ENMT482 Assignment 2

Dominic McNicholas, Oliver Handforth, and Linus Ritchie, Group 307

1 Introduction

This report details the design, simulation, and completion of an automated espresso procedure using a Universal Robots (UR5e) programmed with the RoboDK Python API. The objective is an end-to-end procedure that spans grinding, distribution, tamping, extraction, and cup handling. The project uses a Mazzer grinder doser, a PUQpress automatic tamper, and a Rancilio Silvia single boiler espresso machine.

Three interchangeable end effectors could be attached: A Mazzer push tool for tasks like actuating the grinder doser lever and pushing buttons. A cup tool to move paper cups and serve to the user. And a modified portafilter for the extraction of the coffee. Each of these tools is attached to the pneumatic UR5e tool flange for secure and repeatable handling. The process includes two digital espresso scales. One scale is used to verify the grind dose, and the other is used to measure the amount of coffee dispensed. The UR5e's 6-DoF kinematics and ± 0.03 mm pose repeatability allow for accurate tool-centred interactions in each appliance.

Detailed methodology includes the development of frame-and-pose transformations, path planning combining joint and linear motions, and automatic generation of circular and linear motion paths. These features combine to complete manipulations such as tool insertion and button pressing. The report further discusses program optimisation, system feasibility, and practical considerations around robot performance compared to manual operation.

2 Frames and poses

2.1 Automatic pose generation

Early in the project, it was clear that calculating transforms manually for each desired pose would be time-consuming and make iteration during testing not possible. To rectify this, a script was written that automatically calculates the transform required to position any of the tools in any of the given local frames/local frame points. [1]. The script does this using a `pandas` DataFrame initially loaded from the Excel sheet `points.xlsx`. `points.xlsx` contains the coordinates of every frame outlined in the points sheet. Every frame in `points.xlsx` that has the `global` and `origin` attributes is considered the 'parent' frame of its associated appliance. The steps to compute the rotation for each parent frame are as follows:

1. Identify all points defined in the global frame that share the same `fixture`. These are used to define the orientation of the frame.
2. For each of these points, find their corresponding coordinates in the local frame. This leaves two sets of vectors that describe the same points, one set in the global frame and the other in the local frame.
3. For the Mazzer frame:
   - Use `scipy.spatial.transform.Rotation.align_vectors()` with the two sets of vectors to compute the rotation between local and global points.
   - Extract the Euler angles using `scipy.spatial.transform.Rotation.as_euler('XYZ')`.
4. For the Cup frame:
   - Manually define the rotation as:
     - A 90° rotation about the global Y-axis.
     - Followed by an additional rotation about the new X-axis, derived from local and global definitions of the reference point, keys 17 and 2 from [2] respectively.

- Extract the Euler angles using `as_euler('XYZ')`.

5. For all other frames, we know rotation is only around the Z-axis so,
   - Set the Z values of the local and global vectors to zero to avoid calculating extra rotation due to mismeasurement of reference points in the Z-axis.
   - Use `align_vectors()` to compute the rotation.
   - Extract the Euler angles using `as_euler('XYZ')`
6. Save the computed Euler angles for each frame back to the `pandas` DataFrame.

Once the DataFrame with parent frame rotation angles has been constructed, it can be used to construct the full HT matrix required to define any of the specified tools in an arbitrary pose within any of the specified frames. All HT matrices are calculated with `robomath.TxyzRxyz_2_Pose`, all inverses are calculated with `robomath.invH()`. The steps to construct the pose are as follows:

1. Calculate an optional tool offset transform, for small adjustments in the tool reference frame, and take its inverse.
2. Depending on the tool key:
   - If no tool is specified, use the identity matrix as the tool transform.
   - For special cases (e.g., the angled basket of the Rancilio Tool) use rotation data from the DataFrame applied to the standard positional offset transform. Invert this to get the tool transform
   - For all other tools, the transform is retrieved directly from the standard positional offset and inverted.
3. Calculate an option local offset transform for small adjustments in the local reference frame.
4. Fetch the pose of the specified frame from the dataFrame and combine it with the local transform to get the frame's pose relative to its parent.
5. If the frame is marked as a global origin, this combined transform is final. Otherwise, find the global pose of the parent frame and left multiply it by the current transform to get the global pose.

2.2 Equipment frames

As noted earlier, HT matrices were generated automatically. However, selected examples are included below, as the cup frame required a more manual approach, and the project documentation specifies that these should be provided.

2.2.1 Cup frame

Visual inspection of the Cup Frame, Figure 1, shows that it is a compound rotation away from the base reference frame. It appears to have been rotated 90° about the global Y-axis such that the X-axis points 'up', and is possibly rotated around this new X-axis. As the single reference point given in both local and global coordinates is insufficient to uniquely define this compound rotation [3], it was assumed that the frame had been rotated such that the X-axis points directly up, and only one further rotation had been done, defined by the aforementioned reference point.

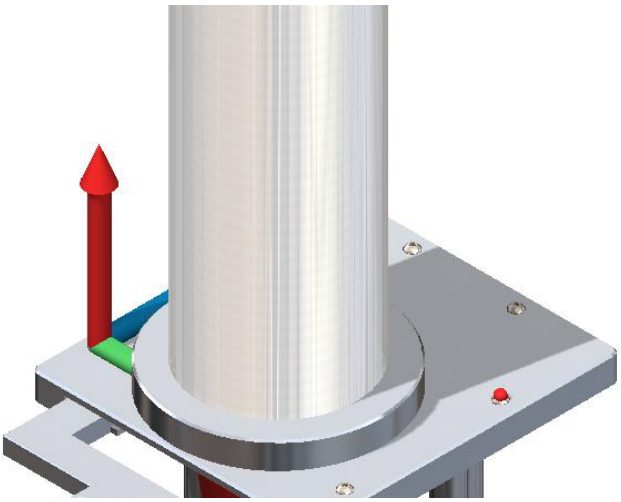

Figure 1: Cup Frame with reference point [2]

To determine the rotation of the local frame about its local, rotated X-axis, the difference in angles between the reference point in the local YZ-plane and the global XY-plane is used. This comparison works assuming that the local YZ-plane and the global XY-plane are parallel, which is true following the previous assumption of a pure 90° about the global Y-axis.

