University *of Ljubljana*
Faculty *of Electrical Engineering*

TEAM SINGULARITAS

Robothon® Grand Challenge

PROJECT REPORT

Sebastjan Šlajpah

Franci Ovčak

Timotej Klemenčič

Peter Kmecl

TEAM SINGULARITAS

ROBOTHON® GRAND CHALLENGE
UNIVERSITY OF LJUBLJANA, FACULTY OF ELECTRICAL ENGINEERING, SLOVENIA

Sebastjan Šlajpah, Franci Ovčak, Timotej Klemenčič, Peter Kmecl

- Universal Robots UR5e
- Robotiq Hand-e
- multigrip fingers
- web camera
- user interface
- force based interaction
- toys as e-waste

Abstract: For the challenge collaborative robot Universal Robots UR5e was selected. The robot was equipped with gripper Hand-E from Robotiq with custom designed fingers and additional web camera. Manipulator was places on an aluminum table with area for the task board to be placed.
The software consists of four parts: graphical user interface, machine vision, communication architecture based on ROS, and programs on the robot for solving individual tasks. The robot was controlled via graphical user interface where different task order can be selected. At first the robot performs localisation of the task board. After, the tasks are solved in selected order. Tasks solutions have integrated functionalities of force/torque sensors thus providing flexible and robust performance.
In the transferability challenge we follow a 3-years old boy through his ordinary day. During the day he comes in contact with many different electronic devices: from a radio alarm clock in the morning, car keys of his father's car, toys and interactive books during play time, to TV's remote controller while watching cartoon before sleep. The children's' world of today is filled with electronic devices and many of them use batteries as a power source, sadly with the potential to become future e-waste after kids are tired of playing with them. Responsibility is of course on the parents' side to pass the toys on to other kids or, in case that the toy is damaged beyond repair, take care of proper disposal thus providing an example to future generations.

1. Introduction

Robothon® is an international competition and benchmarking event for measuring state-of-the-art performance in robot manipulation. This year challenge included handling, sorting, and manipulating e-waste.
In this document we present the hardware components used to successfully completed given task. Special attention is given to the design of grippers.
In software section we present overall architecture of our system. We also present the graphical user interface and use of camera for task board localisation. Each task is described in step-by-step manner with additional figures and highlighted features.
Quick start guide can be used for someone who would like to run our program. Is describes step-by-step procedures and instructions to successfully use our solution.
The last chapter includes presentation of our transferability demo. We presented toys as a possible future e-waste problem with some proposal how to minimize the problem.

2. Hardware

Robotic cell is composed of

- collaborative robot Universal Robots UR5e (software PolyScope 5.11) with integrated force/torque sensor mounted on aluminium table,
- collaborative gripper Robotiq Hand-E,
- custom 3D printed fingers,
- web camera Logitech C922 PRO mounted on the gripper,
- task board provided by Robothon®,
- personal computer.

The whole system is presented in Figure 1.

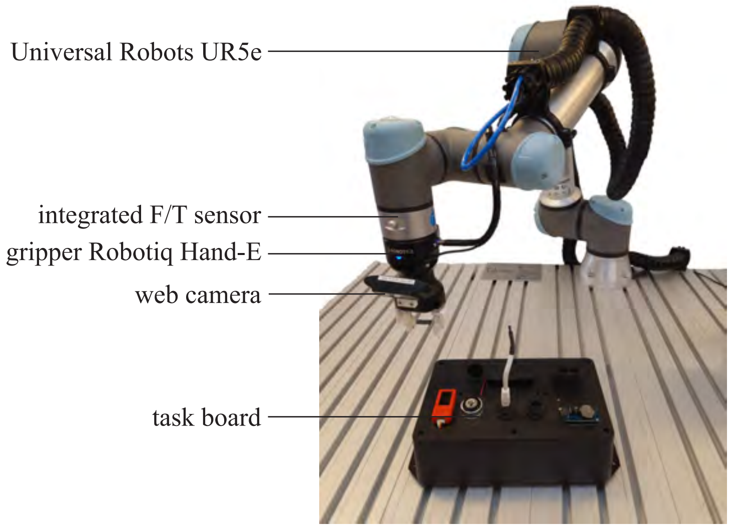


Figure 1: Robotic system

2.1 Universal Robot UR5e

For the manipulator we selected 6 DoF collaborative robot UR5e from Universal Robots as it brings ultimate flexibility for the challenge. The robot has a payload of up to 5 kg and a reach of 850 mm. With the PolyScope interface the robot enables quick and easy programming of even more complex applications.

The main feature of the robot is integrated force/torque sensor on top of the robot. Force/torque sensor with functionalities from Force Copilot by Robotiq provides additional feedback about interaction between the robot and the environment and thus enables implementations of different force-based functionalities, e.g, pressing the button or turning the key by moving the robot until threshold force is exceeded; using force-feedback to solve peg–in–hole problem while inserting batteries. All of those functionalities upgrade the system to be more robust and more flexible in case of errors from vision system or perhaps with problems while gripping some objects.

2.2 Gripper design

Collaborative robotic gripper Hand-E by Robotiq provides a basis for the gripper design. Two finger parallel gripper was upgraded with custom designed finger and additional web camera used for robotic vision. The whole gripper design can be seen on Figure 2.

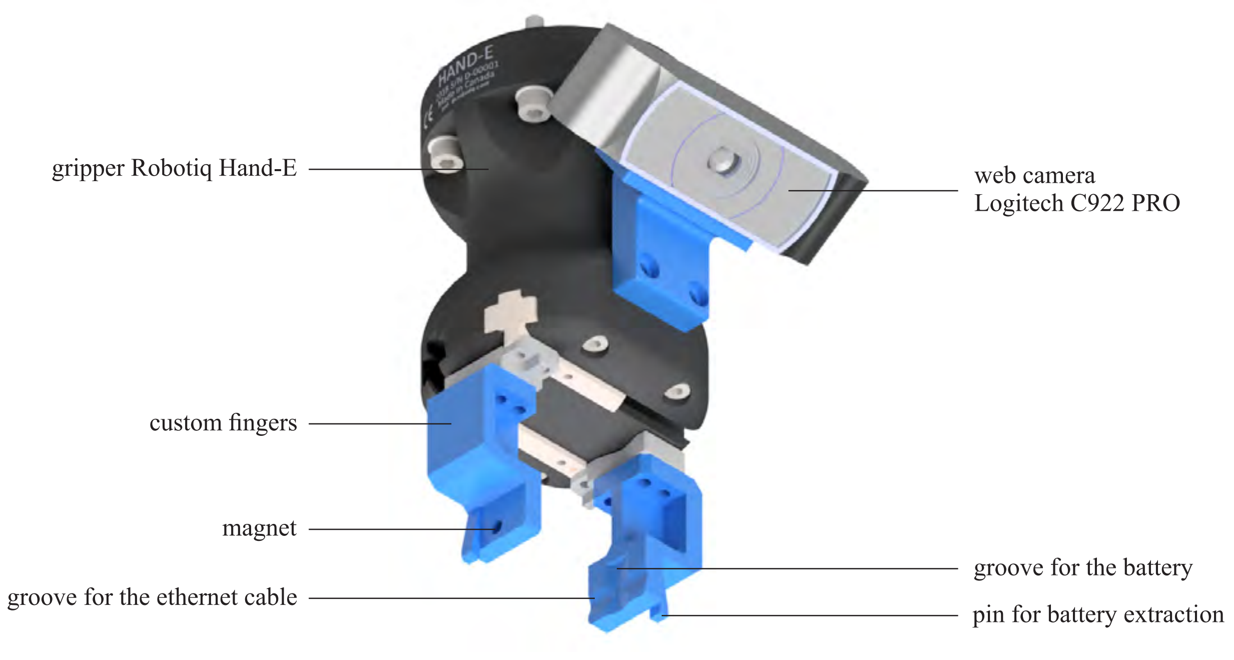


Figure 2: Gripper design used in the robotic system

2.2.1 Finger design

The basic design of the fingers represent classical flat gripping surface. Fingers were 3D printed with PLA plastic so the design needed to be upgraded in a way to ensure rigidity and strength of the fingers while gripping objects. In addition, four small modifications were introduced for successful (or less stressful for the material) completion of tasks.

