TEAM ECOBOT

AUTHORS

HARIHARAN THANGARAJ

JESU JOHN JOLES THADEUS

SAHITH KALISETY

SRINIVASAN BALASUBRAMANIAN

TEAM ECOBOT

ROBOTHON® 2023 GRAND CHALLENGE
HOCHSCHULE SCHMALKALDEN UNIVERSITY OF APPLIED SCIENCES

Hariharan Thangaraj, Jesu John Joles Thadeus, Sahith Kalisety, Srinivasan Balasubramanian

Abstract: The workflow of the challenge focuses on the successful execution of the tasks with respect to the task board and for this purpose, the Universal Robot UR5e is selected. The robot is mounted on an aluminium table alongside the task board. The Hand-E adaptive gripper by Robotiq provides the base for the gripper and it is further upgraded for the challenge with custom-designed and 3D-printed gripper fingers and a mount for the camera. The Robotic vision part of the challenge is handled using the Intel Realsense D435i camera. The workflow is in the following order: Camera and Gripper calibration, Task board localization (micro and macro pose), feature detection & image processing, followed by the execution of the tasks in the selected order.

The trial involves the tasks of button pressing, reading data from the display & moving the slider to the corresponding position, plugging the probe into the test port, door opening and circuit probing, wrapping the cable around the given pegs and finally pressing the 'Stop Trial' button. These tasks replicate many of the real-time applications. The manipulator is programmed in such a way that all these tasks are executed sequentially in the mentioned order.

The Transferability challenge is about the application of these task execution methods and logics to identify and segregate the e-waste. Effective e-waste segregation is essential to minimize the risks associated with the disposal of electronic waste. We have selected the use case, in which the robot identifies the batteries from a discarded remote controller, checks them with a multimeter to see how much residual charge is available and segregate it as e-waste or not.

Keywords: Universal Robot UR5e, Intel Realsense D435i Camera, Robotic Vision & Image Processing, Custom Gripper.

1. INTRODUCTION

Robothon® is an international competition with 20 teams competing internationally from different universities and organisations. This global competition brings together roboticists to create transferable physical robotics manipulation skills with their robot platform towards pressing industrial challenges. This year's competition focuses on manipulation skills involving – Locating Objects, Perceiving and Manipulating Dials, Plugging in Probes, Accessing & Probing Circuits and Cleaning Up of Workspace.

The Report is divided into 4 sections-

- Hardware Components
- Software Dependencies
- Robot Tasks
- Transferability task [BYOD]
- Quick start guide

2. HARDWARE COMPONENTS

The robotic system is composed of the following components,

1. Universal Robot UR5e mounted on an aluminium table.
2. Robotiq Gripper
3. 3-D printed custom grippers
4. Intel Realsense D435i Camera
5. Taskboard
6. Workstation

The complete system is depicted in the below figure.

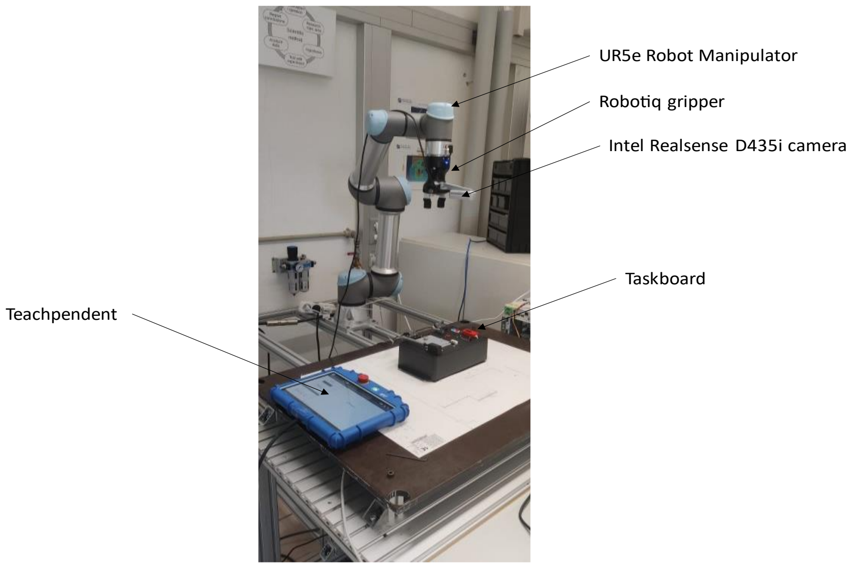

Figure 1: Manipulator setup

2.1 Universal Robot UR5e

We have selected Universal Robot's UR5e collaborative robot as the manipulator for this task especially for its flexibility and safety. It is a 6-DOF robot with a payload of 5 kg and a reach of 850 mm. Its six-axis design provides high flexibility and precision, allowing it to reach into tight spaces and perform complex manoeuvres. Additionally, the UR5e is designed to work safely alongside human operators, with built-in safety features like force sensing and collision detection.

2.2 Gripper Design

The basis of the gripper design is provided by the Hand-E adaptive gripper by Robotiq. The Hand-E's high accuracy and parallel stroke make it perfect for the precise execution of the Robothon Tasks. The gripper is then fitted with custom-designed, 3D-printed fingers and a mount for the camera for robotic vision. The Gripper comes with 'Auto Calibration' feature, means the gripper is capable of automatically defining the closed position of the fingers based on their size.

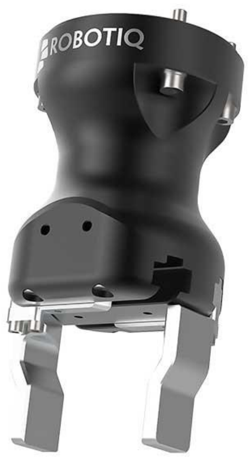

Figure 2.1: Gripper with custom-printed fingers and camera mount

2.2.1 3D-Printed Custom Gripper Fingers

The initial gripper design is developed with the classical model where the gripping surface is flat. Later, to complete the tasks efficiently, two features are implemented in the base design.

1. **Rectangular slot for Cable handling:** The gripper fingers are provided with a horizontal groove so that the probe cable can be handled efficiently and the probe wire can be stowed properly around the given pegs.
2. **Circular Slot to Hold the Magnet:** The fingertip is also given a slot feature to hold the Neodymium magnet, which assists in opening the taskboard door.

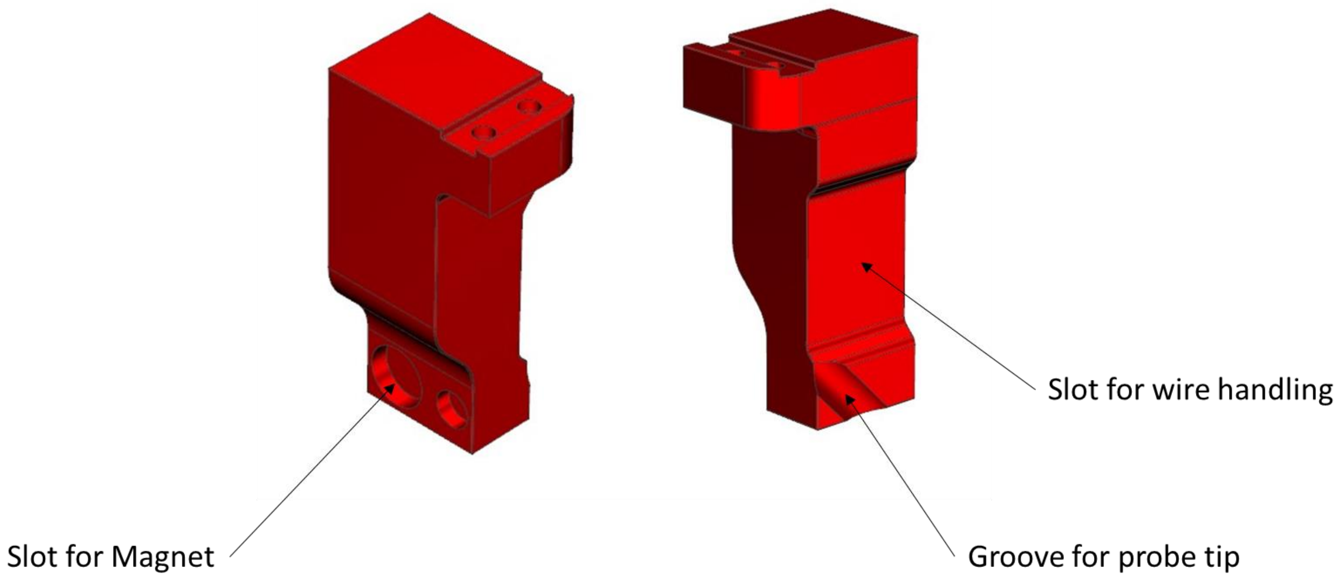

Figure 2.2: Custom finger design

2.2.2 3D-Printed Custom Camera Mount

The base gripper is then fitted with a vertically oriented, 3D-printed camera mount. This is done to orient the camera axis with that of the TCP.

