



Robotics practicals

Topic 6 - ABB IRB120

Microengineering - Spring Semester 2021

| | | |
|-------------------|-------------------------------|--|
| Assistants | Luzius Kronig Louis Munier | luzius.kronig@epfl.ch louis.munier@epfl.ch |
| Place | ME A0 407 | |

1 Objectives

This practical work (PW) aims at giving the possibility to familiarize students with an industrial robot currently used in the industry.

Students will have the opportunity to discover the potential that serial robots offer.

Students will learn the basics through simple exercises using the manual control, then they will programmatically define robot actions. It is awaited that the students prepare the PW at home before coming to the PW. The expected workload of each student is the following: 2h preparation, 4h PW, 2h report.

2 Robot

IRB 120 is a small 6-axis industrial robot and the smallest robot from ABB with only 25kg. The IRB 120 is a serial robot with six rotational joints, 3 for positioning end-effector (extremity of the robot) and 3 for orienting it. All the degrees of freedom (DOF) are actuated through AC motor and their relative position is monitored by optical encoders. The robot can be programmed either by the console or by a PC interface via a virtual environment.

The robot is used for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors. The robot handles a payload of up to 3kg (4kg with its wrist down) and with its reach of 580mm, the robot is able to carry out a series of operations using flexible rather than hard automated solutions.

For the PW the robot will be equipped with a pneumatic gripper holding a pen.

The main features of the robot ABB IRB 120 are presented thereafter.



Figure 1: The ABB IRB 120 – [cgtrader.com]

IRB 120

Specification

| Variants | Reach | Payload | Armload |
|---------------|--------|-------------|---------|
| IRB 120-3/0.6 | 580 mm | 3 kg (4kg)* | 0.3 kg |

Features

| | |
|--------------------------|---|
| Integrated signal supply | 10 signals on wrist |
| Integrated air supply | 4 air on wrist (5 bar) |
| Position repeatability | 0.01 mm |
| Robot mounting | Any angle |
| Degree of protection | IP30 |
| Controllers | IRC5 Compact / IRC5 Single cabinet or Panel mounted |

Movement

| Axis movements | Working range | Maximum speed |
|-----------------|----------------|---------------|
| Axis 1 Rotation | +165° to -165° | 250 °/s |
| Axis 2 Arm | +110° to -110° | 250 °/s |
| Axis 3 Arm | +70° to -90° | 250 °/s |
| Axis 4 Wrist | +160° to -160° | 320 °/s |
| Axis 5 Bend | +120° to -120° | 320 °/s |
| Axis 6 Turn | +400° to -400° | 420 °/s |

Performance

1 kg picking cycle

| | |
|---------------------------|---------------------|
| 25 x 300 x 25 mm | 0.58 s |
| Max TCP velocity | 6.2 m/s |
| Max TCP acceleration | 28 m/s ² |
| Acceleration time 0-1 m/s | 0.07 s |

Electrical connections

| | |
|--------------------|---------------------|
| Supply voltage | 200-600 V, 50/60 Hz |
| Rated power | |
| Transformer rating | 3.0 kVA |
| Power consumption | 0.25 kW |

Physical

| | |
|------------------------|--------------|
| Dimension robot base | 180 x 180 mm |
| Dimension robot height | 700 mm |
| Weight | 25 kg |