1. **Horizontal groove for Ethernet cable**: RJ45 connector at the end of Ethernet cable has a locking latch that is covered with protective rubber. In order to press the latch all the way the fingers needs to move the latch beyond the rubber covering the latch joint (alternative is to provide high gripping force to squeeze the rubber). A 0.8 mm vertical groove was added to right finger to compensate for the additional rubber on the RJ45 connector.
2. **Pins for battery extractions**: To extract the batteries from the battery holder, the batteries need to be pushed toward the negative end to release them. For that we added two small pins at the end of the fingers that enables us to simultaneously press and release both batteries at the same time. In addition, those pins are used also for pressing buttons, sliding battery cover, and releasing coin battery.
3. **Small magnet**: To prevent problems with the robot configuration at different task board orientations,

all the elements on the board needed to be manipulated from the top. That means that the batteries needed to be rotated from horizontal to vertical orientation. In order to do that we added small magnet to the left finger that provided enough magnetic force to keep the battery attached to the finger while transferring the battery to upright position. In this way we used the gravity to provide force for rotation of the battery while the magnet represented the axis of rotation.

4. **Vertical groove for handling batteries**: To provide repeatable grip of the battery in the upright position, additional vertical groove was added to the right finger thus providing three points of contact between the battery and the fingers. Flat design of the gripping surface would provide only two points of contact that can be problematic from the point of repeatability.

2.2.2 Camera

For the camera we used widely accessible web camera Logitech C922 PRO. Camera has a full HD 1080p@30 fps sensor with 78° field of view. It has auto-focus and light correction. Camera in combination with open source libraries for image processing represents a low-cost counterpart to different industrial grade cameras (with the price tag 10× − 100× lower compared to industrial cameras).
We mounted the camera directly on the gripper under 45° angle via 3D printed custom designed interface. Camera mounted on the robot provides additional flexibility of the system as the working are is not limited to camera's field-of-view. In addition, there is no additional gantry for the camera providing more space for robot motion.

3. Software

The software part of the solution, as seen in Figure 3, is composed of:

- Universal Robots controller software PolyScope 5.11;
- URCap for gripper Robotiq Hand-E;
- URCap Force Copilot by Robotiq;
- graphical user interface designed with PyQt;
- image recognition software designed in Python with included OpenCV library.

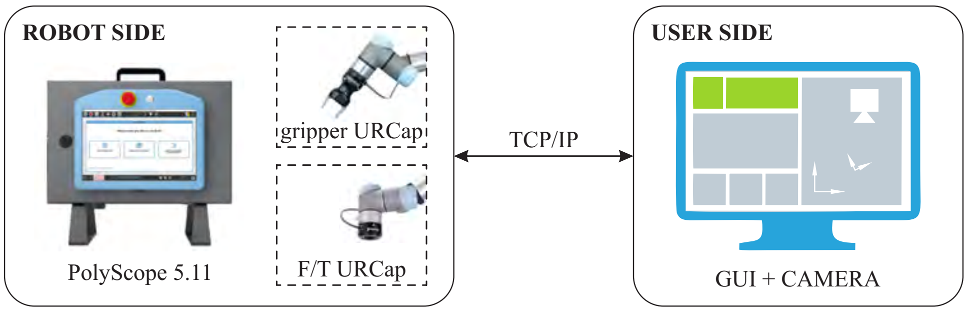


Figure 3: Software used for the application can be divided into two parts: a) software used on the side of the robot, and b) software used on a user's side (on a PC).

TCP/IP protocol is used for communication between master PC with graphical user interface and the robot controller.
Overall architecture is presented in Figure 6).
On the master side... GUI....camera....
On the robot side a main program consisted of TCP/IP client from which different subprograms are called after initial handshake with the server. Total 10 subprograms are defined:

1. move the robot to snapshot position for task board macro localisation;
2. update the definition of user coordinate system with the recognised pose of the task board;

