Robotics Practical - Report

Topic 6 - ABB IRB120

Group 29:

Simon GILGIEN 253797

Jean LESUR 284531

Filip SLEZAK 286557

Assistants:

Luzius KRONIG

Louis MUNIER

Date : March 3, 2021

Contents

1 Introduction

This Practical Work aims at familiarizing us with the basics of industrial robots. We will implement two simple applications : a Maze escape and a TicTacToe game. This will be done using the two different code development tools (the Control Pad and the RAPID suite).

2 Theoretical questions

1. What kind of robot is the IRB120? (serial, parallel)

   The ABB IRB120 is a serial robot.

2. How many degrees of freedom does ABB IRB120 have and what type are they?

   The robot itself has 6 degrees of freedom. All of those DOFs are rotational DOF. In this practical work, the robot uses a gripper that add another DOF, which is translational.

3. Compare serial and parallel robots in terms of : Workspace, Inertia, Stiffness, Precision, Control, Singularities.

Robot Type	Serial	Parallel
Workspace	large	small
Inertia	high	low
Stiffness	low	high
Precision	low	high
Control	simple	hard
Singularities	exist	shouldn't exist

   Justifications:

   - Workspace: for comparable robots, the serial one can deploy the tools further from its base
   - Inertia: the serial robot has to move the motors controlling the kinematic chain while the parallel one has most motors at its base
   - Stiffness: the parallel robot is stiffer as the mechanism forms loops to the base
   - Precision: lower for the serial robot as uncertainty accumulates through the chain of motors
   - Control: simpler for a serial robot as we have a direct control of position - the inverse dynamics is more straight forward
   - Singularities: both robots may have singularities but they are more likely in the serial robot (particularly at the edge of its workspace) and in the case of a parallel mechanism we usually try to avoid them during conception as they may result in the unwanted gain of a degree of freedom.

4. What are the possible applications of ABB IRB 120?

   This robot aims at manipulating or assembling small packages or devices in the industry. Especially in the electronic, food and medical domains.

5. What is the end-effector position repeatability? What is the maximum payload?

   The end effector position repeatability is given in the datasheet as 0.01mm. The maximum payload is equal to 3kg in general use but can weight up to 4kg if the wrist is aligned vertically (alleviating the load on the last link).

6. What kind of actuator is used for the gripper?

   The gripper is a pneumatic gripper, that is, the opening/closing motion is generated by air compression and decompression in cylinders. In our case, we had a crayon attached with adhesive tape as our tool.

7. What kind of objects can you grasp with that gripper (orientation, size etc.)?

   The pneumatic actuator allows for a constant grip force for objects of different shapes and sizes but their weight should be similar.

8. Is this gripper well dimensioned for the robot? Please justify.

   This gripper is particularly long and takes up a great portion of the robot's workspace so it restricts the working area (we had to reposition our drawing paper although we were already near its base).

9. Why is important to define the center of gravity for the new tool?

   It is important in order to adapt the internal dynamics model of the robot, allowing smoother operations.

10. What is the advantage of having defined a new work object?

    We change the reference point of our robot for the internal calculations to be made according to our point of interest, the tool, which is more intuitive for the operator to control.

11. Explain the arguments of these functions:

    a. **MoveL p10, v1000, z50, tool0** : The instruction moveL tells the robot to move the Tool Center Point (TCP) along a linear path. The first argument (**p10**) defines the target towards which the robot will go following a linear path from its current position. The second argument (**v1000**) defines the speed of the robot. The example value is a predefined value. The third argument (**z50**) defines the distance to the target at which the next instruction should be executed. Executing the next instruction before the robot is exactly at the target allows to move at a constant speed, without taking sharp corner implying high accelerations. The last argument (**tool0**) gives an information about the tool mounted on the robot. It contains the position of the Tool Center point w.r.t. the wrist of the robot, as well as the mass and inertia of the tool to be used in the dynamic model of the robot.

    b. **Offs(p10, 50, 50, 0)** : The function Offs apply an offset to a point given as first argument. Here, the function returns a point with a displacement of 50 mm on both the x and y axis from the point **p10**. The first argument is the reference point, and the next three arguments are the displacement along the x, y and z axis given in millimeters in the object coordinate system.