Environment

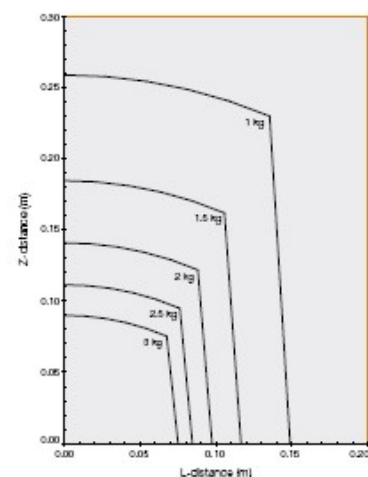
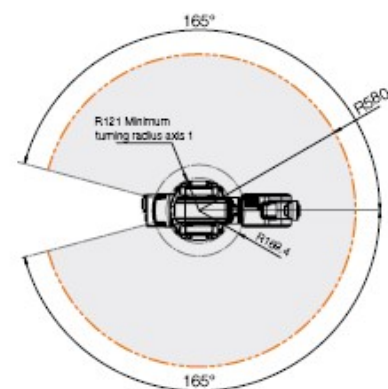
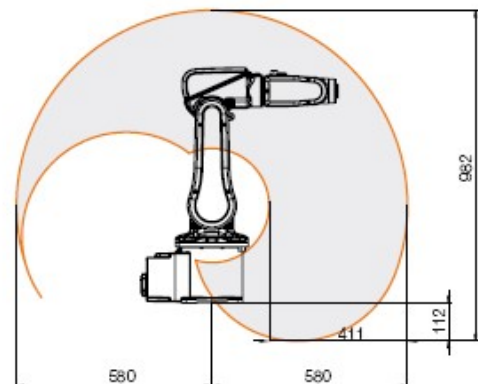
Ambient temperature for Robot manipulator:

| | |
|-------------------------------------|---|
| During operation | +5°C (41°F) to +45°C (122°F) |
| Relative transportation and storage | -25°C (-13°F) to +55°C (131°F) |
| For short periods | up to +70°C (158°F) |
| Relative humidity | Max 95% |
| Noise level | Max 70 dB (A) |
| Safety | Safety and emergency stops 2-channel safety circuits supervision 3-position enabling device |
| Emission | EMC/EMI-shielded |

* With vertical wrist

Data and dimensions may be changed without notice

Working range at wrist center & load diagram



3 Safety



Despite IRB 120 is a relatively light robot it is extremely powerful even when working at low speeds. That's why dealing with this tool requires attention and caution. Therefore, the following rules are to be respected **at all time**:

1. Always maintain a safety distance of at least 1m when the robot is moving or while you manipulate the console.
2. Only one person is allowed to control the robot at the time. This person has to stop the robot immediately if anyone enters the workspace of the robot.
3. Accurately verify your program before running it.
4. Always keep an eye on the robot, its workspace can be surprisingly big.
5. **If anything goes wrong, stop the robot immediately.**

Accidents always appear when people start to be confident using the robot. Keep these rules in mind.

4 Schedule

- Introduction
- Safety
- Exercise 1: Theoretical questions
- Introduction to the robot use
 - Trajectories and coordinate systems
 - Tool and world objects
- Exercise 2: Maze
- Introduction to Robot Studio
- Exercise 3: TicTacToe game

5 Report

A report of the whole practical work must be written and uploaded to *Moodle*¹ at latest one week after the end of the session (Monday 17:00).

The report should not be longer than 4 pages of content, first page and annexes excluded. The following elements, at least, are expected to figure in the report:

- First page including:
 - Title of the practical work
 - Date
 - Group number and member's names
 - Name of the assistant(s)
- Introduction
- Answers to all theoretical questions.
- First exercise: Maze. This section must contain, at least, the following elements:
 - What was requested?
 - What was done?
 - What were the results?
- Second exercise: TicTacToe. This section must contain, at least, the following elements:
 - What was requested?
 - What was done?
 - How is the code structured and how is it working?
 - What were the results?
- Conclusion
- Annex: Code of all the programs developed during the session

The report shall be written in English.

¹ <https://moodle.epfl.ch/mod/assign/view.php?id=835982>

6 Exercise 1: Theoretical questions



Home preparation: Questions 1, 2, 3, 4, 5, 6, 11



For the report, copy the question ([Q]) and put your answer ([A]) beneath

6.1 General questions

1. What kind of robot is the IRB120? (serial, parallel)
2. How many degrees of freedom does ABB IRB120 have and what type are they?
3. Compare serial and parallel robots in terms of
 - a. Workspace
 - b. Inertia
 - c. Stiffness
 - d. Precision
 - e. Control
 - f. Singularities
4. What are the possible applications of ABB IRB 120?
5. What is the end-effector position repeatability? What is the maximum payload?