First project the global vector onto the XY-plane, Equation 1, then compute its angle in the global frame with respect to the Y-axis, using the global X and Y axes as reference, Equation 2. Finally, compute the angle of the vector in the local frame, Equation 3, and calculate the revelative rotation of the frame as the difference of the two angles, Equation 4.

$$\vec{v}_{\text{proj}} = [v_x, v_y, 0] \tag{1}$$

$$\theta_1 = \arctan 2(-\vec{v}_{\text{proj}} \cdot \hat{x},\ \vec{v}_{\text{proj}} \cdot \hat{y}) = 32.602^\circ \tag{2}$$

$$\theta_2 = \arctan 2(v_z^{\text{local}},\ v_y^{\text{local}}) = 32.367^\circ \tag{3}$$

$$\theta = \theta_1 - \theta_2 = 0.235^\circ \tag{4}$$

Despite θ being a small value, it was included as ENMT482 Technical Officer Rodney Elliott swears by the math. Equation 5 gives the rotation matrix of the Cup Frame.

$$R_{\text{cup}} = R_y(90^\circ) \times R_x(x^\circ) = \begin{bmatrix} 0 & -0.004 & 1.000 \\ 0 & 1.000 & -0.004 \\ -1 & 0 & 0 \end{bmatrix} \tag{5}$$

Again, this final rotation matrix was decomposed, using `as('XYZ')`, into the intrinsic Euler angles (90.000°,-89.765° and 90.000°) that `robomath.TxyzRxyz_2_Pose` uses to construct HT objects [4]. The HT matrix resulting from these Euler angles and the translation of the Cup Frame is 6.

$$HT_{\text{Cup}} = \begin{bmatrix} 0.000 & -0.004 & -1.000 & -589.900 \\ 0.000 & 1.000 & -0.004 & -221.100 \\ 1.000 & 0.000 & 0.000 & 212.900 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{6}$$

2.2.2 SwitchRocker (lowered) frame

The 'SwitchRocker (lowered)' frame defines the coordinate system for the hot water dispensing button on the Rancilio Silvia single boiler espresso machine. This button is used by the robot to dispense hot water directly

into the portafilter during automated beverage preparation, and is physically mounted on the machine's front control panel, see Figure 2.

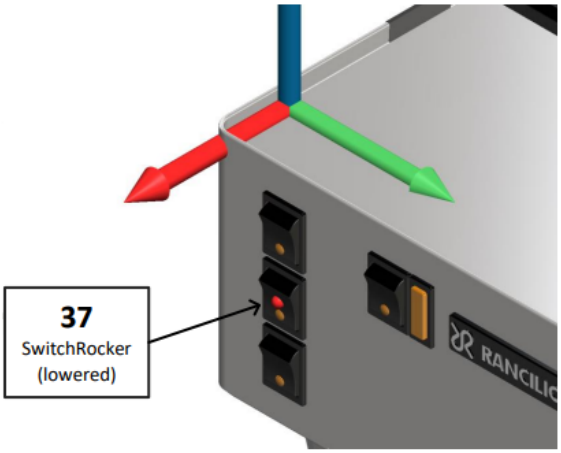

Figure 2: SwitchRocker (lowered) frame with reference point [2].

To accurately operate the SwitchRocker button, a dedicated equipment frame was calculated. This ensures precise and repeatable robot motions for hot water dispensing.

To determine the orientation of the frame, two or more corresponding feature points on the flat front panel were measured in both global and local frames. The frame definition is dominated by rotation about the global Z-axis, corresponding to the panel surface. To ignore small vertical misalignments, all vectors are projected onto the XY-plane as shown in Equation 7.

$$\vec{v}_i^{\,proj} = [x_i, y_i, 0] \tag{7}$$

The optimal planar rotation matrix $R_z(\gamma)$, aligning the set of local vectors $\vec{v}_i^{\,local}$ to their global counterparts $\vec{v}_i^{\,global}$, is found by minimising the mean squared error as given in Equation 8.

$$R = \underset{R \in SO(2)}{\arg\min} \sum_i \left\| R\vec{v}_i^{\,local} - \vec{v}_i^{\,global} \right\|^2 \tag{8}$$

where $SO(2)$ is used since all vectors lie in the plane and only Z-axis rotation is meaningful.

The solution for R corresponds to a Z-axis Euler angle γ (rotation about panel normal), found in code by passing projected local and global vectors into `rotation = sp.Rotation.align_vectors(global_vectors, local_vectors)`. This angle is stored for the frame, and is used in constructing the homogeneous transformation matrix, Equation 9.

$$HT_{\text{SwitchRocker}} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & t_x \\ \sin\gamma & \cos\gamma & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

where $[t_x, t_y, t_z]$ defines the origin in world coordinates.

The computed homogeneous transformation matrix for the SwitchRocker (lowered) frame is given in Equation 10.

$$HT^{\text{rancilio}}_{\text{SwitchRocker (lowered)}} = \begin{bmatrix} 0.500 & -0.866 & 0.000 & -420.544 \\ 0.866 & 0.500 & 0.000 & -418.986 \\ 0.000 & 0.000 & 1.000 & 288.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{10}$$

This homogeneous transform matrix is used by the robot's path planner to target the SwitchRocker, as seen in the code snippet below. This ensures the SwitchRocker button is correctly pressed in the automated beverage-making process.

```
UR5.MoveJ(tf.pose(points_df, 37, tool=id.Mazzer_Tip_Tool, pos_x=15, pos_z=5, theta_x=0,
theta_y=-90, theta_z=180), blocking=True)
```

This method provides repeatable accuracy for robotic interaction with front-panel controls, supporting collision avoidance and successful task execution during hot water dispensing. The approach is easily generalised to other equipment frames on the machine.

