Sami Farhat
STUDENT ID: 40097236

LAB 5

Extending the TCP client to manipulate Ned2

Performed on 21th February 2023

Embedded Systems
COEN 421
WINTER 2023

Concordia University
7th March 2023
Section U-UJ

SUBMITTED TO: CRISTIAN TIRIOLO

2. Objectives

The main objectives in this lab were to:

- Develop a deeper understanding of a TCP connection.
- Improve our connection with the robot.
- Perform a specific task with the robot.

3. Theory

This lab is a continuation of the previous lab where we use a JSON format packet to send commands to the robot from the client side. However, this time, the server send values to the robot. In order to do tat , we send the required values throught the request JSON packet, in the same way as the previous lab. For reminder, The format takes this format {“command” : ”<desired_command>” , “param_list “ : [params]}

4. Task/Results/Discussion

In this lab , most of the functions that needed to be coded are function that make the robot move. The functions that will be implemented are the following:

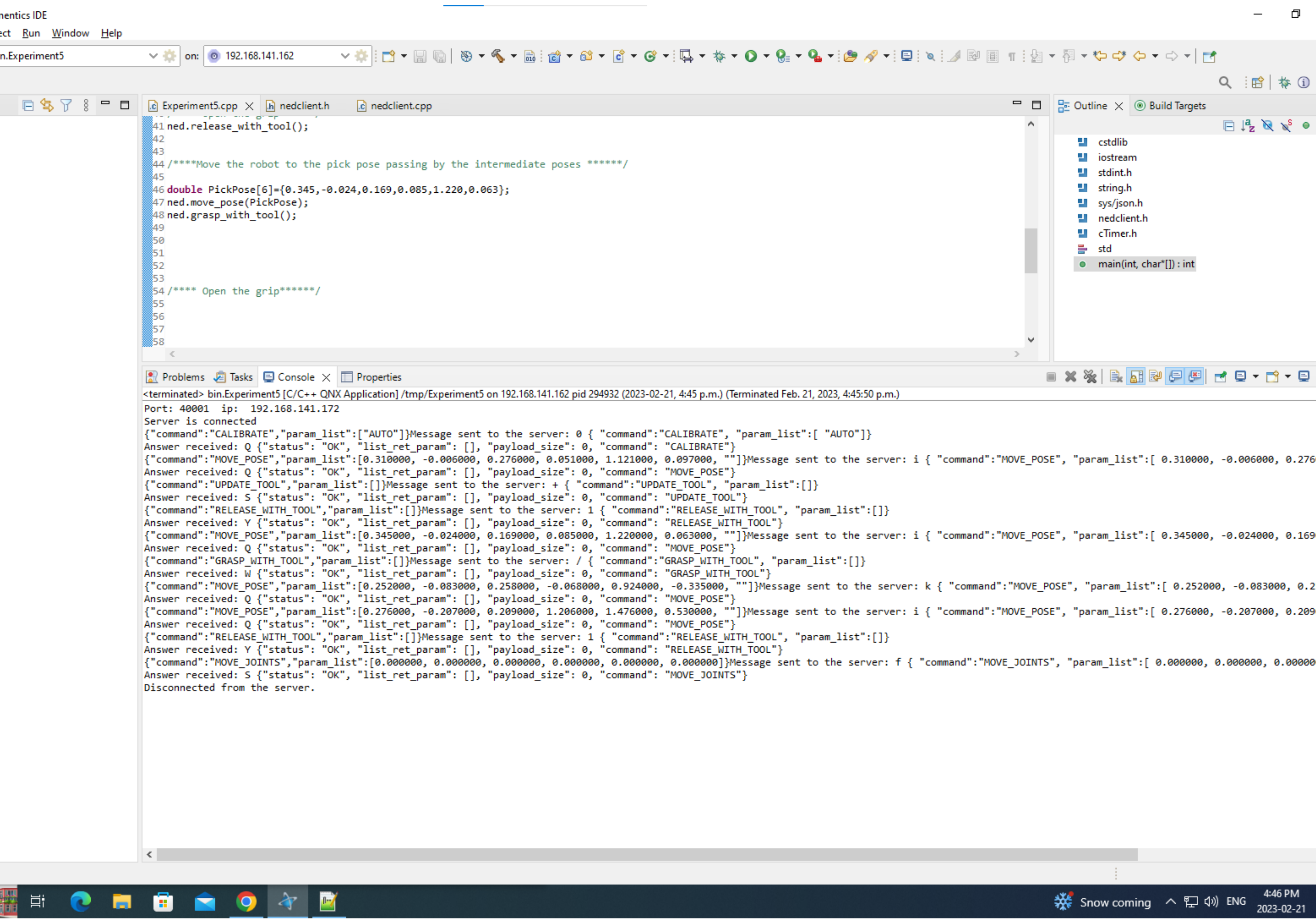
- Move_joints()
- Move_pose()
- Update_tool()
- Grasp_with_tool()
- Release_with_tool()

This lab made use of the same NEDCLIENT class that was used in lab 4, however we added some extra functions in order to make the robot move.

Move_joint and move_pose both take an array of double as inputs. Given the foregoing, we take that array and convert it into an array of char in order to write it into the send_command that what coded in the last lab.

