The University Of Sheffield.

Automatic Control & Systems Engineering

ACS6503: Manipulator Robotics Laboratory Sheet 2

Updated: 20 November 2023

Contents

1. Introduction

This lab exercise is to familiarise and operate the Mover6 robot arm. This arm is a 6-DOF robot arm comprised of rotational joints. In this exercise you will control it through a variety of different schemes, and under different environments familiarising you with the operation of an arm which is typical in small manufacturing applications.

Dr Jonathan Aitken

Module Leader
jonathan.aitken@sheffield.ac.uk

2. Laboratory Procedures

In this laboratory you'll be using a mixture of different control methodologies for simulating and controlling the Mover6 arm. Further information can be found on this arm in the documentation which is available through Blackboard. You should familiarise yourself with this information before you begin using the equipment.

2.1 Starting the Software

Open the CProg programme either from the start menu or from the icon on the desktop.

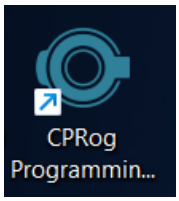

Figure 1: CProg Shortcut

When opened the software will bring up the robot selection screen:

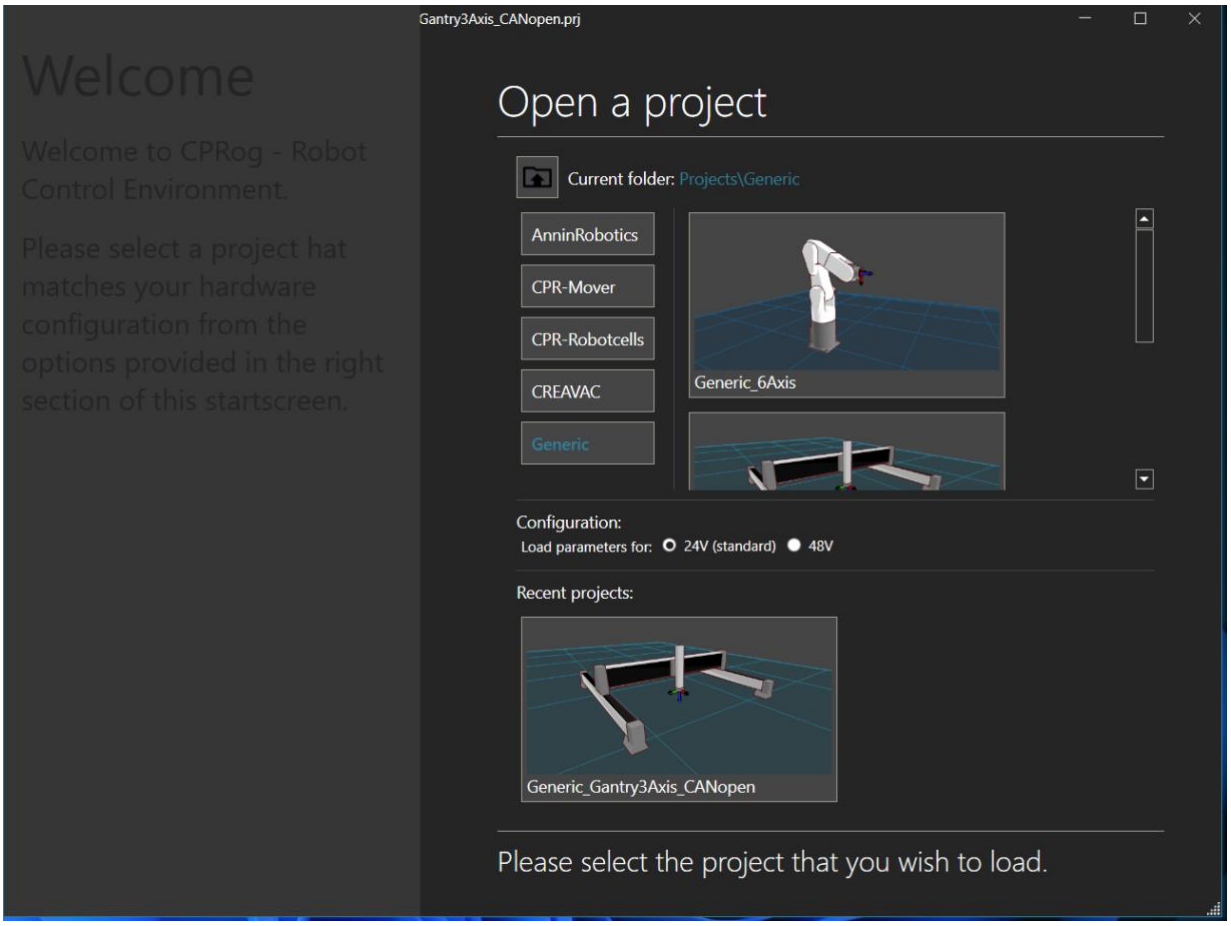

Figure 2: Robot Selection Screen

The robot that we'll be using the is the CPR-Mover6. Select "CPR-Mover" then select "Mover6" as shown below:

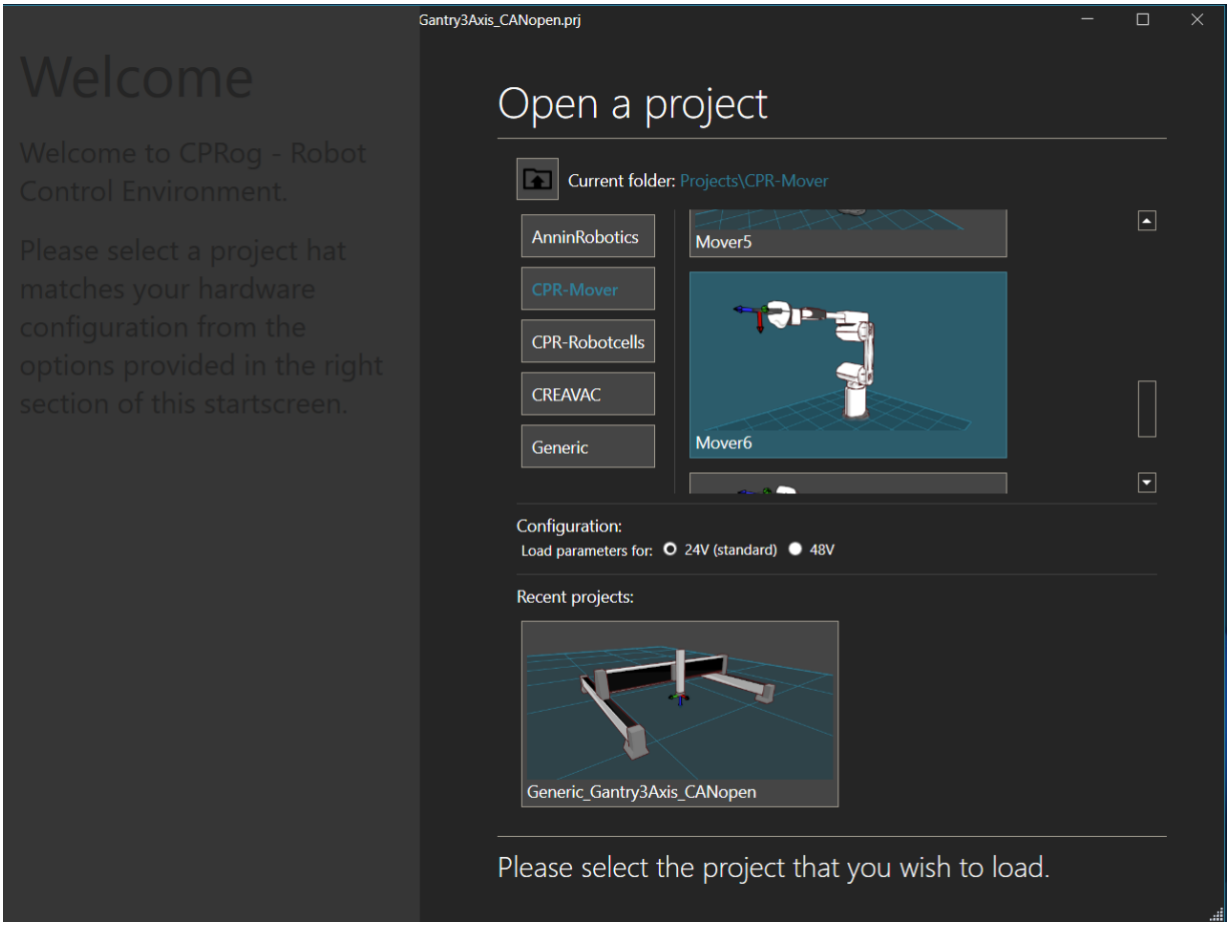

Figure 3: Selecting Mover6

2.2 Familiarisation with the Environment

The CPRog programming environment provides the potential to program and control the Mover6 robot. It is the simplest of the programming environments that we will look at during this lab. This is the first example that we will see of software that can work both online and offline. Online it will move the robot, and the 3D model will represent real movements. In offline mode, the 3D model will reflect the motions the robot will perform; it can be used to experiment with how the robot will perform. You should also refer to the **USER MANUAL**, this will be included on Blackboard.

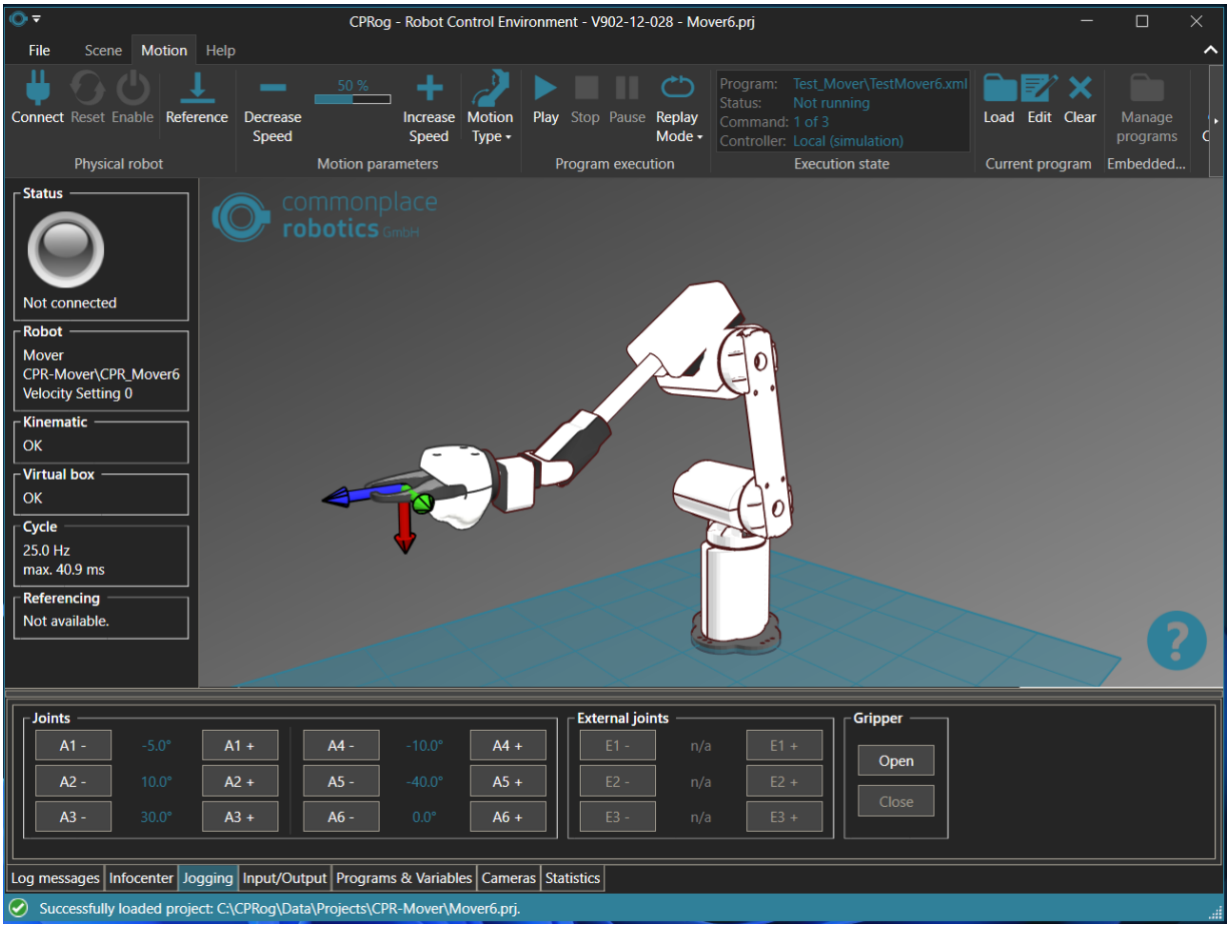

Figure 4: CProg User Interface

Figure 4 shows the main programming interface of the CProg software. In the upper area of the program there are a collection of toolbars and ribbons that control the functionality.

In the “Scene” ribbon, as shown in Figure 5, the 3D view of the camera of the robot shown in the main model window can be changed, using options such as “Select”, “Move within x, y plane”, “Rotate about z axis”, “Zoom” and “Reset view”. This provides mouse-based control of the environment.

