposes	✓
8	The robot shall have access to spare parts	✓
9	The total cost of ownership shall be less than that of a human worker	–

TABLE 6.1: Achieved stakeholder requirements

For the first requirement the process was fully automated except for one step. The robot was unable to remove the old SFP from the DU but were otherwise capable of a full repair cycle. There was a solution for removal of the old SFP but due to time restriction it was never implemented.
The second requirement was not verified with other optical adapters but should be fulfilled provided it use the same cable.
We were unable to verify requirement No. 3 due to receiving a fully installed robot and thus never had the opportunity to study the installation process. From research done on the subject it is doubtful if this requirement would have been met.
The last requirement depends on the time frame, country and location of the base station. These variables coupled with the difficulty of getting a price for the robotic arm and the price of a technician makes it hard to calculate if this requirement is fulfilled.

6.2 Cable Replacement Process

The results of the replacement are tabulated as figures in Appendix E. The following steps were able to be achieved using the current prototype:

1. Step 1: Go to initial position.
2. Step 2: Move to a position in front of the SFP at a safe distance.
3. Step 3: Move to gripping position near the SFP.
4. Step 4: Grip the cable. This step utilizes the pressure sensor and measures pressure values gradually to command the gripper to stop at a correct interval.
5. Step 5: Activate the solenoid to push the lock and grip the cable out.
6. Step 6: Move to a safe point in space.
7. Step 7: Move besides the rack and take a photo for the computer vision algorithm for hole detection.
8. Step 8: Move to the helper station.
9. Step 9: Move into the new SFP and connect the cable.
10. Step 10: Move back.
11. Step 11: Move in front of the port.
12. Step 12: Find the hole with algorithm 1.
13. Step 13: Insert SFP in the hole with algorithm 2.
14. Step 14: Release the gripper and move back to the initial position.

The current procedure was demonstrated to the stakeholder and was accepted as a proof-of-concept for further research and development on the idea of creating an autonomous repair robot for the Ericsson base station units.

6.3 Hole Finding and Insertion Algorithm Experiments

The result for the *Hole Finding Algorithm*, Alg. 1, is shown in Fig. 6.1. At record 78, 110 and 152 the force $F_{\mathrm{zs}} \geq F_{\mathrm{tzs}}$ and the torque $|T_{\mathrm{xs}}| \geq T_{\mathrm{txs}}$ and $|T_{\mathrm{ys}}| \ngeq T_{\mathrm{tys}}$, and therefore a correction in y_{r} by $\Delta y_{\mathrm{r}} = 0.5\,\mathrm{mm}$ is made. Note that before this correction is made, the robotic arm moves back in z_{r} by $d_{\mathrm{sr}} = 1.5\,\mathrm{mm}$ to ensure that the SFP module does not get stuck at the DU. Similar to these records, at record 196 and 254 a correction in x_{r} by $\Delta x_{\mathrm{r}} = 0.5\,\mathrm{mm}$ is made, since $F_{\mathrm{zs}} \geq F_{\mathrm{tzs}}$ and the torque $|T_{\mathrm{xs}}| \ngeq T_{\mathrm{txs}}$ and $|T_{\mathrm{ys}}| \geq T_{\mathrm{tys}}$ at these records. Beyond the last mentioned record the torque values stay in their bonds and the robotic arm finally passes the threshold d_{ftr}, which indicates that the slot is found.

The result for the *inserting algorithm*, Alg. 2, is shown in Fig. 6.2. Almost at all records ≥ 476 either $|F_{\mathrm{ys}}| \geq F_{\mathrm{tys}}$ or $|F_{\mathrm{xs}}| \geq F_{\mathrm{txs}}$, and therefore a lot of corrections in y_{r} or x_{r} respectively are made. This mainly indicates that the correction $\Delta x_{\mathrm{r}} = \Delta y_{\mathrm{r}} = 0.1\,\mathrm{mm}$ is set too low, so that with each correction in position there is a high chance that the forces still exceed their bounds. Moreover, it can be observed that way more corrections in x_r than in y_r are made, since x_r falls by $\Delta = 1.2\,\mathrm{mm}$ while y_r falls by $\Delta = 0.3\,\mathrm{mm}$ after record 476. During the inserting operation a spike of F_zs is expected with a maximum value of around 22 N and can be seen at record 510. This is due to the reasons that the SFP is equipped with a spring mechanism to ensure a secure hold in the hole. Finally, at record 515 the SFP is fully inserted, which is indicated by $z_\mathrm{r} \geq d_\mathrm{itr}$. The point d_itr was found after manually inserting the SFP in the slot and measuring its position in K_r.

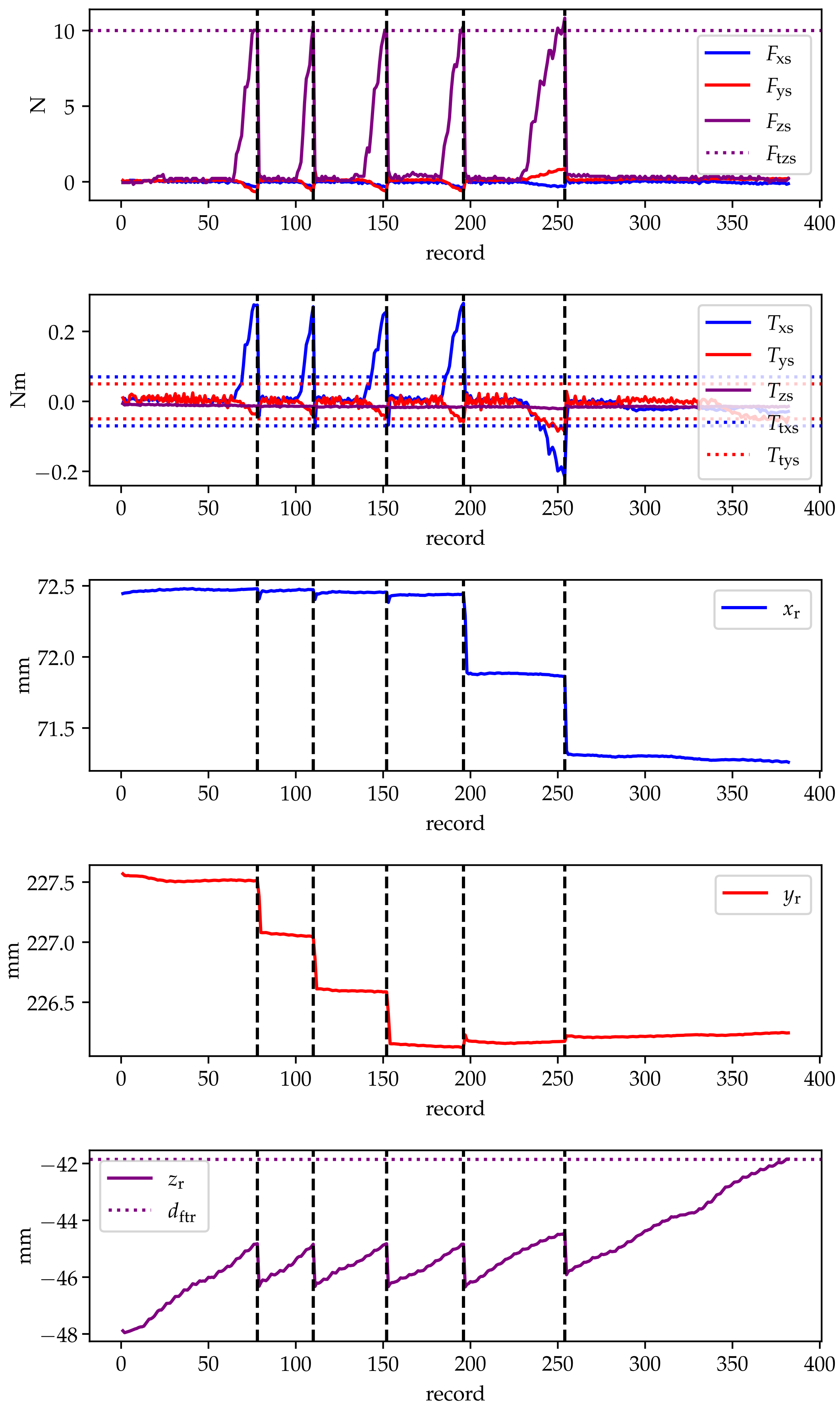


FIGURE 6.1: Experiment with the finding hole algorithm, Alg. 1. The vertical black dashed lines mark records where a correction in position (x_r, y_r) was made.