3 First exercise : The Maze

The Maze exercise was mainly to discover the basic motion functions (Move L and MoveC, respectively for linear and circular motion) of the robot.

Given any initial pose of the robot in its workspace, we had to make the robot move its Tool Center Point inside a maze. The maze was pretty simple : it had a unique possible path and was only made of straight and perfectly roundish corridors. The use of MoveL and MoveC was thus pretty evident.

In order to achieve this, we moved the robot in places of interest in the maze (mainly at the corners of the corridor) with the help of the jogging command on the Control Pad. By combining MoveL and MoveC commands, we ran through the sequence of interest points in one go to draw the desired path.

On the first try, we did not consider the 'z' argument of the command. As a consequence, the robot was not moving exactly through the points we had defined, thus resulting in colliding with the walls. Once the 'z' argument was set to 0, we were able to perfectly draw the path in the maze !

You will find the code for this exercise in appendix.

4 Second exercise : TicTacToe

The goal of the second exercise was to implement a TicTacToe game on the robot. The robot was used to draw the Xs and Os in the grid, according to the commands given by the 2 human payers on the Control Pad. In addition to the game's rules, some additional requirements came from the practical's assignment (selection of Best Of X, indication of what player needs to make a move).

We implemented this exercise using a bottom-up approach. That is, we began to write the two necessary functions to draw the Xs and the Os. Then, we focused on a simple game loop between the two players and when this loop was perfectly working, we moved to the implementation of the Best Of X functionality.

The drawing paper we had to play on was composed of 12 small grids, each cell of the grid was 2 by 2 centimeters. To move from one cell to another, or from one grid to another, we used the `Offs` function along with the simple functions DIV and MOD .

Optimization is admittedly lacking in some places but it was not required, the performance is acceptable as is. As we had trouble defining a new workspace on the Controller (it was not taken into account), we had offsett problems when moving from one grid to another.

You will find the code for this exercise in appendix.

5 Conclusion

In conclusion, we were glad to attend this Practical Work despite the fact that it was more coding in an unknown language than using the robot. However, we discovered the basics of industrial robots : how to make the robot move with the Control Pad (in joint or work space), how to edit and launch code on the robot directly from the Control Pad and also how to use the RAPID code editor.

