Robot Assignment 5

Group members: Théo de Morais and Jesús Sánchez

Working Hours

The table below shows the **total hours** we dedicated to this project, along with a description of the tasks completed each day.

Day / Hours	Task Done	Person
07/04/2025 9:00-13:00 **4 hours**	1-Read all the instructions to understand the task. 2-Start with global variables and basic loops in RAPID.	Jesus and Theo
11/04/2025 9:00-13:00 **4 hours**	1-Robot connection with python 2-Modify rapid variables with python 3-Reading QR codes from the pucks	Jesus and Theo
14/04/2025 9:00-13:00 **4 hours**	1-Test python communication with the robot 2-Try to pick a move pucks to different known positions. 3-Study camera distortion to enhance the coordinates of the pucks.	Jesus and Theo
21/04/2025 9:00-13:00 **4 hours**	Modify python and rapid code to receive a coordinate that will be determined by the calibration function that we have to implement.	Jesus and Theo
22/04/2025 9:00-13:00 **4 hours**	Make a calibration function in python and determine the transformation matrix.	Jesus and Theo
24/04/2025 9:00-13:00 **4 hours**	Continue with the calibration function and try to fix issues with the coordinates.	Jesus and Theo
28/04/2025 9:00-13:00 **4 hours**	Fix issues with communication between python and robotstudio.	Jesus and Theo
02/05/2025 9:00-13:00 **4 hours**	Test robot puck detection using the python image acquisition.	Jesus and Theo
05/05/2025 9:00-13:00 **4 hours**	Enhance the transformation matrix to detect puck coordinates in a better way.	Jesus and Theo

06/05/2025 9:00-13:00 **4 hours**	1-Perform a final test of the program and show it to the professor. 2-Finish writing the report for the assignment.	Jesus and Theo

A. First Version

We began this project by expanding the code from Assignment 3. We implemented a main loop that listens for variable changes. Specifically, we introduced three key variables that we plan to modify using Python in the future:

- **WPW (What Python Wants):** This variable represents the desired command from the Python script. The main loop continuously monitors it for changes. Depending on its value, different actions will be triggered.
- **WRD (What Robot Does):** This variable determines the specific routine that the robot will execute. When the main loop detects a change in WPW, it copies its value to WRD and then resets WPW to its initial value (0).
- **Pos:** This variable defines the target position where we want the robot to move the puck.

```
18      VAR num WPW := 0;
19      VAR num WRD := 0;
20      VAR robtarget pos;
```

```
44  PROC MainLoop()
45      TPWrite "MainLoop starts";
46      WHILE TRUE DO
47          ! Move to safe position
48          MoveJ Offs(targets{3}, 0, 0, safeHeight), vFast, z50, tGripper\WObj:=wobjTableN;
49
50          WRD := 0;    ! Robot does nothing, waits
51          WHILE (WPW = 0) DO
52              ! Wait for Python command
53              TPWrite "Robot waits for Python to set WPW";
54              WaitTime 1;  ! Add delay to prevent log flooding
55          ENDWHILE
56
57          WaitUntil (WPW <> 0);
58          TPWrite "Python wants to do task WPW = "\Num:=WPW;
59          WRD := WPW;
60          WPW := 0;
61
62          TEST WRD
63              CASE -1:
64                  TPWrite "Exiting MainLoop";
65                  RETURN;          ! Quit MainLoop
66              CASE 0:
67                  WaitTime 0.1;    ! Should not happen
68              CASE 1:
69                  movePuck p2;
70              DEFAULT:
71                  TPWrite "Unknown command: "\Num:=WRD;
```

B. Simple Python Connection

Once we had a basic RAPID program, we decided to test the connection between a simple Python script and the robot. To do this, we wrote a basic Python program using the *rwuis* library.

```python
from rwsuis import RWS
import time

nolbertIp = "http://152.94.160.198"
robot = RWS.RWS(nolbertIp)

wpw = robot.get_rapid_variable("WPW")
print(f"[Status] WPW: {wpw}")

target = robot.get_rapid_variable("target_K0")
print(f"[Status] target: {target}")

print("Program has ended.")
```

The main functionality of the script is to read variables from the RAPID program and print their values in Python.

Finally, we verified that all the variables had the same values in both Python and RAPID.