2.3 Tool frames

2.3.1 Rancilio

The Rancilio tool, Figure 3, is an espresso portafilter mounted to the pneumatic attachment required for interfacing with the UR5e tool attachment point. As the top face of the portafilter is slanted compared to the portafilter handle, additional rotations are required to orient the top face flat. A flat top face is required to place the portafilter on and in the equipment. On Figure 3, θ represents the rotation of the portafilter face around the Y-axis of the point at the centre of the basket, and d is the depth of the basket.

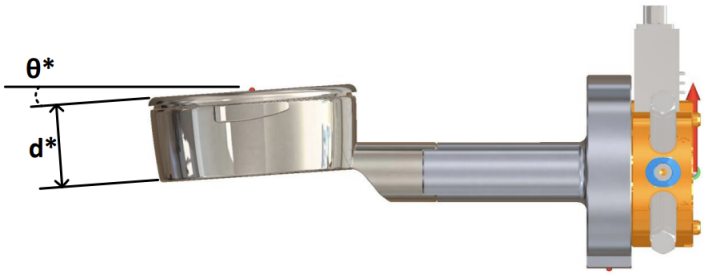

Figure 3: Rancilio frame with reference angles and lengths [2]

To calculate the HT to the flattened portafilter simply rotate the standard translated frame in the Y-axis before applying the 50° rotation between the UR5's tool atachment point and the Rancilio tool's (and all other tool's) pneumatic attachment point, Equation 11[1]

$$HT_{\text{Rancilio Basket}} = R_y(\theta) \times R_z(50°) + T(\text{basket}) = \begin{bmatrix} 0.643 & -0.766 & -0.030 & 28.700 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.019 & -0.023 & 1.000 & 146.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{11}$$

To calculate the transform between the tool attachment point and the base of the Rancilio tool, the basket transform is simply translated in the tool X-axis by d, Equation 12. For robot 3, $d = 29.3mm$.

$$HT_{\text{Rancilio Base}} = HT_{\text{Rancilio Basket}} + T_x(d) = \begin{bmatrix} 0.643 & -0.766 & -0.030 & -0.600 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.019 & -0.023 & 1.000 & 146.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{12}$$

Note that in code, both of these transforms are calculated without the 50° rotation between the tool attachment point and the tool frames and is inverted before being applied to the transform stack that defines the UR5's global pose. The 50° rotation is applied after this inverse by right multiplying the transform stack by $R_z(50°)$.

[1] '$R(x)$' denotes either a 3×3 rotation matrix, or a 4×4 HT matrix with zero translation component. The notation '$T(x)$' denotes a 4×4 HT matrix with zero rotation component.

2.3.2 Tool offsets

2.4 Poses to achieve tasks

2.4.1 Circular paths

Several tasks, most crucially, inserting the Rancilio portafilter into the Rancilio machine, require precise circular movements to complete. Again, for testing efficiency and ease of use, manually calculating the many poses that define circular rotation was determined to be inefficient. So a further helper function, `generate_circular_path`, [1], was written to compute n poses on a circular arc from any initial pose P, with an arbitrary centre C, and an arbitrary arc length $n\phi$. This function was restricted to rotations parallel to the global XY-plane.

To calculate the rotated pose, extract the translations and rotations of the centre of rotation pose using `robomath.Pose_2_TxyzRxyz` isolate purely the translated component of the centre of rotation pose C_{T} use `robomath.Pose_2_TxyzRxyz` to extract the translations and rotations of the pose. Then `robomath.TxyzRxyz_2_Pose` with only the extracted translations to reconstruct the centre of rotation without rotation, this process is equivalent to Equation 13

$$\text{if } C = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ then } C_{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & t_x \\ 0 & 0 & 0 & t_y \\ 0 & 0 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Translate the initial pose to the centre of rotation following Equation 14.

$$P_{\mathrm{C}} = C_{\mathrm{T}}^{-1} \times P \tag{14}$$

Rotate by ϕ following Equation 15.

$$P_{\mathrm{C}\phi} = R_z(\phi) \times P_{\mathrm{C}} \tag{15}$$

Finally, transform back to the global frame, Equation 16. The final rotation matrix, $R(\phi \text{ or } 0)$ is set to ϕ if the desired motion rotates the end effector to keep it aligned throughout rotation, as is done when rotating the Rancilio portafilter into the Rancilio machine. $R(\phi \text{ or } 0)$ is set to 0 when the end effector should maintain its angle through the motion so it doesn't slow motion or spin past its $\pm 360°$ limit. This is used for spinning the WDT tool.

$$P_{\phi} = P_{\mathrm{C}} \times P_{\mathrm{C}\phi} \times R_z(\phi \text{ or } 0) \tag{16}$$

Example poses generated by this process, used to complete task i, spinning the WDT are explained below. To generate the initial pose, P_0, point xx is selected as the target, with the mazzer tool tip selected as the tool, the pose is rotated 180° around the local X-axis and offset in the Z-axis 6mm to clear bolts on the WDT. This pose, is showed left of Figure 4. The centre of rotation, C, was set to $HT_{\mathrm{WDT}}^{\mathrm{World}}$, the overall angle $n\phi$ was set to 180° and two steps were computed. R_Z was set to 0° to not overspin the UR5's final joint, note how the mazzer dose bar does not change orientation in Figure 4. The function generates poses P_{90} and P_{180}, Figure 4 centre and right respectively.