6 Annexes

6.1 Code for the Maze

```
MODULE Module1
TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[243.16,-171.596,185.38],
                    [0.990529,-0.000492899,0.00463643,0.137226]],
                    [[0,0,0],[1,0,0,0]]];
TASK PERS tooldata tool1:=[TRUE,[[0,0,290],[1,0,0,0]],[1,[0,0,150],
                    [1,0,0,0],0,0,0]];
CONST robtarget p10:=[[-11.01,-6.64,-0.49],
                    [0.0726311,-0.124891,-0.989502,0.00359438],
                    [-1,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p30:=[[28.56,73.33,165.11],
                    [0.0726047,-0.124945,-0.989497,0.00358957],
                    [-1,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p20:=[[28.76,120.24,102.69],
                    [0.0725569,-0.124944,-0.989501,0.00362694],
                    [-1,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p40:=[[-34.62,194.26,-4.18],
                    [0.0726758,-0.124946,-0.989492,0.00361997],
                    [0,-1,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p60:=[[-3.70,196.16,-2.21],
                    [0.072713,-0.124909,-0.989494,0.00361764],
                    [0,-1,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p50:=[[-19.53,206.26,-3.20],
                    [0.0727152,-0.124919,-0.989492,0.00362148],
                    [0,-1,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p80:=[[79.41,193.10,-3.24],
                    [0.0727355,-0.124838,-0.989501,0.00361403],
                    [0,-1,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p70:=[[37.55,161.02,-2.22],
```

```
                    [0.072727,-0.124898,-0.989494,0.00361669],
                    [0,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p90:=[[75.26,319.35,-2.24],
                    [0.0727507,-0.124826,-0.989501,0.00361007],
                    [0,0,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p100:=[[200.31,328.24,-2.26],
                    [0.0727756,-0.124825,-0.9895,0.00362433],
                    [0,0,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p110:=[[211.10,117.20,-3.27],
                    [0.0727722,-0.12481,-0.989502,0.00363251],
                    [0,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p120:=[[119.94,21.60,-5.30],
                    [0.0728087,-0.124822,-0.989498,0.00356747],
                    [-1,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p130:=[[73.11,46.74,-5.34],
                    [0.0727427,-0.124889,-0.989494,0.003501],
                    [-1,-1,-1,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p140:=[[150.24,193.83,-4.35],
                    [0.0728168,-0.12487,-0.989491,0.00343902],
                    [0,-1,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p150:=[[147.21,288.82,-3.35],
                    [0.0728569,-0.124867,-0.989489,0.00345241],
                    [0,0,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p160:=[[147.18,288.81,71.65],
                    [0.0729089,-0.124868,-0.989485,0.00347673],
                    [0,0,0,0],
                    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
PROC main()
        MoveL p10, v1000, z5, tool1\WObj:=wobj1;
        MoveL p40, v1000, z5, tool1\WObj:=wobj1;
        MoveC p50, p60, v1000, z0, tool1\WObj:=wobj1;
        MoveC p70, p80, v1000, z0, tool1\WObj:=wobj1;
        MoveL p90, v1000, z5, tool1\WObj:=wobj1;
        MoveL p100, v1000, z5, tool1\WObj:=wobj1;
        MoveL p110, v1000, z5, tool1\WObj:=wobj1;
        MoveL p120, v1000, z5, tool1\WObj:=wobj1;
        MoveL p130, v1000, z5, tool1\WObj:=wobj1;
        MoveL p140, v1000, z5, tool1\WObj:=wobj1;
        MoveL p150, v1000, z5, tool1\WObj:=wobj1;
```

```
        MoveL p160, v1000, z5, tool1\WObj:=wobj1;
ENDPROC
ENDMODULE
```