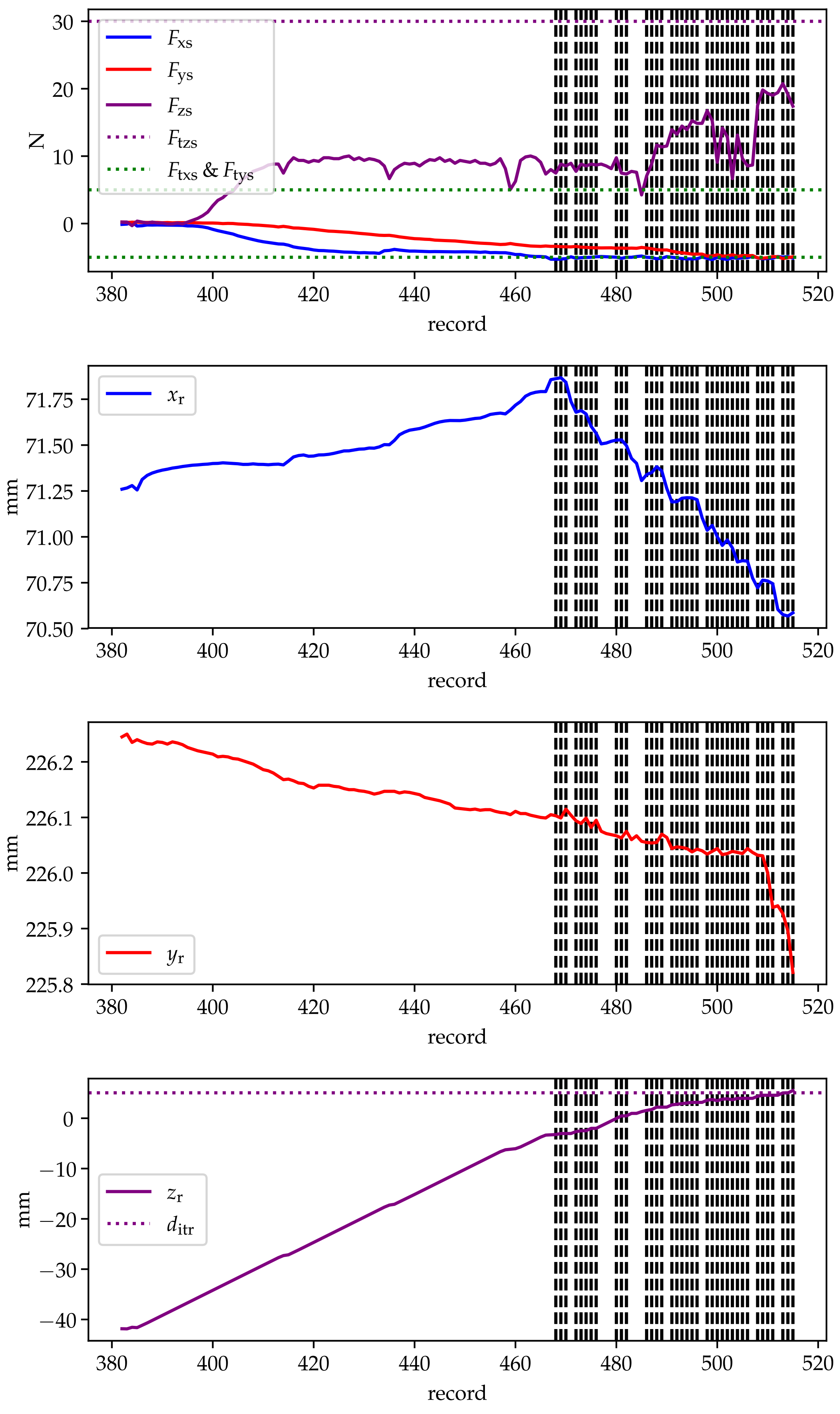


FIGURE 6.2: Experiment with the inserting algorithm, Alg. 2. The vertical black dashed lines mark records where a correction in position (x_r, y_r) was made.

Chapter 7

Discussions and Conclusions

This chapter presents the conclusions and starts a discussion about the prospects of work an achieved results at each level of system design and implementation.

7.1 IRC5

Currently the robot runs the risk of encountering singularities and out of reach errors. These problems are potentially fatal errors since they need manual input to be resolved. In this project they were circumvented by programming the desired positions in advance but a future improvement could be to create a function that identifies singularities and out of reach errors before they occur and maps alternate paths and robot configurations.

7.2 Gripper

Regarding the construction of the gripper, a longer nut could have been used to offset the experienced shear forces when the gripping occurs at a position not completely perpendicular and in line with the center axis of the gripper. Currently there are problems with having the gripper work in a vertical stance, i.e. having a top and bottom pincer instead of right and left pincer. The middle link that is directly moved by the nut is inclined to tilt slightly due to gravity. This sometimes causes the nut to jam, thus preventing the gripping function. An increased width on that nut would prevent this 4.3.

Although enough force was exerted by the gripper for this application, lubrication can been applied at various points, especially between the motor axle and the driving nut, to reduce friction which in turn allows for higher force output.
Improvements could be done to the gripper construction with regards to fitting tolerances and cutting unnecessary material to make the gripper smaller and lighter.

Another thing that could be substantially improved is the casing and placement of the micro controllers and the PCB:s on the gripper. They are currently not protected from dirt and collisions.

The gripping force is only implemented via the raw ADC values generated by the pressure sensor. Possible improvements is implementing a secondary pressure sensor on the other pincer and taking an average value which allows for redundancy in the system as well as future protection against wrongful gripping. As the pressure sensor behaves non-linearly, it may be good to map the values given to actual force enacted to further gain understanding of the system as a whole.

Considering the self-tuning aspects of the gripper, it was determined that it worked well enough for this project as it ensured a final gripping force in the desired range, however further implementations could be made to safeguard against the gripping force exceeding the limit, possibly by reducing the motor speed.

7.3 6-axis force-torque sensor

Improvements could be made regarding how the measurements from the 6-axis force-torque sensor are made. Firstly with regards to how aligned the coordinate system of the sensor matches up with the coordinate system used in the project, and secondly with reducing jitter from the readings by taking multiple values and averaging them out instead of instantaneous values, especially for the function where re-calibration occurs.

7.4 Computer Vision

Due to the not having a specific time requirement the CV algorithm was not properly optimized. The speed of the algorithm could have improved if the input image had a lower resolution while ensuring that the important features of the SFP are still readily visible. Due to the way the PiCamera works the higher resolution does not necessarily mean more detail.

The accuracy could have been improved in various ways. Firstly it could be improved by attaching a lens to the camera to zoom in on the focused area make sure that it is as detailed as possible, eliminating a lot of the need for a high resolution. A high resolution with a lens would probably be the absolutely best option in terms of accuracy though. It should be noted however that a lens may result in slight distortions of the image which would make the Line Detection much harder to implement.

The algorithm could also be expanded to, once vertical and horizontal alignment was assured, run the Line Filtration again but with no angular margin. This would increase accuracy by a lot since there can never be a slightly tilted line that is misidentified as the furthest edge line, which in the current iteration causes errors of about 0.3 mm.

7.5 Control Algorithm

Conclusive tests were only made for the four disjoint cases and one of the interconnected cases, where an offset was introduced in positive x and y simultaneously. Given the assumptions made when developing the control algorithm, the results ought to be symmetrical and thus applicable throughout the defined application space, however this has not been independently confirmed. It was empirically verified that the implemented algorithm successfully fulfills its task.

The advantages of the proposed algorithms are the easy implementation and the fact that they can handle most cases sketched in Fig. 5.2 and 5.3. However, since only the position is changed but not the angles of the joints of the robotic arm, the latter requires that the estimated position by the camera matches quite well the actual position of the slot and that the angle $\alpha \approx 90°$ and $\gamma_{\text{xms}}, \gamma_{\text{yms}}, \gamma_{\text{zms}} \approx 0°$. These requirements can be achieved with a precise robotic arm like the ABB IRB 120 and a gripping mechanism that prevents the cable from bending too much so that the SFP module stays aligned with the gripper when gripping it. Overall, the main disadvantage are the potential high cost of the complete system that can fulfill the mentioned requirements and the observation that the threshold values have to be find for each installation setting, which can be time consuming.

