5. USER GUIDE

This section describes how to get the implementation running so it can be replicated later by anyone with access to the files and required components.

5.1 Robot Cell Setup

The robot should be mounted as shown in figure 20 where the z axis of the robot is orthogonal to the working table. Pickup area and build platform may be anywhere within the reach on the same side of the robot as long as they are not close to the robot's base and the areas are positioned such that the robot's **shortest path from one area to another is not over the robot's base**. The latter requirement is for preventing the robot from moving over its base, which may put cabling and the camera in danger. On the UR teach pendant set payload to 1.07 kg (weight of the gripper and Raspberry).

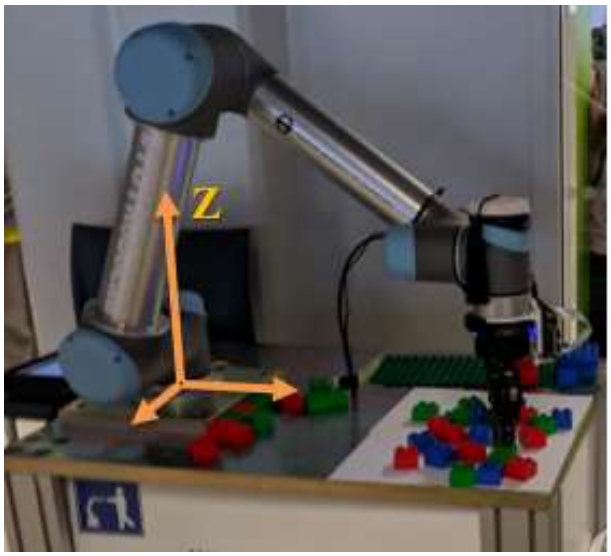
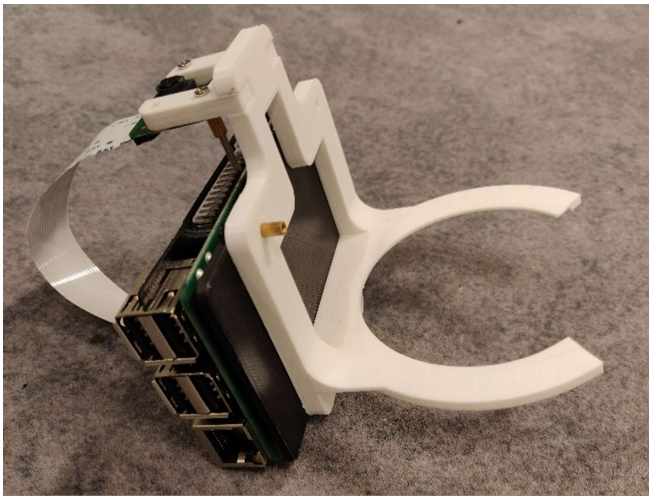


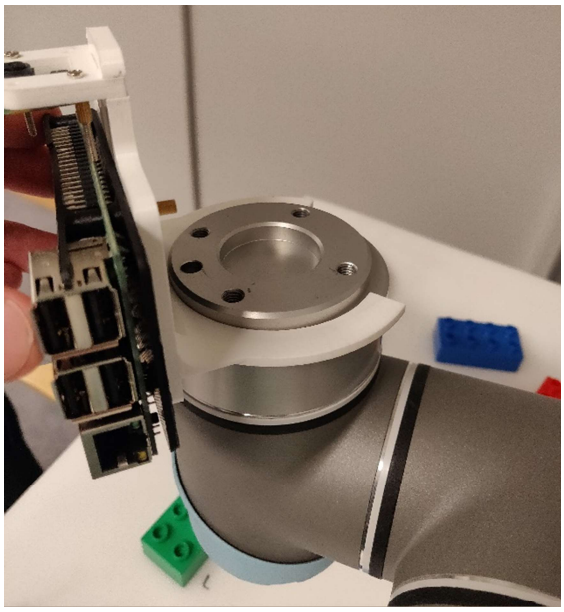
Figure 20. *UR5 mounted on a plane parallel to the working table.*

The build platform (rectangular) should be attached to the table in some way to prevent it from sliding out of position during operation. Double sided tape works well.