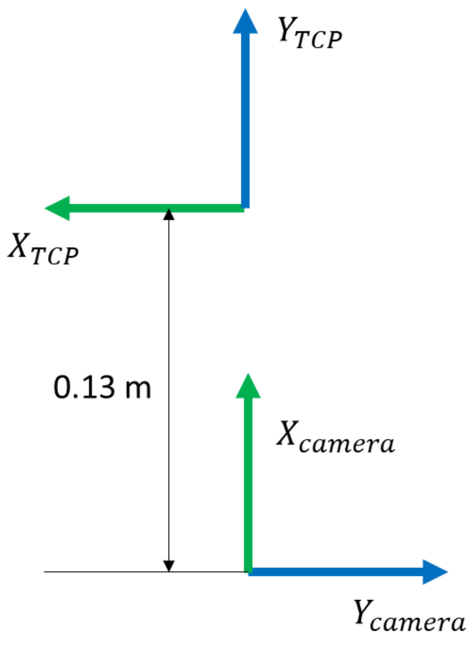

Figure 2.3: Camera offset and orientation w.r.t TCP

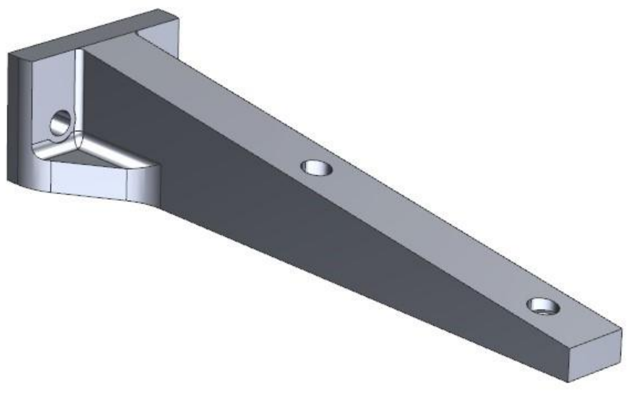

Figure 2.3: Custom camera mount design

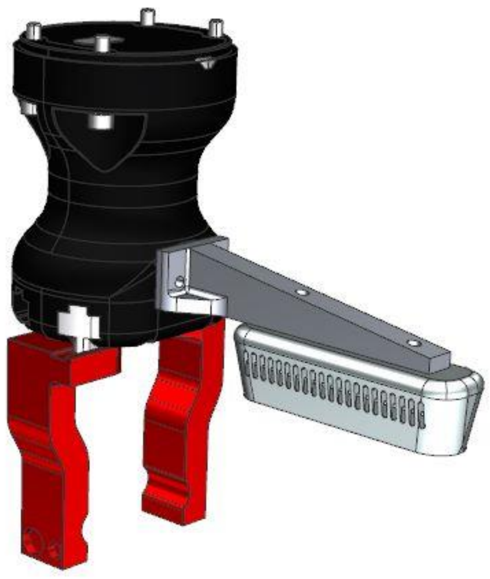

Figure 2.4: Overall gripper assembly with camera mount and fingers

2.4 Intel Realsense D435i Camera

The Robotic Vision part of the challenge is handled with the Intel Realsense D435i Camera. The camera combines a wide field of view and a global shutter sensor, making it a preferred solution for robotic navigation and object recognition applications. The camera has a 1920×1080 @30fps sensor with a 69° × 42° field of view. It also comes with integrated depth-sensing hardware and software. The camera is mounted onto the vertically oriented, custom 3D-printed camera mount.

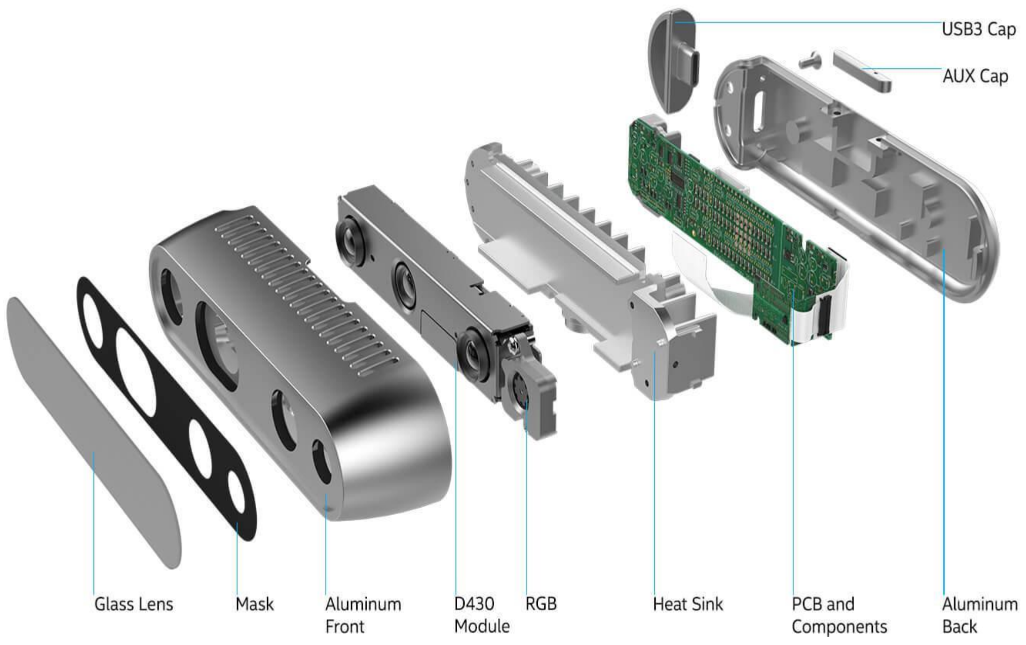

Figure 2.5: Exploded view of the camera sensor

2.5 Robothon Taskboard

The taskboard provided by the Robothon team replicates many of the real-time applications. It mainly focuses on the following tasks:

1. Object detection
2. Perceiving setpoint data from the screen and moving the slider to the corresponding position
3. Probing the circuit and
4. Wrapping the cable around the pegs.

The manipulator is programmed in such a way that these tasks are executed sequentially in the order mentioned.

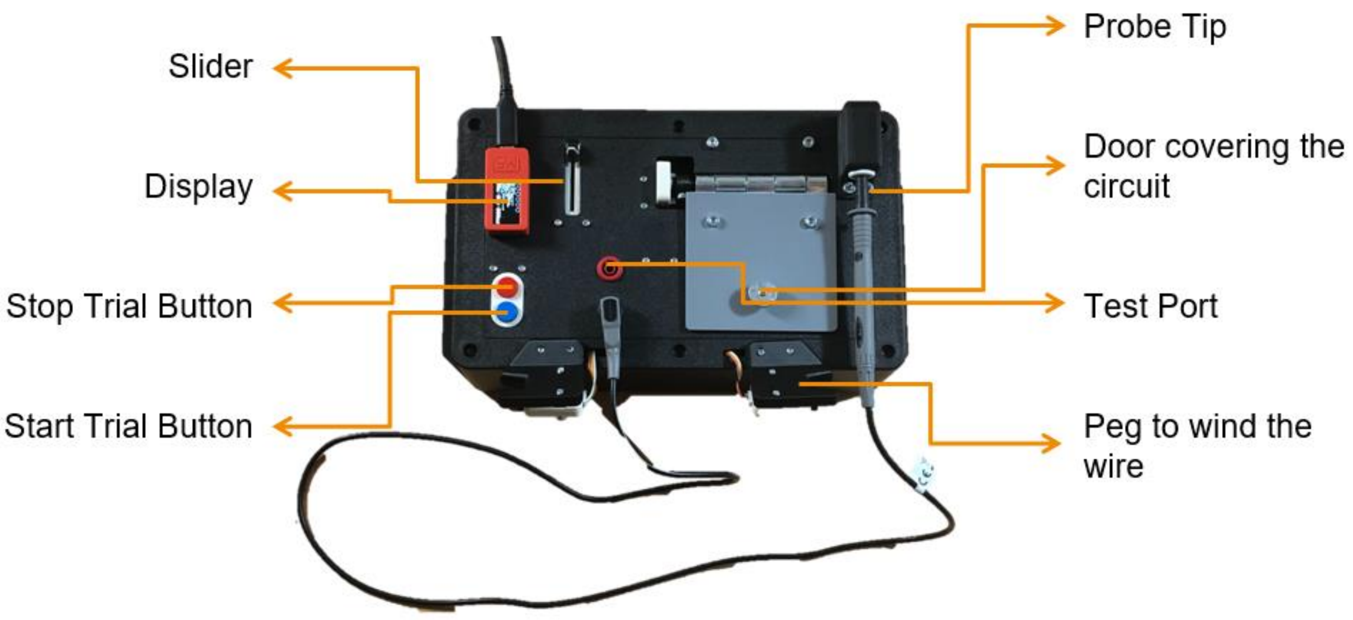

Figure 2.6: Taskboard

3. SOFTWARE DEPENDENCIES

The solution to the challenge is mainly dependent on the following software,

- **PolyScope 5.11** - Universal Robots controller software
- **Intel Realsense SDK** – for Camera calibration
- **Python 3.10.10 with the following libraries**
    - **OpenCV** – for Image processing and object detection
    - **Tensorflow** – for Digit detection
    - **Ur-rtde** – for Robot control
    - **pyrealsense2** – for Camera control