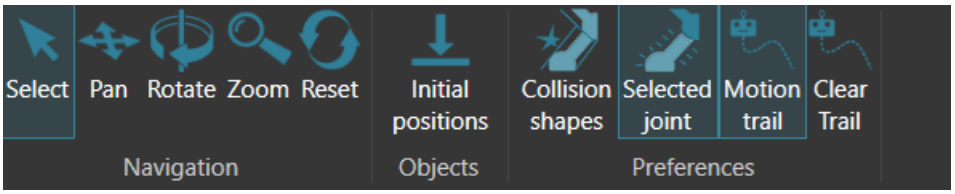

Figure 5: Scene Ribbon, Navigation Toolbar

The “Motion” ribbon sets up the main programming and movement functionality of the robot, it is shown in Figure 6. Across the toolbar are a collection of different buttons that control the robot operation.

Each tab from the left to the right defines different forms of operation. Which will be further defined.

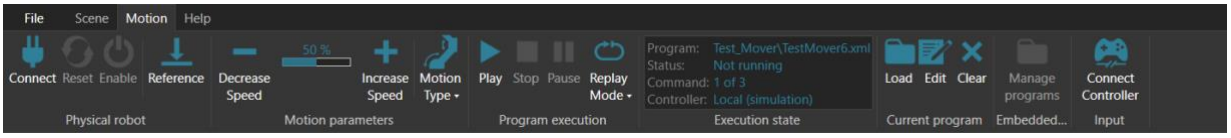

Figure 6: Motion Ribbon, All Toolbars

2.1.1 Connection

The “Connection” tab controls connection from the robot to the PC, and is shown in Figure 7.

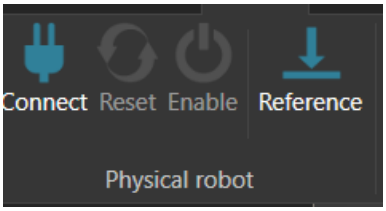

Figure 7: Connection Tab

The three buttons on the left of Figure 7 determine connection to the robot. Once the CProg software is stated the Connection panel on the left-hand side of the screen will look as shown in Figure 8. The left-hand button in the “Connection” tab will connect the PC to the robot, upon clicking it the left-hand pane will move to Figure 9. The right-hand button in the “Connection” tab will enable the motors of the robot, moving the left-hand pane to Figure 10. Any errors can be reset with the centre button, e.g. the robot may say “Overcurrent” if so press “Reset” and then “Enable” if necessary. Once the light is green the robot motors are live.

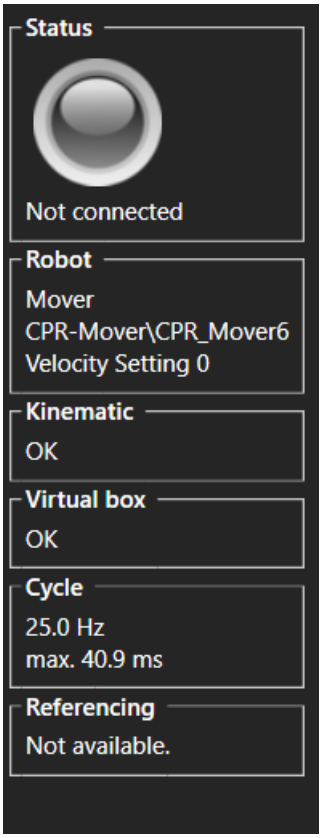

Figure 8: CProg Software once Booted

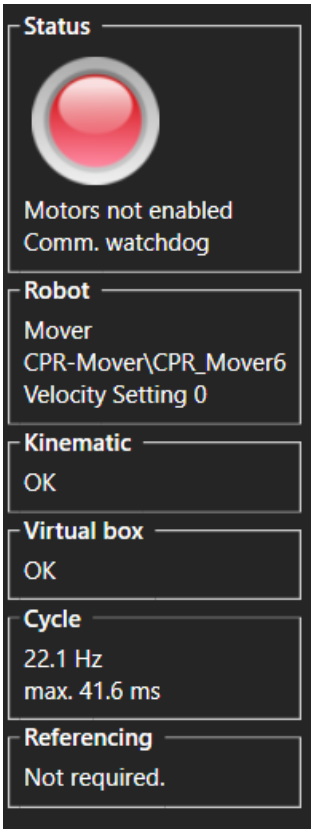

Figure 9: State When Connected

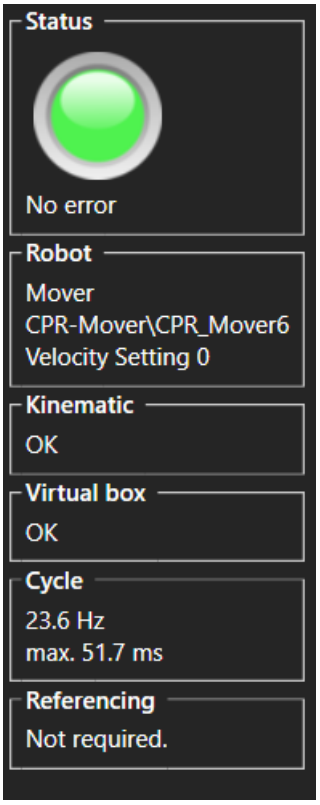

Figure 10: State Once Activated

Task 1: Connect to the robot and activate the motors.

2.1.2 Motion Type

This tab defines the selection of external controls for the robot. For example, the joypad will be lit if one is available and connected. In this lab we will not be using one.

The dropdown list allows definition of the jog frame of the robot. The possible options are shown in Figure 11. These option define the functionality of the jogging buttons at the bottom of the screen. “Joint” is the easiest, each button then moves one of the joints independently.

“Base” provides six buttons, these move the end effector in the frame of the base; i.e. in an x, y, z direction with rotations about each axis relative to the base. If this option is selected, then all joints will move as a movement of the end-effector in the motion is desired. This uses the inbuilt inverse kinematics to control all joints simultaneously.

“Tool” provides six buttons, these move the end effector in the frame of the end effector, i.e in an x, y, z directions with rotations about each axis relative to the end effector. NB: These may then be different from “Cart Base”.

Figure 11: Jog Frame Selection

2.1.3 Motion parameters

The "Motion parameters" tab is shown in Figure 12. This controls the overall speed with which the robot will move as a multiplier of the demand velocity. For example, if the robot is programmed to move at 50% speed (i.e. full speed), but the override is set to 60% it'll move at 30% speed. The plus and minus signs will adjust this.