Chapter 8

Future Work

This chapter processes the leads and needs to solve or better solve the problem. The chapter will go through areas such as system hardware, software and control and discuss which parts that can be improved and which parts that are currently missing to satisfy all initial requirements. This while also highlighting areas that has potential to grow towards even greater functionalities.

8.1 Unfinished Functionalities

This section discusses what remaining functionalities are left to satisfy the initial requirements. Many of the these were delivered. However, a few pieces of the replacement process are still missing.

Hardware and Mechanics

One missing functionality still is the removal of the old SFP from the SFP port in the DU. The hook-like piece on one of the pincers is yet to be integrated, see 4.3. The length and material of the piece is still not decided due to not having time to test it together with the claw movements and forces.

Software and Control

Algorithms and parameters regarding the SFP removal are missing to complete the whole replacement cycle. Both in the Raspberry Pi's and the IRC5's code. In order to remove the SFP, the ABB arm's wrist holding the gripper needs to be able to tilt to get close enough to it's hinge mechanism. The code for this movement with appropriate safety precautions is still not implemented.

In order to speed up the process, especially with regards to the hole finding algorithm and insertion algorithms, it would be beneficial to instruct the robotic arm to move to the desired final location and measure the experienced forces and torques and sending an interrupt request to the robotic arm instead of moving a short distance followed by measurements which may or may not result in adjustments.

Computer Vision

The Computer Vision algorithm, in it's current form, only returns the estimated movements required to correct any errors that might occur. It is not verified to be able to use these movements to properly control the robot and adjust for errors correctly, and has no code implemented to do this in the cycle. It also doesn't share the robot arm's coordinate system and would need an axis rotation similar to the sensor.

8.2 Expanding the solution

A few ideas on how this solution can be expanded will be discussed in this section.

Possible Reduction of Redundant Functionalities

Since total cost of a commercial application of the system is of interest, one should explore the possibility of using an arm with fewer degrees of freedom and/or a force sensor with fewer sensory axes, thus enabling less costly components.

Movable base

One possibility to expand on this solution is to implement the robotic arm on a movable base so it may serve multiple cabinets within a close proximity. Care must be taken to ensure that the position of the robotic arm remains sturdy when in operation, and its positioning needs to be fairly precise and repeatable.

Modular end effector

To further enhance the versatility of this solution, it may be beneficial to explore the possibility of an autonomous end effector replacement. This way, multiple end effectors, each designed for a particular repair scenario, could be stored at a convenient location and autonomously be acquired by the robotic arm as necessary.

Computer Vision

Several large parts of potential future improvements lies within Computer Vision. This project only implements CV for the cable insertion but with more development it could be expanded to work with all of the critically controlled phases of the procedure. Both cable extraction and SFP insertion, for example, could have used Computer Vision to verify and correct positions, eliminating the need for hard positions and force/torque reliant control. This would require extensive amounts of work to implement however, specifically to attune existing CV methods to be able to properly identify an SFP in a DU which has other DUs right next to it, which the helper station did not.

Another way to solve this issue could be to train a deep learning classifier with localization. While probably not as accurate as regular Computer Vision, deep learning would be very applicable in this specific scenario and could, if execution time is not an issue, be combined with Computer vision to narrow down which area to further scan for lines.

Implementing these functionalities would eliminate the need for a 6-axis Force Sensor and would lead to the robot only requiring a sensor that senses forces in the Z direction, which would lower costs substantially.

One early wish that was scrapped was for the robot to be able to also change the optical cables. This is a task that definitely cannot be solved without some form of Computer Vision and/or Deep Learning due to the amount of cabling that needs to be managed by the robot. There is no current concept of how to implement this however so this area would need to be explored from the start.

Appendix A

Base Station Components

Components related to the base station are presented below.

A.1 SFP-module

The abbreviation SFP stands for "small form-factor pluggable" and is used for both telecommunication and data communications applications by transmitting information from one place to another. This is done with the help of light pulses that are sent through an optical fiber cable, where the SFP is the transceiver and receiver at each end of the cable [43]. Different SFP work with different wavelengths at an appointed distance which also controls what type of SFP that should be used [44]. The SFP-module can be seen in Figure A.1.

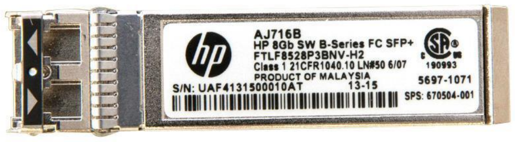


FIGURE A.1: SFP+ model AJ716B created by Hewlett-Packard [45]

The dimensions of the SFP module can vary depending on the type. The SFP module within this scope is a called SFP+ module and have dimensions according to Table A.1.

TABLE A.1: General dimensions of a SFP+ transceiver [44].

Dimension	**Value**	**Unit**
Height	56.5	[mm]
Width	13.4	[mm]
Depth	8.5	[mm]

A.2 Fiber Optic cable

Fiber optic cables are used in telecommunications infrastructure, where it enables transmission of high-speed voice, video, and data traffic in enterprise and service provider networks. Depending on the type of application and the reach to be achieved, various types of fiber may be considered and deployed [46].
An optical fiber is a flexible filament of very clear glass capable of carrying information in the form of light. The two main elements in an optical fiber cable are its core and cladding. The core is made out of silica glass, where the light transmission is held.

The cladding is the layer that completely surrounds the core. The refractive index of the cladding is less than the index of the core, such that when the light in the core strikes the interface of the cladding at a bouncing angle, it gets trapped in the core by total internal reflection. This allows the light to travel in the proper direction down the length of the fiber to its destination [46]. The architecture within an optical fiber cable is shown in Figure A.2.

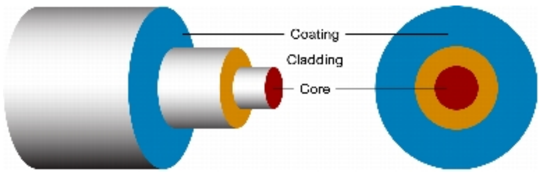


FIGURE A.2: Optical fiber cable architecture [46].

A.3 Digital Unit

A digital unit (DU) is a device that provides switching, traffic management, timing, base-band processing and radio interfacing for the radio base station [47].

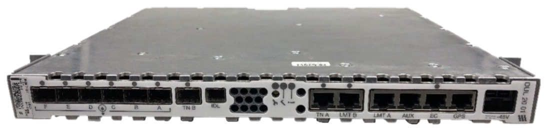


FIGURE A.3: The Ericsson digital unit DUL 20 01[47]

The DU is equipped with 6 radio interfaces labeled A-F in Figure A.3. It is into these slots the SFP adapters are placed. When the SFP converter is faulty the signal between optical and electrical signals fails causing slower or interrupted service [48].

Appendix B

PCB layout

PCB circuits used in the project are presented in this appendix.

B.1 Solenoid circuit

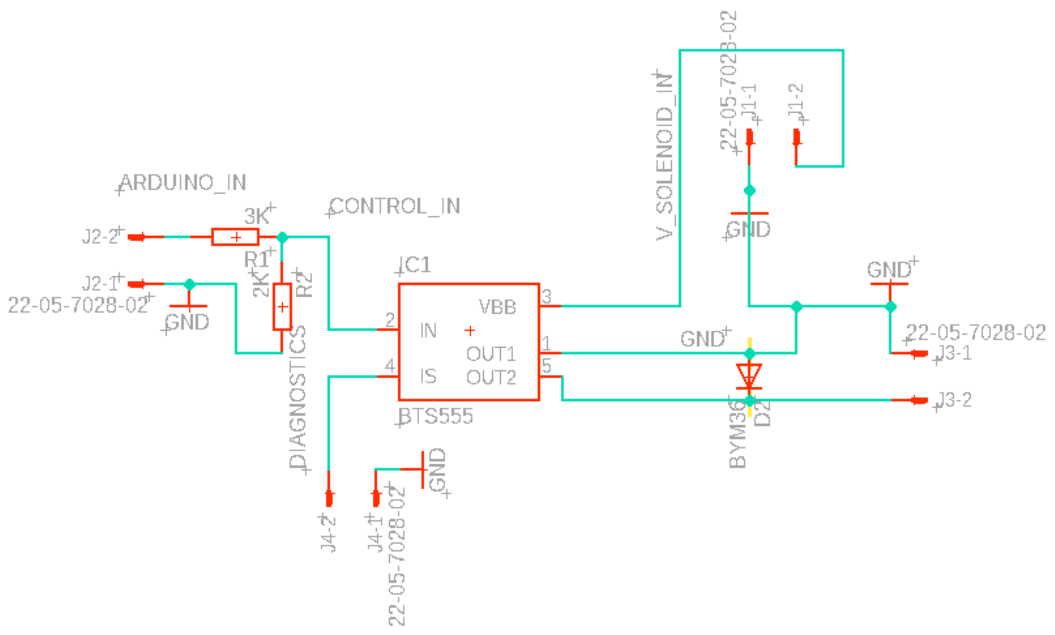


FIGURE B.1: Solenoid circuit diagram.