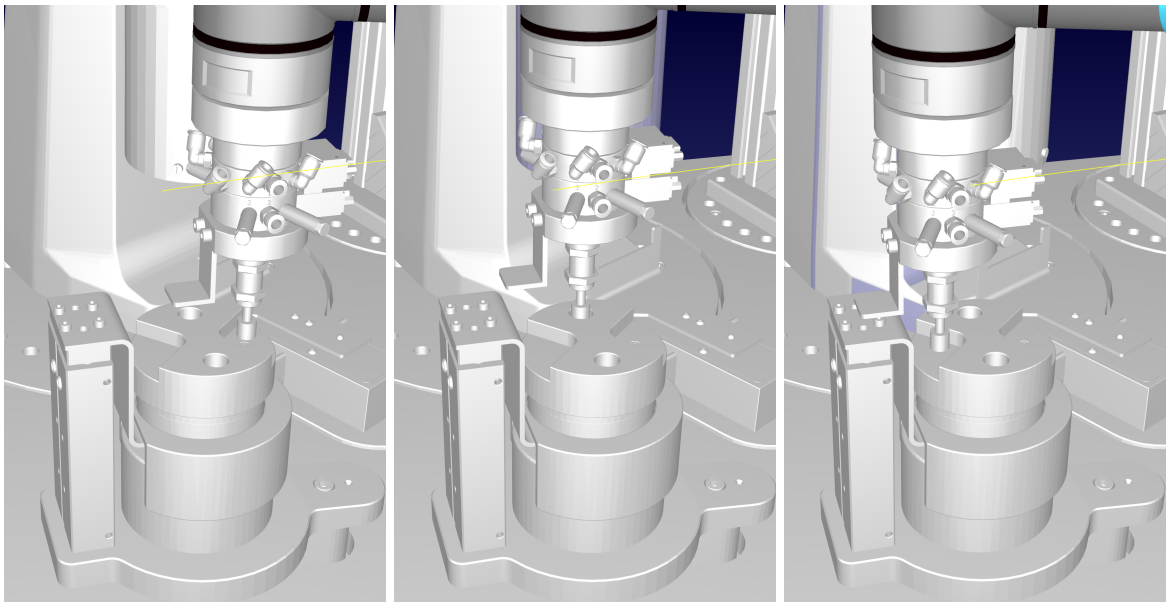

Figure 4: RoboDK simulation of sample poses used for spinning the WDT.

$$P_0 = \begin{bmatrix} -0.643 & 0.765 & 0 & 561.701 \\ 0.765 & 0.643 & 0 & -100.926 \\ 0 & 0 & -1 & 278.820 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_{90} = \begin{bmatrix} -0.643 & 0.765 & 0 & 590.526 \\ 0.765 & 0.643 & 0 & -127.699 \\ 0 & 0 & -1 & 278.820 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_{180} = \begin{bmatrix} 0 & 0.765 & 0 & 617.299 \\ 0 & 0.643 & 0 & -98.874 \\ 0 & 0 & -1 & 278.820 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.4.2 Linear Paths

The `pose` function constructs a global homogeneous transformation matrix that represents the position and orientation of the robot's tool relative to the robot base frame. This matrix encodes all necessary translations and rotations to precisely position the end effector during task execution.

Mathematically, the function converts user inputs, including a base frame key, a tool identifier, and local positional and rotational offsets. A transform is completed through the following sequence:

1. Convert local rotation offsets $(\theta_x, \theta_y, \theta_z)$ and optional tool rotation offsets (off_θ_x,off_$\theta_y$off_θ_z) from degrees to radians.

2. Construct the tool offset transform T_{tool_off}, accounting for fine experimental adjustments in the tool's reference frame, and invert this transform to correctly adjust the tool pose, as shown in Equation 17.

$$T_{\text{tool\_off}} = \text{inv}(\texttt{TxyzRxyz\_2\_Pose(off}_x\texttt{, off}_y\texttt{, off}_z\texttt{, off\_}\theta_x\texttt{, 0, 0)}) \tag{17}$$

3. Determine the tool transform T_{tool} according to the specified tool key, with special cases (such as the Rancilio portafilter basket), or otherwise, default to an identity transform.

4. Calculate the local positional and rotational offset transform relative to the base frame as given in Equation 18.

$$T_{\text{local}} = \texttt{TxyzRxyz\_2\_Pose(pos}_x\texttt{, pos}_y\texttt{, pos}_z\texttt{, }\theta_x\texttt{, }\theta_y\texttt{, }\theta_z\texttt{)} \tag{18}$$

5. Retrieve the base frame transform T_{frame} from the calibration data DataFrame using the frame key, which includes position and Euler angles, Equation 19.

$$T_{\text{frame}} = \texttt{TxyzRxyz\_2\_Pose(X, Y, Z, Euler}_{\texttt{R\_x}}\texttt{, Euler}_{\texttt{R\_y}}\texttt{, Euler}_{\texttt{R\_z}}\texttt{)} \tag{19}$$

6. Combine these transforms multiplicatively to produce the full global pose, as expressed in Equation 20.

$$T_{combined} = \begin{cases} T_{frame} \times T_{local} \times T_{tool\_off} \times T_{tool} \times R_z(50^\circ - \text{off}\_\theta_z), & \text{if frame is global origin} \\ T_{parent} \times T_{frame} \times T_{local} \times T_{tool\_off} \times T_{tool} \times R_z(50^\circ - \text{off}\_\theta_z), & \text{otherwise} \end{cases} \tag{20}$$

where $R_z(\cdot)$ is a fixed rotation around the Z-axis correcting for tool mounting.

The resulting 4×4 matrix is returned and used by the robot controller for accurate tool placement.