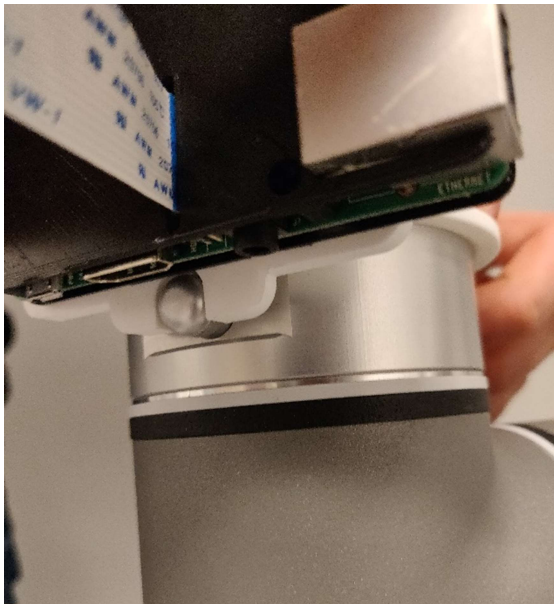
Print the 3D models and assemble the camera fixture, Raspberry Pi and the camera module (figure 21a). Place the Raspberry mount at the end of the robot arm (figure 21b) so that the tool base plate connector locks the mount in place stopping it from rotating (figure 21c). Tighten the mount in between the arm and the tool using the fitting ring. If the Robotiq gripper has Robotiq camera module installed leave the fitting ring out to make room for the fixture (figure 21d). Finally install the Robotiq gripper and the cables can be left free or attached to the arm itself (figure 21e).



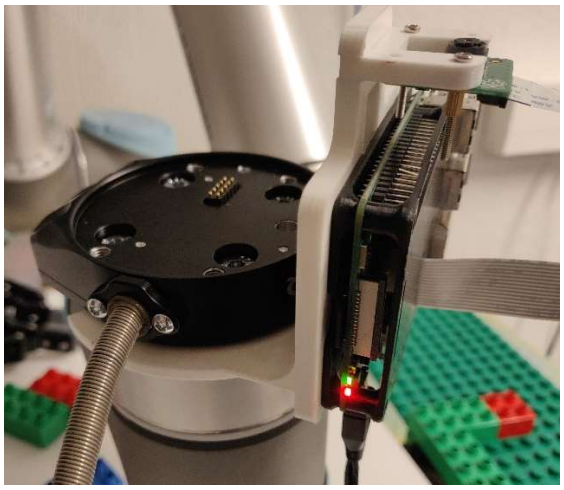
a)



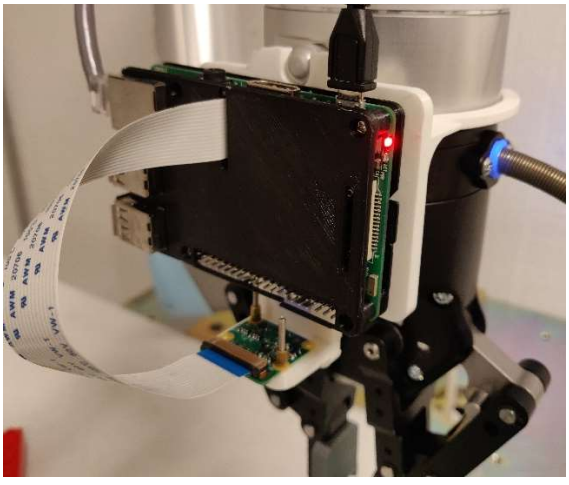
b)



c)



d)



e)

Figure 21. *Installing the mount.*

Finally, a switch or a router is recommended for connecting Raspberry, robot and host PC. Connection from the robot itself is always wired through ethernet and the robot's IP is set according to the networking settings accessible from the teach pendant. Raspberry has Wi-Fi capabilities and by taking advantage of that there is one less cable to worry about.

5.2 Software Setup

Software installation is two-part: Raspberry Pi camera server setup and host PC setup. Both systems run their programs using **Python 2.7**. **UR software version** must be higher or equal to **3.4** and lower than 4.

The file *raspberry_setup.md* describes the steps to install auto-bootable camera server and its Python dependencies on Raspberry Pi and a blank SD-card. On host PC install Python 2.7 and Python package manager PIP. Then Python dependencies can be installed by running command *pip install -r requirements_host.txt* that installs Python packages listed in the text file.

User configurable values are collected into a file named ***config.yml*** located in the root of the project folder. The file is formatted as YAML. The file contains values for

- TCP – Tool center point for gripper and camera.
- Gripper closed amount in various situations.
- Calibration color –color name for the calibration brick.
- Simulation toggle – if true the system is runnable using the official UR5-simulator.
- Travel height – Height above the platform used to travel between locations.
- 2x2 brick dimensions – Ideal dimensions of 2x2 DUPLO brick.
- Calibration data file paths.
- Model name – name of a model file to be built. The file should be in the root folder of the project.
- Camera parameters.
- Gripper definition script file path – Path to a script file used as template for activating the Robotiq gripper.
- **Network configuration** – IP and port values for UR, Raspberry and host PC.