The override can be used to adjust the jog speeds for the robot. Smaller override values will provide smaller jog movements.

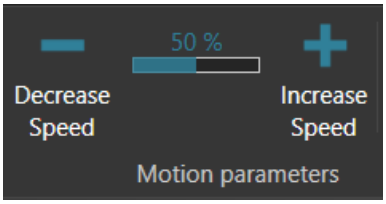

Figure 12: Motion parameters tab

2.1.4 Run Program

The "Run Program" tab, Figure 13, controls the motion of the robot from scripted code. The tab itself contains buttons to Play, Stop and Pause the currently selected file. The dropdown box allows the script to be executed in several modes:

- Single – Run through the program once
- Step – Iterate through the program line by line
- Repeat – Loop through the program.

The filename of the active program is shown below the dropdown list and can be selected through the "Edit Program Tab".

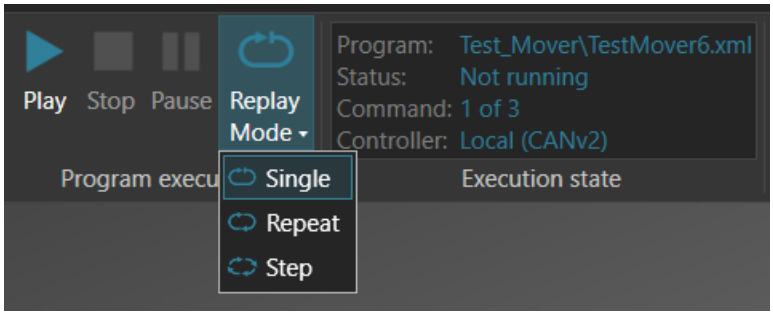

Figure 13: Run Program Tab

2.1.5 Edit Program on PC

The "Edit Program on PC" tab is shown in Figure 14. It contains three buttons that control the loaded program which controls the operation of the arm. The left-hand symbol allows a sample program to be loaded. The middle symbol opens the Program editor which contains allows a series of commands to be programmed into the arm. The right-hand button deletes all commands on the current program.

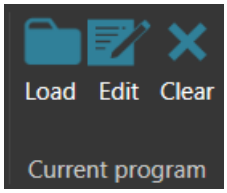

Figure 14: Edit Program Tab

2.1.6 Jogging

Jogging the robot implies that it is being moved, the "Input" tab gave some potential options for how the robot may be jogged, these configure the Jog Section; depending on the selection each will look as indicated below; the frame is then defined as in the "Input" tab, as defined in the "Input" section above. In each of these, the buttons will be lit as shown when the robot is active, otherwise they will be greyed-out. The Open and Close buttons for the gripper will be greyed-out depending on the current state – e.g. If gripper is closed, Open will be lit and Close greyed.

2.1.6.1 Base Jogging

The bottom window will be configured as shown in Figure 15 when "Cart Base" is selected in Figure 11. There is a button which toggles the gripper open or fully closed. The other 12 buttons increase and decrease the values of translation and rotation about the X, Y and Z axis when aligned to the base of the robot.

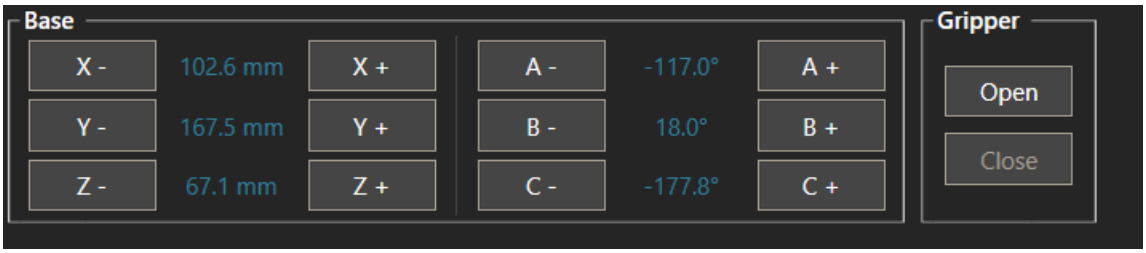

Figure 15: Base or Cartesian Jogging

2.1.6.2 Tool Jogging

The bottom window will be configured as shown in Figure 16 when "Cart Tool" is selected in Figure 11. There is a button which toggles the gripper open or fully closed. The other 12 buttons increase and decrease the values of translation and rotation about the X, Y and Z axis when aligned to the tool of the robot.

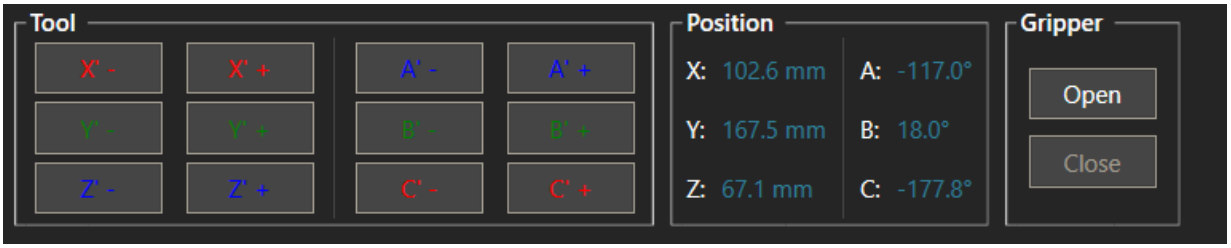

Figure 16: Tool Jogging

2.1.6.3 Joint Jogging

The bottom window will be configured as shown in Figure 17 when "Joint" is selected in Figure 11. There is a button which toggles the gripper open or fully closed. The other 12 buttons increase and decrease the values of rotation about each joint of the arm.

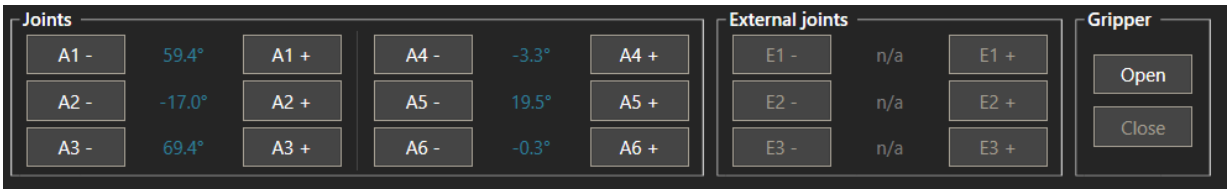

Figure 17: Joint Jogging

Task 2: Explore the difference between the different types of joint input, jogging the robot in each dimension. Explore the difference between "Cart Base" and "Cart Tool".

