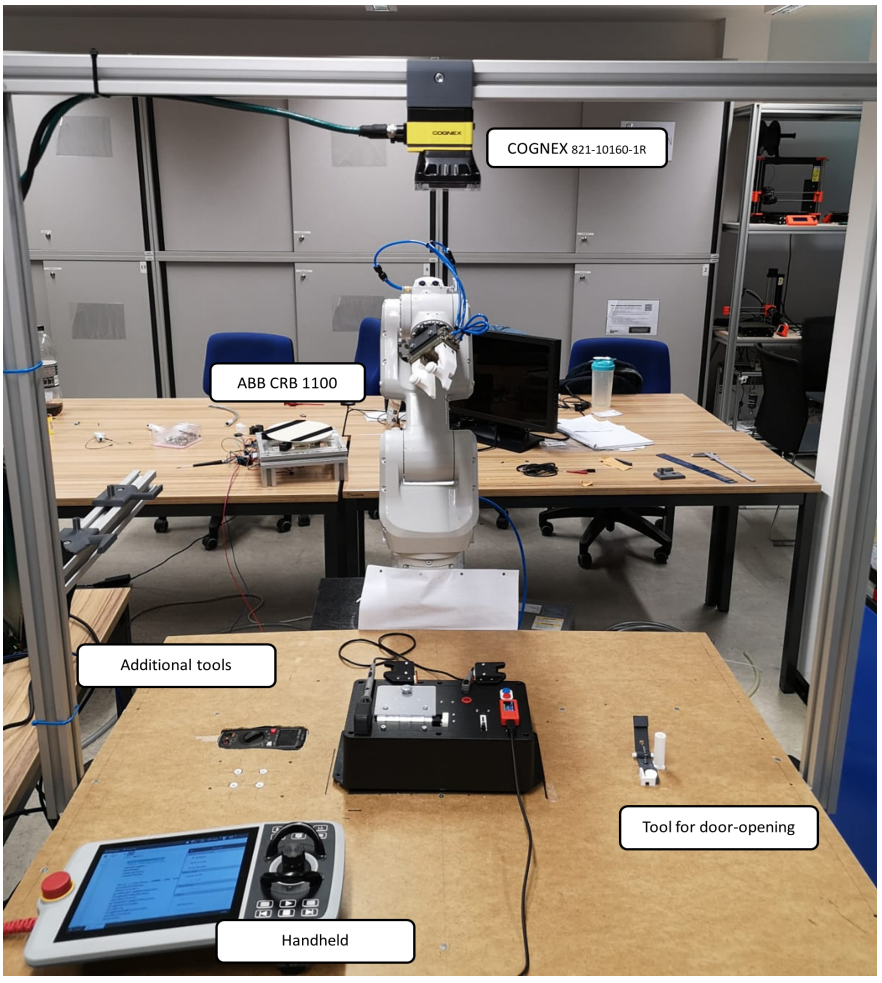
Robothon-2023-Grand-Challenge-TechWi(e)nBot-Team-Report

This page deals with the main-task (Taskboard). To see the report of the BYOD-report, click here.

Hardware Setup

The setup consists of



- Base (Table consists of aluminum profiles)
- ABB CRB 1100 + OmnicorC30 control unit
- Schunk parallel gripper (MPC 075) with 3d printed plates
- COGNEX 821-10160-1R camera
- (optional) turntable as additional axis (self-crafted) turntable
- additional tools in order to fullfill the tasks

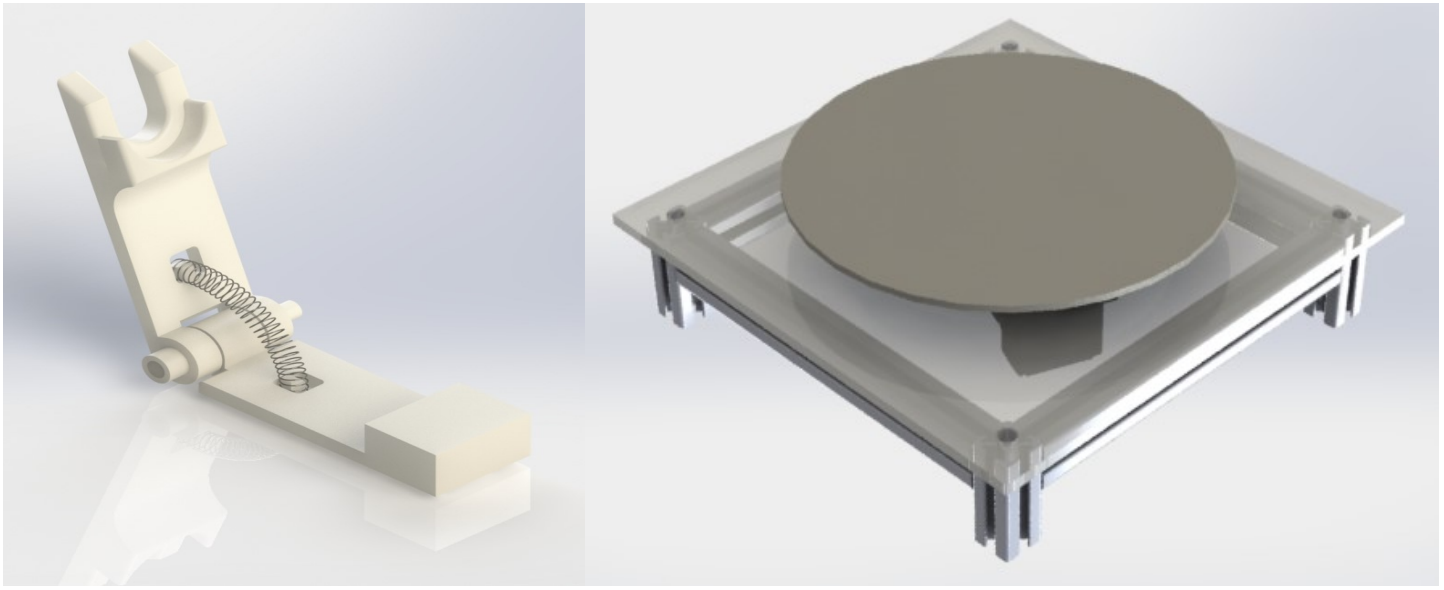
General Approach

Our goal was to successfully complete all of the tasks with perfection. However, our robot's range is not