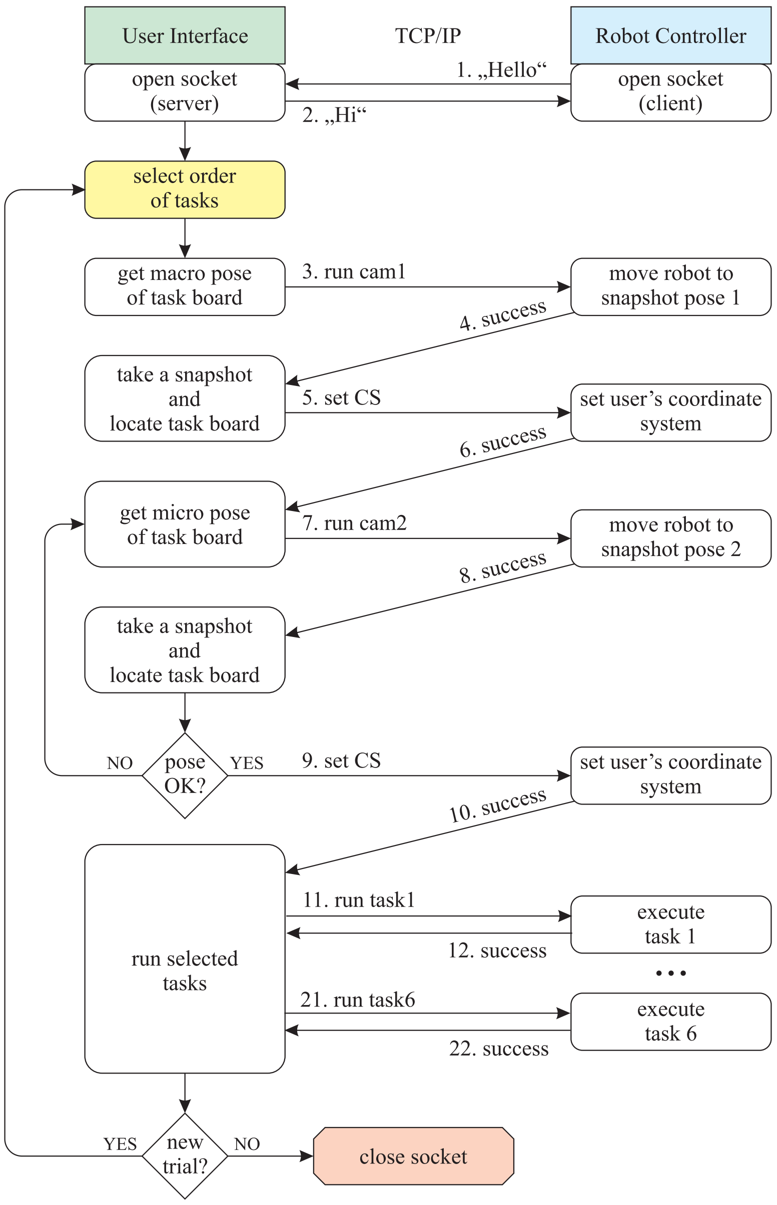


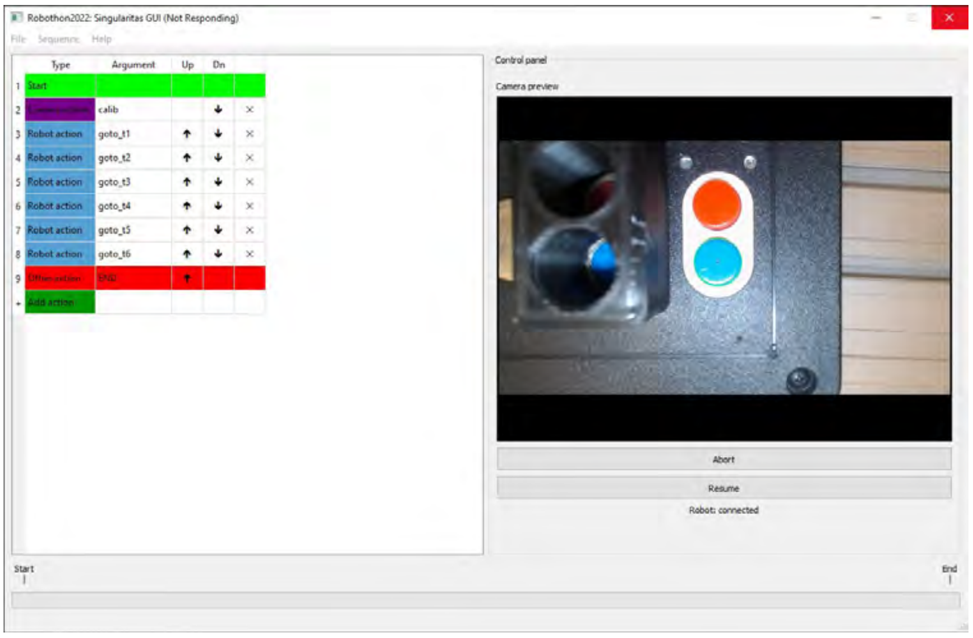
Figure 4: Software architecture with marked connections. On the left side are the instructions from the master (user interface) and on the right side are instructions for the robot. Yellow block represents an input from the user.

3. move the robot to new snapshot position for task board micro localisation;
4. if newly defined pose is sufficient, update user coordinate system;
5. subprograms for individual tasks (6 in total).

After individual task is finished robot sends a confirmation to the server in order to continue with the execution of the main program.

3.1 Graphical user interface

We have developed an intuitive graphical user interface (GUI), that enables us to load, save, edit, run and monitor task sequences for the Robothon grand challenge taskboard. The GUI, seen on figure 5, can be split into two parts - the sequence editor, and the control panel. Many GUI features can also be accessed from the top menu.



	Type	Argument	Up	Dn	
1	Start				
2	Camera action	calib		↓	×
3	Robot action	goto_t1	↑	↓	×
4	Robot action	goto_t2	↑	↓	×
5	Robot action	goto_t3	↑	↓	×
6	Robot action	goto_t4	↑	↓	×
7	Robot action	goto_t5	↑	↓	×
8	Robot action	goto_t6	↑	↓	×
9	Other action	END	↑		
+	Add action				

Figure 5: The graphical user interface features a sequence editor (left) and a robot control panel with a live camera feed (right).

3.2 Sequence editor

The sequence editor enables the user to program a custom sequence for the Robothon grand challenge taskboard. It consists of a table-like structure, where each row represents a task. The rows also contain controls for each task.
The editor provides the following controls for each task:

- DELETE (represented by the X) - deletes the task
- UP (represented by the up arrow) - moves task one row up
- DOWN (represented by the down arrow) - moves task one row down
- COMMENT / UNCOMMENT (represented by the /) - disables or enables the task respectively

A new task can also be added by clicking on the “Add action” button.
The editor supports three types of task - Camera, Robot and Other. We can edit the type of task by double clicking on the “Type” column in the corresponding row. Any entry starting with the letter “C” will be auto corrected to “Camera action”, similarly any entry starting with “R” will be corrected to “Robot action”. Any other entry will be labeled as “Other action”.
Camera actions are actions, that require the use of the camera. Currently only the “calib” argument is supported, which executes the user coordinate system homing sequence. WARNING: Even though it is labeled as a camera action, the robot still receives commands and moves during this action.
Robot actions are actions, that run tasks on the robot. The “Argument” Column is sent to the robot as a command. This column can also be edited with a double click.

The current valid robot arguments (commands) are:

- goto_snap - moves to the first user coordinate system homing point (high over the workspace)
- goto_snap2 - moves to the second user coordinate system homing point (over the board)
- goto_snap3 - moves to the second user coordinate system homing point (over the board) WARNING: FOR HOMING SEQUENCE ONLY, MANUAL ENTRY IN SEQUENCE MAY RESULT IN DAMAGE OF ROBOT AND OTHER EQUIPMENT
- goto_t1 - executes the blue button press task

- goto_t2 - executes the key task
- goto_t3 - executes the ethernet cable task
- goto_t4 - executes the AA batteries task
- goto_t5 - executes the button cell battery task
- goto_t6 - executes the red button press task

Tasks labeled as "Other tasks" don't use the camera or the robot. Mostly these include sequence control actions such as "PAUSE" and "END".
The sequence can also be cleared from the top menu (Sequence -> Clear sequence).

3.2.1 Saving and loading sequences

Each sequence may be saved in a *.seq* file. This file is in *json* format and includes the current position, state, type and argument of each task. The save and load commands can be found in the to menu under the "File" category. Each command opens a system dialog for its respective action.

3.3 Robot control panel

The robot control panel enables us to control the execution of the sequence on the robot. The top part also gives us the live camera feed during the user coordinate system homing procedure.
The control panel features two buttons and a robot connection indicator. While the robot is connected, the indicator states so, and the "Start" button is enabled. If the robot disconnects for any reason, the button becomes disabled, and the indicator changes to "disconnected".
A user may start the loaded sequence by pressing the "Start" button. During execution, the "Start" button becomes the "Abort" button, and the "Pause" button becomes enabled. Both the "Abort" and "Pause" buttons will stop sequence execution after the currently running task is finished. If the "Pause" button is pressed, we can resume execution with the press of the "Resume" button, which replaces the "Pause" button.
Current sequence progress is displayed on the bottom with a progress bar.