B.2 Flash circuit

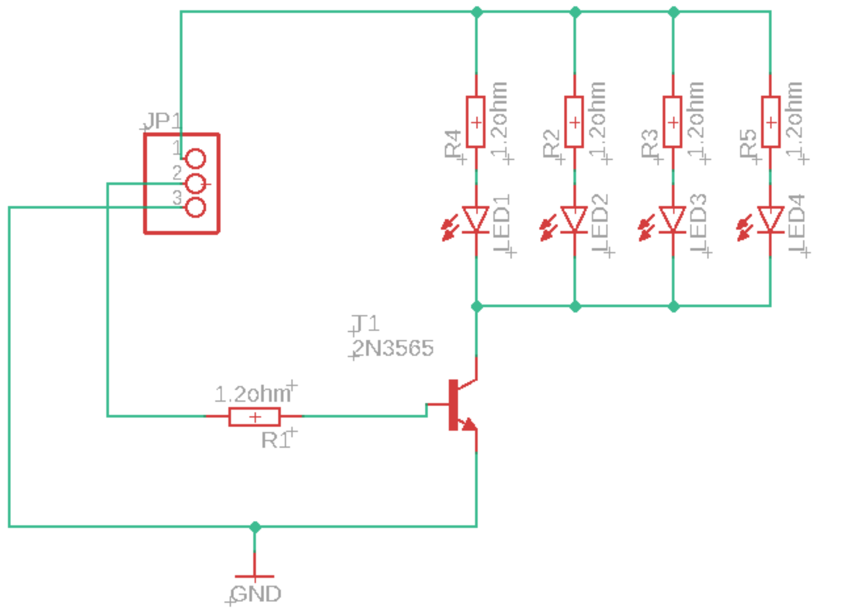


FIGURE B.2: Flash circuit diagram.

B.3 Pressure sensor circuit

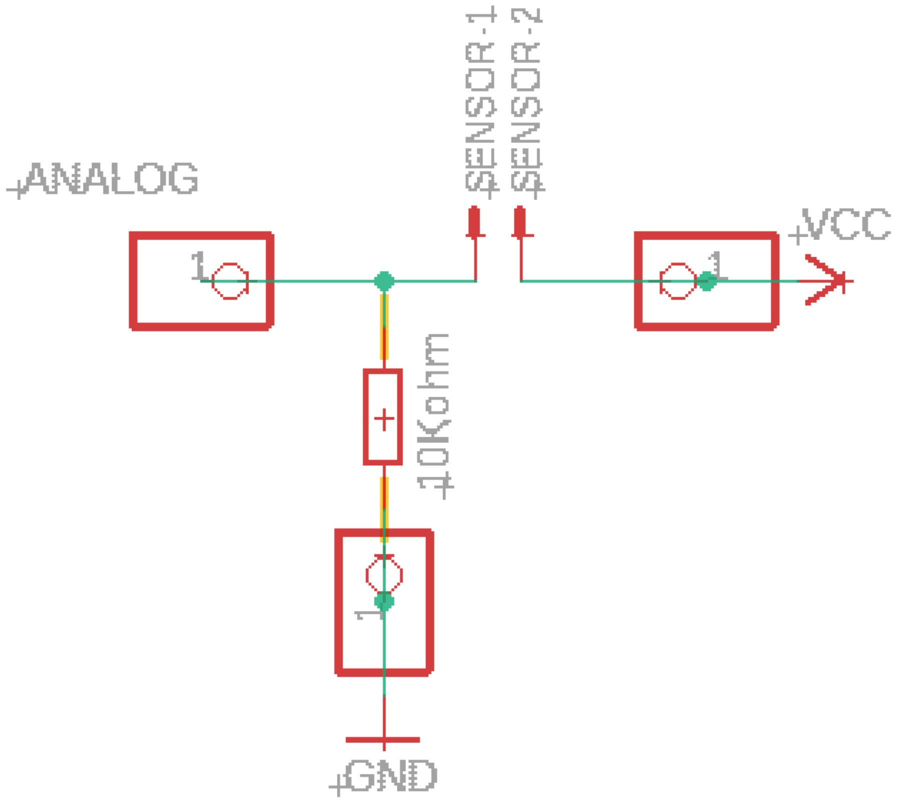


FIGURE B.3: Pressure sensor diagram.

Appendix C

Components

This Appendix will cover information of the components that were used in this project with a motivation to why these components were chosen.

C.1 Actuator

A stepper motor was chosen as the actuator for the gripper mechanism because the control does not need tuning nor encoder, thus making the implementation less time consuming and that the operation results with small errors [49]. The stepper motor that was chosen was "Hybrid Stepper Motor 6540-13" from SONCEBOZ and can be seen in Figure C.1.

FIGURE C.1: Hybrid Stepper Motor 6540-13 from SONCEBOZ.

This stepper motor was chosen since it operates within 40 V, which can be supplied from the DU as well as delivering a holding torque of 130 mNm, which is within the force requirement to replacement the SFP which was presented in the result section.

C.2 Voltage Regulator

Only one power input should be present to secure as simplistic installation of the robot as possible. The voltage input should be equal to the most required voltage in the tool which will then be divided internally as required. A switched variable voltage regulator, LM2596 from Luxorparts as seen in Figure C.2 was chosen to divide the voltage.

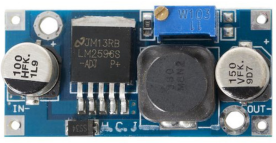


FIGURE C.2: Switched Variable Voltage Regulator, LM2596 from Luxorparts.

This voltage regulator was chosen for the voltage and current range as seen in Table C.1.

Input	3.2 − 30 V
Output	1.5 − 28 V
Current	0 − 2 A

TABLE C.1: Voltage Regulator data.

The switched voltage regulator satisfied all needs in regards of current and voltage to the components utilized on the tool.

C.3 Pressure Sensor

To prevent any damage of the optical cables during the replacement procedure, the pressure in which the cables are exposed to by the pincers must be limited. This was done by mounting a force-sensing resistor behind the touching surface of the pincers. The force sensor that was chosen was I.E.E. Strain Gauge 11.7 mm and can be seen in Figure C.3.

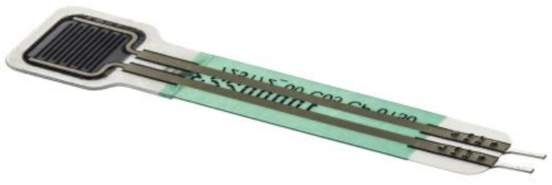


FIGURE C.3: Pressure sensor, I.E.E. Strain Gauge 11.7 mm.

The sensor had a service life greater than 10 million cycles and was cheap relative to competitive options, which resulted in the use of this sensor.

C.4 Bearings

FIGURE C.4: Ballbearing 624 from SKF.

The chosen ball bearings to support the linkage mechanism is SKF 624 single row bearing. This was based on the factors seen in table C.2 and the decision was made according to the methodology in section 4.3.

Basic static load rating C_0	0.29 kN
Inner diameter d	4 mm
Outer diameter D	13 mm
Static safety factor s_0	3

TABLE C.2: Calculation factors and dimensions for SKF 624.

C.5 Solenoid

The solenoid, GM - Miniature solenoid was chosen for the size and capability of delivering a force high enough to unlock the optical cable through the solenoid link mechanism. The solenoid can be seen in Figure C.5.

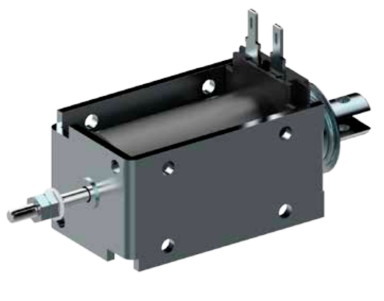


FIGURE C.5: GM - Miniature solenoid [50].