Using this function, task-specific poses can easily be generated by varying the offsets. For instance, pressing the `SwitchRocker` button (frame key 37) on the Rancilio espresso machine involves moving between four primary poses, P_0, P_{press}, P_{release}, and P_{back}, given as follows:

$$P_0 = \begin{bmatrix} 0.663 & 0.557 & -0.500 & -361.648 \\ -0.383 & -0.321 & -0.866 & -316.942 \\ -0.643 & 0.766 & 0.0 & 293.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_{\text{press}} = \begin{bmatrix} 0.663 & 0.557 & -0.500 & -371.646 \\ -0.383 & -0.321 & -0.866 & -334.264 \\ -0.643 & 0.766 & 0.0 & 293.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{\text{release}} = \begin{bmatrix} 0.663 & 0.557 & -0.500 & -371.646 \\ -0.383 & -0.321 & -0.866 & -334.264 \\ -0.643 & 0.766 & 0.0 & 283.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_{\text{back}} = \begin{bmatrix} 0.663 & 0.557 & -0.500 & -361.648 \\ -0.383 & -0.321 & -0.866 & -316.942 \\ -0.643 & 0.766 & 0.0 & 293.000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The robot execution function `P()` [1], and performs the operation by waiting until the scale confirms the target weight of hot water (30±0.1 g) has been dispensed before switching off the button. This logic ensures precision and repeatability in the amount of water dispensed.

Figure 5 visualises these four poses: initial approach, button press, button release, and retraction of the end effector.

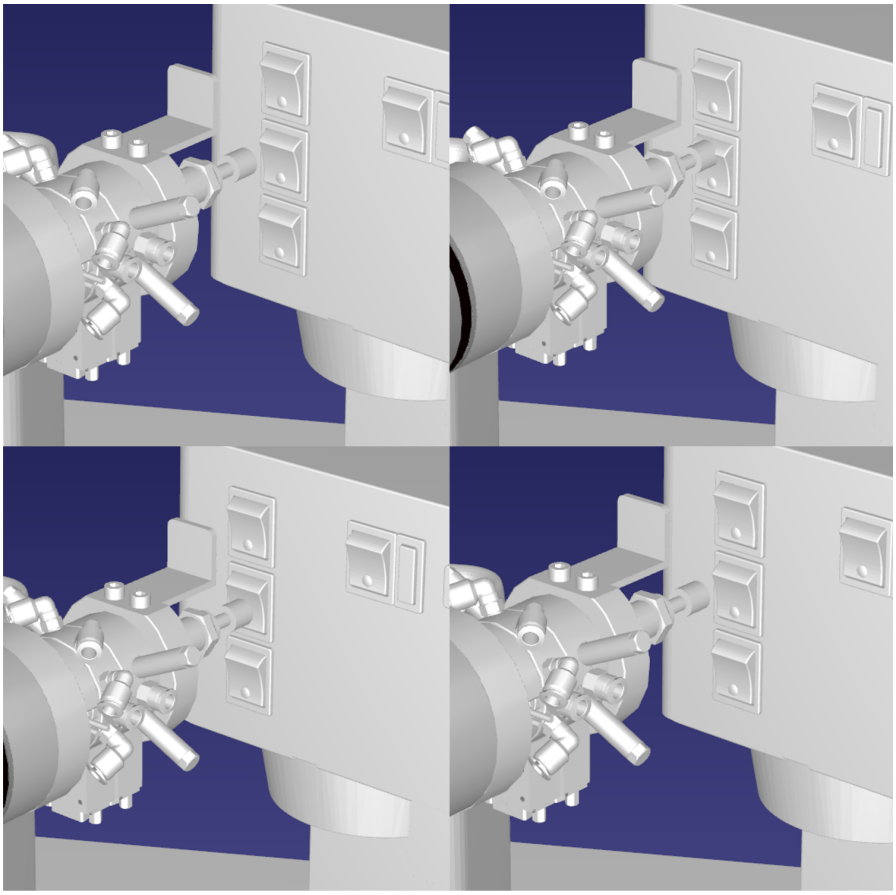

Figure 5: Poses for button operation: initial approach, button press, button release, and retraction.

This approach enables precise and reliable control of the SwitchRocker button, integrating it into the automated espresso-making process for consistent and repeatable operation.

3 Completing the required tasks

3.1 Toolpath

3.1.1 General Movements

The general concept for positioning the robot was to use rough moves to get into an approximate position, then work within the local coordinates for fine control. To transition from appliance to appliance, `MoveJ()` was used to reach the approximate approach pose. The joint angles were generated by manually manipulating the robot in the simulation. One particularly effective method was to position the robot into positions in the local frames by running code, then adjust the robot to an approach position. This was done by altering joint angles or doing linear moves in either the tool or the robot's global frame.

To transition to the approximate approach pose, intermediate points were selected by manipulating the robot in the simulation to find intermediate points to ensure no interference between the robot and the tools, workplace features or itself. It was also noted that intermediates were chosen to be near the original path of the robot to ensure high speeds.

One more novel way was to position the robot in a position where a rotation would not interfere with any workplace features, then only pivot a single joint. This simplified movements and reduced the slower wrist movements. This was used for the large sweeping moves, such as to quickly move the mazzer tool from the WDT to the cup dispenser.

3.1.2 Rancillio Insertion

`MoveC()` was unable to be used to insert/remove the Rancilio tool. The `MoveC()` resulted in a rotation that did not correspond precisely to the centre of the Rancilio tool, resulting in a slight bobble of the tool. This problem was solved by calculating many small incremental poses from the start pose and the end pose. A `MoveL()` command was then used to interpolate between these intermediates. This ensured that the tool centre stayed centred relative to the Rancilio machine gasket.

3.1.3 Left Hand Configuration

The first tasks, operating the Mazzer grinder and the Mazzer scale, are easier to complete with the robot in left-hand configuration. As a result, the robot was kept in this configuration for all other tasks, as switching configurations takes time, of which the goal was to reduce.

3.1.4 Optimisations

It was quickly realised that wrist movements (6th axis) resulted in a reduced speed of the robot. This knowledge was exploited to optimise the code for greater speed. The method was to use the angle of the 6th axis that was set when a tool change took place (206.18 degrees), as this angle would have to be adjusted when changing tools. This angle was used for all Mazzer operations, as for the majority of the time, the Mazzer orientation did not matter due to the end effectors' symmetry. This enabled a significant increase from around 14 minutes to around 11.5 minutes.