6.2 Robot introduction

6. What kind of actuator is used for the gripper?
7. What kind of objects can you grasp with that gripper (orientation, size etc.)?
8. Is this gripper well dimensioned for the robot? Please justify.
9. Why is important to define the centre of gravity for the new tool?
10. What is the advantage of having defined a new work object?

6.3 Maze


11. Explain the arguments of these functions:
 - a. MoveL p10, v1000, z50, tool0
 - b. Offs(p10, 50, 50, 0)


7 Exercise 2 – Maze


 *Don't forget to save a copy of your code and put it in the appendix of the report.*


The task consists of making the robot draw a line into a maze. Using the embedded programming, make a program that does all the path (from start to end), starting from any point in the robot workspace. Try different values for speed and zone data to understand how these parameters behave together.

8 Exercise 3 – TicTacToe Game

 *Home preparation: Understand the basic principles of the RAPID language (explained in detail in the document “RAPID overview”, available on Moodle), especially how to create and call functions.*

 *Home preparation: Read, understand and be able to implement the selected RAPID functions. You should be able to efficiently search in the RAPID datasheet (“RAPID full datasheet”, available on Moodle)*

 *Home preparation: Create a pseudocode for the TicTacToe Game*

 *Don't forget to save a copy of your code and put it in the appendix of the report.*

Students are asked to implement a 2 player's version of the famous game TicTacToe² (Jeu du Morpion). An A3 format sheet containing 9 empty boards will be provided at the beginning of the session. The application should respect the following specifications:

- 2 Players: Player 1 has the X and player 2 the O.
- Switch the player who starts at each set
- Game Mode selection: BO3, BO5 or BO7 (BO3 = Best of 3 games)
- Points: players are granted 0 points for a loss, ½ point for a draw and 1 point for a win.
- Indication of who starts, whose turn and who wins the set
- Indication of the game status after each set. (1-0, 2-1, etc.)
- Indication of the winner of the match
- Errors handling during the game (wrong box or already used box selection)

² For more information and rules: <https://fr.wikipedia.org/wiki/Tic-tac-toe>

9 Appendix: tutorials

9.1 Introduction to the robot use (console)

9.1.1 Moving the robot

The assistant will explain how to move the robot. The different operation modes (Axis 1-3, Axis 4-6, linear, rotational) can be found by clicking in the bottom-right corner of the console. You can also change the tool object, the work object and the coordinates origin (robot base, world) here.

9.1.2 Initialize the encoders

1. Move to the origin position (all axes at 0°)
2. Navigate to ABB→calibration→Rob1→Update counters→Yes→Update→Update

9.1.3 Create a module

1. Navigate to ABB→Program editor→Modules→File→New module
2. Change the module name to “MainModule”
3. Click Ok→Ok

9.1.4 Define a tool object

4. Navigate to ABB→Jogging→Tool→New→Ok
5. Select the new tool (tool1)→Edit→Change value
 - a. Tool end position (x, y, z): (0, 0, 230) [mm]
 - b. Tool mass (mass): 2kg
 - c. Tool center of gravity (x, y, z): (0, 0, 115) [mm]
6. Click OK

9.1.5 Define a work object

1. Navigate to ABB→Jogging→Work→New→Ok
2. Select the new tool (wobj1)→Edit→Define→User method: 3 points
3. Place the tool end on the origin (X1) and click “Modify position”
4. Place the tool end on the X axis (X2) and click “Modify position”
5. Place the tool end on the Y axis (Y1) and click “Modify position”
6. Click OK