C.6 Arduino

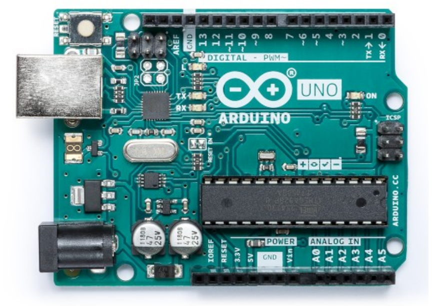


FIGURE C.6: The Arduino UNO open-source microcontroller board, accessed from [10].

Arduino is an open-source software and hardware platform. The hardware is built upon the micro-controller ATmega328p. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 V battery, though it accepts voltages between 6 V and 20 V[10]. The platform was chosen because of its clock peripherals being able to trigger interrupts and its ease to use with external hardware through the Digital and analog pins.

C.7 Raspberry Pi Model 3B

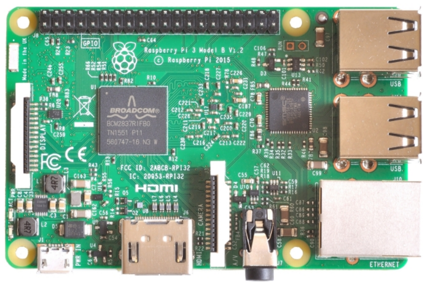


FIGURE C.7: The Raspberry Pi Model 3B single-board computer, accessed from [11].

Raspberry Pi 3 Model B is a single-board computer with a 64-bit quad-core processor, network connection through Ethernet and WiFi and four USB-ports. The Raspberry pi also features 40 GPIO headers which can be used for external electronics. The board hosts a Operating system, Raspbian which is recommended by the Raspberry Pi foundation for normal use of the hardware. The board was chosen to be the main system because of its processing capabilities, the integrated PiCamera connection and the variety of connection abilities[11].

C.8 PiCamera Module 2

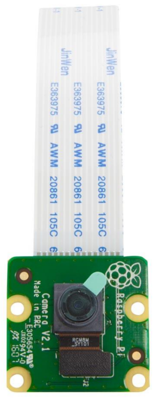


FIGURE C.8: The PiCamera Module.

The PiCamera Module 2 is a camera extension module for the Raspberry Pi that connects to a specific camera connector on the board. It weighs 3g, has a still resolution of 8 Megapixels, can record videos at 1080p 30 fps and can take pictures with as high as 3280 × 2464 pixels resolution. It has a lot of integrated functionalities with the Pi which can be used to preview before taking a picture, setting resolution, setting frame rate and various other functions. The module was chosen because of the small size, high compatibility and great specifications.

C.9 6-axis Force Sensor

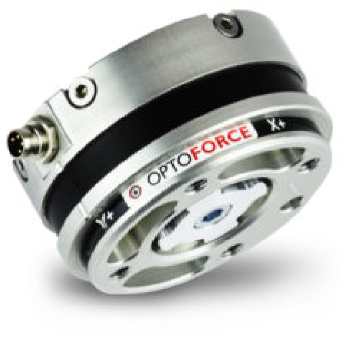


FIGURE C.9: The 6-axis Force Sensor.

The 6-axis Force Sensor, made by OptoForce, measures forces and torques in all three axes. It is connected to and powered from a DAQ device that translates the data to different kinds of communication interfaces. The sensor was chosen due to there being one available for us to borrow, its ability to also measure torques and it being somewhat renowned within the industry.

C.10 Motor driver

The motor driver, X-NUCLEO-IHM14A1 was chosen as driver to control the stepper motor and can be seen in Figure C.10.

FIGURE C.10: Stepper motor driver, X-NUCLEO-IHM14A1 from Nucleo.

This component was chosen due to it being compatible with Arduino UNO which was chosen as the controller. The functions of the driver was provided by the data sheet and could be operated accordingly.

Appendix D

Software

The following sections cover software related topics.

D.1 RobotStudio

A Graphical User Interface (GUI) software tool used for configuration and offline programming of ABB robots with RAPID programming as the basic programming language.

D.2 Socket Communication

Sockets are used to a establish a communication between a server and a client. In the setup in Fig. 4.1b the Raspberry Pi is the client, whereas the Ericsson DU, the IRC5 and the F/T sensor acts as a server. This configuration is chosen like this to be able to easily recover a connection. Therefore, whenever the Raspberry Pi looses the connection to one of the servers, the respective server will automatically be recreated and the Raspberry Pi is able to reconnect with it, and then is able to continue with the repair procedure.

In Fig. D.1 the communication flow between the Raspberry Pi and the IRC5 is shown as an example. First, the IRC5 has to create a socket. This is done with the following RAPID commands [51]:

1. *SocketCreate:* Create an endpoint for communication.
2. *SocketBind:* Bind a socket to the specified server IP-address and port number.
3. *SocketListen:* Start listening for incoming connections and thereby start acting as a server.
4. *SocketAccept:* Accept incoming connection requests. Block until there is a connection request from the client.

After these steps, the Raspberry Pi is able to connect with the IRC5 by sending an *connect* request. As long as a working connection exist, both sockets communicate with each other as described in section 4.2. After the repair process is successfully completed, the Raspbery Pi will disconnect from the IRC5, which will in consequence open a new socket and stay in the state of listening to new connection requests.

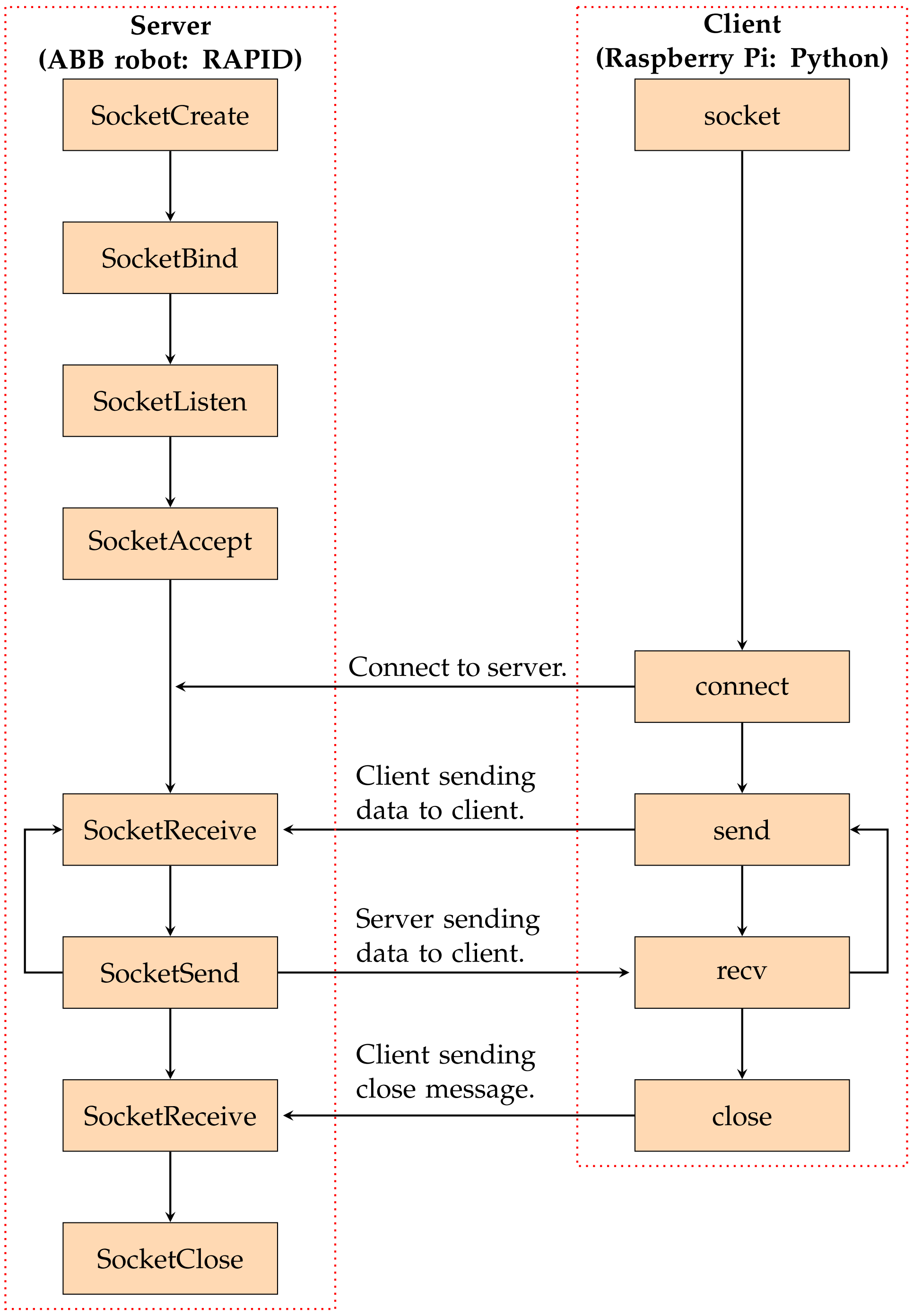


FIGURE D.1: Socket communication between the ABB robot (server) using RAPID [51] and the Raspberry Pi (client) using Python [52]. The Graphic was adapted from [53].