6.2 Code for the Tic Tac Toe

```
MODULE Module1
TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[243.16,-171.596,185.38],
                      [0.990529,-0.000492899,0.00463643,0.137226]],
                      [[0,0,0],[1,0,0,0]]];
TASK PERS tooldata tool1:=[TRUE,[[0,0,290],[1,0,0,0]],[1,[0,0,150],
                      [1,0,0,0],0,0,0]];
CONST robtarget p10:=[[-10.30,-13.32,-5.45],
                      [0.0729905,-0.124794,-0.989488,0.00348412],
                      [-1,-1,-1,0],
                      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

VAR num bon;
VAR num player := 1;
VAR num firstplayer := 1;
VAR num score{2} := [0,0];
VAR num gridNB := 0;

PROC main()
    TPReadNum bon, "BO3, BO5, BO7 ?";
    TPWrite NumToStr(bon, 0);

    WHILE (bon <> 3 AND bon <> 5 AND bon <> 7) DO
        TPReadNum bon, "BO3, BO5, BO7 ?";
    ENDWHILE
    WaitTime(2);

    WHILE bon <> 0 DO
        player := firstplayer;
        IF firstplayer = 1 THEN
            firstplayer := 2;
        ELSE
            firstplayer := 1;
        ENDIF

        IF gridNB < 12 THEN
            run_single_game Offs(p10, (gridNB DIV 4) * 77,
                                      (gridNB MOD 4) * 77, 0);
            gridNB := gridNB + 1;
        ELSE
            TPWrite "Error : no more grids available !";
        ENDIF
        bon := bon - 1;
```

```
    ENDWHILE
ENDPROC

PROC run_single_game(robtarget gridOffset)
    VAR num grid{9} := [0,0,0,0,0,0,0,0,0];
    VAR bool win := FALSE;
    VAR bool posValid;
    VAR num position;
    VAR robtarget caseOffset;
    VAR num count := 0;

    WHILE win <> TRUE AND count < 9 DO
        posValid := FALSE;

        WHILE posValid = FALSE DO
            IF player = 1 THEN
                TPWrite "Player 1 must play";
            ELSE
                TPWrite "Player 2 must play";
            ENDIF

            TPReadNum position, "What position ?";
            IF position <= 9 AND position >= 1 THEN
               IF grid{position} = 0 THEN
                   posValid := TRUE;
               ENDIF
            ENDIF
        ENDWHILE

        caseOffset := Offs(gridOffset, ((position-1) DIV 3) * 20,
                           ((position-1) MOD 3) * 20, 0);
        grid{position} := player;

        IF player = 1 THEN
            draw_cross(caseOffset);
        ELSE
            draw_circle(caseOffset);
        ENDIF

        MoveL Offs(gridOffset, 0, 0, 150),
              v1000, z0, tool1\WObj:=wobj1;

        count := count + 1;

        IF grid{1} = grid{2} AND grid{2} = grid{3}
                             AND grid{1} = player THEN
            win := TRUE;
```

```
        ELSEIF grid{4} = grid{5} AND grid{5} = grid{6}
                                 AND grid{4} = player THEN
            win := TRUE;
        ELSEIF grid{7} = grid{8} AND grid{8} = grid{9}
                                 AND grid{7} = player THEN
            win := TRUE;
        ELSEIF grid{1} = grid{4} AND grid{4} = grid{7}
                                 AND grid{1} = player THEN
            win := TRUE;
        ELSEIF grid{2} = grid{5} AND grid{5} = grid{8}
                                 AND grid{2} = player THEN
            win := TRUE;
        ELSEIF grid{3} = grid{6} AND grid{6} = grid{9}
                                 AND grid{3} = player THEN
            win := TRUE;
        ELSEIF grid{1} = grid{5} AND grid{5} = grid{9}
                                 AND grid{1} = player THEN
            win := TRUE;
        ELSEIF grid{3} = grid{5} AND grid{5} = grid{7}
                                 AND grid{3} = player THEN
            win := TRUE;
        ENDIF

        IF win THEN
            score{player} := score{player} + 1;
            TPWrite "This is a win !";
        ENDIF

        IF player = 1 THEN
            player := 2;
        ELSE
            player := 1;
        ENDIF
    ENDWHILE

    IF (count = 9 AND win = FALSE) THEN
        TPWrite "This is a draw !";
        score{1} := score{1} + 0.5;
        score{2} := score{2} + 0.5;
    ENDIF
    TPWrite "Player 1 score is :"\Num:=score{1};
    TPWrite "Player 2 score is :"\Num:=score{2};
ENDPROC

PROC draw_cross(robtarget offset)
    MoveL Offs(offset,5,5,10), v1000, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset,5,5,0), v200, z0, tool1\WObj:=wobj1;
```

```
    MoveL Offs(offset,15,15,0),v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset,15,15,10),v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset,5,15,10), v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset, 5, 15,0), v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset, 15,5,0), v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset,15,5,10), v1000, z0, tool1\WObj:=wobj1;
ENDPROC

PROC draw_circle(robtarget offset)
    MoveL Offs(offset, 10, 5, 10), v1000, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset, 10, 5, 0), v200, z0, tool1\WObj:=wobj1;
    MoveC Offs(offset, 15, 10, 0), Offs(offset, 10, 15, 0),
          v200, z0, tool1\WObj:=wobj1;
    MoveC Offs(offset, 5, 10, 0), Offs(offset, 10, 5, 0),
          v200, z0, tool1\WObj:=wobj1;
    MoveL Offs(offset, 10, 5, 10), v1000, z0, tool1\WObj:=wobj1;
ENDPROC
ENDMODULE
```