Default values should work in most cases and the user would only need to modify the network parameters. If using router use IP addresses that the router has assigned to the connected devices. Ensure that *config.yml* is identical in both systems before continuing.

5.3 Lego Digital Designer Setup

Download and install Lego Digital Designer (LDD) from official LEGO site. Find the LEGO Company folder in ***/AppData/*** and create folder named ***UserPalettes*** under the LEGO Digital Designer. After that copy the provided ***LegoAssembler.lxf*** file to the location you just created. This is a template that provides only the supported bricks.

5.4 Creating a Model

Create a 3D model using LDD or use the provided ***demo.ldr*** file. Note that after opening LDD, choose the ***Free Build*** – mode. Then click on the ***Filter bricks by boxes*** option on the lower left-hand corner and choose ***LegoAssembler.*** This is the template that includes only the bricks that are supported.

Export your model from LDD as a LDraw (.ldr) file to the project's root folder. Name it as what was defined for the model name in *config.yml*.

As mentioned earlier there is a one good so-called rule of thumb to remember when creating the model. The robot can successfully place and build same way as a person using two fingers on one hand could. If a person fails to place the brick and the structure collapses, the robot will fail the same operation as well. This should be kept in mind when designing the model.

5.5 Running the Program

The program features five subprograms: build a model, teach platform, preview taught platform, calibrate camera and calibrate colors. Python files can be run using command *python name_of_the_file.py*. Make sure you have initialized the robot and the gripper and edited the config file before running the program.

5.5.1 Color Calibration

Text labels are attached to color signatures in a user-assisted manner. Place DUPLOs of various colors under the camera and the run the program. Running python file *do_color_calibration.py* starts the guided color calibration. First the program captures an image from the camera and then the user is prompted to select and label the bricks seen in the image. Labeling process:

1. Using 4 points the user defines quadrilateral area.
2. The points and connecting lines are drawn to the image.
3. User is prompted to give a name to the color within the area.
4. A color signature is saved, along with the given name, to a file named *color_definitions.yml*.

Previous definitions of same name are overwritten.

5.5.2 Teaching the Platform

This subprogram is used to teach where the build platform and the pickup area are. Run the main program *python start_main.py* and select the *teach platform* subprogram. Place 2x2 DUPLOs on all 4 corners of the build platform and follow instructions that are shown on the teach pendant. The teach procedure

1. Select build platform origin according to figure 22
2. Teach it by moving the gripper to grab the 2x2 brick by its studs. See figure 22
3. Teach the diagonal opposite corner keeping the tool orientation same.
4. Teach the last 2 corners in same manner.
5. Teach two corners of the pickup area (ground level).

After the subprogram has finished a file named *platform_calibr.yml* is saved in the root folder. To preview the taught points run the main program and select subprogram *preview taught platform*.

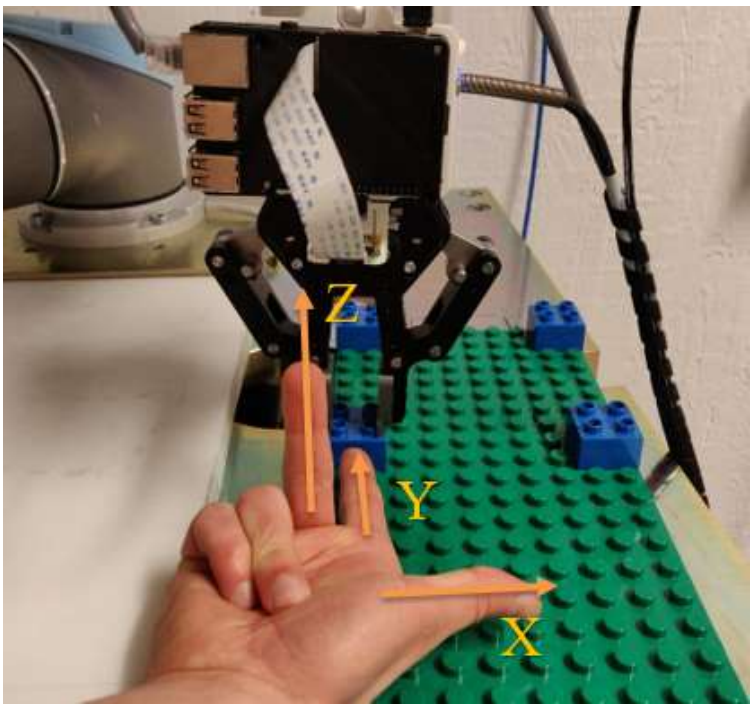


Figure 22. *Right hand rule to find the first corner during build platform teaching.*

Figure 22 demonstrates how to choose starting corner for build platform teaching. Align the gripper such that its y-axis aligns with y-axis of the right-hand coordinate frame. Keep the gripper orientation constant for all following corners. Note that in the figure only part of the dark green platform is used.