- **NumPy** – for Computation

- **NX-CAD** – Design and modelling
- **Z-Suite** – Creating custom G and M codes for 3D Printing.
- **Zortrax M-300** – 3D printer with feed material Z - ABS

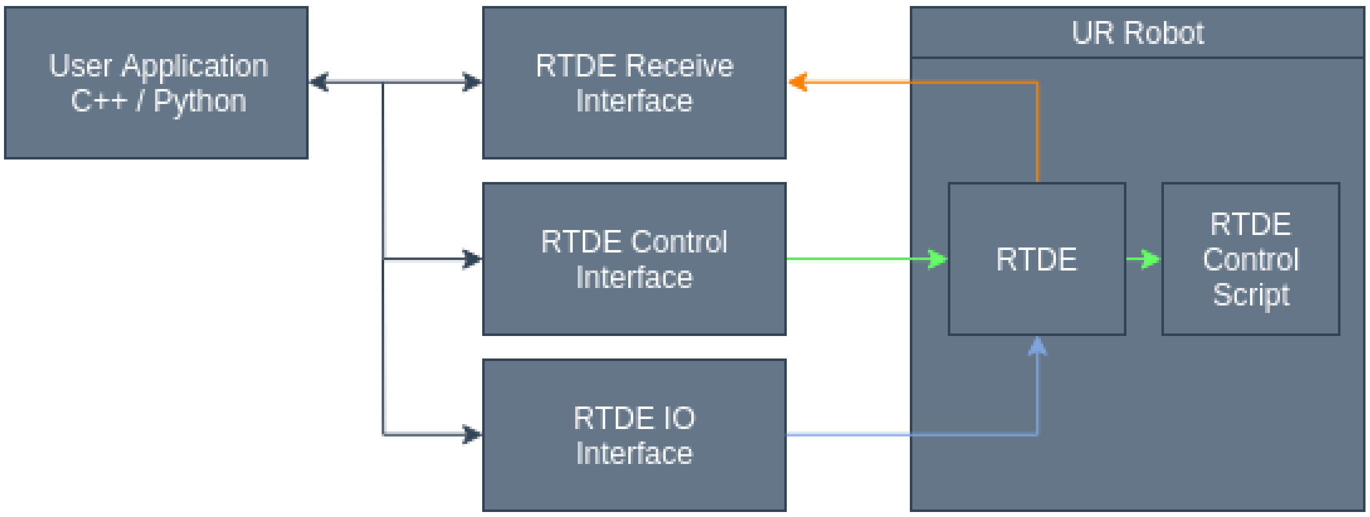

Figure 3.1: Communication architecture with RTDE library

The communication between the Workstation and the robot controller is established using the TCP/IP Protocol. After the initial handshake with the server, task-specific subprograms are called from the main program. The following are the task-specific subprograms,

a. **realsense_camera.py:** To initialize the realsense camera and to capture the depth and colour information.
b. **robotiq_gripper.py:** To calibrate and control the gripper fingers.
c. **Vision_Program.py:** Board Localization; to identify the red button centre and M5 button centre.
d. **ProjectionPlane.py**: To convert the pixel coordinates to world coordinates.
e. **MicroZoom.py:** To identify the red and blue button centres and calculate the orientation of the task board.
f. **Triangle_Class.py and main_triangle_class.py**: To identify the coloured triangles from the display and calculate the distance between them.
g. **RPYtoRV.py:** To convert roll, pitch and yaw values of the robot to rotational vectors.