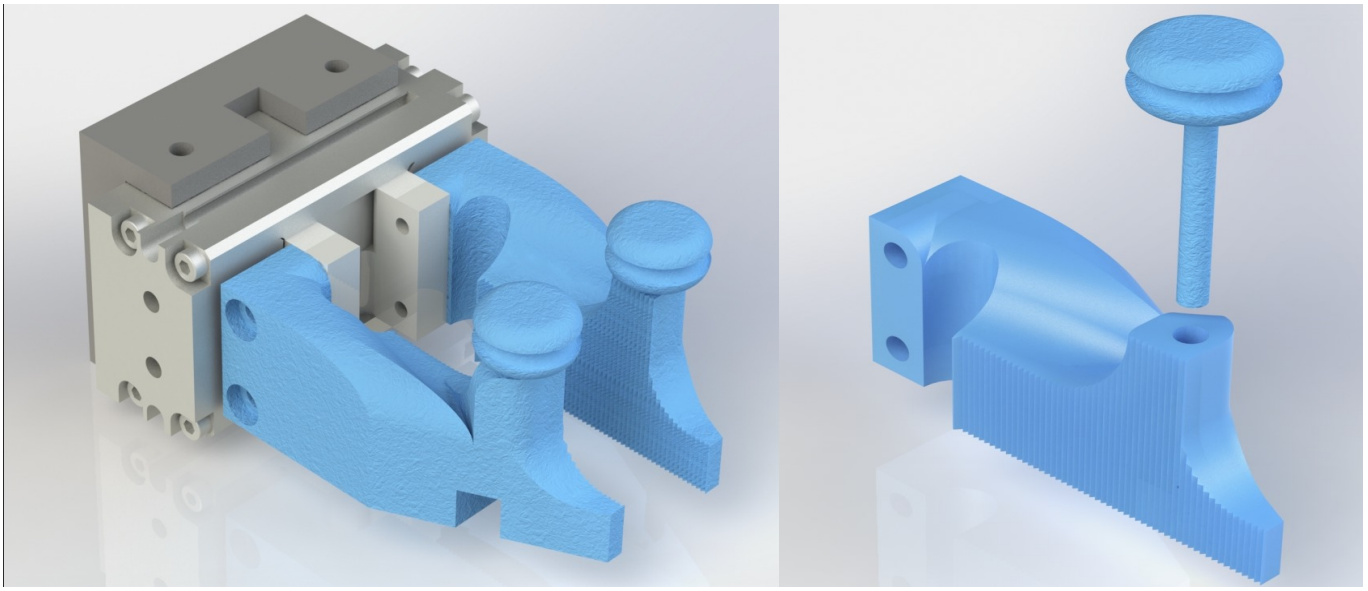
sufficient to solve the task from any given orientation of the taskboard without constraints.

constraints, or see below

To overcome this limitation, we decided to create tools, such as a gear-spring mechanism to open the door, and an additional axis to rotate the whole taskboard to a suitable orientation. The additional axis is controlled by the robot control system and does not require any human interaction. While there are specific orientations of the taskboard in which the additional tools are not necessary, we wanted to ensure that our robot could adapt to any random orientation of the taskboard. However, at the time of submission the integration of the turntable to the robot system does not work with a sufficient reliability. Thus, we did not use it in our final submission video. Nevertheless we made a great effort in the developing process, which is why we still want to present it in our report.



Despite the tools we presented, the gripping mechanism itself was the most crucial part of our design. To perform all of the requested tasks efficiently, we needed to create a gripper plate that was significantly different from any existing types on the market. Our solution, especially for the wire-winding task, was to incorporate pulleys into the plates. These pulleys, when in a closed configuration, created a small gap for the wire, allowing it to be held tightly while still maintaining the necessary freedom of motion to wind the wire.



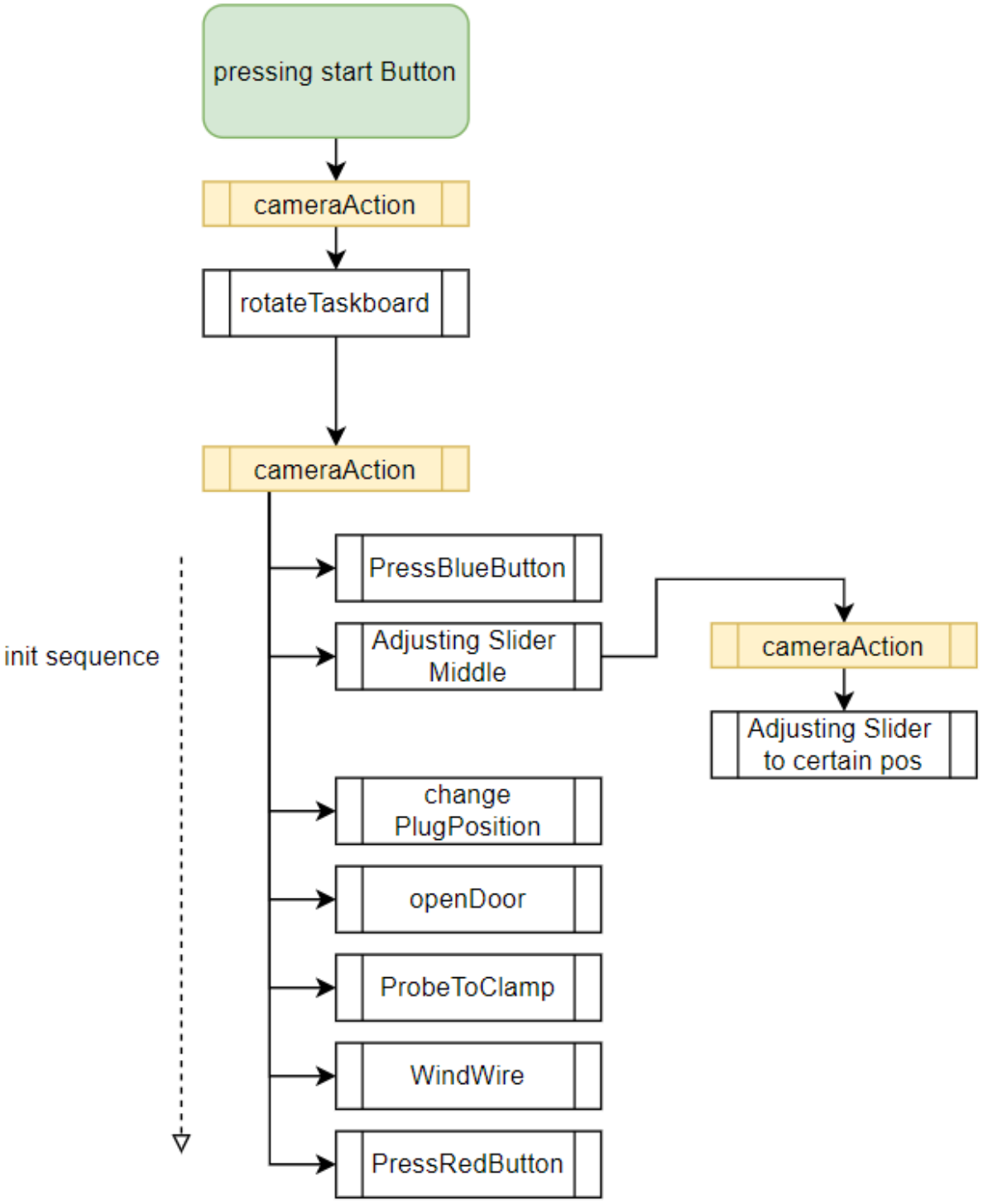
Why do we use such a complex construction for the wire-gripper part?:

The reason why we decided to use pulleys is that from our point of view it is the method which damages the wire least. Other options - like simply grabbing it or using a bigger gap which allows some motion - would still evoke very high friction and would destroy the wire over time. With our solution, friction is neglectable and as a matter of fact the service life of the wires increases.

Software solution

Methods

Our approach was to split all of the sub-tasks into separate methods to make them independent of any given sequence. However, there is an initialization sequence that was provided by the task.



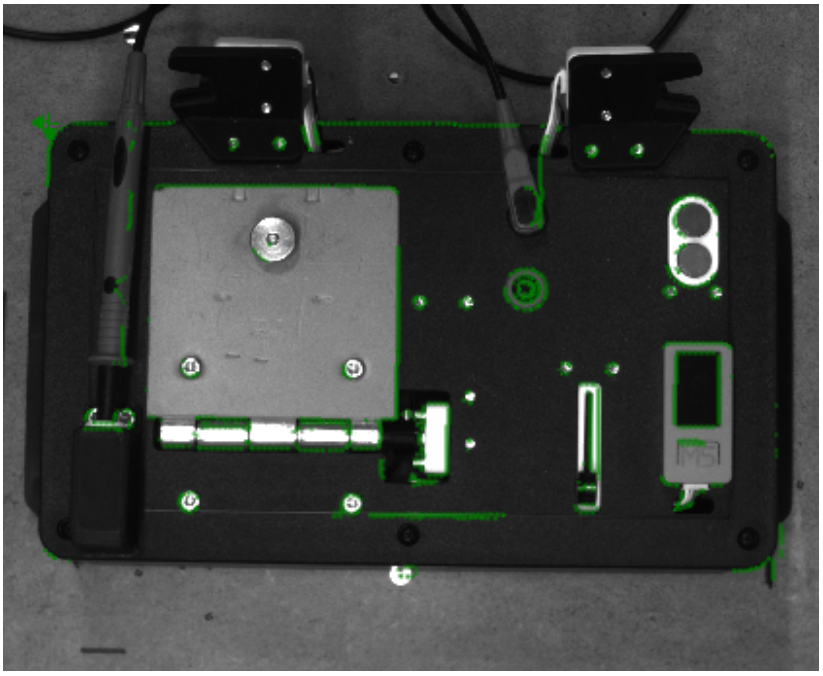
In the flowchart above, you can see that the process 'camera action' is repeated three times. This indicates that a specific camera job is loaded onto the camera, executed, and the results are received. More information about the camera (vision) jobs will be provided in the next paragraphs.

Vision System: Board localisation

We used the Cognex Vision framework in order to locate the taskboard and its orientation. In detail, we generated several different "jobs", which got called by ABB's rapid-code. Specifically, we used the Cognex PatMax RedLine Pattern algorithm.

- "newTaskboard.job" -> result: position and orientation of the taskboard
- "TaskboardOnTable.job" ->result: position and orientation of the taskboard
- "BYOD.job" ->result: position and orientation of the BYOD

In simple terms, the Cognex software generates an additional coordinate system that refers to the origin of the calibration coordinate system. To use the coordinate system of the taskboard, the object frame is copied from the camera target to a coordinate system that is available in ABB's RobotStudio IDE.



The origin of the taskboard coordinate system is set to the upper left corner (indicated by the green symbol), and all other positions are "hardcoded" with respect to this point.

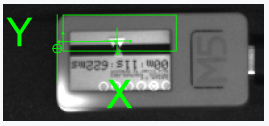
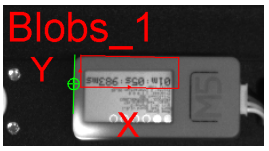
Vision System: Slider adjustment

Seperated from the localisation of the taskboard, there is another camera job to execute

- "SliderDisplay.job" ->result: distance beetween green and yellow triangle

In order to detect the triangles, we used what are called 'blobs.' Blobs are a collection of several pixels with a common brightness. Since the blue screen of the display has a different brightness and number of pixels than the triangles, it is possible to separate them.