D.3 Serial Communication

Serial communication is used between the Raspberry Pi and the Arduino. The Arduino has a built-in serial library which is taken advantage of, and for the Raspberry Pi the PySerial environment was utilized. Serial communication implies that data is sent bit-by-bit over one connection, and care must be taken to implement an interpreter for when the message is complete.

On the Arduino, an interrupt is generated whenever there is an incoming message, which is then parsed and the corresponding function is called. When the function has completed, a message containing system update parameters is created and returned over the serial connection.
For the Raspberry Pi, a thread is established that periodically polls the serial communication for any incoming messages. When polling the connection, an interlock is activated to prevent any sending of messages to occur at the same time. If a message is received, this is parsed and the appropriate evaluation function is called depending on the message ID. A visual guide for the serial communication is provided in figure D.2.

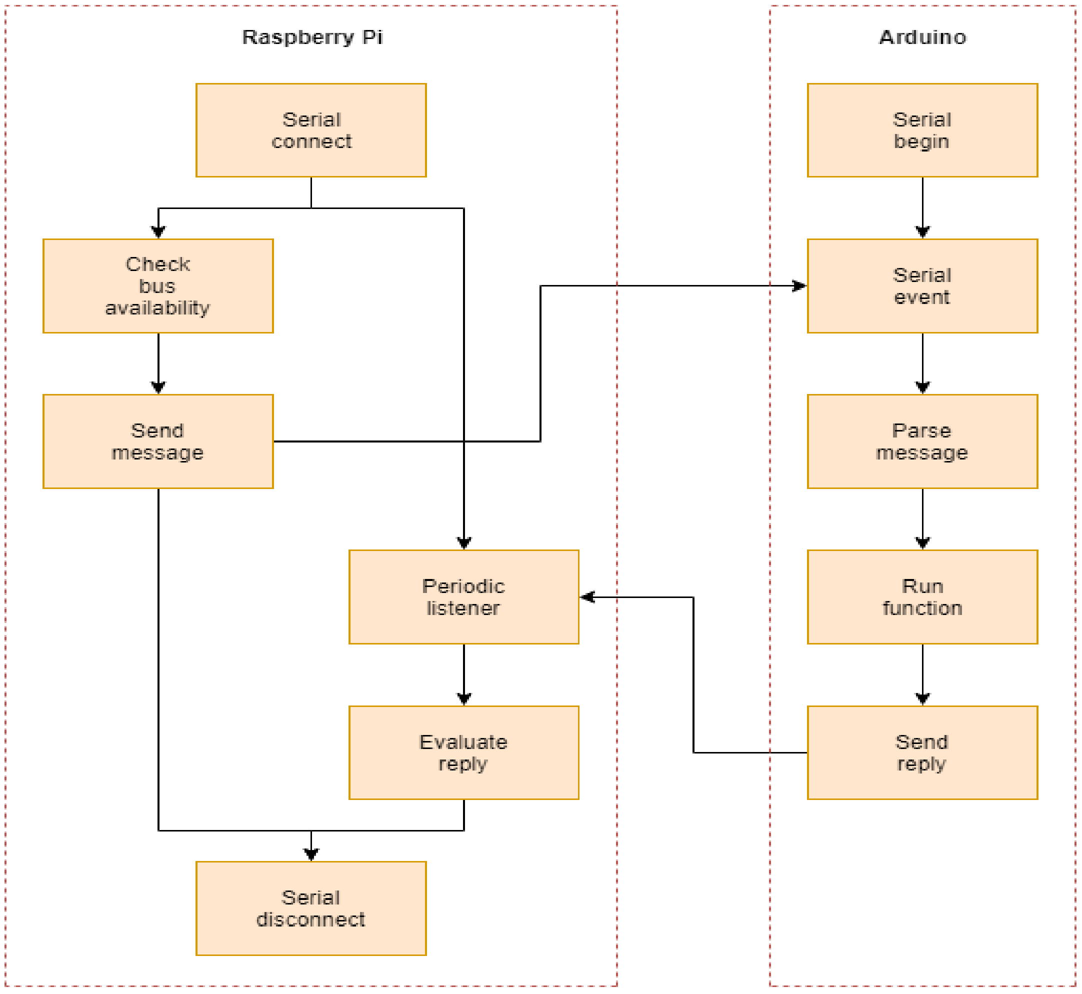
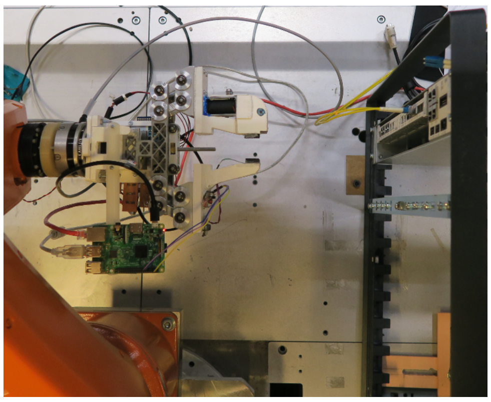


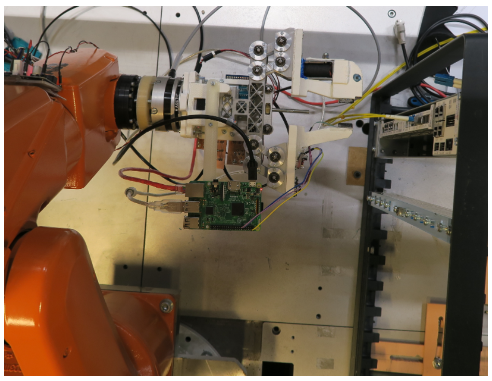
FIGURE D.2: Serial communication between the Raspberry Pi and the Arduino. Graphic created in *draw.io*.

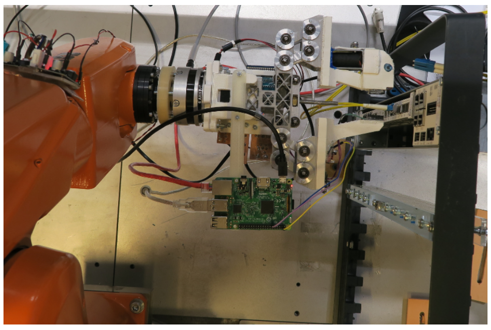
Appendix E

Process

This appendix presents overview of the switching process and achieved results, see Fig. E.1.


(A) Step 1) Start from the initial position.


(B) Step 2) Move in front of the SFP.


(C) Step 3) Move to the SFP in gripping position.