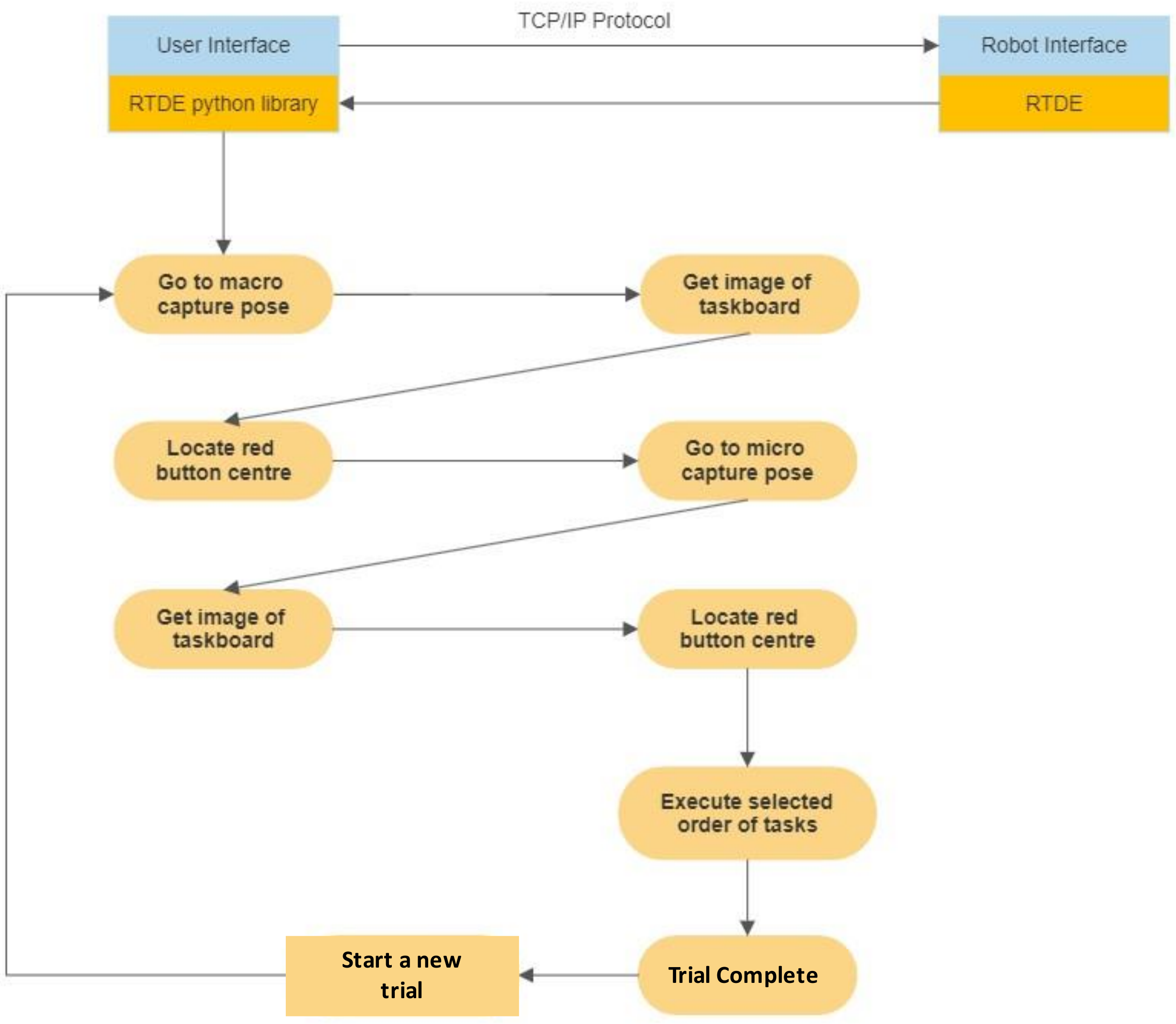

Figure 3.2: Software architecture in the form of a flow chart

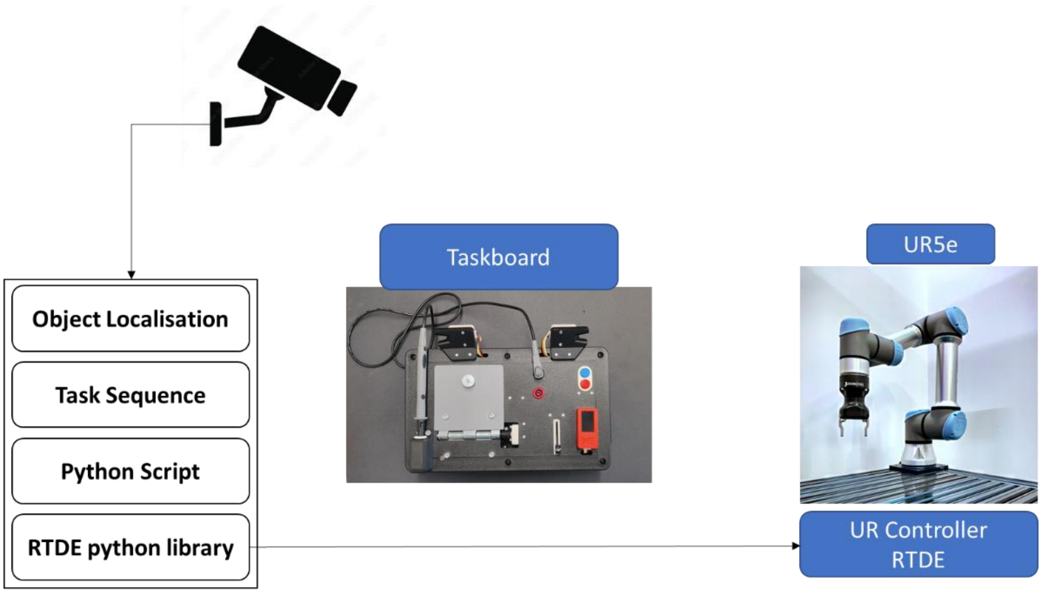

Figure 3.3: General architecture of the proposed solution

3.1 Task board localisation

For accurate and precise robotic manipulation of objects, the relationship between the positions of a robotic arm's end-effector (hand) and its camera (eye) in three-dimensional space must be determined. This Hand-eye calibration helps in reducing the risk of errors. Camera calibration is required to ascertain the intrinsic camera parameters, such as the focal length, principal point, and lens distortion. A calibration tool available in the **RealSense SDK** is used to calibrate the camera.

An image of the Task Board is captured using the Realsense Camera. The Realsense SDK then uses the '**Depth Projection Technique**' to create a depth map from the image collected. This depth map holds information about how far away each pixel in the image is from the camera.

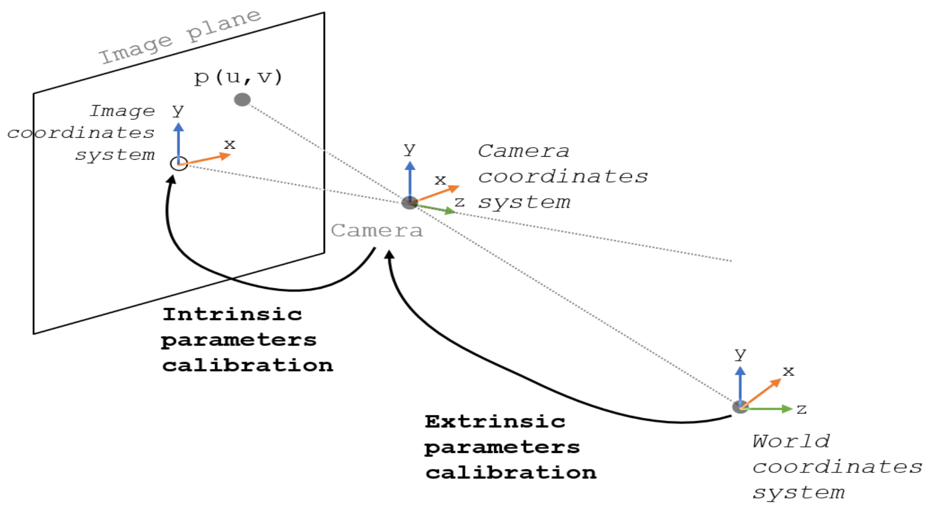
Image plane
p(u,v)
Image coordinates system
y
x
y
x
z
Camera coordinates system
Camera
Intrinsic parameters calibration
y
x
z
Extrinsic parameters calibration
World coordinates system

Figure 3.3: Depth projection technique

Taskboard localisation using the camera can be divided into the following steps.

a) Camera calibration

Camera calibration is carried out using **Intel realsense SDK**. For intel realsense camera, calibration is done to get the depth and colour intrinsic. Steps for calibration are provided by Intel and a link is provided below for reference. In addition to camera calibration, depth and colour frames are aligned to each other to eliminate projection differences between them.

https://dev.intelrealsense.com/docs/calibration

b) Macro capture pose

The camera is positioned at a height of 550 mm above the task board to detect the red button. The depth and colour frames are obtained from the camera in the macro pose. Then the red and M5 button pixel centres are detected using OpenCV and NumPy libraries. The red button centre in robot coordinates is calculated from the red button pixel centre using the '**depth projection technique**' using the camera.

Due to the camera perspective, errors are inevitable, and the actual location of the button can vary from the true pose so another position estimation is necessary for accuracy.

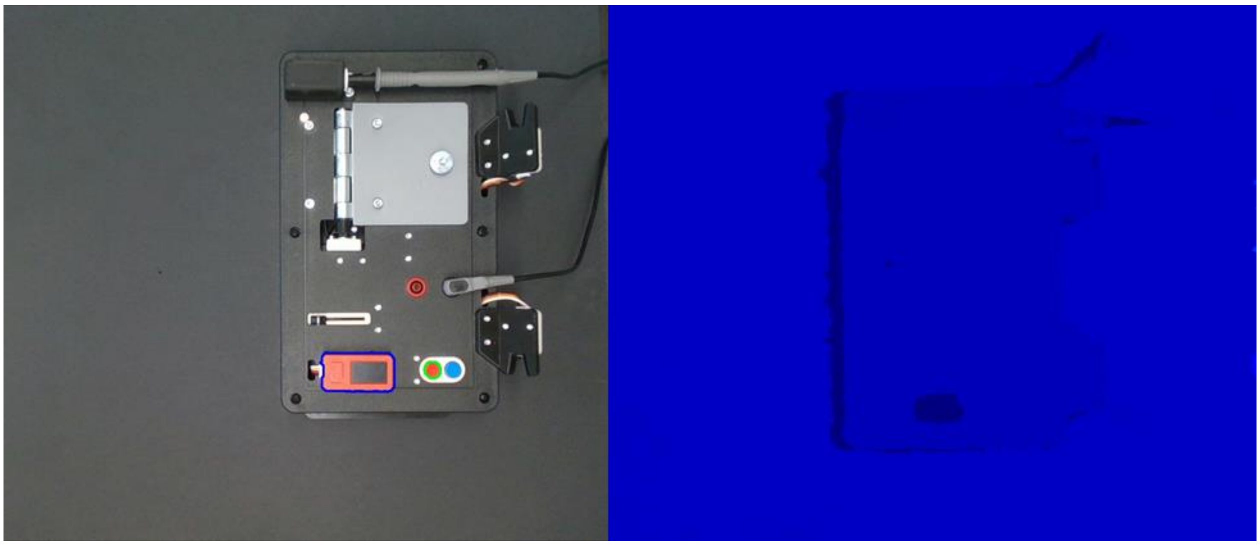

Figure 3.4: Macro capture pose with RGB and depth frame

c) Micro capture pose

To eliminate the errors, micro localisation is carried out where the robot is moved in such a way that the camera centre is aligned with the red button and positioned at a height of about 250 mm from the robot's origin. The red button centre is calculated again to minimise the errors in the red button position. This allows us to get the true location of the red button in the world/robot coordinates. Then using geometric and trigonometric relations, the location of other features is calculated with respect to the red button centre.

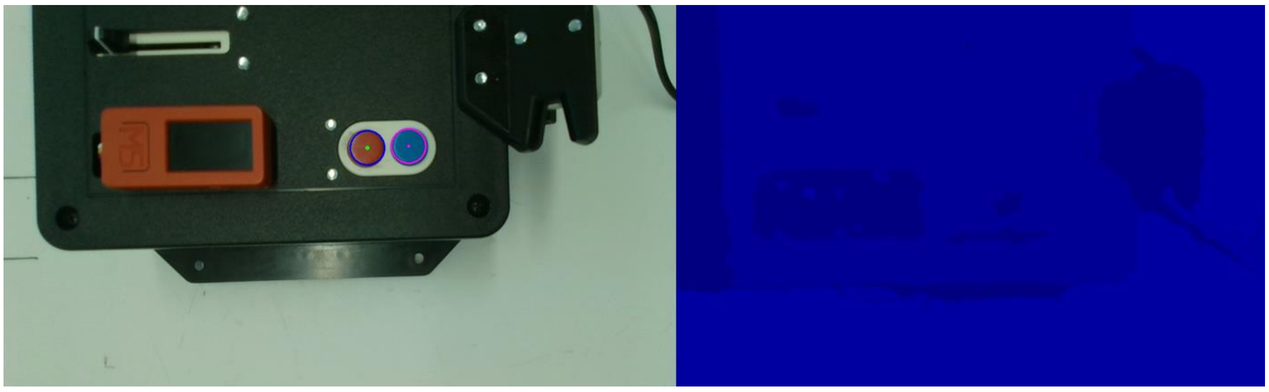

Figure 3.5: Micro capture pose with RGB and depth frame

d) Rotation angle computation algorithm

During the process of Taskboard Localization, the red button centre in the robot coordinates is calculated from the red button pixel centre (u_1, v_1). In the same way, the blue button pixel centre (u_2, v_2) is also calculated using image processing techniques. The orientation (θ) in which the taskboard is positioned is calculated from the identified points using mathematical and geometric methods as shown in Figure 3.4.