Since the orientation and position are already known by the system, the only important information is the distance between the triangles. Unfortunately, there are several edge cases that need to be considered:

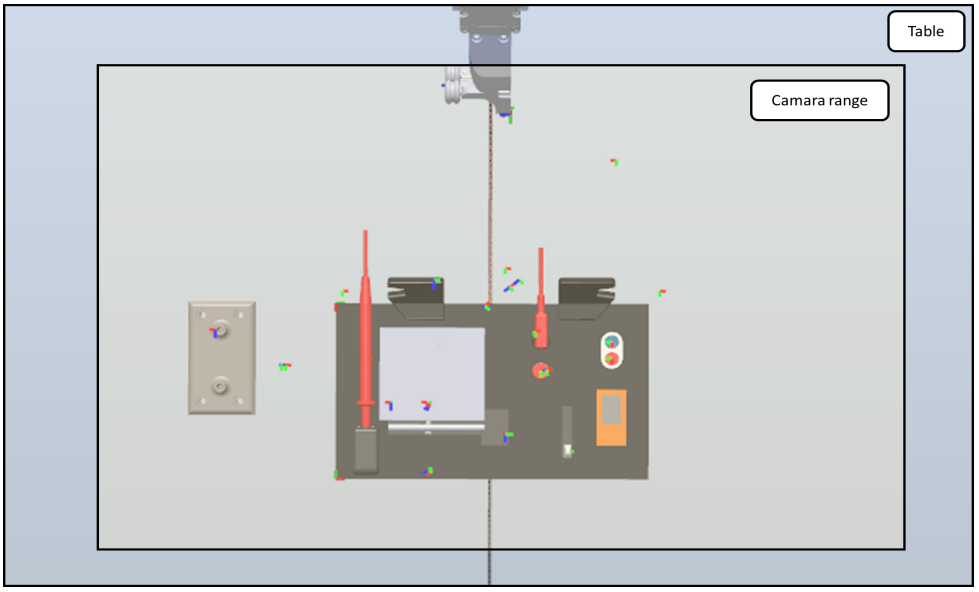
Case	definition	Image	sequence
regular	camera can distinguish between both triangles and reed the distance beetween them correctly		robot moves slider to determined position
overlap	triangles overlap. So the blob algorythm can´t find 2 seperate triangles, since they merged to one big collection of pixels		robot moves slider 5mm from middle postion in both directions. This sequence is indicated by a short blue blink of the robot´s built-in lamp
done	A rare case, in which the triangles overlap so tight, that the task is already done by simply moving to the middle triangle.		should be none. To avoid errors due to false interpretation, we decided to start the same sequence like in "overlap".

Constraints due to Setup

Since we use only one robot to solve the task (asumption that turntable is not implemented to the solution yet), there are some constraints which have to be considered while placing the taskboard to a specific position/orientation

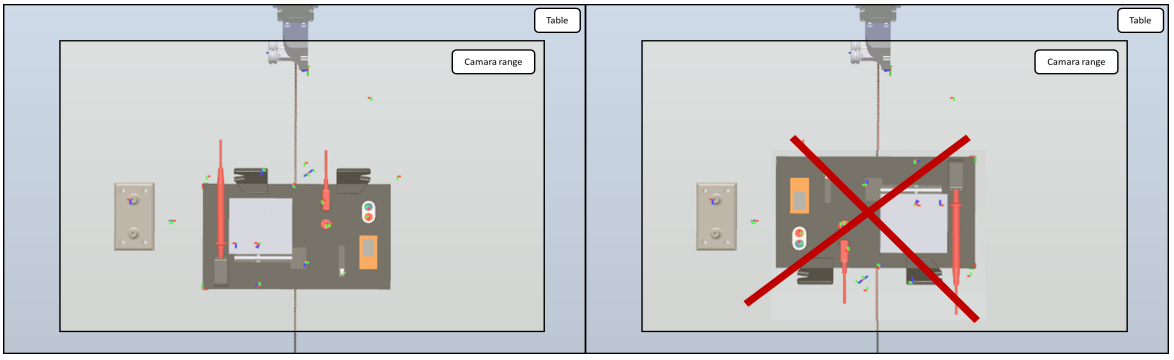
camera frame

First of all, the area of positionining is constrainted by the field which is covered by the camera. Due to the chosen height of the camera and its technical properties, we are able to locate the taskboard in a area which looks like this:



Range of robot

Second topic to consider is the range of motion of the ABB CRB 1100. In general the range of the TCP0 (center of flange) is about 74cm. In addition, to solve the given tasks, the range of "free positioning" decreases. In particular, the taskboard must be placed in an orientation, so that the "wind wire" area faces the robot´s base. With our tools and configuration, it is not possible to rotate the taskboard so that the winding area is orientated away from the base.

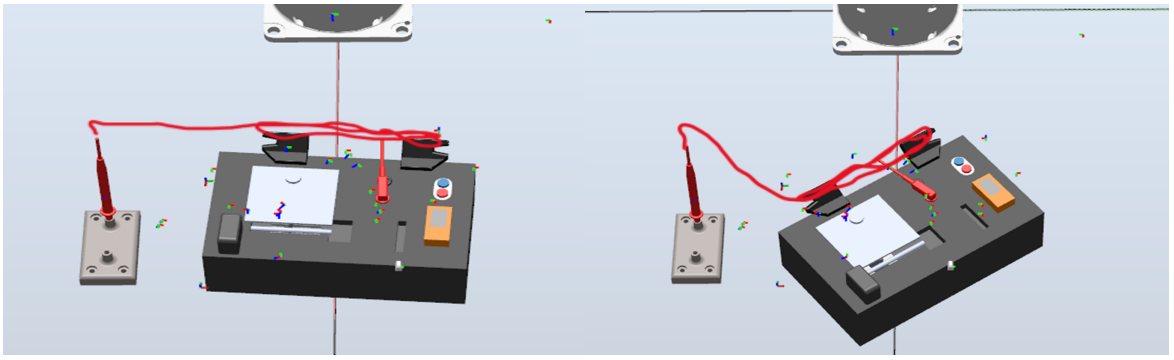


Another aspect to consider is that the taskboard must maintain a minimum distance from the robot's base. If this minimum is undercut, the robot's joints would collide, and the robot would stop. While holding this minimum distance, it is nearly impossible to open the door by simply grabbing the knob and performing a circular movement that is out of the robot's reach. Hence, opening the door and winding the wire would be mutually exclusive. Therefore, we created the 'door-opener' tool, which bypasses the problem of limited range. With this tool, we are now able to open the door from the front side.