3.4 Task board localisation

Description of work related to the camera: a) calibration, b) task board recognition, c) segmentation (key fob, buttons, cell battery, battery department, ethernet cable + connectors)

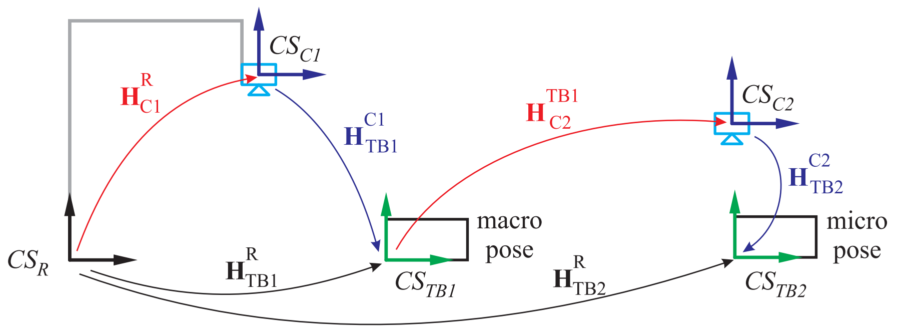


Figure 6: Transformations between objects used for task board detection with corresponding coordinate systems CS. CS_R is a robot's CS, CS_{C1} is a camera pose for detecting macro pose of task board CS_{TB1}, and CS_{C2} is a camera pose for detecting micro pose of task board CS_{TB2}

3.4.1 Camera calibration

For calibration, two fixed transformation needs to be determined:

1. transformation $\mathbf{H}_{C1}^{R}$ between camera and robot used for macro localisation of task board;
2. transformation $\mathbf{H}_{C2}^{TB}$ between camera pose for micro localisation regarding to macro pose of task board.

$$\mathbf{H}_{C1}^{R} = \mathbf{H}_{TB0}^{R}{\mathbf{H}_{TB0}^{C1}}^{-1} \tag{1}$$

where $\mathbf{H}_{TB0}^{R}$ denotes pose of the task board regarding to robot's coordinate system acquired by defining user coordinate system on the task board. $\mathbf{H}_{TB0}^{C1}$ represent the detected pose of the task board by the camera.

$$\mathbf{H}_{C2}^{TB0} = (\mathbf{H}_{C1}^{R}\mathbf{H}_{TB0}^{C1})^{-1}\mathbf{H}_{TB0}^{R}{\mathbf{H}_{TB0}^{C2}}^{-1} \tag{2}$$

3.4.2 Task board localisation

Localisation is done in two-step procedure.

1. **Macro pose**: First, a macro pose is detected regarding to recognised buttons where camera is locate approximately 600 mm above task board. Due to camera perspective, errors during calibrations (determining factor between pixels and mm), and external factors (ambient lightning), this macro pose can deviate from true pose for more than 5 mm.
2. **Micro pose**: To overcome aforementioned errors, a micro localisation was introduces where robots moves over the buttons at a distance around 100 mm. Robot is adapting its pose regarding to the error between detected position of the button and the centre of the image. With this iterative procedure sufficient accuracy was provided.

Estimation of task board's macro pose $\mathbf{H}_{TB1}^{R}$ is described with

$$\mathbf{H}_{TB1}^{R} = \mathbf{H}_{C1}^{R}\mathbf{H}_{TB1}^{C1} \tag{3}$$

Micro pose of the task board $\mathbf{H}_{TB2}^{R}$ is determined with

$$\mathbf{H}_{TB2}^{R} = \mathbf{H}_{TB1}^{R}\mathbf{H}_{C2}^{TB1}\mathbf{H}_{TB2}^{C2} \tag{4}$$

3.4.3 Features detection

Features we look for in an object are the blue and red buttons (start button and stop button), finding circles or buttons is done with a Hough transform (CHT), which returns several candidates, to find the prime candidate we used a two-step filtering. The first filtering separates candidates that do not have a nearby candidate. After the extraction, the next step is to check the colour of the surface of the circle for blue or red (looking at the HSV channels). If a candidate is either of these two, colors it is stored in the FIFO associated register (redFifo, blueFifo). Once the FIFO is filled with the sufficient data, the true position is determent from averaging all values and thus eliminating camera noise.

3.4.4 Estimating the pose of the object $\mathbf{H}_{TB1}^{C1}$ and $\mathbf{H}_{TB2}^{C1}$

To obtain more accurate transformations, it is first necessary to calibrate the camera to obtain the intrinsic parameters and the lens distortion. To estimate or find these parameters, we use the calibration with chess board pattern Figure x. The image of the pattern by was captured at different positions and angles.
The process of estimating $\mathbf{H}_{TB1}^{C1}$ is done once the positions of the buttons in the picture had been found a coordinate system was positioned on the blue button, where the x axis is pointing in the same direction as the line between the two centres of the circle, angle of the task board was calculated using the equation 5 (Figure 7, left). The coordinate system is then mapped from the camera into 3D (pixels to mm) space via known scalar

factors, and this is then moved to the task board grove (Figure 7, right). After this, a transformation (Eq. (??)) to the robot coordinate system must be performed to obtain the task board position.

$$atan2(\frac{y}{x}) \tag{5}$$

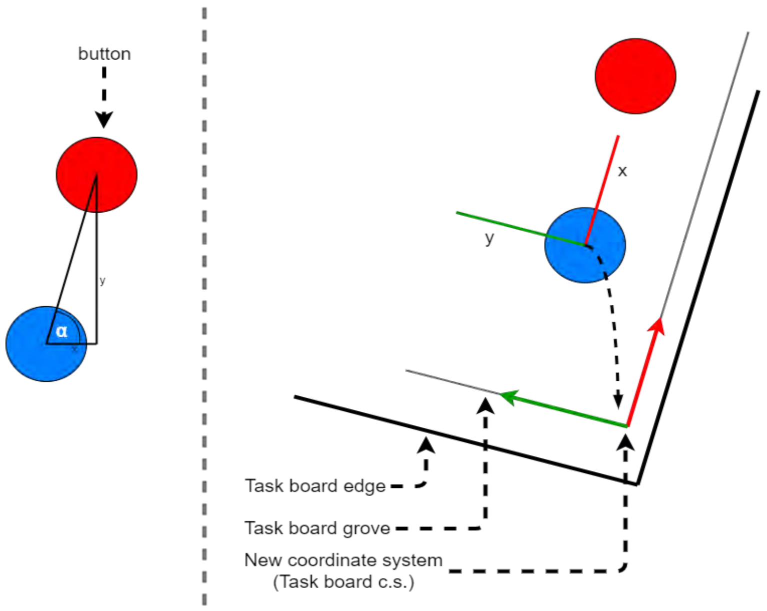


Figure 7: Angle between the blue and red buttons equal to the rotation of the task board (left); and transformation from the knob to the groove of the task board (right)

In order to increase precision and minimise error, the $\mathbf{H}^{C1}_{TB2}$ assessment is carried out in a more complex manner. Where we first position the camera 100 mm above the estimated position of the buttons, and then start aligning the position of the cameras until the blue button is at the centre of the image, we then calculate the position of the button in 3D space. The same step is repeated for the red button.The calculation of the position of the new coordinate system is done in the same way as in the first estimation (macro pose) of the coordinate system, except that different scalar factors are used instead.
The reason for this positioning (centring the button) comes from determining the exact angle and position of the task board, as we want to have an estimated rotation below $< 0.5°$ (at 300 mm the error is ± 2.5 mm) and a centre CS error < 0.5 mm. The misestimation of both is only known when inserting the key (large distance from the CS, which causes large errors due to small error rotation) and we need the highest accuracy and repeatability to perform the process successfully.

3.5 Tasks

In the following sections solutions of different tasks are described. In addition, a key features of individual tasks are highlighted.

3.5.1 Button press

Workflow for button press (Figure 8):

a) move the robot above the button;
b) approach the button within 2 mm clearance;
c) move the robot in $+z$ direction of the gripper until the measured interaction force $F_z > 5$ N;
d) move the robot away from the button.

FEATURE: Button press has no predefined final position but it is pressed using force feedback.