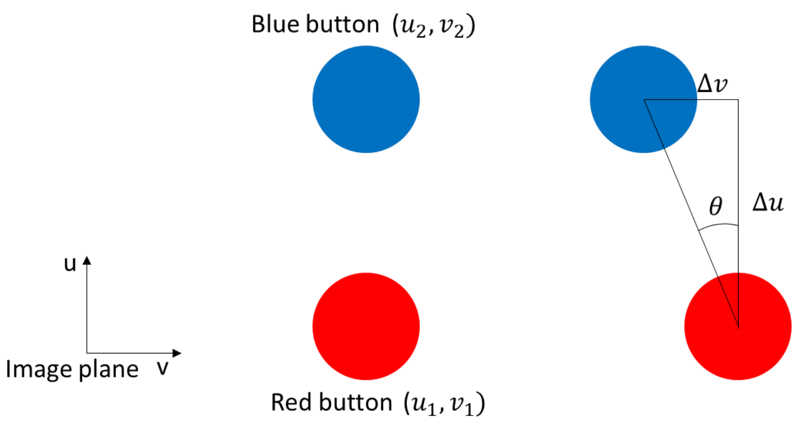

Figure 3.6: Determination of taskboard orientation

$$\theta = \tan^{-1}\left(\frac{v_2 - v_1}{u_2 - u_1}\right)$$

e) Slider arrow detection algorithm

Initially, the position of the red arrow (current position of the slider) with respect to the yellow arrow (first target position for the slider) in the display is read using the camera, by moving the robot to 'capture position'. By using image processing techniques, the distance between these two arrows is calculated in "pixels". A "scaling factor" is calculated by dividing this pixel length by the screen length in pixels, which is further used to find the actual distance by which the slider needs to be moved. Once this is done, the same technique is iterated for the green arrow (second target position for the slider) and the distance to be moved is calculated.

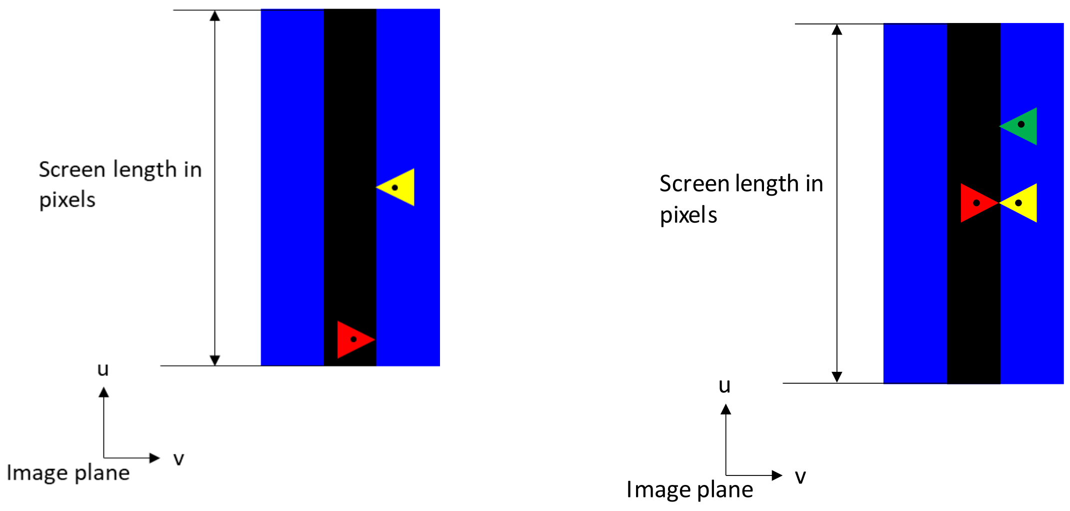

Figure 3.7: Algorithm for detecting the slider arrow

$$Yellow\ arrow\ centre\ = (u_2, v_2)$$
$$Red\ arrow\ centre\ = (u_1, v_1)$$
$$distance\ to\ be\ moved\ \ = u_1 - u_2$$
$$Slider\ motion\ range\ in\ mm = l_1$$
$$Screen\ length\ in\ pixels = p_1$$
$$slider\ displacement\ = (u_1 - u_2)\frac{l_1}{p_1}$$

$$Green\ arrow\ centre\ = (u_3, v_3)$$
$$Red\ arrow\ centre\ = (u_1, v_1)$$
$$distance\ to\ be\ moved\ \ = u_1 - u_3$$
$$Slider\ motion\ range\ in\ mm = l_1$$
$$Screen\ length\ in\ pixels = p_1$$
$$slider\ displacement\ = (u_1 - u_3)\frac{l_1}{p_1}$$

4. ROBOT TASKS

The solutions to various tasks are presented below.

Task 1: Button Pressing

1. The blue button centre is calculated from the red button using trigonometry.
2. The robot is first moved above the centre of the blue button with the gripper closed.
3. The robot is then lowered in the -z-direction to press the blue button on the task board.
4. After pressing the blue button, the gripper is moved above 100 mm in the z-direction above the taskboard.

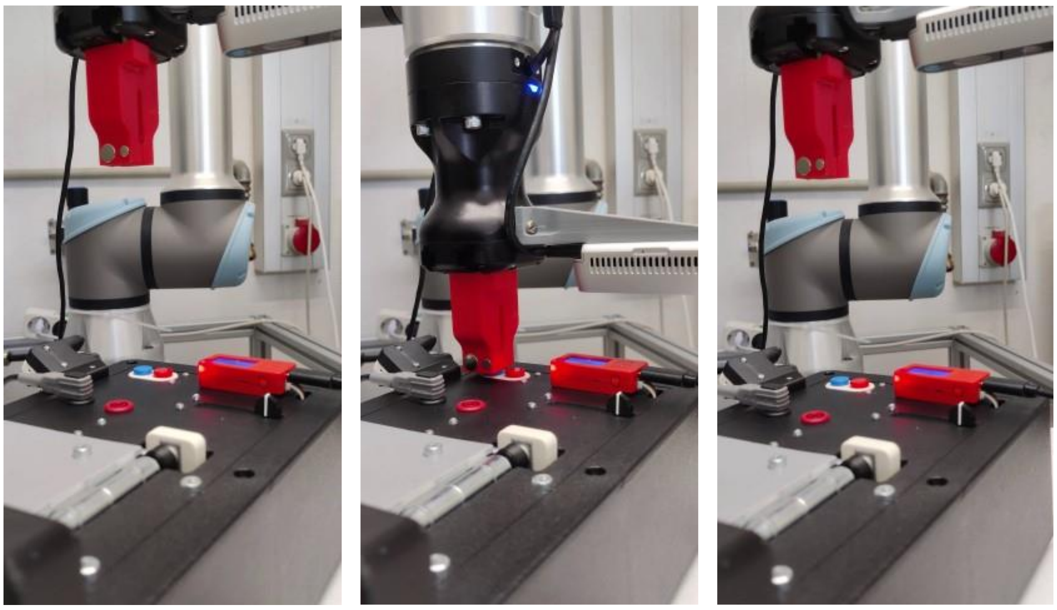

Figure 4.1: Blue button actuation

Task 2: Slider Motion

1. The robot is moved to the ‘capture position’ to read the display.
2. From the display, the yellow arrow position is detected and the relative distance to move the slider is computed using the positions of the red and yellow arrows.
3. The robot is moved to the start position of the slider.

4. It is then lowered, and the gripper is closed to clasp the slider and then moved to match the red arrow with the yellow one.
5. Gripper is opened, and the robot is once again moved to the capture position to read the display for the green arrow.
6. Relative distance between the red and green arrows is calculated.
7. Robot is moved to the current slider position and the gripper is closed.
8. It is then moved to match the red arrow with the green arrow to complete the task.