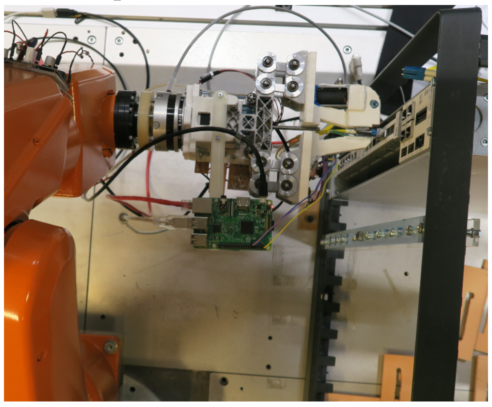
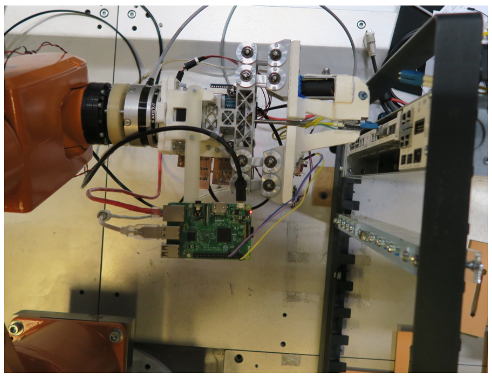
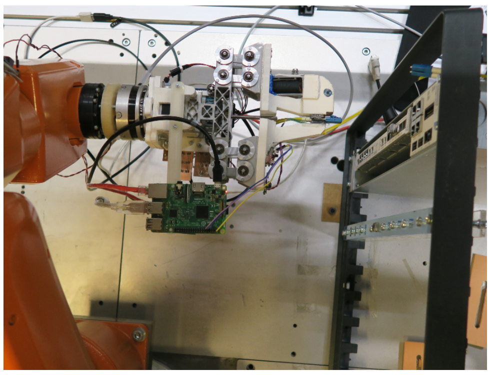
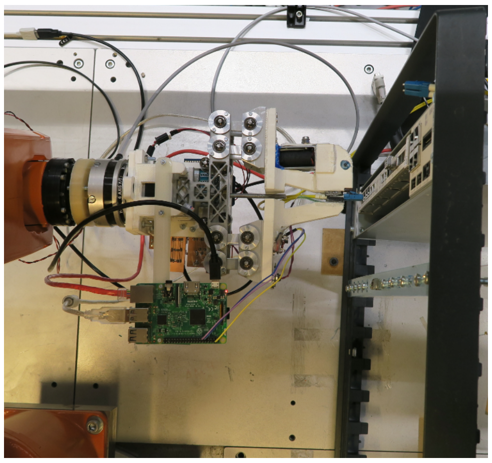
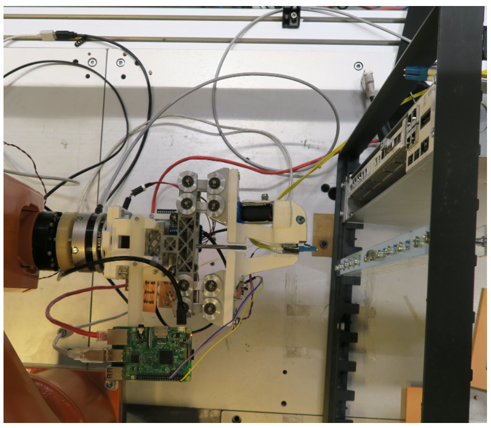

(D) Step 4) Grip the cable.

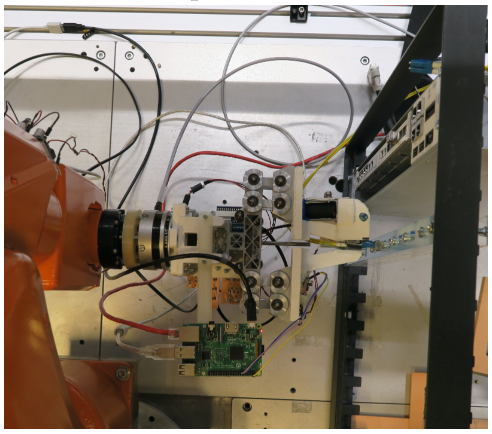
FIGURE E.1: The whole process which was shown at the demo.


(E) Step 5) Activate solenoid & move back & deactivate solenoid.


(F) Step 6) Move back even further.


(G) Step 7) Move beside the helper station and take a photo (whose use is yet to be implemented).


(H) Step 8) Move in front of the helper station.


(I) Step 9) Move into new SFP.

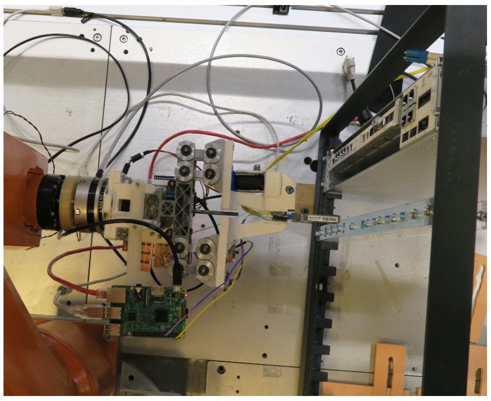
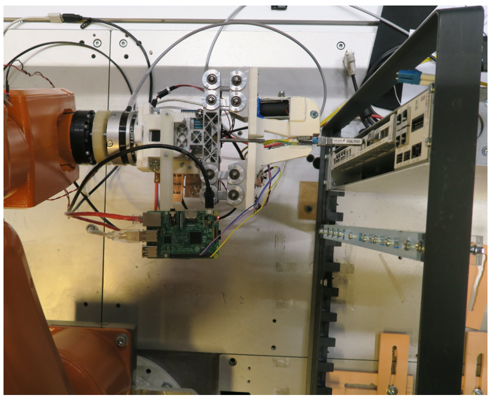
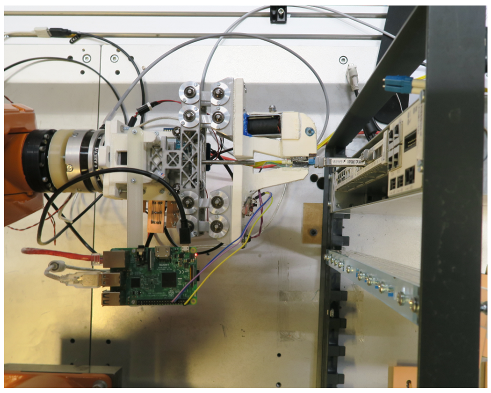
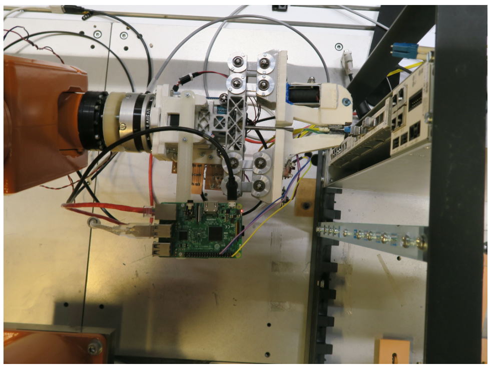

(J) Step 10) Move back.

FIGURE E.1: The whole process which was shown at the demo.


(K) Step 11) Move again in front of the SFP port.


(L) Step 12) Find hole with Alg. 1.


(M) Step 13) Insert SFP with Alg. 2

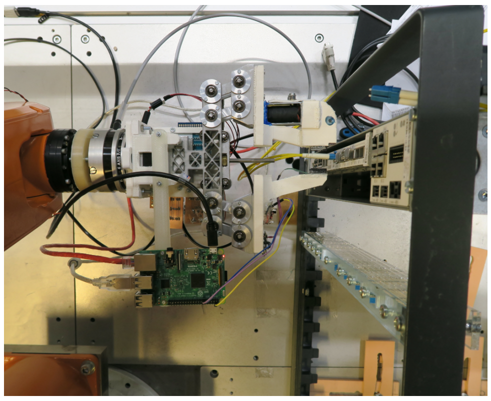

(N) Step 14) Release Gripper & move back to the initial position.

FIGURE E.1: The whole process which was shown at the demo.

Bibliography

[1] Erik Kyrkjebo, Pål Liljebäck, and Aksel A Transeth. "A robotic concept for remote inspection and maintenance on oil platforms". In: *ASME 2009 28th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers. 2009, pp. 667–674.

[2] Sadegh Tabatabaei Hossein Tabatabaei. *Dorna robotic arm*. 2017. URL: `https://www.kickstarter.com/projects/775197166/dorna-fast-powerful-and-precise-robotic-arm` (visited on 04/26/2018).

[3] RS-online. *Robot Arm Construction Kits*. 2018. URL: `https://uk.rs-online.com/web/c/robots-robot-parts/robot-construction-kits/robot-arm-construction-kits/?sra=p&intcmp=UK-WEB-_-L1-PB3-_-Mar-18-_-Robots_Robot_Parts` (visited on 2018).

[4] NASA. *Space Shuttle Canadarm Robotic Arm Marks 25 Years in Space*. 2006. URL: `https://www.nasa.gov/mission_pages/shuttle/behindscenes/rms_anniversary.html` (visited on 05/07/2018).

[5] Universal Robots. *UR5 technical specification*. 2017. URL: `https://www.universal-robots.com/media/1801303/eng_199901_ur5_tech_spec_web_a4.pdf` (visited on 05/16/2018).