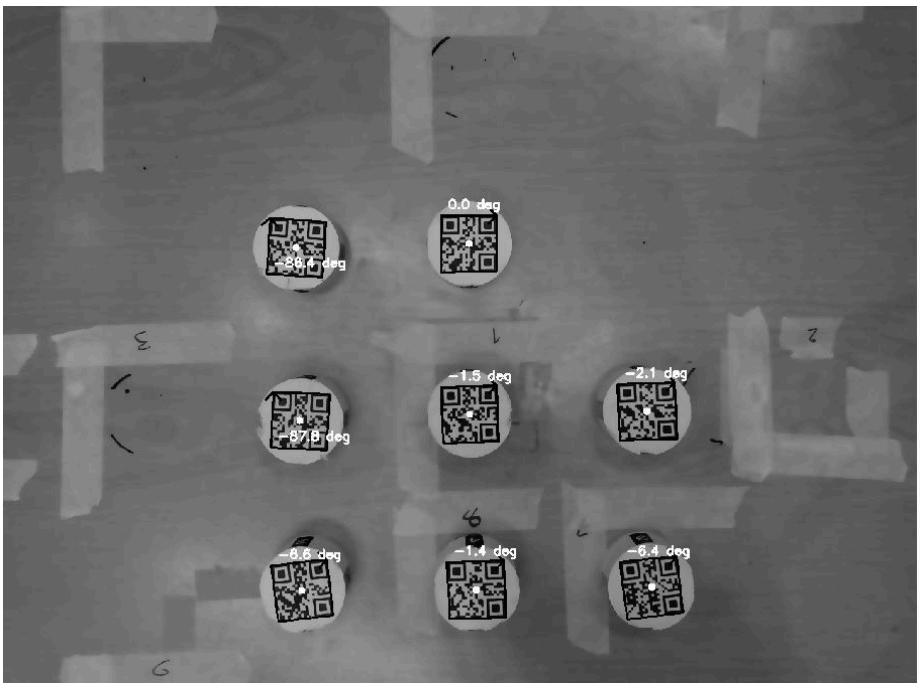
```
PS C:\Users\jesus\ELE610\py> & C:/Users/jesus/AppData/Local/Programs/Python/Python311/
[Status] WPW: 0
[Status] target: [[0,0,0],[0,1,0,0],[0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
Program has ended.
```

C. Taking Pictures

The next step was to verify that the camera was functioning as expected. To do this, we captured images of the pucks from different positions. The results matched our expectations.

D. Using Pyzbar

To move the pucks based on their position detected by the camera, we first needed to obtain their coordinates. For this purpose, we used the **Pyzbar** library, which allows us to detect the coordinates of the four corners of a QR code. Using these corner positions, we calculated the center of the puck by averaging the coordinates.



E. Transformation Matrix

To move the robot based on the position of the puck detected by the camera, we needed to convert image coordinates (from the camera) into real-world table coordinates. This process is called coordinate transformation.

Since the camera was mounted at a fixed height directly above the center of the table, we assumed a simple linear transformation would be sufficient. Using a set of known point correspondences (coordinates in both image and table frames), we calculated a transformation matrix using least squares estimation.

We used the *numpy.linalg.lstsq* function in Python to compute the transformation matrix **T**, which maps image coordinates to table coordinates according to the following model:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Where (xi,yi)(x_i, y_i)(xi,yi) are the coordinates in the image, and (xt,yt)(x_t, y_t)(xt,yt) are the corresponding coordinates on the table.

This matrix allowed us to accurately map the detected puck positions from the camera image to physical positions the robot could use for movement.

$$T_{250} = \begin{bmatrix} -0.0017 & 0.2198 & -62.7952 \\ -0.2194 & -0.0035 & 145.6839 \\ 0.0 & -0.0 & 1.0 \end{bmatrix}$$

$$T_{500} = \begin{bmatrix} 0.0066 & 0.422 & -174.9793 \\ -0.4098 & -0.0071 & 272.8828 \\ 0.0 & -0.0 & 1.0 \end{bmatrix}$$

F. Interpolation to obtain transformation matrix

In this section, we used the transformation matrices previously obtained at different heights. To calculate the transformation matrix at a new height, we applied the following formula:

$$T(z) = \frac{z_2 - z}{z_2 - z_1} T(z_1) + \frac{z - z_1}{z_2 - z_1} T(z_2)$$

Using the transformation matrices at **z = 250 mm** and **z = 500 mm**, we interpolated to obtain the transformation matrix at **z = 350 mm**.

$$T_{350} = \begin{bmatrix} 0.00162 & 0.30068 & -107.66884 \\ -0.29556 & -0.00494 & 196.56346 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

G. All parts together

In this section, we began establishing communication between the image acquisition system and the robotic component. As a starting point, we developed a simple program that detects the position of a single puck and moves it to a predefined location.

H. Multiple pucks

In the previous section, we successfully moved a puck from a random position to a predefined target. In this section, we enhanced our program to detect multiple pucks and stack them at a single location.

I. Image Viewer App

Before reaching the 40-hour time cap, we had some free time, so we consolidated all the code developed in previous steps into an image viewer application. This app provides a user-friendly interface for interacting with the functionalities we implemented earlier.

Through this application, users can easily obtain mastership on the FlexPendant, capture images, detect pucks, and even move them.

RB5 Group V


ImageViewer : gray scale image
File
Scale
Edit
Function
Camera
Robot
Request mastership
Start RAPID code
Launch Task01
Detect puck
Collect one detected puck
Collect all detected pucks
pixmap: (w,h) = (1280,960)
(x,y) = (233,114): gray = 188