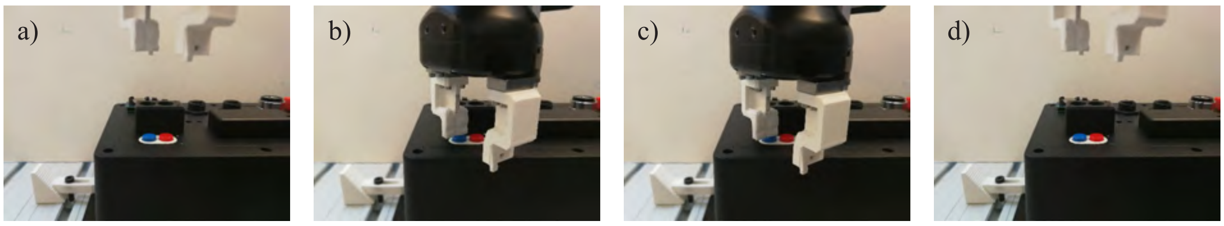


Figure 8: Pressing the blue button

3.5.2 Key manipulation

Workflow for key manipulation (Figure 9):

a) move the robot above the key in key holder;
b) move the robot to the key and grip it;
c) move the key from the key holder;
d) transfer the key to above the key switch preferably right from the keyhole;
e) to detect the upper side of the key switch, move the key in $+z$ direction of the gripper until the measured interaction force $F_z > 10$ N;
f) use the key as a force probe to detect the edge of the keyhole: move the key simultaneously in $+x$ and $+z$ direction of the gripper while maintaining constant interaction force of $F_x = F_z = 5$ N; when the interaction force exceeds $F > 7$ N, stop the movement as the key is aligned with the the edge of keyhole;
g) lift the key for 3 mm from current position;
h) rotate the key for $90°$ around tool's $+z$ axis;
i) push the key in tool's $+z$ axis until contact force $F_z = 5$ N is detected;
j) use the key as a force probe to detect the guiding pin in the keyhole: move the key simultaneously in $+x$ and $+z$ direction of the gripper while maintaining constant interaction force of $F_x = F_z = 5$ N; when the interaction force exceeds $F > 7$ N, stop the movement as the key has a contact with a guiding pin;
k) rotate the key for $-85°$ around tool's $+z$ axis; after that use force control to apply constant force in tool's $+z$ axis $F_z = 5$ N and constant torque around tool's $+z$ axis $M_z = 0.3$ Nm; when the torque is exceeded the key is aligned with the keyhole;
l) to insert the key into the key switch, move the robot in $+Z$ direction of the gripper until the measured interaction force $F_Z > 20$ N;
m) to compensate for possible misalignment, slightly open the gripper;
n) rotate the robot around $+z$ axis of the gripper until the key switch is activated (the interaction force $F_z > 12$ N);
o) rotate the key back to the initial position;
p) move the robot away from the key switch.

FEATURE: Robot uses key as a force probe to detect the edge of the key hole and the guiding pin in the keyhole and with this correct location of the keyhole.

3.5.3 Ethernet cable manipulation

Workflow for ethernet cable manipulation (Figure 10):

a) move the robot next to the cable from the left side; in this way when approached to the RJ45 connector the cable will be gently pushed aside;

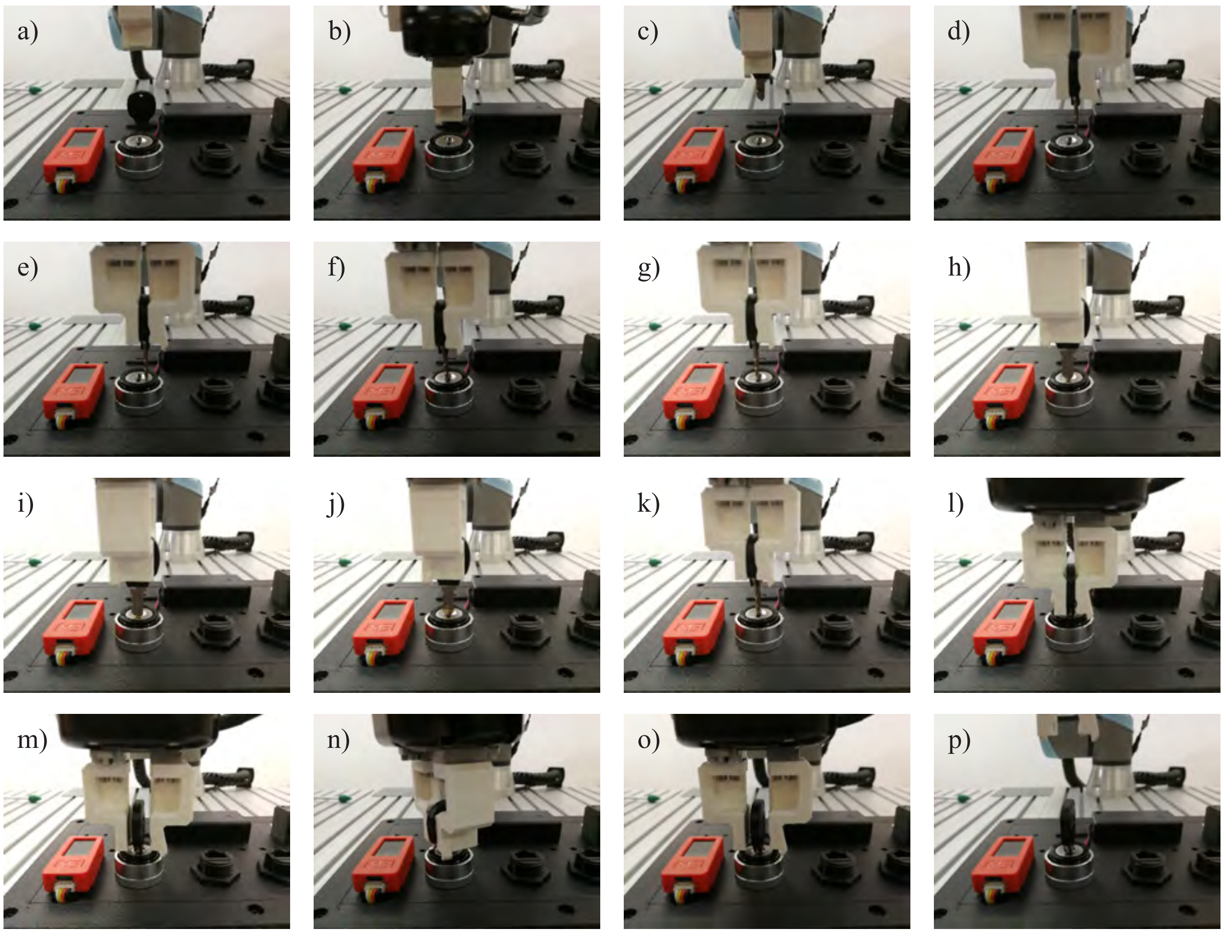


Figure 9: Key manipulation from the key holder to the key switch and activating the key switch by rotating the key to the right

b) swipe the robot to the RJ45 connector and grab it with the right side of the fingers where is the groove for the latch protection cap of the connector; in this way, the impact of the cable on the orientation of the gripped connector is minimised;
c) transfer the cable from the left socket connector to above the right one;
d) to insert the RJ45 connector into the socket, move the robot in $+z$ direction of the gripper until the measured interaction force $F_z > 10$ N;
e) open the gripper;
f) move the robot away from the right socket connector.

FEATURE: To ensure a proper connector insertion, a force feedback is used.

3.5.4 AA battery manipulation

AA battery manipulation can be segmented into four parts:

1. open the cover of the battery holder;
2. detach the battery from the battery holder;
3. manipulate batteries from the battery holder;
4. press the batteries to confirm insert.