[6] Macron Dynamics Inc. *What Is Linear Robotics?* 2018. URL: `https://www.linearmotiontips.com/when-do-you-need-a-gantry-robot/` (visited on 2018).

[7] WEISS. *GRIPKIT BY WEISS ROBOTICS, Smart Gripping Solutions for Universal Robots*. 2018. URL: `https://www.weiss-robotics.com/gripkit/en/` (visited on 05/16/2018).

[8] Bjartmar Freyr Erlingsson et al. "Axiomatic Design of a linear motion robotic claw with interchangeable grippers". In: *Procedia CIRP* 53 (2016), pp. 213–218.

[9] Jin-Siang Shaw and Vipul Dubey. "Design of servo actuated robotic gripper using force control for range of objects". In: *Advanced Robotics and Intelligent Systems (ARIS)* (2016).

[10] Arduino. *Arduino Documentation*. 2018. URL: `https://store.arduino.cc/` (visited on 05/07/2018).

[11] *Raspberry Pi 3, Model B*. Raspberry Pi Foundation. URL: `https://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf`.

[12] *Gertduino Board*. element14. URL: `https://www.element14.com/community/servlet/JiveServlet/previewBody/64534-102-2-287165/User%20manual%20Gerduino%205.6.pdf`.

[13] Richard Szeliski. *Computer Vision Algorithms and Applications*. eng. Texts in Computer Science. 2011. ISBN: 1-84882-935-3.

[14] Rafael C Gonzalez. *Digital image processing*. eng. 3. ed.. Upper Saddle River, N.J.: Pearson Prentice Hall, 2008. ISBN: 978-0-13-505267-9.

[15] Davinia Font et al. "A proposal for automatic fruit harvesting by combining a low cost stereovision camera and a robotic arm". eng. In: *Sensors (Basel, Switzerland)* 14.7 (June 2014). ISSN: 1424-8220.

[16] Dinesh Nair. *ITU Radio Regulations, Section IV. Radio Stations and Systems*. URL: https://www.techbriefs.com/component/content/article/14925?start=1 (visited on 05/17/2018).

[17] OpenCV. *OpenCV Introduction*. 2018. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html (visited on 12/12/2018).

[18] OpenCV. *OpenCV Introduction*. 2018. URL: https://docs.opencv.org/3.4/d1/dfb/intro.html (visited on 12/12/2018).

[19] ABB. *Product Manual IRB120*. Version 3HAC035728-001 Revision: G. 2009-2013.

[20] Michael Dawson-Haggerty. *open-abb-driver*. 2018. URL: https://github.com/robotics/open_abb/ (visited on 12/16/2018).

[21] Richard Gordon Budynas, J Keith Nisbett, et al. *Shigley's mechanical engineering design*. Vol. 8. McGraw-Hill New York, 2008.

[22] Takeshi Furuhashi, Nobuyoshi Morita, and Masayuki Matsuura. "Research on dynamics of four-bar linkage with clearances at turning pairs: 1st report, general theory using continuous contact model". In: *Bulletin of JSME* 21.153 (1978), pp. 518–523.

[23] SKF. *Ballbearings*. 2018. URL: http://www.skf.com/binary/21-121486/Rolling-bearings---17000-EN.pdf.

[24] Inc. Fenner Drives. *Ninjaflex*. 2018. URL: https://ninjatek.com/ninjaflex/ (visited on 12/09/2018).

[25] LLC Engineers Edge. *Coefficient of Friction Equation and Table Chart*. URL: https://www.engineersedge.com/coeffients_of_friction.htm (visited on 12/12/2018).

[26] ISLIKER MAGNETE AG T. *Miniatur Solenoids (D-Frame)*. URL: http://www.elmatik.ee/info/pdf/Muud/GMS-2608.pdf (visited on 11/07/2018).

[27] Manualzz. *User Guide for the Universal Robots OptoForce Kit*. 2017. URL: https://manualzz.com/doc/27806766/user-guide-for-the-universal-robots-optoforce-kit (visited on 12/12/2018).

[28] How To Geek. *What's the Difference Between TCP and UDP?* 2017. URL: https://www.howtogeek.com/190014/htg-explains-what-is-the-difference-between-tcp-and-udp/ (visited on 12/14/2018).

[29] Various Authors. *SWIG Executive Summary*. 2018. URL: http://www.swig.org/exec.html (visited on 12/14/2018).

[30] OpenCV. *OpenCV Introduction*. 2018. URL: https://docs.opencv.org/3.4.4/d4/d13/tutorial_py_filtering.html (visited on 12/12/2018).

[31] OpenCV. *OpenCV Introduction*. 2018. URL: https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html (visited on 12/12/2018).

[32] ABB. *Product specification IRB 120*. URL: https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC035960-001&LanguageCode=en&DocumentPartId=&Action=Launch (visited on 12/16/2018).

[33] Mario Peña-Cabrera et al. "Machine vision approach for robotic assembly". In: *Assembly Automation* 25.3 (2005), pp. 204–216.

[34] Jun Miura and Katsushi Ikeuchi. "Task-oriented generation of visual sensing strategies in assembly tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.2 (1998), pp. 126–138.

[35] Wyatt S Newman, Yonghong Zhao, and Yoh-Han Pao. "Interpretation of force and moment signals for compliant peg-in-hole assembly". In: *ICRA*. 2001, pp. 571–576.

[36] Hong Qiao and SK Tso. "Three-step precise robotic peg-hole insertion operation with symmetric regular polyhedral objects". In: *International Journal of Production Research* 37.15 (1999), pp. 3541–3563.

[37] Young-Loul Kim, Hee-Chan Song, and Jae-Bok Song. "Hole detection algorithm for chamferless square peg-in-hole based on shape recognition using F/T sensor". In: *International journal of precision engineering and manufacturing* 15.3 (2014), pp. 425–432.

[38] Seung-kook Yun. "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?" In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 1647–1652.

[39] Gregory G Slabaugh. "Computing Euler angles from a rotation matrix". In: *Retrieved on August* 6.2000 (1999), pp. 39–63.

[40] A.R.M.S. group. *A.R.M.S.* 2018. URL: https://github.com/EricssonResearch/arms (visited on 12/16/2018).

[41] Debasis Pradhan. *Unit Testing versus Functional Tests*. 2018. URL: http://www.softwaretestingtricks.com/2007/01/unit-testing-versus-functional-tests.html (visited on 12/16/2018).

[42] Joe O'Conor. *"The Magic Tricks of Testing" by Sandi Metz*. 2018. URL: http://jnoconor.github.io/blog/2013/10/07/the-magic-tricks-of-testing-by-sandi-metz/ (visited on 12/16/2018).

[43] Haider Khalid. *SFP Transceivers Explained*. 2018. URL: https://ourtechplanet.com/sfp-transceivers-explained/ (visited on 05/16/2018).

[44] Cisco Systems Inc. *Cisco 10GBASE SFP+ Modules Data Sheet*. 2017. URL: https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/data_sheet_c78-455693.html?referring_site=RE&pos=2&page=https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/gigabit-ethernet-gbic-sfp-modules/product_data_sheet0900aecd8033f885.html (visited on 05/07/2018).

[45] Memory4less. *670504-001 HP B-Series 8Gbps Short Wave Fibre Channel SFP Transceiver Module*. 2018. URL: http://www.memory4less.com/hp-network-transceiver-670504-001 (visited on 05/07/2018).

[46] Cisco Systems Inc. *Fiber Types in Gigabit Optical Communications*. 2008. URL: https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/white_paper_c11-463661.html (visited on 05/07/2018).

[47] *Digital unit Description*. Ericsson. 2016.

[48] *Handling SFP Modules and Optical Cables*. Ericsson. 2014.

[49] Orientalmotor. *Everything You Need to Know About Stepper Motors*. URL: https://www.orientalmotor.com/stepper-motors/technology/everything-about-stepper-motors.html (visited on 12/12/2018).

[50] ISLIKER MAGNETE. *GM - Miniature solenoid*. URL: http://www.islikermagnete.ch/download/prospekt/GM.pdf (visited on 12/16/2018).

[51] ABB. *Technical reference manual. RAPID Instructions, Functions and Data types*. Version 3HAC 16581-1. 2004-2010.

[52] Python Software Foundation. *Python 3.7.0 documentation*. 2001-2018. (Visited on 09/20/2018).

[53] IBM. *How sockets work*. URL: https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_72/rzab6/howdosockets.htm (visited on 09/20/2018).