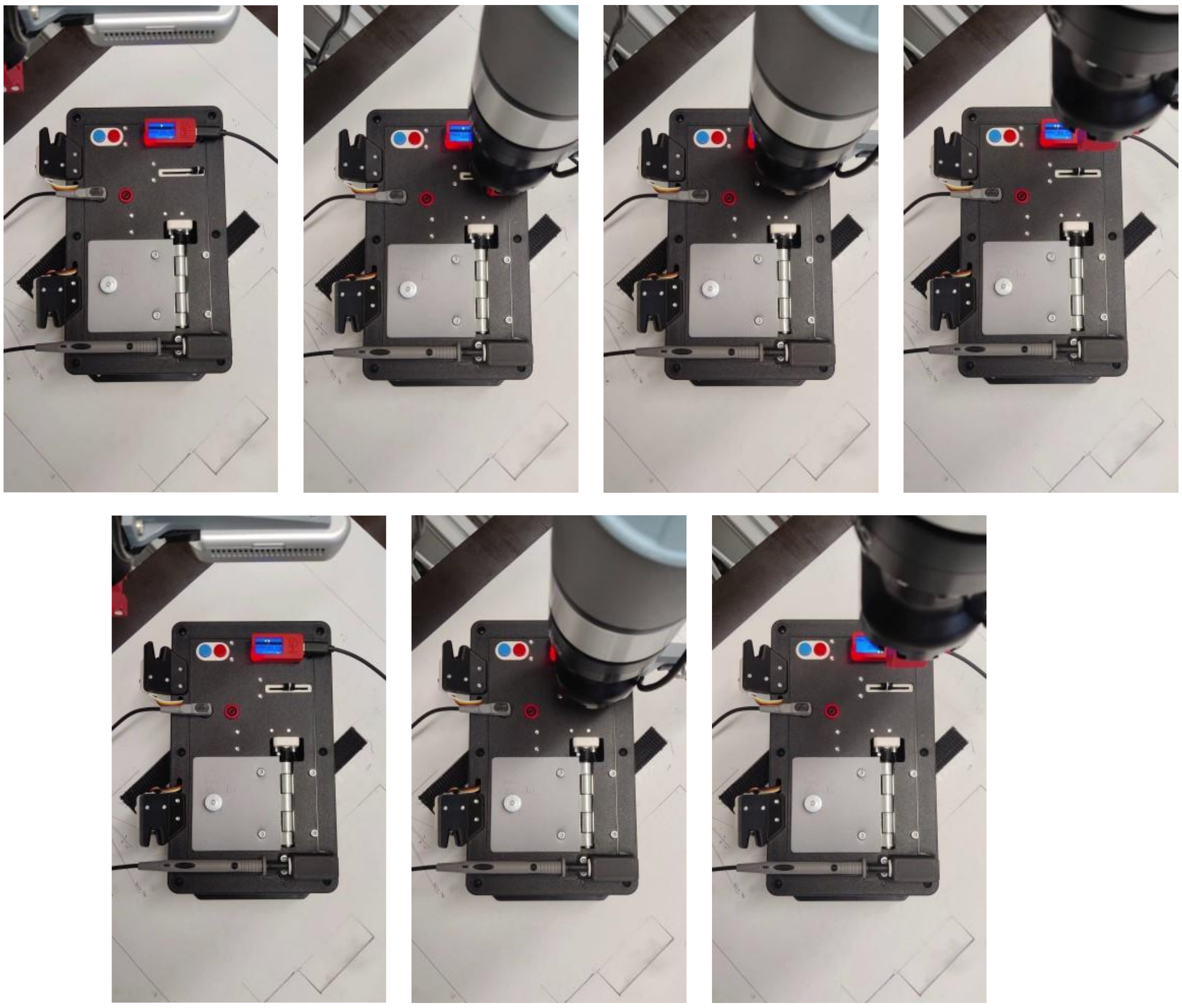
Figure 4.2: Slider motion

Task 3: Plugging Probe into Test Port

1. The location of the black test port and the red test port is calculated from the red button centre using trigonometric relations.
2. Gripper is opened and the robot is moved above the black test port.
3. The robot is lowered, the gripper is then closed, and the robot picks up the probe from the black test port.
4. Next, the robot is moved to the red test port location and the probe is inserted into it.

5. Gripper is then opened, and the robot is moved in the z-direction away from the taskboard.

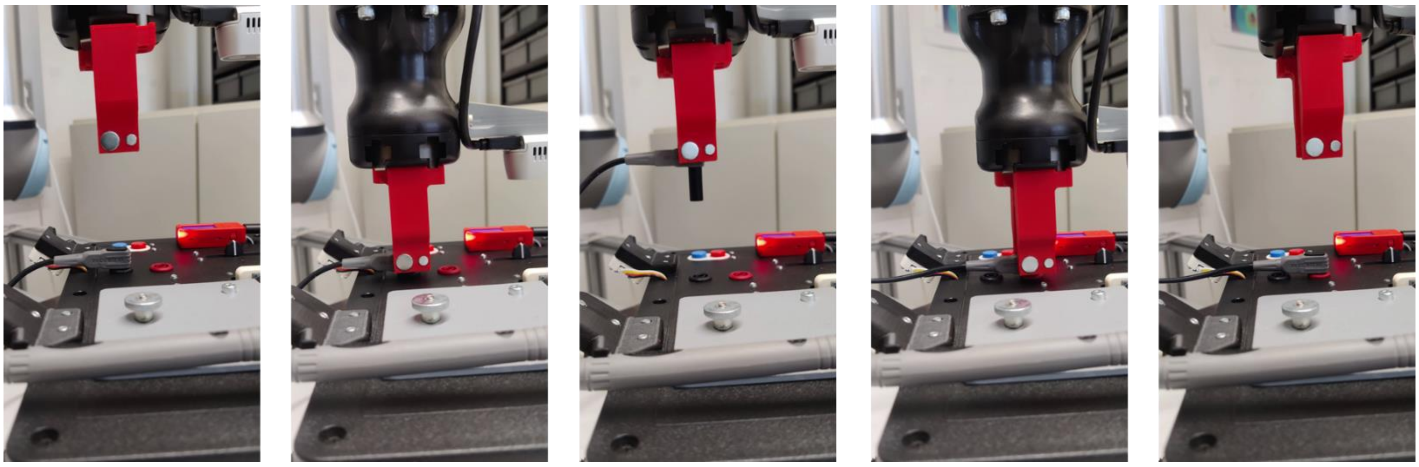
Figure 4.3: Probe insertion into the test port

Task 4: Door Opening and Probe Circuit

This task can be subdivided into 2 parts: 4a) Door opening 4b) Probing the circuit

4.a) Door Opening:

1. The robot is moved near the doorknob such that knob is attracted by the magnetic field.
2. Robot is then moved in a circular path at a slower speed to open the door so that the doorknob does not slip away from the magnetic field.
3. Robot then moves in front of the door and moves to open the door furthermore
4. After that robot is moved to the probe pick-up position.

4.b) Probing the circuit:

1. Robot is positioned at the probe pick-up point.
2. RPY value of (45,0,180) is applied to the robot TCP to achieve the probe pick-up orientation.
3. It is then lowered to probe level with the gripper open.
4. Gripper is then closed and the robot is moved in y-direction to remove the probe tip from the slot.
5. After that it is again moved up and a pitch of -180° is applied to orient the probe in the z-direction.
6. Robot is moved to the probing point and then lowered in the z-direction to probe the circuit.
7. After that robot moves away from the task board and places the probe on the work table at a predefined position.