Workflow for opening the cover (Figure 11):

a) move the robot above the battery holder;
b) tilt the gripper that only one finger will be in contact with the cover;

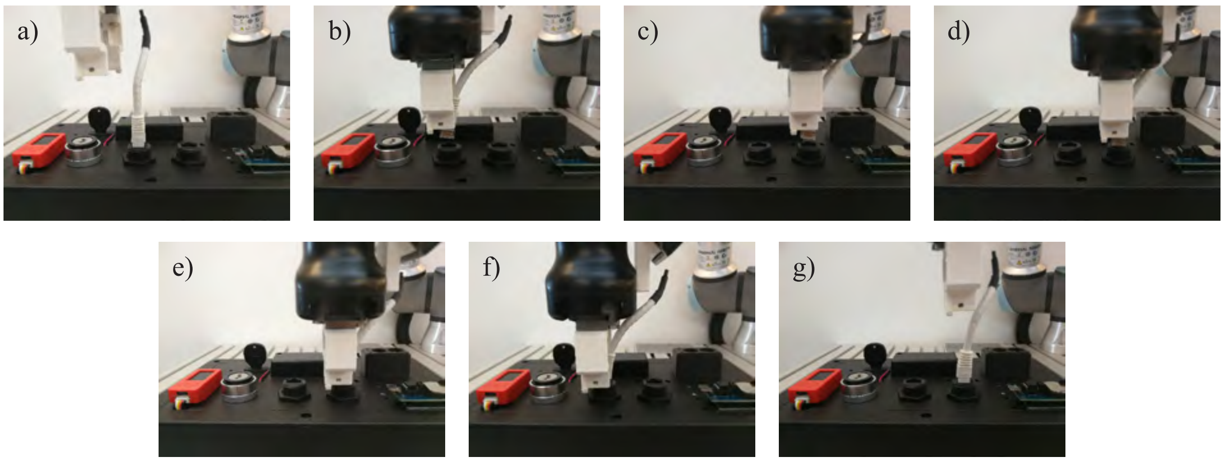


Figure 10: Ethernet cable manipulation

c) move the gripper in $+z$ direction of the gripper until the measured force is 15 N;
d) swipe the gripper while still pressing on the cover; the distance should be long enough to unhook the cover but still short enough not to hit the key;
e) lift the gripper above the cover;
f) reorient the gripper;
g) lower the gripper to grip the cover;
h) lift the cover;
i) transfer the cover on the side of the task board;
j) release the cover;
k) lift the gripper away from the cover.

FEATURE: Constant force was used to unhook and open the cover (no fixed positions were used).

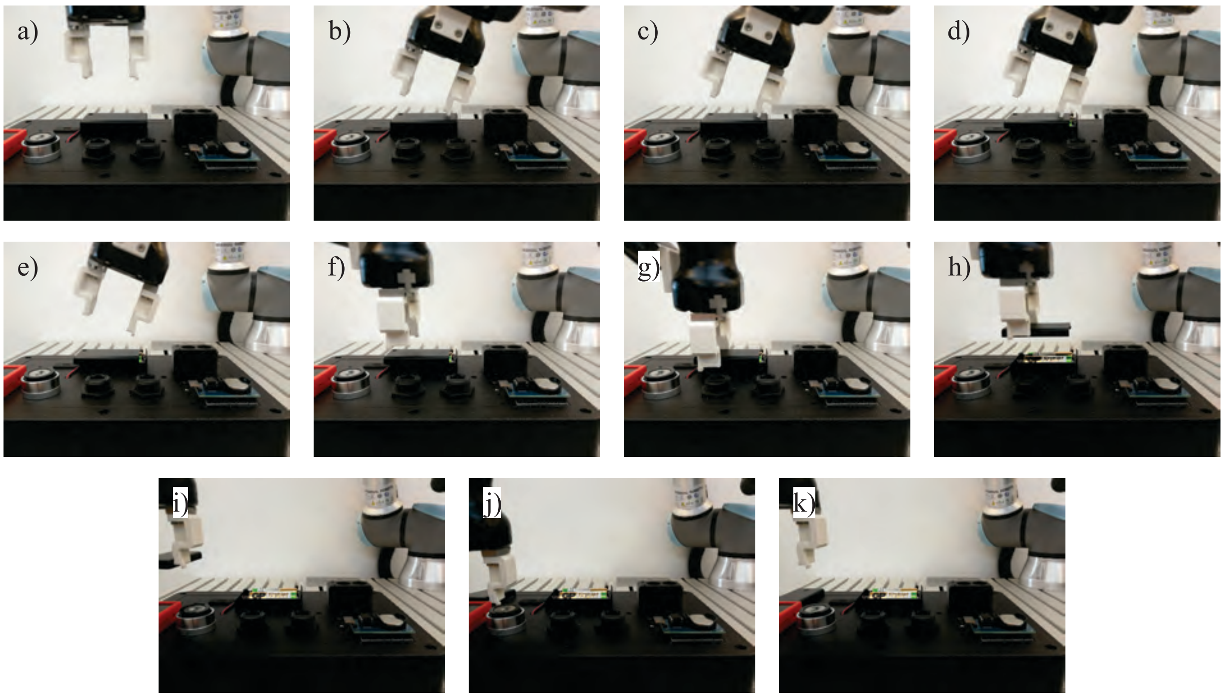


Figure 11: Open the battery cover

Workflow for detaching the the batteries from the cradle (Figure 12):

a) move the robot above the batteries and open the gripper;
b) lower the gripper until the force contact with the batteries is made (pins on the fingers are behind each battery);
c) close the gripper to tension both spring in the battery holder by pressing the batteries on each side;
d) move up with the gripper until batteries are above the edge of the cradle;
e) open the gripper for batteries to lay on the edge of the battery holder;
f) close the gripper to prepare it for battery manipulation.

FEATURE: Additional pins on the fingers enable simultaneous detachment of two batteries from the battery holder at the same time.

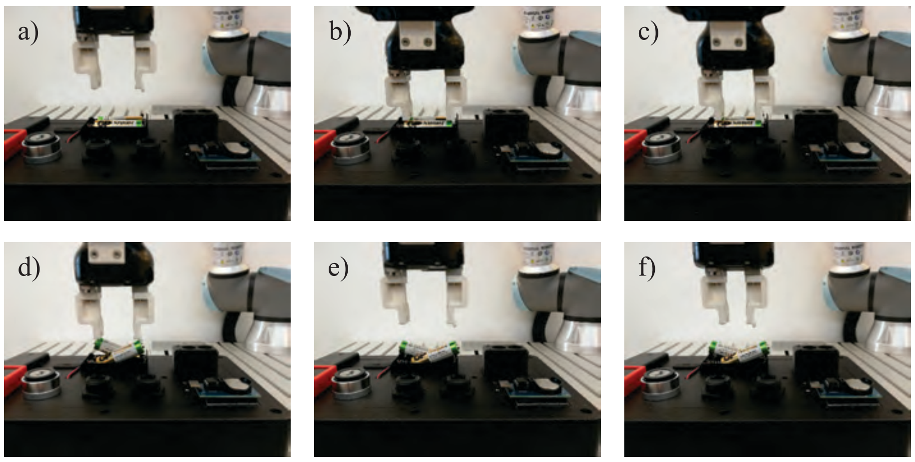


Figure 12: Detach the batteries from the battery holder.