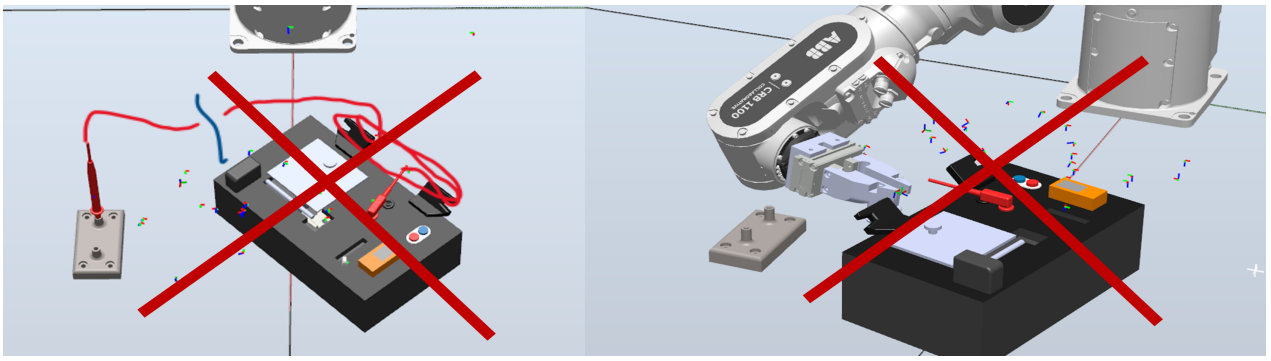
placing probe in temporarily clamp

Due to the fact that we only use one robot, the combination of the winding task and the probe itself cause another constraint. After pluging the probe into the regular clamp, we can not plug it back to the white socket because it would interfere with the winding task. So we have to plug it to an temporarily clamp which allows us to taking it back from a determained position after the winding task. Since we must wind the wire 2 times to guarantee an "always-working solution", the rest of the wire becomes very short. That is the reason why we were forced to reduce the accepted orientation-angle to about -10° (left image) to 45° (right image). To clarify: orienting the long side of the taskboard parallel to the robot´s

base would lead to an angle of 0°. Angle is counted positive counter-clockwise.



If we fall below the bottom limit (left image), the wire will be too short and would tear the probe from the temporarily clamp. If we exceed the the angle in positiv direction (right image), the winding task won`t work due to the limited range of motion of the robot.



New insights

In general, participation in the robothon challenge was a great pleasure - although it was extremly time-consuming. To give a short conclusion, we want to list our most important new insights in the developing-process of new robotic-solutions:

- if the task seems to be impossible: be creative and create a tool to solve it
- first things first: Set up the working station perfectly at the beginning, not in the end. (we had to learn it the hard way)
- mount camera seperatly from the robot. If robot moves fast, the camera wiggles a bit
- wires are hard to control and their behaviour is nearly unpredictable
- wire´s behaviour change tremendously after changing the velocity of the robot
- clean up your working station before using the robot -> it will definitely reduce damages
- treat any suggestion for a solution instead of ruling it out instantly

How to run

To run the software solution, follow this sequence:

- Print all .step files with a 3d printer (slice to gcode)
- Mount gripper
- Install ABB Robot Studio or run directly from FlexPendant via USB

- Import the software solution
  SoftwareSolutionRobot
- Connect the camera to the controller via Profinet
- Select camera IP adress in the IDE
- Store all .job files on the camera
  CamJobs
- change light level to fit to the environment
- calibrate the camera via cognex grid
- calibrate robots workobject coordinate system to the grid
- run the solution via ABBs IDE in hand-operation-mode and check if all trajectories are suitable
- change robots settings to automatic mode in order to operate with full speed

Repository of software modules

The created software solution is based on third parties software:

- ABB robot studio IDE, usage of software package Rapid
- Cognex Vision framework
- Arduino IDE + stepper.h library (stepper motor for turntable)

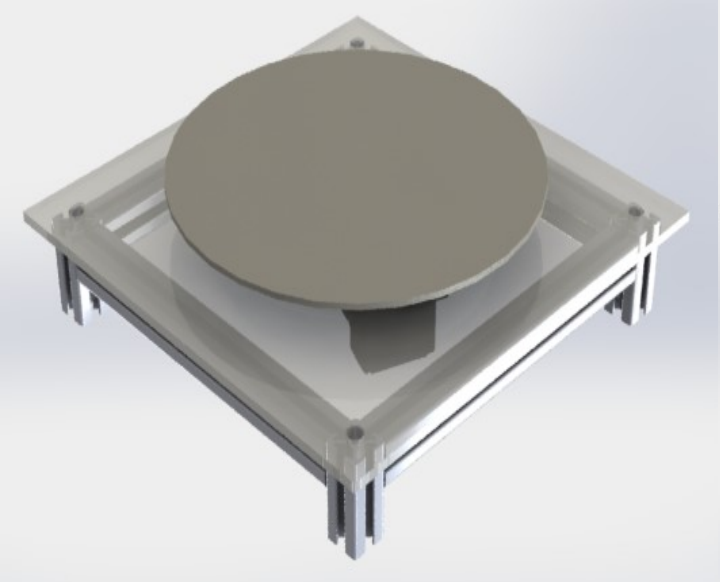
Authors

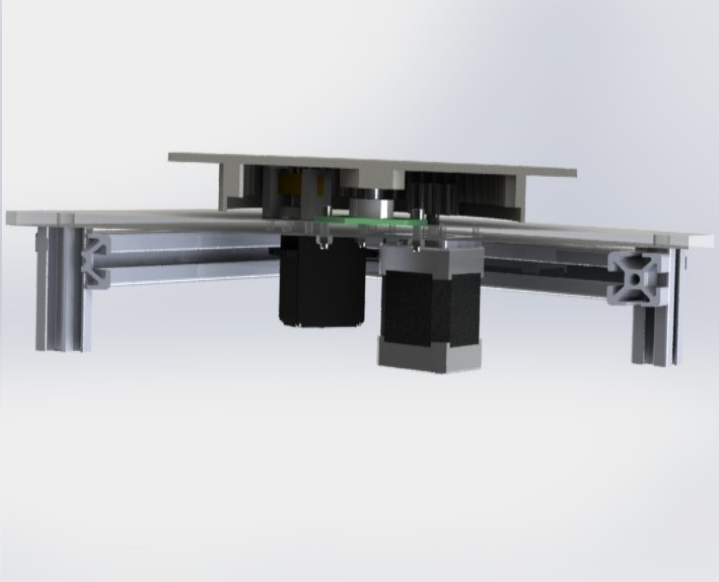
(in alphabetical order)