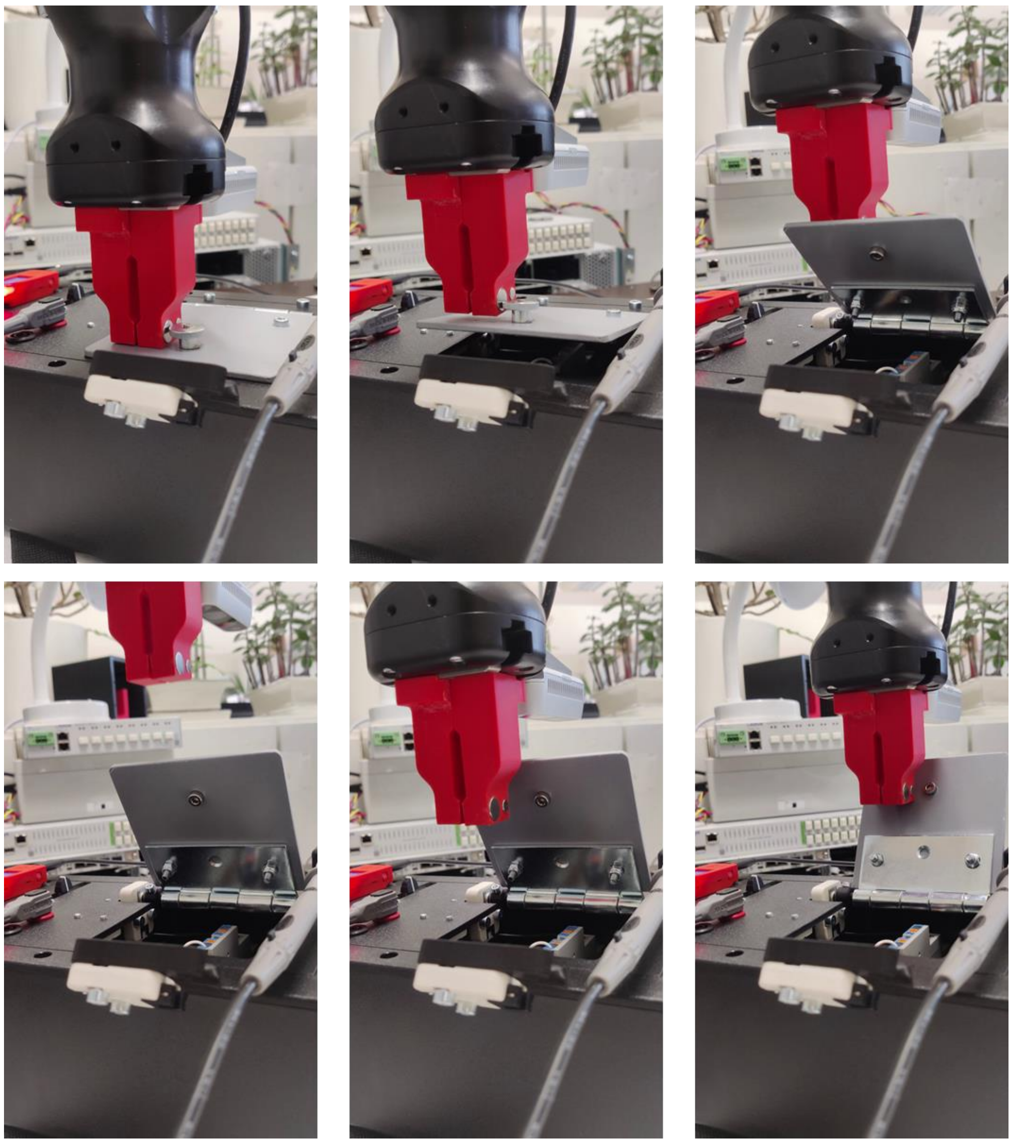
Figure 4.4: Door opening

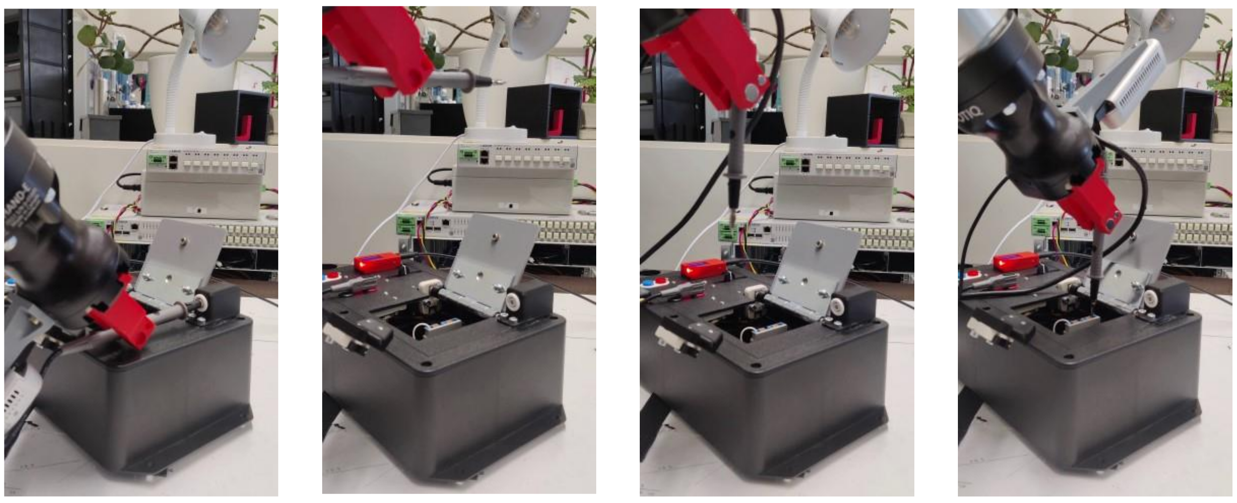
Figure 4.5: Circuit probing

Task 5: Cable Wrapping and Replacing Probe

This task can be subdivided into 2 parts: 5a) Wire Pickup 5b) Wire Stowing

<u>5.a) Wire Pickup</u>

1. Robot is moved to the probe wire approach position.
2. 45 deg roll is applied to the robot TCP to rotate the robot to pick up the wire with grippers open.
3. Robot is moved to the wire pick up point and gripper is closed to pick up the wire.

<u>5.b) Wire Stowing</u>

1. Robot is moved through a sequence of waypoints such that the wire is wrapped around the pegs.
2. After the wire is stowed around the pegs, the robot is moved to the image capture position to detect the probe tip on the table.
3. Once the probe tip is located, the robot is moved down to pick up the object from the table.
4. After reaching the probe, the gripper is closed and the robot is moved to the probe insertion point and the probe is inserted.

5. TRANSFERABILITY DEMO:

As a team, our main focus concerning the transferability challenge was to brainstorm an idea that enables us to incorporate the Robothon tasks in the automation of E-waste processing. We realised that in recent years, the increasing use of batteries in electronic devices, particularly rechargeable batteries in smartphones, laptops, and other portable devices, has contributed significantly to the growing e-waste problem. What's even more dangerous is a charged battery ending up in a landfill. It could create a fire risk, if there is a short circuit between the 2 terminals. So we are obligated to take up this opportuniy to come up with a solution to ensure proper handling of a battery as an e-waste.

<u>PROPOSED SOLUTION: -</u>

Idea → Detection of residual charges in AA batteries and sorting them using robot

The transferability demo done by the robot is carried out in 2 stages:

a) Removal of batteries from an electronic device and placing them into the probing slot.

b) Probing of batteries to determine the residual charges using a multimeter and then sorting them accordingly.

a. <u>Battery removal by robot:</u>

- The e-waste device (in this demo, a remote) is placed onto the fixture.
- Robot is moved to the image capture position and detects the battery slot using the camera
- The battery removal tool is picked up by the robot from the fixture nearby and utilised to loosen the batteries from their slots in the device.

- The batteries removed from the remote are then picked up by the robot.
- The removed batteries are then placed into the battery holder one by one.

b. <u>Probing of battery:</u>

- The battery holder is designed in such a way that one of the multimeter probes is already inserted at its bottom.
- The other probe tip is picked up from a stand and the other end of the battery is probed.
- The value displayed on the multimeter is read by the camera using CNN MNIST algorithm and an output is given based on the reading using a GUI.
- The probe is inserted back into the stand after the process
- The battery is then picked up and placed in the appropriate box, which is sent for e-waste handling.

