Sami Farhat
STUDENT ID: 40097236

LAB 4

Developing a TCP client to read Ned2's sensor.

Performed on 14th February 2023

Embedded Systems
COEN 421
WINTER 2023

Concordia University
7th March 2023
Section U-UJ

"I certify that this submission is my original work and meets.
THE FACULTY'S EXPECTATIONS OF ORIGINALITY"

SUBMITTED TO: CRISTIAN TIRIOLO

2. Objectives

The main objectives in this lab were to:

- Develop a deeper understanding of a TCP connection.
- Create a TCP connection with the robot.
- Send data and receive data to and from the server.

3. Theory

In this lab , we need to create the TCP connection and send the data to the robot. The robot takes requests as form of JSON format. JSON or JavaScript Object Notation is a format use to send data. It is easy for humans to understand and easy for processors to parse. It is mainly used to send data between server and data. Given the foregoing, it is logical that it is used in TCP data transfers.

In order to understand the rest of the lab, we need to properly understand the JSON format. This format is a combination of Objects and Arrays. The Curly brackets encapsulate the object. Inside the object, we use a key-value format. In the case of our lab, the whole request gets encapsulated in a object where the details are key-value pairs. The first key represent command with its value being the command that we want to run. The second key is for the parameter list with its value being a list with all required parameters.

In this lab, we need to program the commands sent by the user to the robot through the connection. To send these commands, we need to encapsulate them in the proper JSON format. The format takes this format {“command” : ”<desired_command>” , “param_list “ : [params]}

4. Task/Results/Discussion

Before explaining the Task and the results, there is some ground layer to do. First, this lab uses the function of a timer. A specific Timer class was used in order to be able to create, initialize a timer and set it so that we are able to keep track of time, or to make a process sleep for a certain amount. To use the timer class, we call a timer object in the main function and use functions of the timer class.

When creating the object, we pass to it arguments which are second and milliseconds. They get used in the constructor to initialize the timer. The other function used is the waitTimer() function, which wait for the required amount of time before continuing. It is in short, a substitute of the .sleep() function.

Now as for the rest of the lab we have the NedClient class to finish. This class is needed to connect the client to the robot using a TCP connection. It starts by initializing the connection, and then connecting to the server, which is the beginning.

Then, as for the hard part, the send_command is a function which will be vital to the rest of the lab. This function will serve as the gateway for all commands. It converts the arguments, create the JSON files which defines the packets. Then it makes sure everything is properly converted before sending the packet to the server using the CSocket object. Lastly, we receive the answer from the server.

When the calibrate auto gets called, the require command and parameters values get created and then they are used in the send_command method described earlier.

```
char cmd[] = "\"CALIBRATE\"";
char parm[] = "[\"AUTO\"]";
```

That method will take the arguments and transform them into the required JSON packet then pass those values through the TCP connection which will therefore calibrate the robot.

NedClient::calibrate_auto() This command requests an automatic calibration of the robot.	
request	{"command": "CALIBRATE", "param_list": [AUTO]}