Sebastian Neuhofer, Christian Scheidl, David Seyser, Julian Smole

Turntable

To reduce the amount of constraints we have in our system, we decided to build a turntable on which we mount the taskboard. The turntable is driven by a stepper motor, with a spur gear mounted on it to transfer the torgue from the motor to the internal spur gear, which acts as the table-top where the taskboard is located.



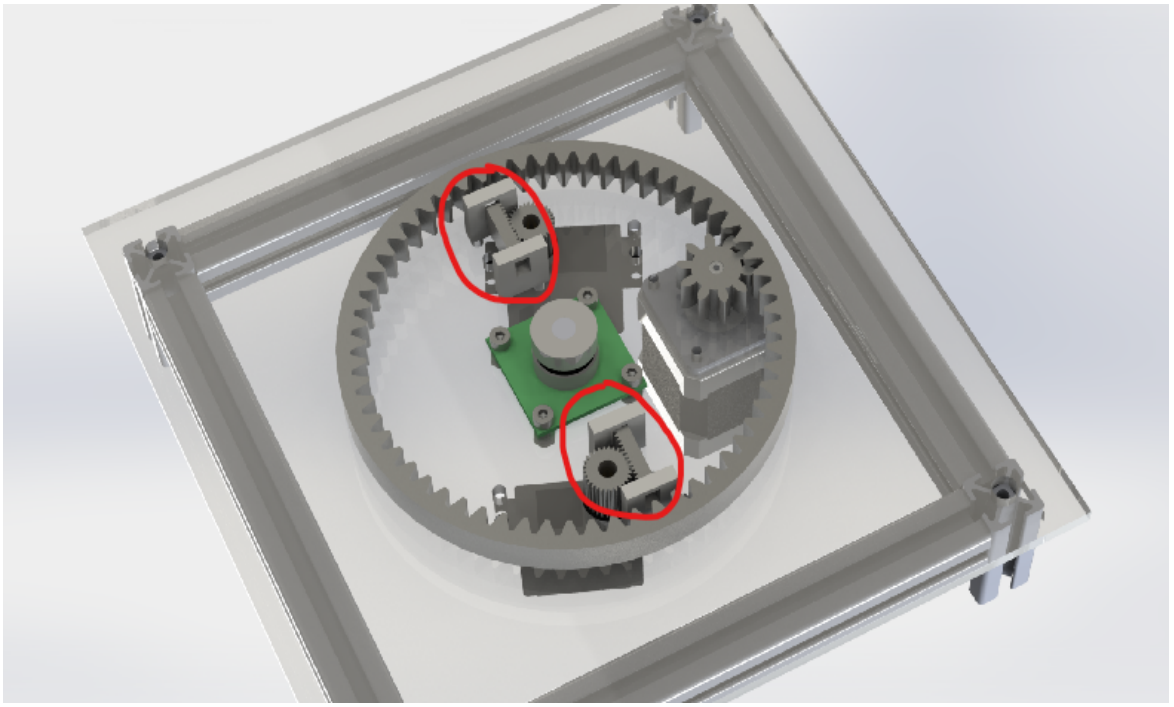


Motor control

The turntable is programmed and wired to an arudino uno, to control the stepper motor we used the Stepper.h library, therefore we only need to provide an angular value to start the turning process, this would mean that we would need a ethernet or TCP/IP connection from ABBs control unit to our arduino, but in the timespan of this project we were not able to achive such a connection and therefor had to circumvent that problem of sending and receiving data from and to the arduino. We resolved this problemby simply sending a digital output signal from the control unit, for a specified period of time, depending on the angular value we received from the camera. In the arduino program we could then create a while-loop, which is only active as long as we receive the signal from the controler, in which we send a turn-signal to the stepper motor. This way we were able to create a functioning turntable which can reposition the taskboard every run.

Break Mechanism

In addition, we decided to build brakes that engage with the internal spur gear with a rack when the table is stationary. This is because the stepper motor use has a small holding-torque and could lead to the robot turning the whole mechanism at some tasks. These breaks are also controlled via the arduino.



Integration of the turntable to the system

At the time of submission, the turntable has not been integrated into the system due to some difficulties that need to be addressed. The main issue is that the turntable's height increases the likelihood of configuration errors for the robot. To resolve this, we suggest embedding the turntable into the working table. Another consideration is that the camera would need to be recalibrated for the new setting, which would require additional time.

In summary, we have proposed a simple 'one-axis' solution to overcome the constraints of orientation. This approach demonstrates the redundancy of a second robot and provides a more cost-effective solution.