2.2 Zeroing the Motors and Setting the Initial Configuration

Each time that you use the robot for each session the motors must be zeroed. This ensures that each joint is set to a known, correct position. Without this it is possible that drift will occur through time and the reliability will be compromised. This is a process commonly known as *mastering* this resets joints to a known tolerance and allows us to work more accurately. Because of the configuration of the Mover6 only joints 1-4 require mastering, joints 5 and 6 are servo driven and will reset to a known position.

In order to master the robot, follow the process defined in **Section 5.6** of the manual. The robot is shown in its home position in Figure 21. Suggested home locations for each joint are shown in the photographs below (Figure 18, Figure 19 and Figure 20 – not that the robot faces the label that says "Front" when in home position), note joints 5 and 6 do not require zeroing as they have encoders that provide accurate position:

Figure 18: Joint 4 Position

Figure 19: Joint 3 Position

Figure 20: Joint 1 and Joint 2 Position

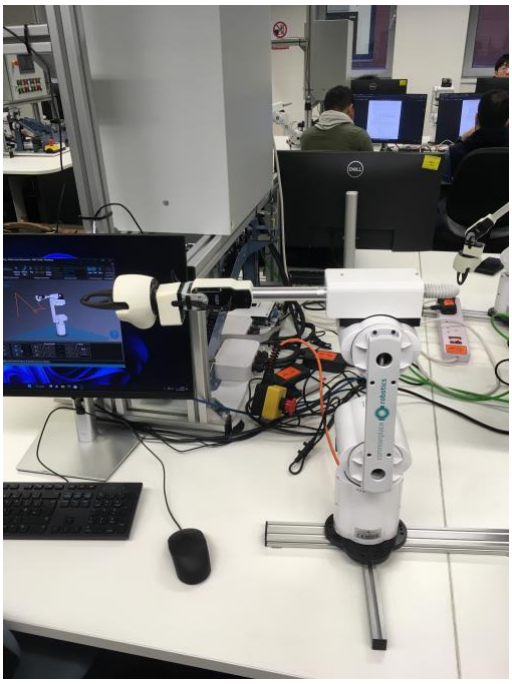

Figure 21: Robot Home Position

Task 3: Develop a process and methodology for zeroing the robot. This can be achieved using the native CPRog software to manually jog the joints to their associated position.

2.3 Programming the Arm

The CProg software can be used as a simulation tool as well as a practical robot interface. This means that we can evaluate how a program will work on the robot before we deploy it. This is important in understanding exactly how the robot will behave and where it will move ***before*** we test it on the

robot. This avoids the potential for any accidental damage as we can understand the behaviour and ensure it is what we desire.

You may turn off the robot having performed the zeroing operation (you'll need to go through this process again before you begin working with it).

You have been provided with a sample program ("SimpleMotion.xml") which demonstrates some different types of motion of the robot. Program files are defined in the Extensible Markup Language (XML) which provides a flexible set of keywords and tags to describe motion predicates. The Program Editor provides a simple interface to write these files so there is no direct need to understand their structure. The sample program is shown in Figure 22. To load this click "Load" and select the file. Then click "Edit" to bring up the programme editor.

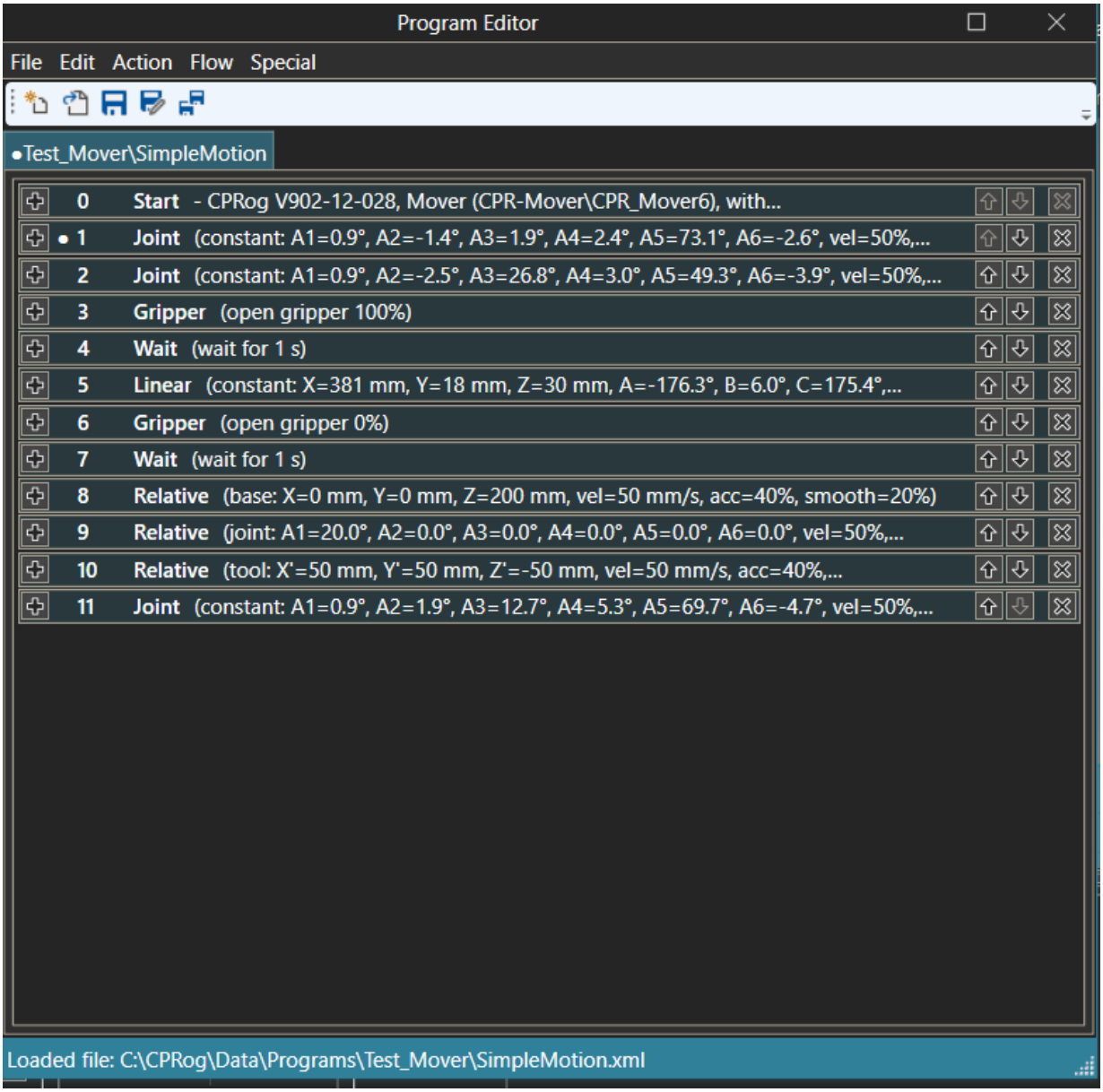

Figure 22: SimpleMotion.xml

Elements can be expanded by clicking the plus button to see more detail:

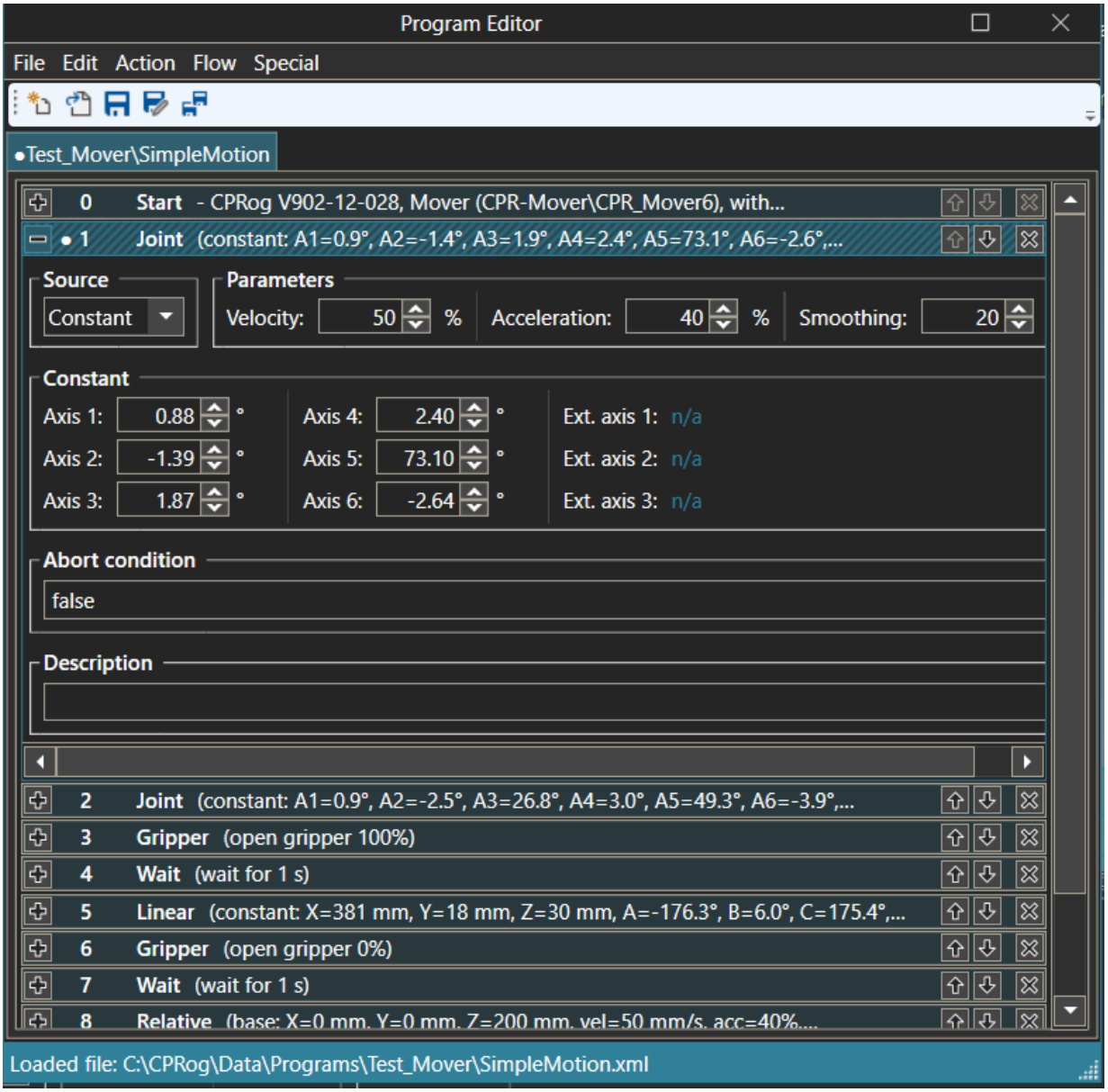

Figure 23: Expanded element showing properties for movement

Task 4: Load the SimpleMotion.xml file into the CProg environment using the "Edit Program Tab". This program is an example file which shows a collection of motions that move the arm, in both joint mode and linear mode. Disconnect the physical robot, and explore the operation of this program. Connect the physical robot and verify that the program performs as it does within the simulator.

If you do not have it the file can be downloaded from:

https://drive.google.com/file/d/1X1FKbfNyeUFnI1SAL2jz16aWzVusGmWL/view?usp=sharing

The robot itself can then be programmed to perform a collection of motions using the movement primitives available within the Program Editor. These are commonly available within the "Action Command" menu (Figure 24) and then can me modified in the "Edit Command" menu (Figure 25).

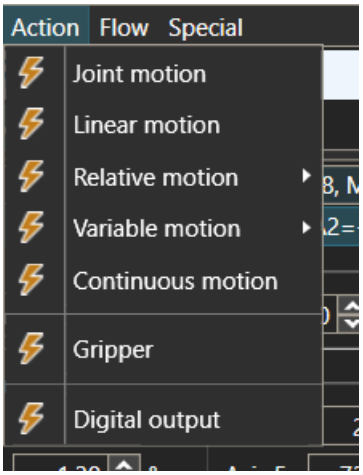

Figure 24: Action Command Menu

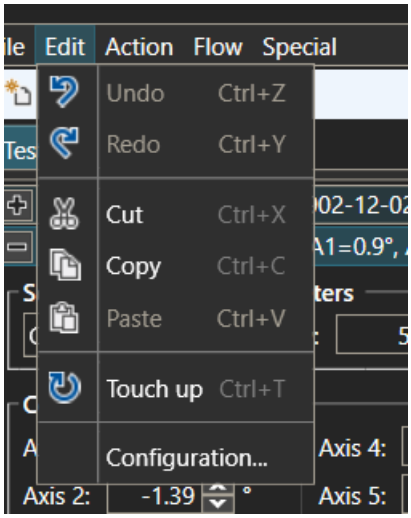

Figure 25: Edit Command Menu

New commands can be added to the program through the "Action Command" menu (Figure 24). There are three primary motion primitives:

1. Linear Motion – Move in a line between the current point and the new desired position and pose; move joints as appropriate using inbuilt inverse kinematics). This is specified in the XML code as shown in Instruction 5 in Figure 22. In this case there are 14 extra components which define the motion:
   1.1. "A1": Joint 1 Angle
   1.2. "A2": Joint 2 Angle
   1.3. "A3": Joint 3 Angle
   1.4. "A4": Joint 4 Angle
   1.5. "A5": Joint 5 Angle
   1.6. "A6": Joint 6 Angle
   1.7. "E1": Set digital output E1
   1.8. "E2": Set digital output E2
   1.9. "E3": Set digital output E3
   1.10. "Vel": Set relative velocity as a percentage of maximum movement. NB: Remember that Override is applied on top of this
   1.11. "Acc": Set acceleration as a percentage of maximum movement
   1.12. "Smooth": Set smoothing of path
   1.13. "condition": Boolean to indicate whether command should be executed or not (if false then command is executed)
   1.14. "Description": Free text field to add comments
2. Joint Motion – Move the joints in a linear fashion between the current joint positions and the new desired joints angles. This is a more robust movement that linear motion as joints will always move between two values. A linear motion may be impossible as the inverse kinematic solution may be impossible to realise as a joint may exceed pre-set limits. In this case there are 14 extra components which define the motion:
   2.1. "X": X position of the end effector

2.2. “Y”: Y position of the end effector
2.3. “Z”: Z position of the end effector
2.4. “A”: Rotation about the Z axis
2.5. “B”: Rotation about the Y axis
2.6. “C”: Rotation about the X axis
2.7. “E1”: Set digital output E1
2.8. “E2”: Set digital output E2
2.9. “E3”: Set digital output E3
2.10. “Vel”: Set relative velocity as a percentage of maximum movement. NB: Remember that Override is applied on top of this
2.11. “Acc”: Set acceleration as a percentage of maximum movement
2.12. “Smooth”: Set smoothing of path
2.13. “condition”: Boolean to indicate whether command should be executed or not (if false then command is executed)
2.14. “Description”: Free text field to add comments

3. Relative Motion – Move in a relative way to the current position. This consists of three types:
   3.1. Relative Base – move to a Cartesian position relative to the current position in the base frame:
      3.1.1. “X”: X displacement in the end effector position
      3.1.2. “Y”: Y displacement in the end effector position
      3.1.3. “Z”: Z displacement in the end effector position
      3.1.4. “Vel”: Set relative velocity as a percentage of maximum movement. NB: Remember that Override is applied on top of this
      3.1.5. “Acc”: Set acceleration as a percentage of maximum movement
      3.1.6. “Smooth”: Set smoothing of path
      3.1.7. “condition”: Boolean to indicate whether command should be executed or not (if false then command is executed)
      3.1.8. “Description”: Free text field to add comments
   3.2. Relative Tool - move to a Cartesian position relative to the current position in the tool frame:
      3.2.1. “X”: X displacement in the end effector position
      3.2.2. “Y”: Y displacement in the end effector position
      3.2.3. “Z”: Z displacement in the end effector position
      3.2.4. “Vel”: Set relative velocity as a percentage of maximum movement. NB: Remember that Override is applied on top of this
      3.2.5. “Acc”: Set acceleration as a percentage of maximum movement
      3.2.6. “Smooth”: Set smoothing of path
      3.2.7. “condition”: Boolean to indicate whether command should be executed or not (if false then command is executed)
      3.2.8. “Description”: Free text field to add comments
   3.3. Relative Joint – move joints relative to current position:
      3.3.1. “A1”: Joint 1 Angle adjustment angle
      3.3.2. “A2”: Joint 2 Angle adjustment angle
      3.3.3. “A3”: Joint 3 Angle adjustment angle
      3.3.4. “A4”: Joint 4 Angle adjustment angle
      3.3.5. “A5”: Joint 5 Angle adjustment angle

3.3.6. “A6”: Joint 6 Angle adjustment angle
3.3.7. “E1”: Set digital output E1 adjustment angle
3.3.8. “E2”: Set digital output E2 adjustment angle
3.3.9. “E3”: Set digital output E3 adjustment angle
3.3.10. “Vel”: Set relative velocity as a percentage of maximum movement. NB: Remember that Override is applied on top of this
3.3.11. “Acc”: Set acceleration as a percentage of maximum movement
3.3.12. “Smooth”: Set smoothing of path
3.3.13. “condition”: Boolean to indicate whether command should be executed or not (if false then command is executed)
3.3.14. “Description”: Free text field to add comments

The “Gripper” commands cause the gripper to open and close to a set amount, the example shows fully closed (“0”) and fully open (“100”). When the gripper is commanded it is important to include a “wait” command which pauses the robot for at least 1s to give the gripper time to operate.

When a command is added, it will use the current position of the robot. This can be changed by using the manual controls to jog the robot into different positions, and then using the “TouchUp” command in Figure 25 which will convert the positions in the line to the new positions of the robot arm. This process of “TouchUp” is used in robotics to teach present positions.

Task 5: Rename the supplied file "BlankMotion.xml" to a suitable name and open it via the program editor. Here we will explore different movement types and styles and combinations by moving the manipulator around the workspace using the different styles of movements.

Move the robot around a cube of positions where it moves to each corner, align the robot so that at each of the 4 corners on top of the cube the gripper points to the centre of the cube. An example of the block configuration to use and end effector position is shown in Figure 26.

Repeat the same movement patterns using each of the motion styles.

What considerations do you have to make about the order that this has to be undertaken in?

Are all of the movement combinations possible in all of the operational modes?

If you do not have the file it can be downloaded from:

https://drive.google.com/file/d/1fPKmhtPaUiY5cD5Fb8Tw5A8T3_Cnnog0/view?usp=sharing