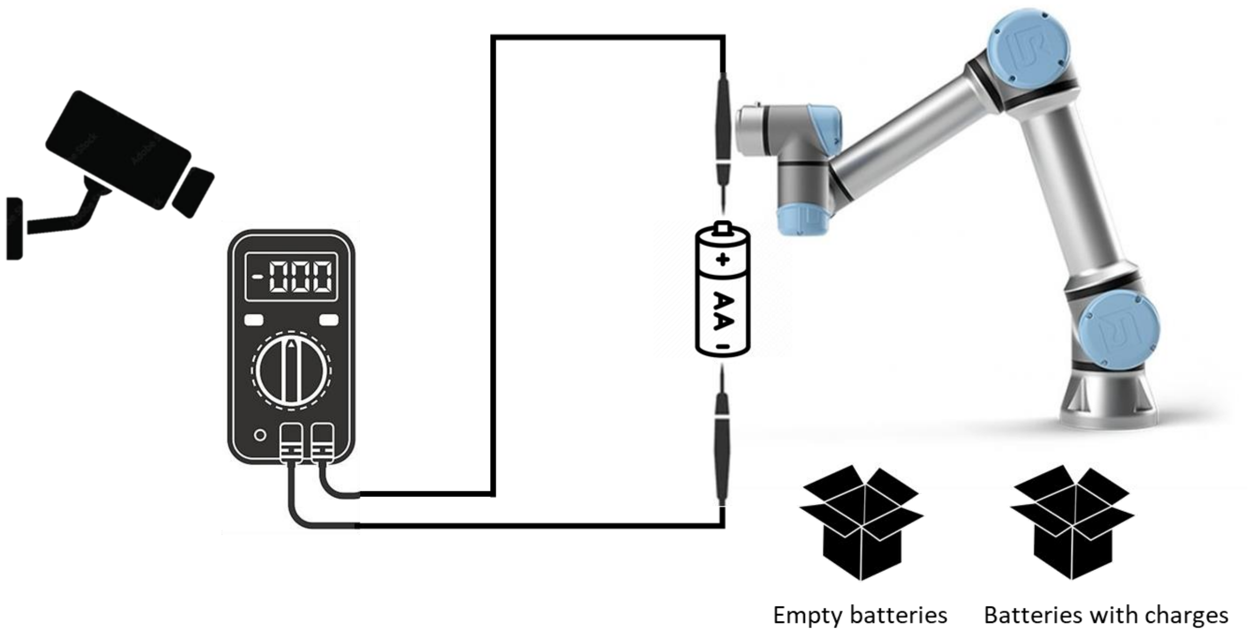

Figure 5.1: Process layout – Transferability challenge

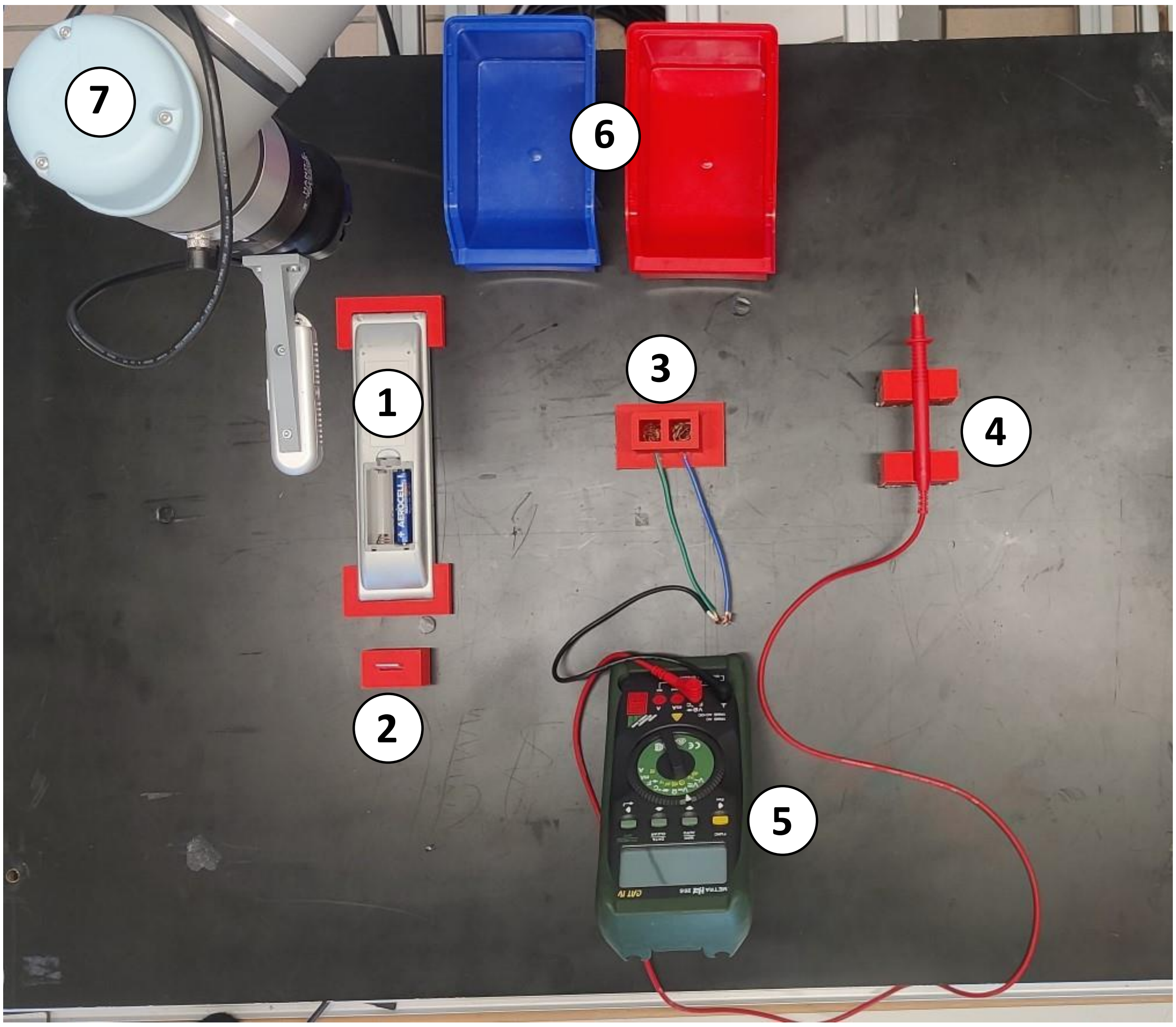

Figure 5.2: Task Setup – Transferability challenge

The following components were used in the demonstration of transferability challenge:

1. Device mounted on a fixture
2. Battery removal tool placed on a fixture
3. Battery holder for probing
4. Multimeter probe placed on a stand
5. Multimeter
6. Collecting bins for batteries
7. UR5e robot with Intel Realsense D435i camera

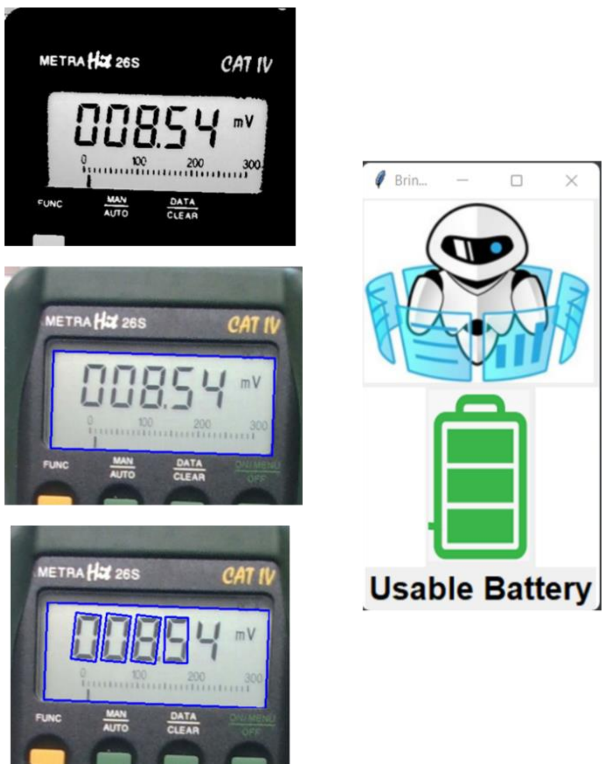

Figure 5.2: Multimeter Digit Detection Using Tensorflow For A Battery With Residual Charge

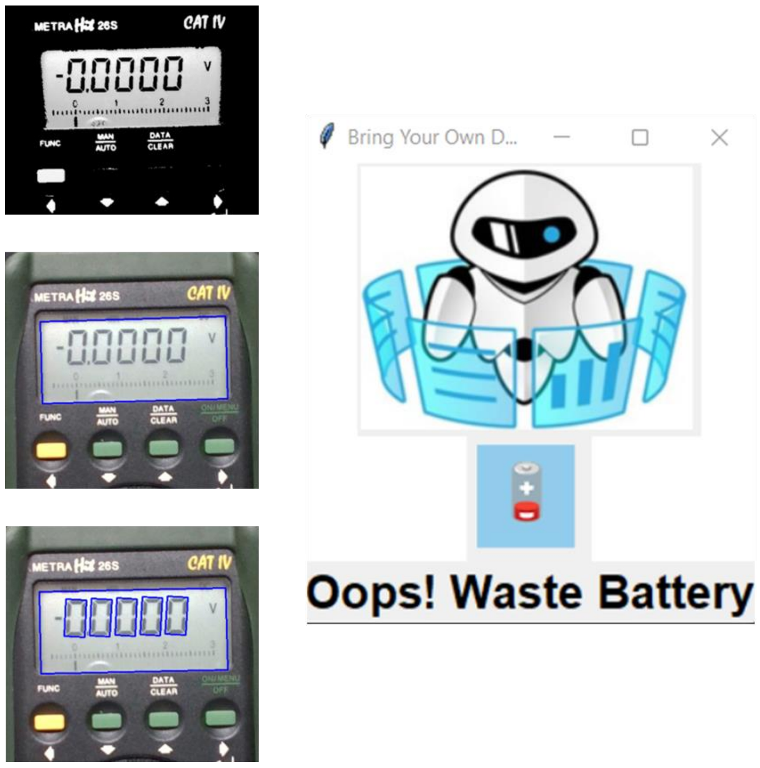

Figure 5.3: Multimeter Digit Detection Using Tensorflow For A Dead Battery

6. QUICK START GUIDE:

1. Turn on the robot controller and bring the robot to the camera pose for capturing the task board.
2. Connect to robot IP using wireless/wired connection and switch to remote control mode in teach pendent.
3. Open Visual Studio Code and load the workspace where the main Python program is present.
4. Position the task board sturdily on the workbench using the velcro strips and connect the camera and task board to the workstation.
5. Choose the order of the execution of tasks by rearranging the task calls in the main program.
6. Run the Python script.
   - a. Feature and orientation detection of the taskboard using the camera.
     - i. To begin with, the robot is positioned in the macro capture pose where the rough location of the red button is detected on the image pixel plane using image processing techniques.
     - ii. Detected pixel point is de-projected into the world coordinates using rs2_deproject_pixel_to_point(...) function in pyrealsense2 library.
     - iii. Robot moves to the micro capture pose - the camera frame origin is located directly above the red button at a height of 250 mm and the red button is detected again to get an accurate location.
     - iv. Pixel point is again de-projected into the world coordinate frame to get the red button centre.
     - v. All the features in the taskboard such as the blue button, slider location, and probe test port are calculated from the red button centre using mathematical and geometric methods.
   - b. The robot will start the trial by pushing the blue button and continue to execute the tasks in the order specified.
   - **c.** Once the task execution is completed, the robot will finish the trial by pushing the red button.