Workflow for transferring the batteries to upright battery holder (Figure 13):
a) move the robot above one battery with opened gripper;
b) lower the gripper to the battery and grip the battery (almost) perpendicular to fingers;
c) move the battery up;
d) open a gripper for a little to allow for gravity to swing the battery down while the magnet in the finger is holding the battery and acts as an axis of rotation;
e) move the gripper in battery direction to orient the battery perpendicular to task board/parallel to fingers;
f) close the gripper and lift the battery from the battery holder;
g) transfer the battery over to the upright battery holder;
h) in case of misalignment between the battery and hole use spiral motion of the robot to find the proper insertion point (peg-in-hole solution using force sensor);
i) insert the battery into the hole;
j) open gripper;
k) move the gripper away from the battery;
l) repeat all of the above steps to transfer the second battery.

FEATURE: To compensate for misaligned grip of the battery a spiral motion of the robot in combination with force sensor is used to align battery with the hole on the battery holder.

Workflow for pressing the batteries to confirm insert (Figure 14):
a) move the robot above the batteries and close the gripper;
b) move the gripper closer to the batteries;
c) press on the batteries with 15 N force in gripper's z axis to activate buttons under them;

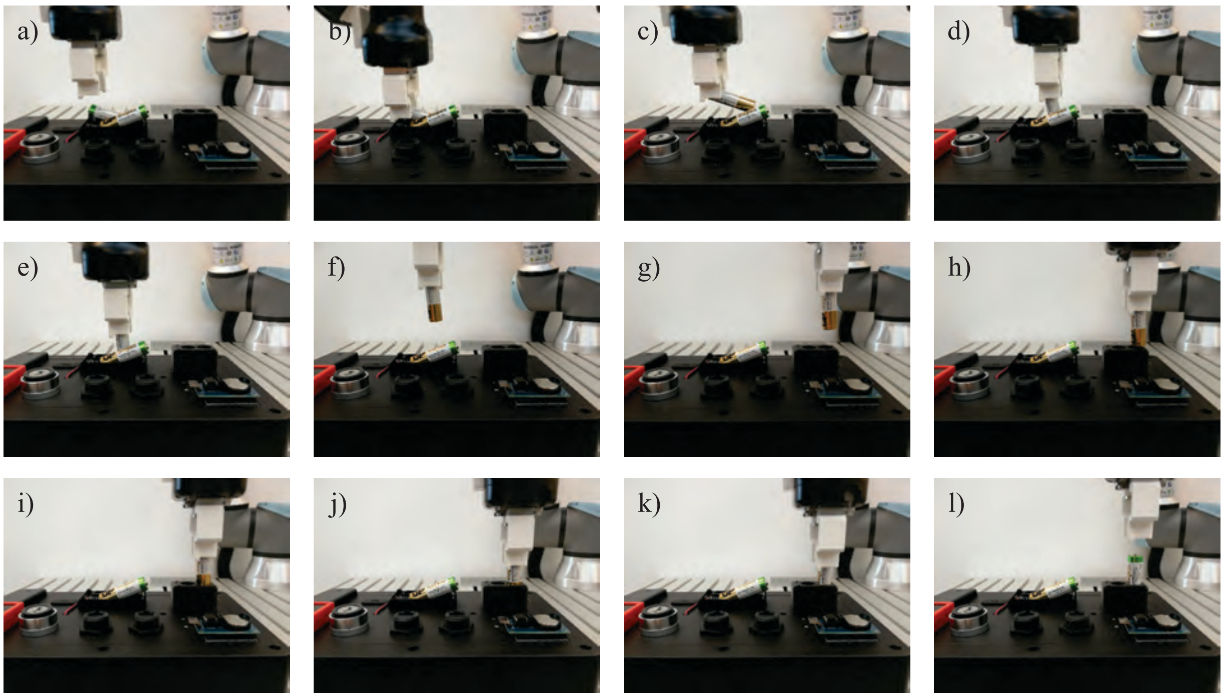


Figure 13: Transfer the battery to battery holder.

d) move the gripper away from the batteries.

FEATURE: For pressing the buttons under batteries a force controlled motion is used with no predefined points.

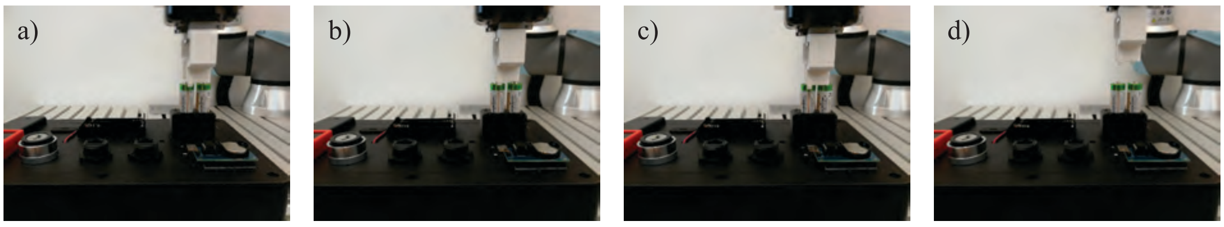


Figure 14: Press buttons with batteries.

3.5.5 Coin battery manipulation

Workflow for coin battery manipulation (Figure 15):

a) move the robot above the cell battery holder;

b) move the robot closer to the battery; then move the robot in $+z$ direction of the gripper until the measured interaction force $F_z > 5$ N to find the correct height of the battery; rotate the gripper so that the pin of the left finger is aligned with the leaf spring holding the battery;

c) press the leaf spring with the force of 10 N to release the coin battery;

d) move the robot up while preventing the battery to be thrown from the battery holder;

e) reorient the gripper for gripping the the battery;

f) grip the battery with the corners of the fingers;

g) lift the battery;

h) move the battery above the task board;

i) tilt the gripper to prevent battery to get stuck on the magnet in the left finger in case of recoil from the board;
j) release the battery;
k) move the robot away from the battery.

FEATURE: To ensure fail-safe release of the coin battery, the leaf spring is being pressed until the pressing force exceeds predefined force threshold.