9.1.6 Program on the console

9.1.6.1 Add the main routine

1. Navigate to ABB→Program editor→Modules
2. File-→New routine
3. Change the name to “main”
4. Click Ok

9.1.6.2 Program

1. Navigate to ABB→Program editor→Routines→main()
2. Add instruction

9.1.6.3 Run the program

1. Navigate to ABB→Program editor→Routines→main()→Debug→PP to main
2. Enable the robot motors and click on the “Play” button. You have to maintain the robot active during the whole execution

9.2 Introduction to Robot Studio (PC)

9.2.1 Setup

1. Start RobotStudio 5.15
2. Add a new controller by going to Controller→Add controller→12-63215 on AOT test
3. Open the RAPID programming tab by opening T_ROB1→MainModule in the tree on the left of the window
4. To write code in the computer, you have to request the write access to the controller. To do so, click on “Request write access” in RobotStudio and then “Grant” on the console

9.2.2 Run the program

1. Check the program by clicking on “Check program”
2. Click on Apply→Yes→Yes
3. On the controller, click on “Rewoke” and run the program as in section 9.1.6.3
4. Once the program has wfinished, you will have to give the control back to the computer, as you have done in section 9.2.1, point 4.

9.3 List of most used RAPID instructions, functions and data types

9.3.1 Data Types:

num - Numeric values (p.1165)

dnum – Numeric values

pos - Positions (table of 3 nums: X, Y and Z) (p.1179)

robtargt - Position data (X, Y, Z, quaternion of the orientation, tooldata, state of external axis) (p.1194)

9.3.2 Instructions:

CRobT - Reads the current position (robtargt) data (p.807)

EXIT - Terminates program execution (p.106)

FOR, ENDFOR - Repeats a given number of times (p.109)

Compact IF - If a condition is met, then... (p.58)

IF, THEN, ELSE, ENDIF - If a condition is met, then ...; otherwise ... (p.131)

MoveC - Moves the robot circularly (p.239)

MoveL - Moves the robot linearly (p.269)

NumToDnum - Converts num to dnum (p.920)

NumToStr - Converts numeric value to string (p.921)

Offs - Displaces a robot position (p.923)

RETURN - Finishes execution of a routine (p.375)

TEST, CASE, DEFAULT, ENDTEST - Depending on the value of an expression (p.560)

TPerase - Erases text printed on the FlexPendant (p.567)

TPreadDnum - Reads a number from the FlexPendant (p.568) (user entry)

TPreadNum - Reads a number from the FlexPendant (p.576) (user entry)

TPwrite - Writes on the FlexPendant (p.580)

ValToStr - Converts a value to a string (p.1099)

WaitTime - Waits a given amount of time (p.695)

WaitUntil - Waits until a condition is met (p.715)

WHILE, ENDWHILE - Repeats as long as (p.723)

9.4 Example of code showing handling vectors in RAPID language

MODULE MainModule

TASK PERS tooldata tool1:=[TRUE,[[0,0,350],[1,0,0,0]],[1,[0,0,150],[1,0,0,0],0,0,0]];

TASK PERS wobjdata wobj1:=[FALSE,TRUE,"",[[225,25,115],[1,0,0,0.2]],[[0,0,0],[1,0,0,0]]];

VAR robtarget p1;

VAR num field;

VAR pos field_vector{3};

PROC init_field_vector()

field_vector{1}:=[20, 0, 0];

field_vector{2}:=[0, 20, 0];

field_vector{3}:=[0, 0, 20];

ENDPROC

PROC main()

init_field_vector;

p1 := CRobT(\Tool:=tool1 \WObj:=wobj0);

TPReadNum field, "Type 1 or 2";

TPWrite "You typed ";

TPWrite NumToStr(field,0);

WaitTime(2);

*MoveL Offs(p1, field_vector{field }.x, field_vector{field }.y, field_vector{field }.z), v5, fine,
tool1\WObj:=wobj0;*

ENDPROC

ENDMODULE