The order of the operations was suboptimal. We changed the order of the operations to reduce the number of tool changes and make the use of tools that were already attached. The Rancillio tool was also inserted and removed while the cup was still in the machine, which was not specified in the original instructions, but made the drop off and pick up of the cup on the Rancillio scale a far simpler movement.

4 Time-and-motion results and discussion

4.0.1 Program Results

The UR5e takes 11 minutes and 30 seconds to make a single coffee with the code produced by Group 12. Rodney takes just 2 minutes and 58 seconds. The initial cost of a UR5e robot is approximately $66,035 NZD [5]. Other expenses, such as maintenance and repairs, will also add costs to the UR5e over the course of its lifetime, but will be ignored for this exercise.

4.0.2 Feasibility of UR5e as a Coffee Robot

People only tend to buy coffee for the first few hours of the day. Altering the working hours to be the same for Rodney and the robot means that the robot will not be able to catch up with Rodney's performance. Rodney can produce far greater volumes of coffee, resulting in his wage being easily covered by the coffee sales. The decision was made to make the idealised assumption that the UR5e will produce coffee 24 hours a day. This assumes that people are continuously purchasing coffee throughout the day.

Metric	Rodney	UR5
Initial Cost ($)	0	66,035
Cost Per Hour ($/h)	28	0.004
Time per Coffee (min)	2.966	11.5
Hours Worked per Day	7	24
Coffees per Day	141.60	125.22
Revenue ($)	686.78	607.30
Daily Profit ($)	284.75	425.02
Days to Break Even	0	155.37

Table 1: Comparison of Rodney and UR5e Coffee Production Metrics

Global Parameters:
NZ Coffee Price = $4.85 [6]
Profit Margin on a Coffee = 70%

The total time with this assumption is 471 days to begin to profit over Rodney. The UR5e is required to work more than 16 hours a day to ever break even. This, however, calculation unlikely as it does not account for the maintenance of the machine and the cyclical nature of coffee demand. Factors that play a huge role in coffee throughput.

The robot arm could be optimised to move faster and have quicker moves. This could be done with a non-collaborative robot arm, where a zone could be fenced off that the arm can operate in. This will allow for quicker moves and potentially the use of a cheaper robot that does not have to have collaborative qualities.

The most significant thing that would help the system the most would be automating tasks such as the spinning of the WDT, the dispensing of the coffee grounds and pulling the shot would reduce the time taken significantly. A more practical method would be purchasing an automatic coffee machine found in many commercial buildings. This, however, does not accomplish the key feature the robotic arm has compared to a conventional coffee manufacturing method: novelty. Many of these tasks would be done far more efficiently without using the robot arm, and if some of these were automated, it would reduce the time taken to produce a coffee and potentially reduce the cost of the robot.

4.0.3 Dear Rodney

People visit coffee shops not just for the caffeine. They instead visit for connection and routine. Rodney's role is not just to make coffee; his job is about care, recognition and delivering human warmth to his loyal customers. Replacing Rodney with a robot arm raises a deeper question: do we want perfect efficiency or imperfect humanity? A robot can craft a flawless drink, but only Rodney can create a meaningful moment around it.

4.0.4 Feasibility Conclusion

There may be ways to increase the viability by changing what robot is used, automating certain tasks and increasing the speed. If the purpose is to produce coffee in a novel way for spectators, it may be viable. As a purely coffee-producing machine in this configuration, it is not viable. It is recommended that this exercise be limited to only the scope of University projects and as a "thinly veiled exercise in linear algebra".

5 Discussion

5.1 Value of Automatic Pose Generation (Setting out a good Workflow)

Taking time to develop a workflow is the best use of time. This project was successful due to the initial time spent developing the Python script for ease of use. Some of the features implemented that made the workflow much easier to implement were:

- Interpreting the points spreadsheet using Pandas increased the portability of the program to different robots
- Linear multi-point interpolation (Move C imitation)
- Using an OO-based ID system to denote the local and sub-local frames for readability.
- Functions for each of the tasks so they can be easily tested independently (commenting out)
- Feature to translate and rotate the tool in the tool coordinates for tasks such as pressing buttons

Having a clear workflow has allowed for rapid development by implementing features to make it easier to work with, read and port to other robots.

5.2 Developing an accurate System Model

Ensuring the robot is using the calibrated settings ensures that the robot provides consistent movements and that the movements are accurate with respect to the given positional points.

If the points generated are accurate to the reality of the machine means that minimal offsets are required. This contributes immensely to the readability of the program. If another developer wanted to refactor or modify the program, these offsets would introduce difficulty in adapting the code.

5.3 Software Limitations

RoboDK limits creativity. For example, there is no simple solution for stopping the robot mid-moment, which would be helpful when defining motion where the end condition is some unknown point along the move (like for the mazzer dosing). This was worked around by creating a set of small linear moves, so that the dispensing of coffee grounds could be achieved with accuracy and without stopping the entire program. This is suboptimal.

Linear moves to approximate a circular move were also used for a different reason in the Rancillio tool insertion. Circular moves, when connected to the UR5, did not match the simulation. This results in the tool bobbling and stalling the machine. This was solved by again adding a series of linear moves to approximate the rotation.

Both these features were limitations of the RoboDK software. These features should be implemented/patched. Learning to overcome these limitations and the software requirements is part of the learning and was critical to overcoming two of the most difficult tasks in the assignment.

References

[1] O. H. Dominic McNicholas, Linus Ritchie, "ENMT482_Assignment_2: RoboDK collaborative robot coffee assignment," https://github.com/domkbm/ENMT482_Assignment_2, 2025, accessed: 2025-10-17.

[2] R. Elliott, "Lab frames 2025," https://gitlab.com/uc_mech_wing/robotics_control_lab/uc-02-2024/robodk_stations/-/blob/master/lab_frames_2025.pdf, 2025, part of the UC-02-2024 project: Coffee Cart Modifications. University of Canterbury.