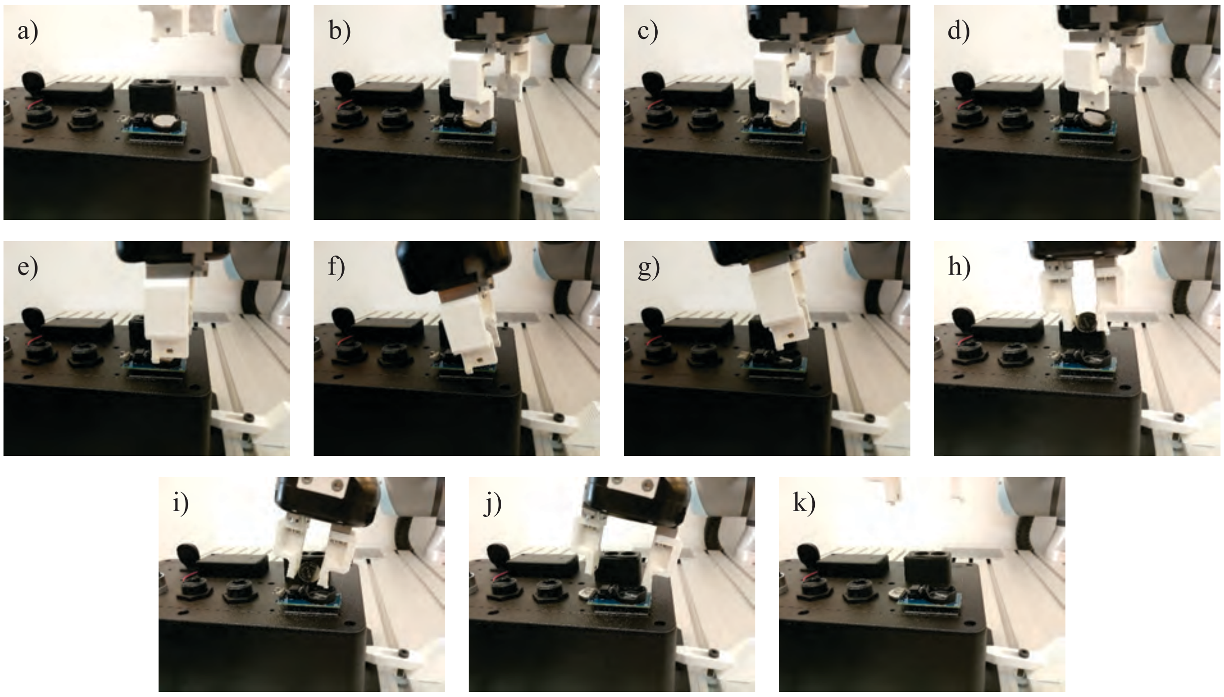


Figure 15: Coin battery manipulation

4. Quick start guide

4.1 Robot

On the robot side proper program with corresponding installation file should be loaded. In current system that file is called `main_tcp.urp`. The robot needs to be connected to network.
For the future reference we assume that the code is loaded on the github repository.

4.2 Running a fixed sequence without the GUI

4.2.1 Software requirements

To run a fixed sequence without the GUI, the following software dependencies need on the computer:

- python 3.6 or newer
- USB webcam drivers

In addition, the following python packages need to be installed in the python environment:

- numpy
- opencv-python
- scipy

4.2.2 Step by step

1. STEP: Clone the github repository and move to the src directory:

```
$ git clone https://github.com/...
$ cd src
```

2. STEP: Mount camera on robot and plug its USB cable to the computer.
3. STEP: Calibrate the camera if needed:
   Use a software like MATLAB to acquire distortion matrices for your USB camera.
   - Open MATLAB.
   - Open the "camera calibrator" app.
   - Take pictures of the calibration pattern.
   - Calibrate the camera.
   - Export the parameters as an openCV xml file.
   - Rename the file to "cameraCalib2.xml" and move it to the src directory of the repository.
4. STEP: Run the headless script:
   - In Windows:

     ```
     $ py headless.py
     ```

   - In Linux:

     ```
     $ python3 headless.py
     ```

5. STEP: Running the robot client code
   - Transfer files from the "robot" directory to the robot
   - Make sure the robot and the computer are connected to the same network
   - Set the IP address at the top of the robot program to the IP address of the computer
   - Run the code. The sequence will start immediately.

4.3 Running a custom sequence with the GUI

To run a fixed sequence without the GUI, the following software dependencies need on the computer:
- python 3.6 or newer
- USB webcam drivers

In addition, the following python packages need to be installed in the python environment:
- numpy
- opencv-python
- scipy
- pyQt5

4.3.1 Step by step

1. STEP: Clone the github repository and move to the src directory:

```
$ git clone https://github.com/...
$ cd src
```

2. STEP: Mount camera on robot and plug its USB cable to the computer.

3. STEP: Calibrate the camera if needed:
   Use a software like MATLAB to acquire distortion matrices for your USB camera.
   - Open MATLAB.
   - Open the “camera calibrator” app.
   - Take pictures of the calibration pattern.
   - Calibrate the camera.
   - Export the parameters as an openCV xml file.
   - Rename the file to “cameraCalib2.xml” and move it to the src directory of the repository.
4. STEP: Run the GUI script:
   - In Windows:

     ```
     $ py main.py
     ```

   - In Linux:

     ```
     $ python3 main.py
     ```

5. STEP: Running the robot client code:
   - Transfer files from the “robot” directory to the robot
   - Make sure the robot and the computer are connected to the same network
   - Set the IP address at the top of the robot program to the IP address of the computer
   - Run the code. The robot will wait for user input on the GUI.
6. STEP: Using the GUI to run a sample sequence:
   For details on how to operate the GUI, refer to the 3.1 Graphical user interface section.
   - Check that the robot status indicator confirms the robot has connected successfully.
   - Load sample sequence (File -> load program sequence...).
   - Edit sample sequence with the “UP”, “DOWN”, “DELETE” and “ADD” buttons.
   - Run the sequence with the “Run sequence” button on the bottom right.

5. Transferability

As a young parents, our children have the privilege to have generous grandparents that wish all the best for them. Sometimes this means that they get carried away with toys and gadgets for their young ones. The transferability challenge introduces different electronic devices that a 3-years old child comes in contact with during an ordinary day. From a radio alarm clock, car keys, TV remote controls, to toys and interactive books. With this we wanted to emphasise the problems of today’s kids world where there are electronic devices on every step.
To demonstrate the transferability of our system, we selected two toys that 3-years old child use on daily basis (see Figure 16):

a) toy robot (4 AA batteries).
b) toy keyboards that comes integrated with the book (2 AA batteries);

For battery extraction We used the same principles as for the task board:

1. force based detection of the cover with manipulation;
2. use of fingers with pins for unhooking two batteries at the same time;
3. use of magnet integrated in the finger to set the battery in upright orientation.

For the toy robot the transfer is quite straightforward; as the robot uses 4 AA batteries, the procedure needs to be duplicated. With the keyboards the batteries are switched left/right, so we could not unhook both of them simultaneously. We extracted the battery one by one, demonstrating usability of custom designed fingers.
Majority of electronic devices children have use batteries as a power source. Sadly, these toys have a large

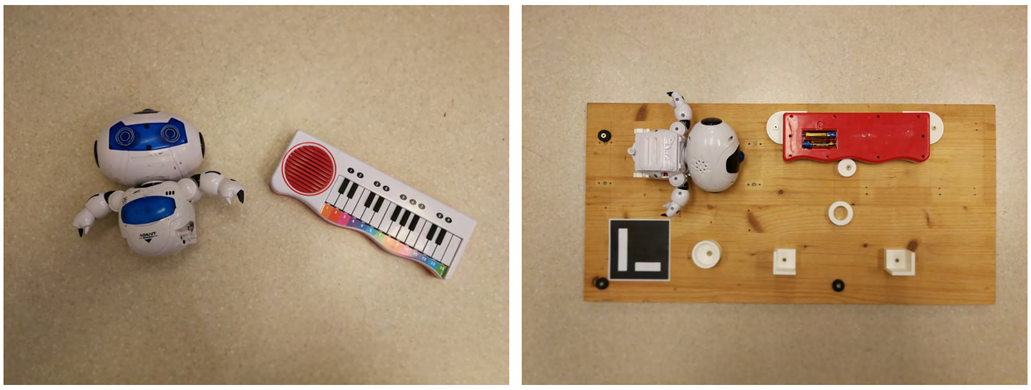


Figure 16: Two toys that children use on a daily basis: a) robot, b) keyboards as part of interactive book. On the right image placement of the items for battery extraction is shown.

potential to become future e-waste after children are tired of playing with them. Responsibility is of course on the parents' side to pass the toys on to other children or, in case that the toy is damaged beyond repair, take care of proper disposal. To further minimize toy's e-waste problem children should play with toys that come from renewable source (e.g., wooden toys) or, even better, to use the most powerful force that is available for free at every step at any time – their imagination.