5.5.3 Calibrating the Camera

For calibration leave only the brick that was placed on the **first taught corner** (has to the color defined for calibration in config). See figure 23. During calibration the arm takes an image of the brick and uses the knowledge of its taught position to create camera calibration. Calibration file is saved to *camera_calibr.yml*. To start run the main program and select the *calibrate camera* subprogam.

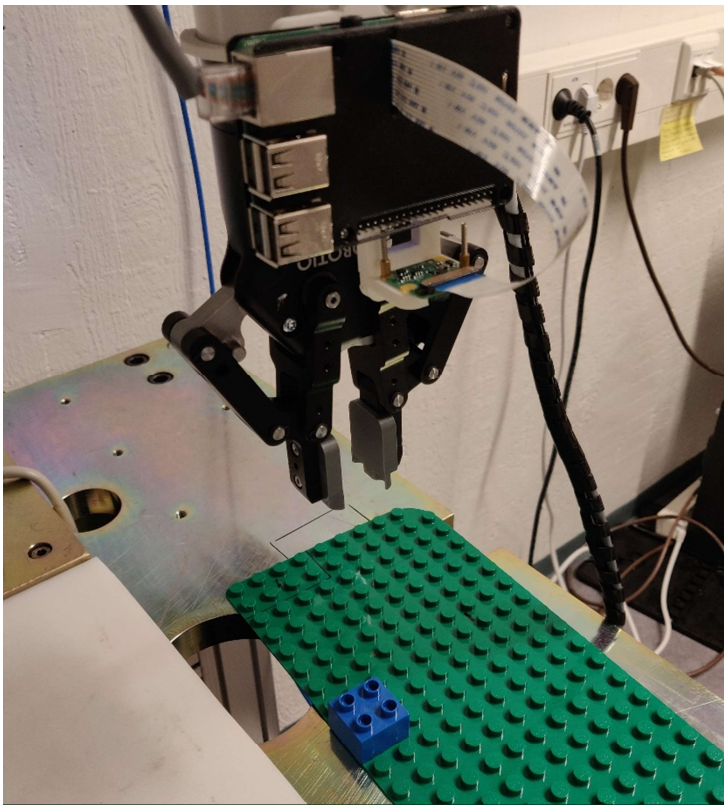


Figure 23. *Calibrating camera with one blue 2x2 DUPLO.*

5.5.4 Building the Model

Building a model requires that the user has successfully completed color calibration, platform teaching and camera calibration. The program informs user with the number of required bricks by size and color. It also displays an error message if the 3D model contains unsupported bricks.

Run the main program and select the *start building* subprogram. Before starting to build the program asks whether to continue from a saved state. If you choose to build a new one the program loads the LDraw model specified in *config.yml*. The robot starts building autonomously until all bricks are built.

Lay required bricks to the pick-up area for the robot to see them. If the robot doesn't see what it's looking for it waits until the user adds the required brick. You can see brick requirements printed to the console when the program is ran.

5.6 Troubleshooting

While running the program there might be some issues that could pop up, they could be easily avoided and if needed, repaired by applying theses simple fixes.

If there is any problem with the build of the structure, one simple problem could be that while constructing the model, the rule of thumb for the designing wasn't taken into an account, and robot physically is not able to place the block, so the solution would be going over the plans and evaluating if it is valid.

There could be situation where robot tries to grasp the block but fails, this issue could be due to not accurate enough machine-vision calibration, and in this case checking if the mount of the camera has kept the same position as it was while calibrating. Or recalibrating camera module should be carried out.

In case when robot cannot place the blocks on the platform, problem might be solved by checking if after placement of bricks, the build platform stays in place, and then recalibrating the platforms.

When there are some problems regarding machine vision, for example, finding the right blocks, color recalibration should fix the problems, it is advisable to recalibrate color every time when the lighting changes.

If robot is staying only at picturing position, make sure that all the blocks are in the pick-up area, that all necessary blocks are provided, bricks aren't too close together to each other, which could lead to problem determining blocks.

Situations when robot triggers a protective stop or a pop-up mentioning that it is close to singularity might be because one of the joints has reached its limits or robot is operating too close to base, so it must be assured that both plates are in reasonable distance from robot.

Most of problems might be caused by inaccuracies of the calibration or due to change of circumstances, so just redoing the process could resolve occurring issues.