[3] L. Euler, "Novi commentarii academiae scientiarum petropolitanae," *Novi Commentarii Academiae Scientiarum Petropolitanae*, vol. 20, pp. 189–207, 1776, enestrom number: E478.

[4] RoboDK, *RoboDK Python API:* `robodk.robomath.TxyzRxyz_2_Pose`, RoboDK, 2025, accessed: 2025-10-02. [Online]. Available: https://robodk.com/doc/en/PythonAPI/robodk.html#robodk.robomath.TxyzRxyz_2_Pose

[5] Standard Bots. (2025) Universal Robots price guide: What to expect (new and used costs). Accessed: Oct. 17, 2025. [Online]. Available: https://standardbots.com/blog/universal-robot-price

[6] Stuff.co.nz. (2025) What's really going on with coffee prices. Accessed: Oct. 17, 2025. [Online]. Available: https://www.stuff.co.nz/money/360575273/whats-really-going-coffee-prices

Appendix A Equipment and tool transforms

None of these transforms were ever explicitly defined, or used, in code, they are merely here due to report specifictaion.

Transform	**Matrix**
$HT^{\text{World}}_{\text{CD\_Top\_Cover fastener (2010 o'clock)}}$	$\begin{bmatrix} 0.000 & -0.004 & -1.000 & -589.900 \\ 0.000 & 1.000 & -0.004 & -221.100 \\ 1.000 & 0.000 & 0.000 & 212.900 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{Mazzer dose hopper lid apex}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 502.100 \\ -0.507 & -0.353 & 0.786 & -420.400 \\ -0.000 & 0.912 & 0.409 & 317.400 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{MS\_Top\_Cover fastener (left)}}$	$\begin{bmatrix} 0.496 & 0.868 & 0.000 & 440.900 \\ -0.868 & 0.496 & 0.000 & -274.400 \\ 0.000 & 0.000 & 1.000 & 41.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{PUQ\_Body lid apex}}$	$\begin{bmatrix} -0.707 & 0.707 & 0.000 & 384.900 \\ -0.707 & -0.707 & 0.000 & 83.600 \\ 0.000 & 0.000 & 1.000 & 270.900 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{TopPanel fastener (left front)}}$	$\begin{bmatrix} 0.500 & -0.866 & 0.000 & -412.000 \\ 0.866 & 0.500 & 0.000 & -479.800 \\ 0.000 & 0.000 & 1.000 & 348.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{MS\_Top\_Cover fastener (left)}}$	$\begin{bmatrix} -0.497 & 0.868 & 0.000 & -385.800 \\ -0.868 & -0.497 & 0.000 & -330.200 \\ 0.000 & 0.000 & 1.000 & 41.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{DT\_Anvil face (centre)}}$	$\begin{bmatrix} -1.000 & 0.001 & 0.000 & 589.500 \\ -0.001 & -1.000 & 0.000 & -99.900 \\ 0.000 & 0.000 & 1.000 & 86.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{cup\_dispenser}}_{\text{CD\_Top\_Cover fastener (1345 o'clock)}}$	$\begin{bmatrix} 0.000 & -0.004 & -1.000 & -680.483 \\ 0.000 & 1.000 & -0.004 & -79.471 \\ 1.000 & 0.000 & 0.000 & 212.900 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

$HT^{\text{cup\_dispenser}}_{\text{CD\_Front\_Index\_Pull (open)}}$	$\begin{bmatrix} 0.000 & -0.004 & -1.000 & -517.891 \\ 0.000 & 1.000 & -0.004 & -150.104 \\ 1.000 & 0.000 & 0.000 & 202.200 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{cup\_dispenser}}_{\text{CD\_Front\_Index\_Pull (shut)}}$	$\begin{bmatrix} 0.000 & -0.004 & -1.000 & -551.491 \\ 0.000 & 1.000 & -0.004 & -150.142 \\ 1.000 & 0.000 & 0.000 & 203.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{cup\_dispenser}}_{\text{Supreme\_Cup rim (centre)}}$	$\begin{bmatrix} 0.000 & -0.004 & -1.000 & -621.692 \\ 0.000 & 1.000 & -0.004 & -150.130 \\ 1.000 & 0.000 & 0.000 & 64.200 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer}}_{\text{Mazzer dose hopper lid edge (2100 o'clock)}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 555.308 \\ -0.507 & -0.353 & 0.786 & -389.104 \\ -0.000 & 0.912 & 0.409 & 317.416 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer}}_{\text{Mazzer bean hopper lid apex}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 562.407 \\ -0.507 & -0.353 & 0.786 & -523.067 \\ -0.000 & 0.912 & 0.409 & 582.739 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer}}_{\text{Mazzer dose lever face (centre)}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 438.590 \\ -0.507 & -0.353 & 0.786 & -480.593 \\ -0.000 & 0.912 & 0.409 & 163.644 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer}}_{\text{Mazzer off button centre (raised)}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 453.014 \\ -0.507 & -0.353 & 0.786 & -526.182 \\ -0.000 & 0.912 & 0.409 & 85.292 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer}}_{\text{Mazzer on button centre (raised)}}$	$\begin{bmatrix} -0.862 & 0.207 & -0.463 & 448.294 \\ -0.507 & -0.353 & 0.786 & -510.268 \\ -0.000 & 0.912 & 0.409 & 86.274 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer\_scale}}_{\text{MS\_Top\_Cover fastener (right)}}$	$\begin{bmatrix} 0.496 & 0.868 & 0.000 & 407.907 \\ -0.868 & 0.496 & 0.000 & -293.253 \\ 0.000 & 0.000 & 1.000 & 41.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer\_scale}}_{\text{MS\_Ball\_Bearing apex}}$	$\begin{bmatrix} 0.496 & 0.868 & 0.000 & 418.847 \\ -0.868 & 0.496 & 0.000 & -273.296 \\ 0.000 & 0.000 & 1.000 & 56.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