Then for the rest of the commands, the proper command and param list must be coded and then formatted properly in order to send them in the send_command function.

Lastly, in the main thread, we call the proper functions in order to make the robot move, pick an object and place it at the appropriate spot on the conveyor belt. In order to make this maneuver we have used exactly the same values as in lab 2 since we know the correct measurements already. The principle of pick pose and safe pick pose were used in order to make sure that the robot moved freely without any injury risk or hazard.



```
ned.release_with_tool();


/****Move the robot to the pick pose passing by the intermediate poses ******/

double PickPose[6]={0.345,-0.024,0.169,0.085,1.220,0.063};
ned.move_pose(PickPose);
ned.grasp_with_tool();





/**** Open the grip******/
```

```
<terminated> bin.Experiment5 [C/C++ QNX Application] /tmp/Experiment5 on 192.168.141.162 pid 294932 (2023-02-21, 4:45 p.m.) (Terminated Feb. 21, 2023, 4:45:50 p.m.)
Port: 40001  ip:  192.168.141.172
Server is connected
{"command":"CALIBRATE","param_list":["AUTO"]}Message sent to the server: 0 { "command":"CALIBRATE", "param_list":[ "AUTO"]}
Answer received: Q {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "CALIBRATE"}
{"command":"MOVE_POSE","param_list":[0.310000, -0.006000, 0.276000, 0.051000, 1.121000, 0.097000, ""]}Message sent to the server: i { "command":"MOVE_POSE", "param_list":[ 0.310000, -0.006000, 0.276
Answer received: Q {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "MOVE_POSE"}
{"command":"UPDATE_TOOL","param_list":[]}Message sent to the server: + { "command":"UPDATE_TOOL", "param_list":[]}
Answer received: S {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "UPDATE_TOOL"}
{"command":"RELEASE_WITH_TOOL","param_list":[]}Message sent to the server: 1 { "command":"RELEASE_WITH_TOOL", "param_list":[]}
Answer received: Y {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "RELEASE_WITH_TOOL"}
{"command":"MOVE_POSE","param_list":[0.345000, -0.024000, 0.169000, 0.085000, 1.220000, 0.063000, ""]}Message sent to the server: i { "command":"MOVE_POSE", "param_list":[ 0.345000, -0.024000, 0.169
Answer received: Q {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "MOVE_POSE"}
{"command":"GRASP_WITH_TOOL","param_list":[]}Message sent to the server: / { "command":"GRASP_WITH_TOOL", "param_list":[]}
Answer received: W {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "GRASP_WITH_TOOL"}
{"command":"MOVE_POSE","param_list":[0.252000, -0.083000, 0.258000, -0.068000, 0.924000, -0.335000, ""]}Message sent to the server: k { "command":"MOVE_POSE", "param_list":[ 0.252000, -0.083000, 0.2
Answer received: Q {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "MOVE_POSE"}
{"command":"MOVE_POSE","param_list":[0.276000, -0.207000, 0.209000, 1.206000, 1.476000, 0.530000, ""]}Message sent to the server: i { "command":"MOVE_POSE", "param_list":[ 0.276000, -0.207000, 0.209
Answer received: Q {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "MOVE_POSE"}
{"command":"RELEASE_WITH_TOOL","param_list":[]}Message sent to the server: 1 { "command":"RELEASE_WITH_TOOL", "param_list":[]}
Answer received: Y {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "RELEASE_WITH_TOOL"}
{"command":"MOVE_JOINTS","param_list":[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000]}Message sent to the server: f { "command":"MOVE_JOINTS", "param_list":[ 0.000000, 0.000000, 0.00000
Answer received: S {"status": "OK", "list_ret_param": [], "payload_size": 0, "command": "MOVE_JOINTS"}
Disconnected from the server.
```

With this screenshot , the output console show that the server is connected , and that the proper functions were called. We can also see that all status are OK, with all the proper commands

5. Questions:

- What is the difference between move_pose() and move_joints()?

The difference is in the efinition of pose and joints. The functions basically do t he same thing ,but pose and joitns are two different set of measure that allow the robots to move differently . These measurement have already been introduced in a previous lab, however, this is a small recap.

Pose: The pose format is a specific position in the XYZ plane. By setting them , the robot is able to accurately go to a specific position in space. When using the move_pose function , we feed the function with every pose needed in the form of an array of doubles and it gets converted then sent to the robot.

Joints: Joints represent degrees in radian. When settings a specific joints , it ask the robot to move a specific part of the robot to a certain degree in order to get to a specific position. When using the Move_joint(), we are feeding the robot with 6 different degrees that get mapped to each part of the robot , then it ask thoses part to rotate at that exact angle in order to get to a specific position.

● Would it be possible to set an arbitrary opening percentage for the grip? Motivate your answer

I believe it would be possible to set an arbitrary opening if we pass a certain parameter inside the JSON packet , given the fact that it does have a parameter arguments. This argument would directly be placed in the param list then it would be decoded and parsed by the code and fed into the robot.

6. Conclusion

The lab was a success. Every single objective was met , the whole process of moving an object was achieved with success. It is possible to say that it was achieved with success since it followed an arbitrary set of instructions that was already tested in previous lab. The object was picked and placed at the correct spot. The functions worked as described without any problem