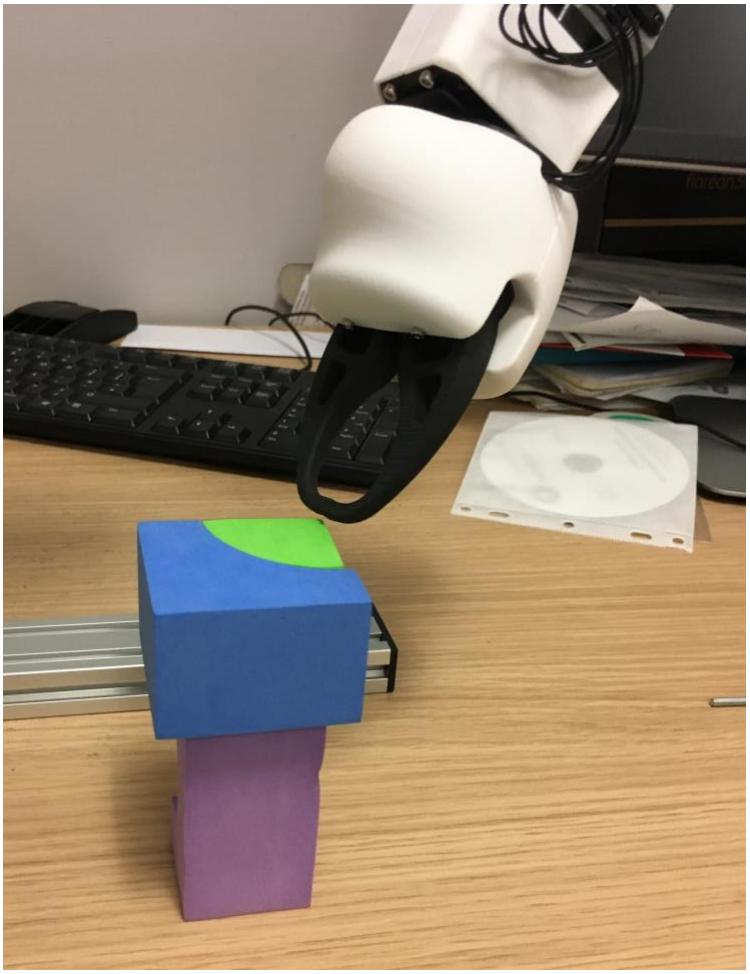

Figure 26: Block Position and Example End Effector Position

Task 6: Create a program that constructs the structure shown in Figure 28 from the resource blocks shown in Figure 27. You should begin by ensuring that the robot joints are correctly mastered. Then

1. ***Decide on the order of handling the blocks in the most effective manner.***
2. ***Setup the environment so that you have a record of the starting block positions with orientations as in Figure 27. This is important as you do not have sensing methodology to change robot operation during the run, so your code is pre-defined to the positions. You could use a piece of paper orientated to the robot in order to define definite positions.***
3. ***Manipulate each block in turn so that is in the correct configuration (Figure 28).***
4. ***As you develop your program, note any inaccuracies that develop; when does the manipulator need re-mastering? When appropriate, go through the process as previously detail and developed within Task 3.***

Complete these steps for the blocks:

1. ***In the simple positions shown in Figure 27***
2. ***Repeat re-orienting blocks:***
    a. ***Place the arch piece so that it stands as an arch***
    b. ***Lie the cylinder on its side***
    c. ***Place the arch piece so that it stands as an arch and the cylinder is on its side***

How do these configurations change the order the pieces must be handled and considerations about how they are moved in the workspace?

NB: Whilst moving toy blocks around sounds a long way from a robotic application in industry, this is where all robot operators start, for example it is common practice within industry to require an operator to build children's structures before they are allowed to move onto handling more expensive or dangerous items!

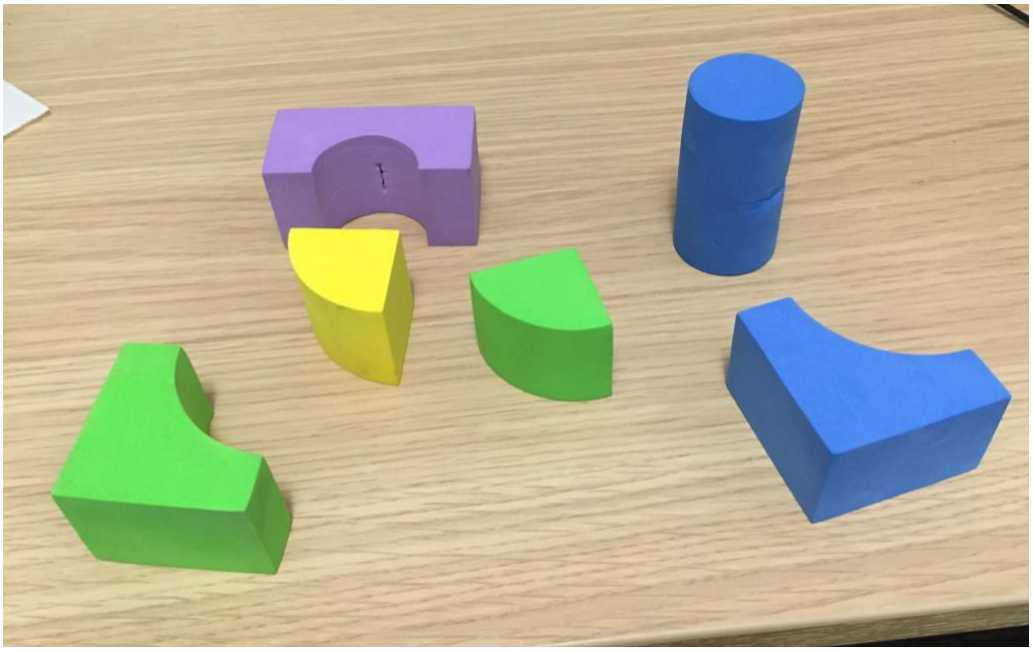

Figure 27: Selection of Blocks

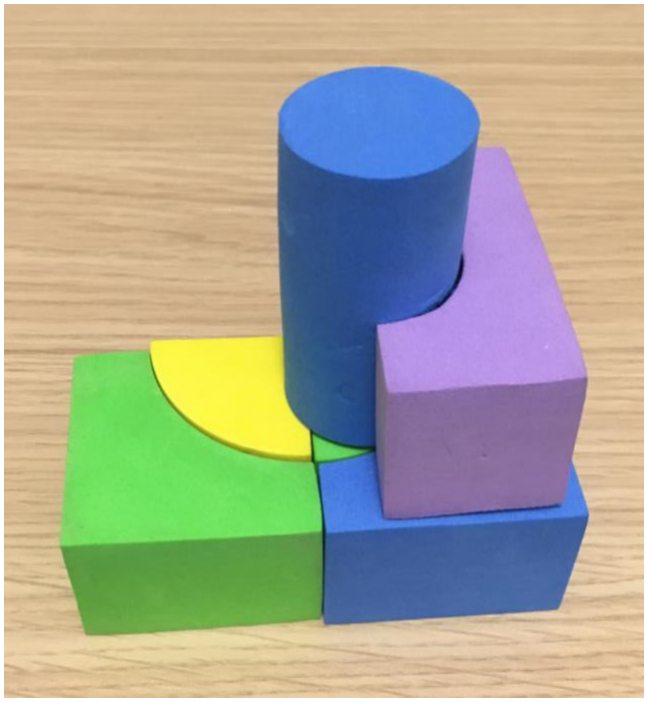

Figure 28: Final Block Form