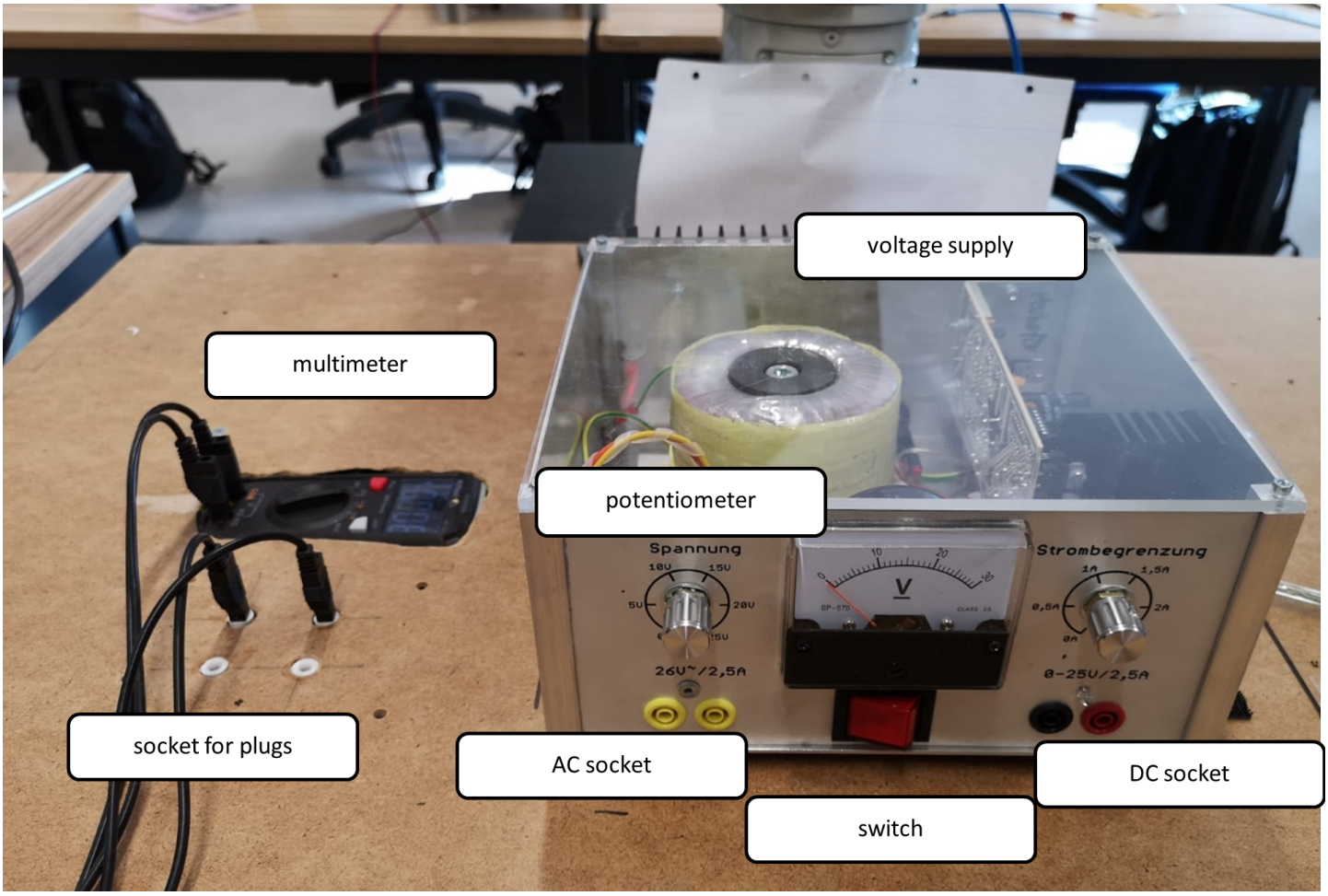
This page deals with the BYOD-Task. To see the report of the Taskboard-report, click here.

BYOD: verification of a voltage supply unit

The selected device for the BYOD task is an old voltage supply unit that has been used in a laboratory. Since these devices require regular verification, an automated solution would decrease the workload for workers in the lab and enhance the service life of the devices by reducing unnecessary exchanges.

Setup

There are many targets which have to be used in order to fullfill the task. The most important targets are highlighted in the following image



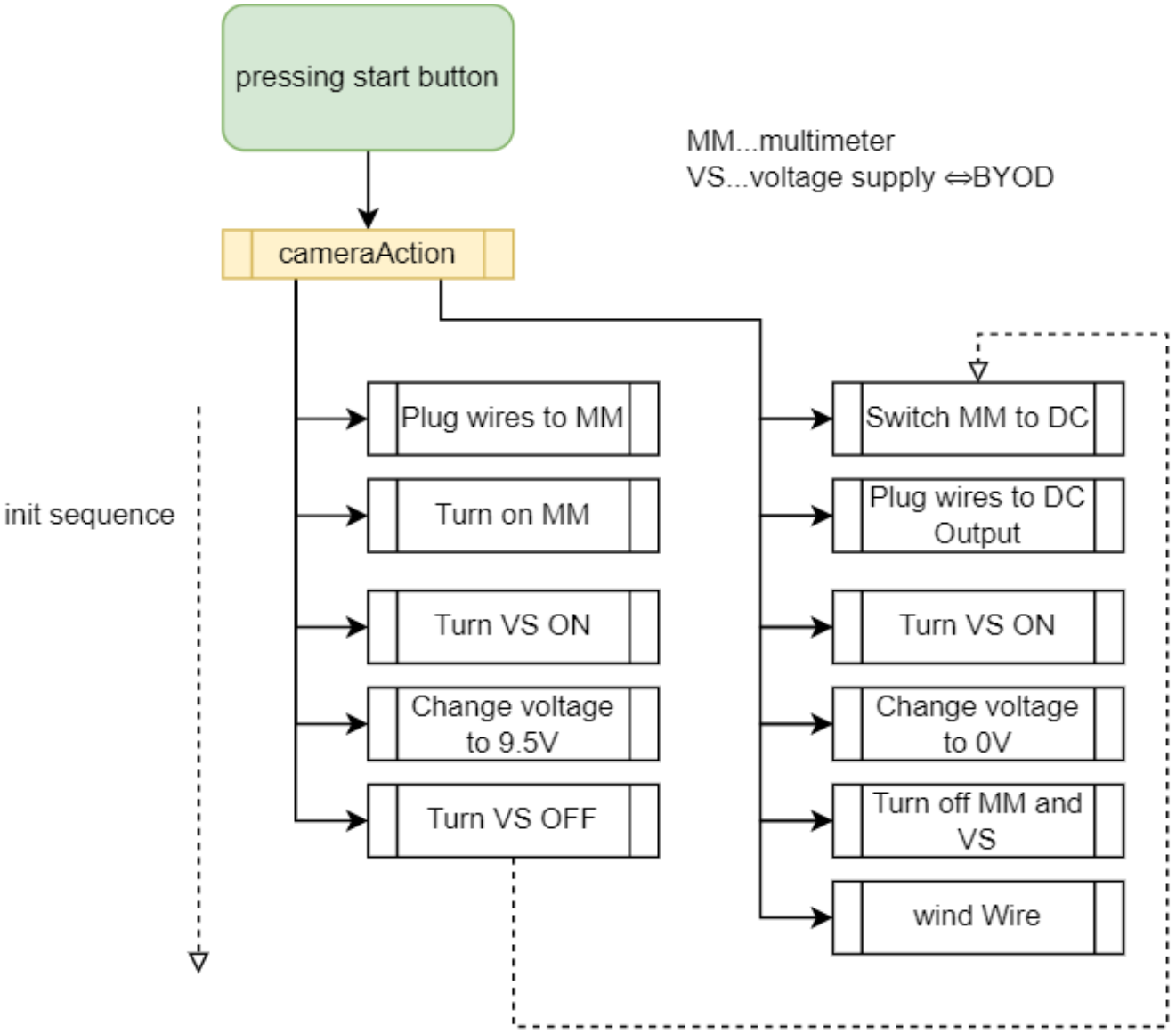
We used the same gripper like in the main-task.