$HT^{\text{mazzer\_scale}}_{\text{MS\_Lock\_Lever (left)}}$	$\begin{bmatrix} 0.496 & 0.868 & 0.000 & 475.222 \\ -0.868 & 0.496 & 0.000 & -291.068 \\ 0.000 & 0.000 & 1.000 & 26.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{mazzer\_scale}}_{\text{MS\_Lock\_Lever (right)}}$	$\begin{bmatrix} 0.496 & 0.868 & 0.000 & 404.842 \\ -0.868 & 0.496 & 0.000 & -331.285 \\ 0.000 & 0.000 & 1.000 & 26.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{puq}}_{\text{PUQ\_Body lid edge (2100 o'clock)}}$	$\begin{bmatrix} -0.707 & 0.707 & 0.000 & 345.033 \\ -0.707 & -0.707 & 0.000 & 123.467 \\ 0.000 & 0.000 & 1.000 & 270.900 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{puq}}_{\text{PUQ\_Body upper clamp (centre)}}$	$\begin{bmatrix} -0.707 & 0.707 & 0.000 & 377.758 \\ -0.707 & -0.707 & 0.000 & 76.317 \\ 0.000 & 0.000 & 1.000 & 137.400 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio}}_{\text{TopPanel fastener (right front)}}$	$\begin{bmatrix} 0.500 & -0.866 & 0.000 & -602.904 \\ 0.866 & 0.500 & 0.000 & -369.616 \\ 0.000 & 0.000 & 1.000 & 348.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio}}_{\text{SwitchRocker (lowered)}}$	$\begin{bmatrix} 0.500 & -0.866 & 0.000 & -420.544 \\ 0.866 & 0.500 & 0.000 & -418.986 \\ 0.000 & 0.000 & 1.000 & 288.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio}}_{\text{GroupGasket (centre)}}$	$\begin{bmatrix} 0.500 & -0.866 & 0.000 & -478.562 \\ 0.866 & 0.500 & 0.000 & -457.893 \\ 0.000 & 0.000 & 1.000 & 202.600 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio\_scale}}_{\text{MS\_Top\_Cover fastener (right)}}$	$\begin{bmatrix} -0.497 & 0.868 & 0.000 & -418.781 \\ -0.868 & -0.497 & 0.000 & -311.325 \\ 0.000 & 0.000 & 1.000 & 41.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio\_scale}}_{\text{RS\_Pan (centre)}}$	$\begin{bmatrix} -0.497 & 0.868 & 0.000 & -480.531 \\ -0.868 & -0.497 & 0.000 & -457.478 \\ 0.000 & 0.000 & 1.000 & 65.940 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio\_scale}}_{\text{MS\_Lock\_Lever (left)}}$	$\begin{bmatrix} -0.497 & 0.868 & 0.000 & -382.760 \\ -0.868 & -0.497 & 0.000 & -368.234 \\ 0.000 & 0.000 & 1.000 & 26.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

$HT^{\text{rancilio\_scale}}_{\text{MS\_Lock\_Lever (right)}}$	$\begin{bmatrix} -0.497 & 0.868 & 0.000 & -453.113 \\ -0.868 & -0.497 & 0.000 & -327.971 \\ 0.000 & 0.000 & 1.000 & 26.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{wdt}}_{\text{DT\_Plate centre drill hole}}$	$\begin{bmatrix} -1.000 & 0.001 & 0.000 & 481.500 \\ -0.001 & -1.000 & 0.000 & -100.000 \\ 0.000 & 0.000 & 1.000 & 43.700 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{wdt}}_{\text{UK\_Rotor fastener (1800 o'clock)}}$	$\begin{bmatrix} -1.000 & 0.001 & 0.000 & 561.701 \\ -0.001 & -1.000 & 0.000 & -100.926 \\ 0.000 & 0.000 & 1.000 & 170.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{Cup\_Holder top face centre (open)}}$	$\begin{bmatrix} 0.643 & -0.766 & 0.000 & -104.500 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 186.620 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{Cup\_Holder top face centre (shut)}}$	$\begin{bmatrix} 0.643 & -0.766 & 0.000 & -53.500 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 186.620 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{YSR-12 cushioning cylinder tip}}$	$\begin{bmatrix} 0.643 & -0.766 & 0.000 & 0.000 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 102.820 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{Mazzer\_Dose\_Bar edge}}$	$\begin{bmatrix} 0.643 & -0.766 & 0.000 & -50.000 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 67.060 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{Rancilio\_Tool\_Adapter ball bearing indent}}$	$\begin{bmatrix} 0.643 & -0.766 & 0.000 & -32.000 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 28.070 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{Tool Connector}}_{\text{VST\_Basket rim (centre)}}$	$\begin{bmatrix} 0.643 & -0.766 & -0.030 & 28.700 \\ 0.766 & 0.643 & 0.000 & 0.000 \\ 0.019 & -0.023 & 1.000 & 146.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{World}}_{\text{PC\_Base front lip (centre)}}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & -150.800 \\ 0.000 & 1.000 & 0.000 & -542.100 \\ 0.000 & 0.000 & 1.000 & 179.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

$HT^{\text{rancilio\_tool\_cleaner}}_{\text{PC\_Body on button centre (raised)}}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & -150.800 \\ 0.000 & 1.000 & 0.000 & -643.800 \\ 0.000 & 0.000 & 1.000 & 196.300 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio\_tool\_cleaner}}_{\text{PC\_Silicone\_Brush (raised)}}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & -195.800 \\ 0.000 & 1.000 & 0.000 & -593.300 \\ 0.000 & 0.000 & 1.000 & 187.200 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
$HT^{\text{rancilio\_tool\_cleaner}}_{\text{PC\_Bristle\_Brush (raised)}}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & -105.800 \\ 0.000 & 1.000 & 0.000 & -593.300 \\ 0.000 & 0.000 & 1.000 & 187.200 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$

Appendix B Handy Link to GitHub

https://github.com/domkbm/ENMT482_Assignment_2