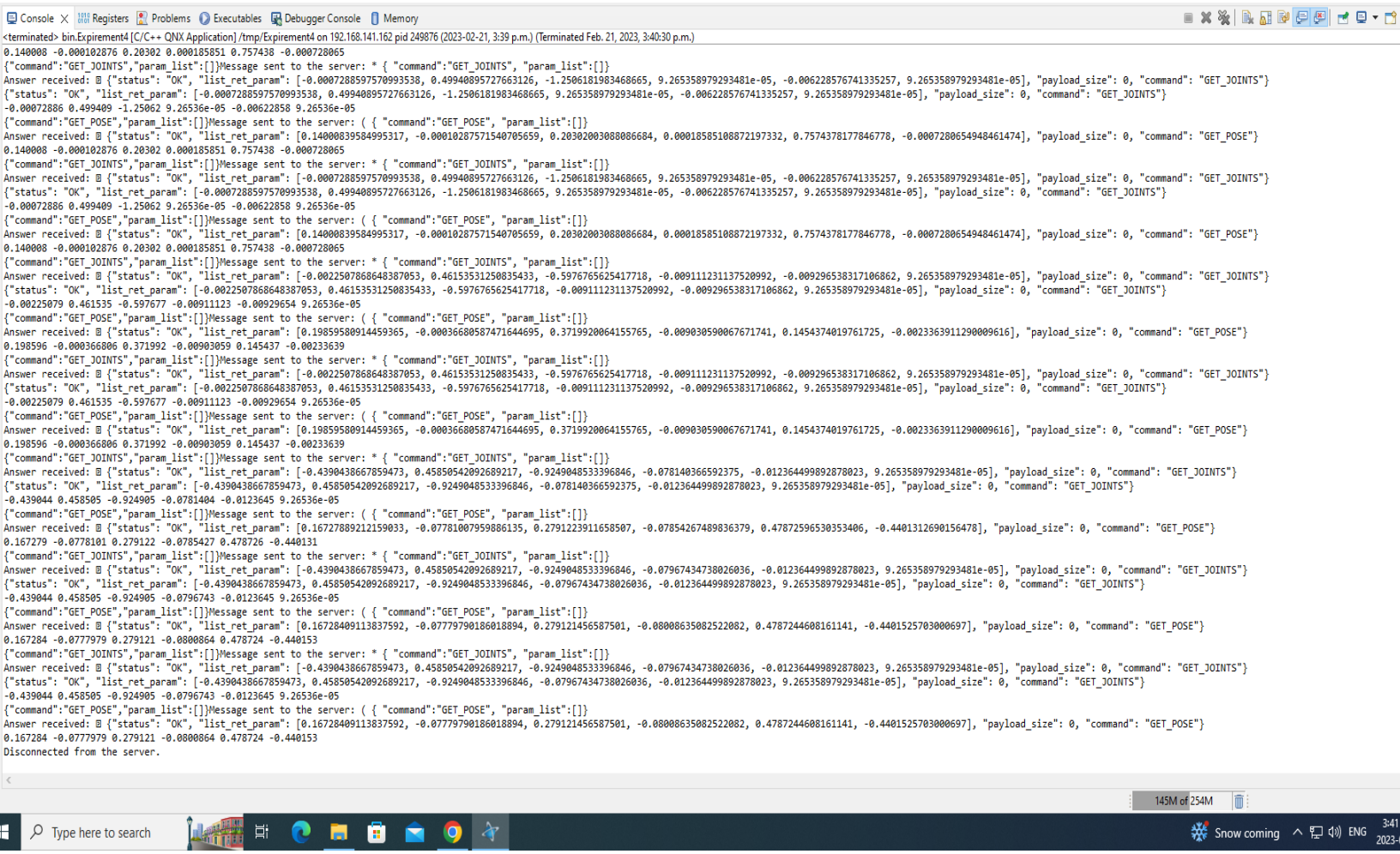
The final JSON would be of this format before being passed on to the robot.

Now for the get_joints and get_poses function , we need an answer from the robot since we want to retrieve data. It gets a little more complicated then. The sending of the request keep the same principle , however, we decode the answer.

Now in order to get the answer, a little bit of cleaning is required because of the formatting of the answer sent by the robot. Then for every single value, we convert it to a double and add it into an array which will be returned by the function. This array will hold the required values. It is the same principle for poses or joints.

Now these are the value gotten when running the get_joints and get pose function in the main. Just for context, the main initialize the IP , the timer, create the NEDCLIENT object, which allow for communication with the robot and then call multiple functions.

Output on the console from the given code. We can see the commands , their answers and also all values for the poses or the joints.



5. Questions:

● Describe the use of the JSON format.

The JSON format is mostly used in web-application since it allows for easy data exchange over the internet. It is lightweight, easy for a human to understand and easy for a machine to parse. The key value pair allows for a quick reading of the data. Given it's lightweight status it also allows for a perfect use with TCP connections. It is also an excellent format in storing data, storing configurations or sending data for updates or commands. It has a lot of useful applications. Another one would be to gather and store temporary dat. When a user answer a form online, it can quickly populate the data in a JSON format making it a quick and efficient way of storing that temporary data.

● Describe how to parse a string in C++ efficiently

For starters, C++ offer multiple useful functions in the standard library which allow for quick parsing of strings. Getline() ; Stoi () ; Stof() and stringstream() are examples. I will explain them briefly.

Getline() : Allows for the user to read a line and store it in a string variable. When the newline character is reached, the variable stores the content. Getline() can also take an extra argument which would be the delimiter. In the context of this lab, we use the "," to delimit and mark a new value.

Stoi() : String to Int , This function takes a string and return an the integer value.

Stof() : String to Float, This function takes a string and return the floating-point value.

Stringstream(): Transform a string into a stream. Can then be used in tandem with Getline()

A string is also an array of character, meaning we could always loop over that string and index every value but it is not the most efficient way .

Finally, the use of regex can also be efficient if a specific pattern is searched.

6. Conclusion

In conclusion, the lab was a success since every objective was met. The lab was a bit harder than expected since it required a more advanced knowledge of C++, but with the right amount of research we were able to get through it. The connection was properly established and the correct result were obtained since we were able to properly calibrate the robot and get the correct values when calling the get pose or the get joints functions.