Software solution

Methods

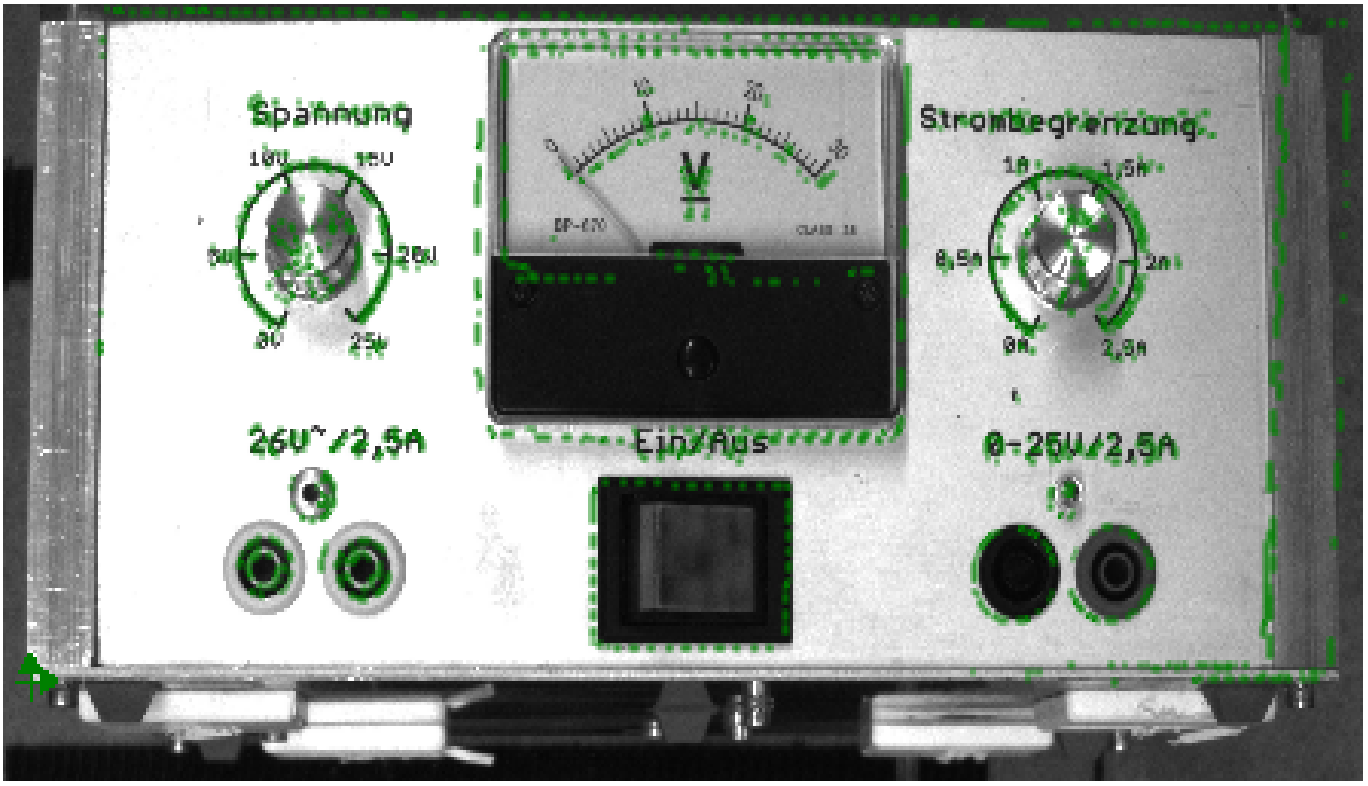
Our approach was to split all the sub-Tasks into seperate "methods", in order to be independent to any given sequence. Of course, there is an init-Sequence which has a "logical" order. You can see the

methods in the following flowchart diagram.



Vision System

Similar to the main task, the position and orientation of the device must be determained. Therefore wie applied a similar alogrythm to detect all information needed.



The origin of the new target coordinate system is set to the bottom left corner and all other targets reffer to this origin.

Difficulties

Since we wanted to utilize as many wire interactions as possible, we created a task that presented some difficulties during operation. One of the biggest challenges was the usage of two different wires. While executing the task, we had to ensure that both wires did not interfere with each other and impede their performance. We developed a well-prepared code sequence to solve this issue, which has been successful in practice. In an iterative process, we developed strategies to cope with the unpredictable behaviour of the wires as well as possible.

Another problem arose due to the small movements of the device. Since the device is not mounted to the table and our table is not well balanced, fast movements from the robot can lead to small changes in the position and orientation of the device during execution. To bypass this problem, we call the method 'camera action' several times during execution. Thus, the position and orientation of the device remains up-to-date. The only compromise we had to make is that execution time will increase since every 'camera action' call needs up to 2 seconds of additional execution time.