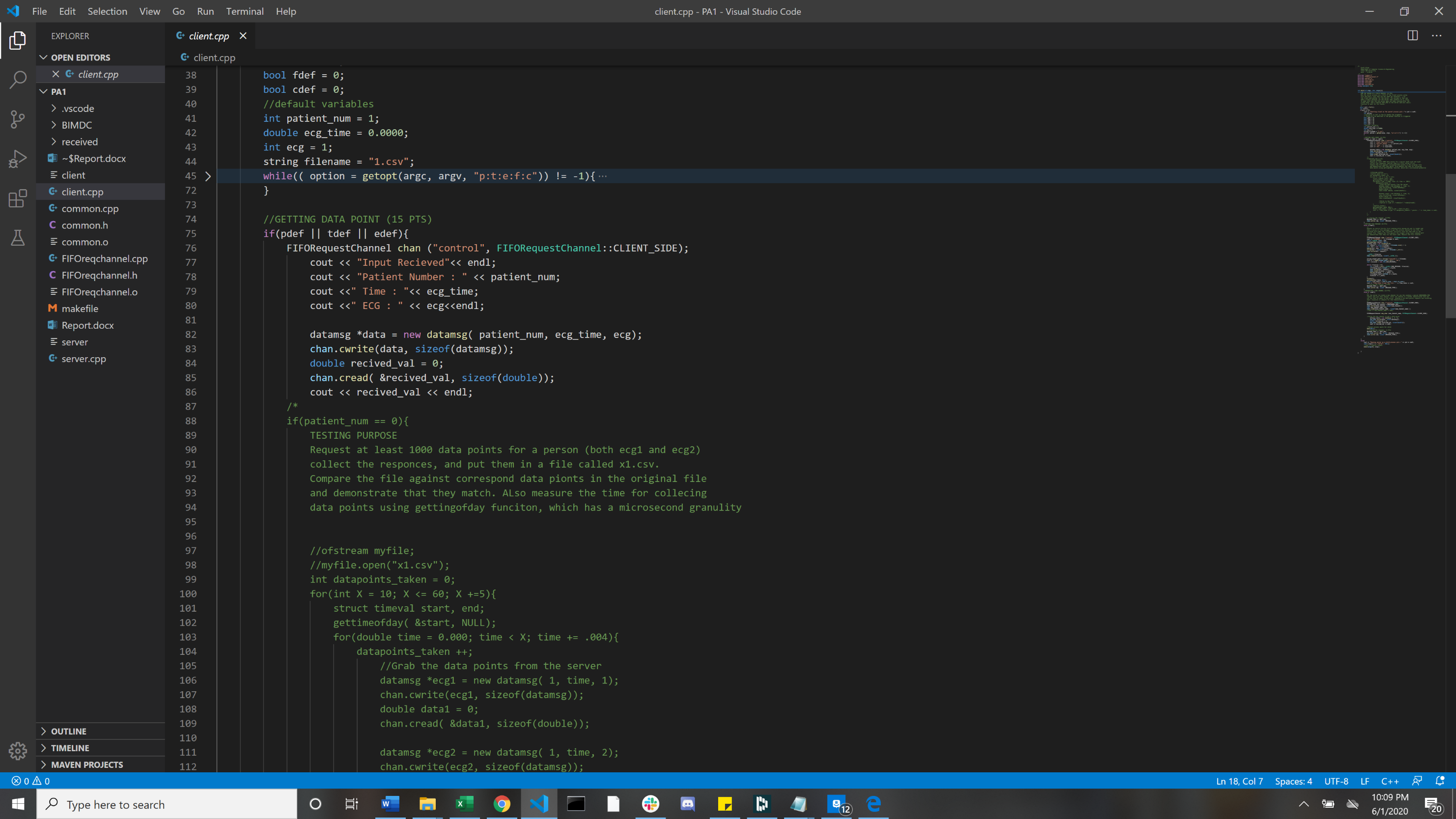
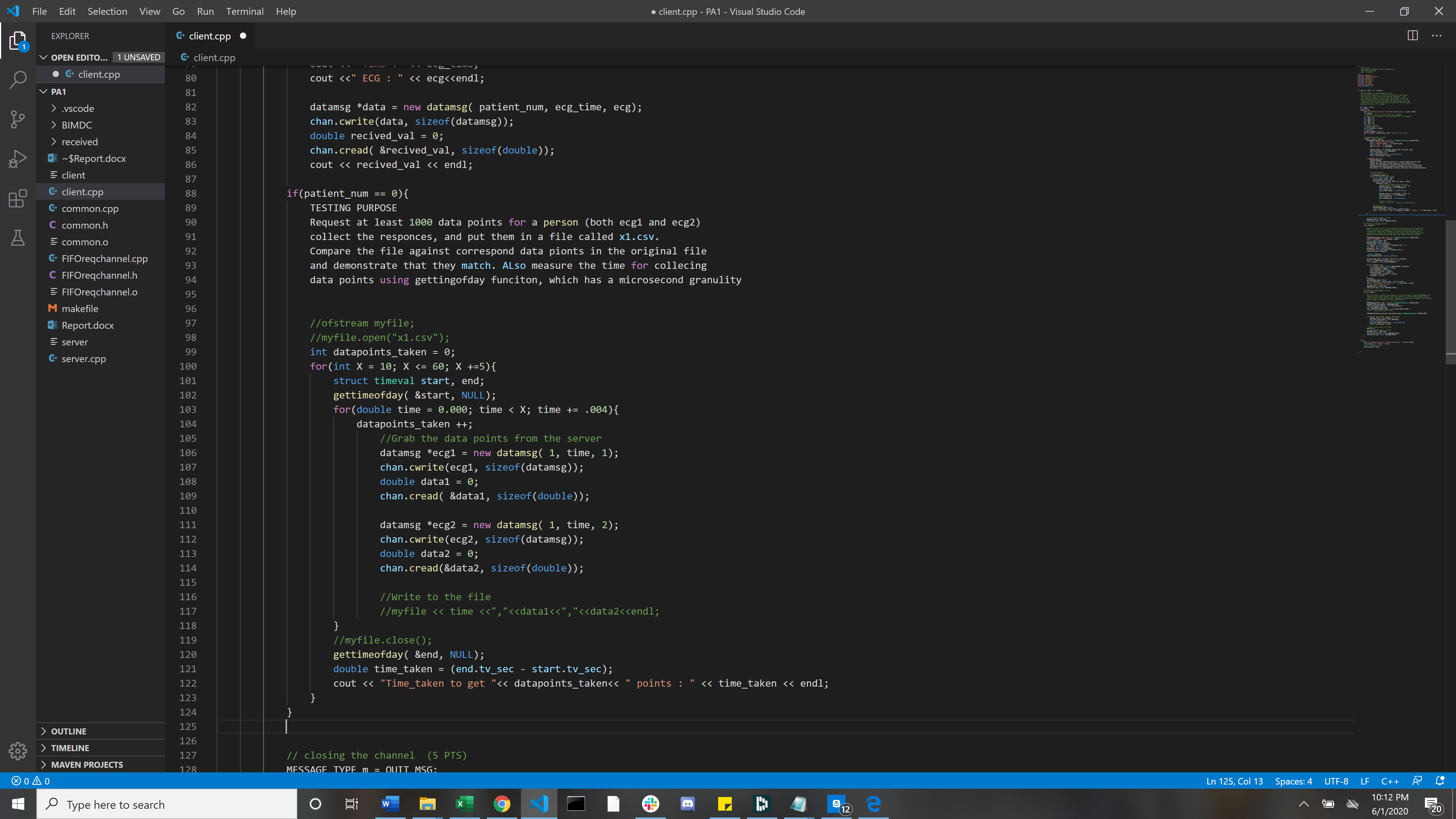
**A Client Process Speaking to a Server Process**

**My tasks:**

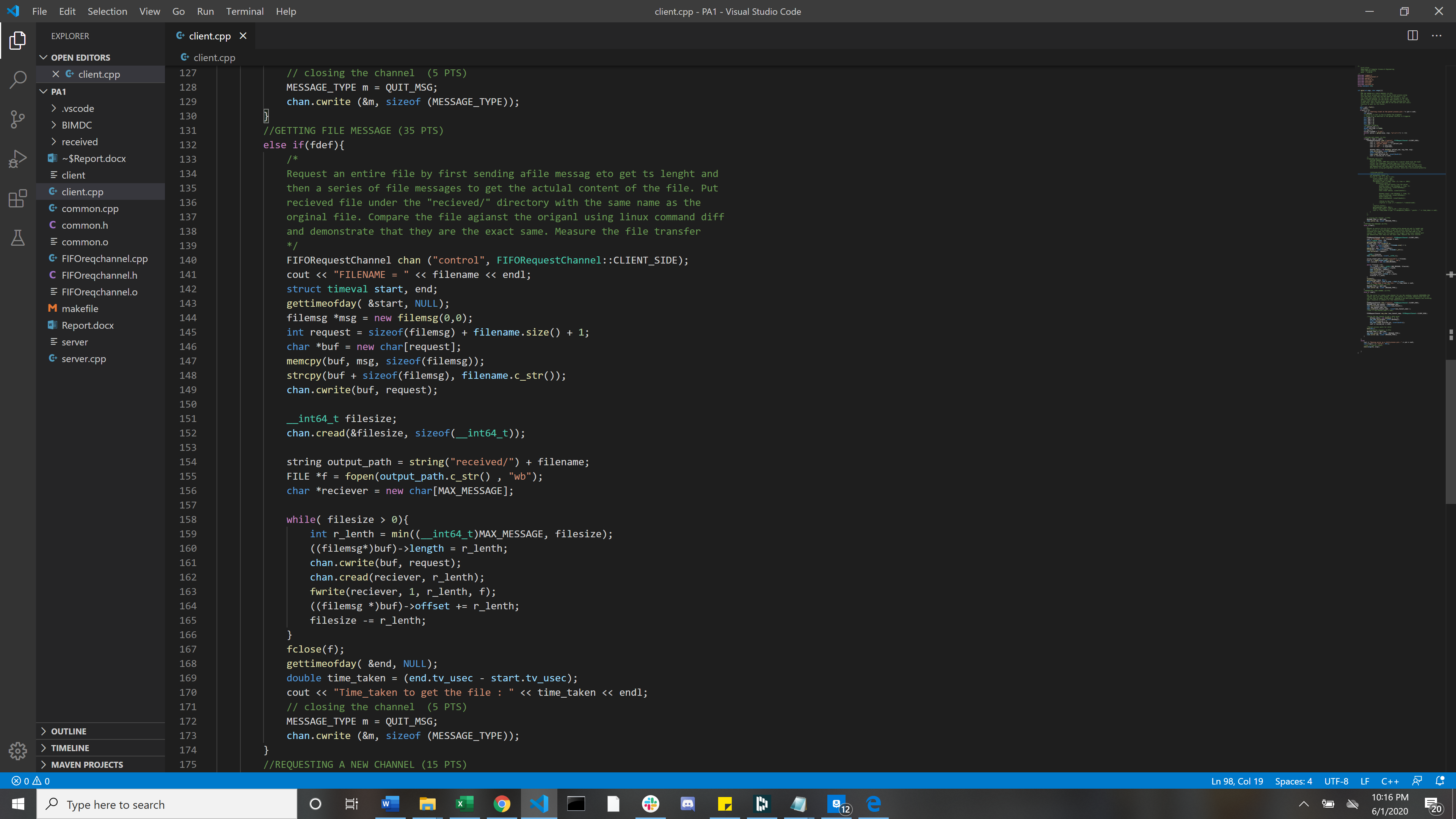
Requesting Data Points (15 pts)

Requesting a new data point was simply just sending data message. This was done with the parameters of the patient number, the time in seconds, and the ECG (1 or 2). After this was done we would write to the server with the data message. Once this was done we created a spot for the value to be stored. Once this was done we would read the value the server sent back. For testing purposes I created a nested for loop to iterate through grabbing a number of data points. This was done with the command ./client -p 0. This was the signal for the code to run through my test case and print out the time taken. The graph of collecting data points is displayed below.



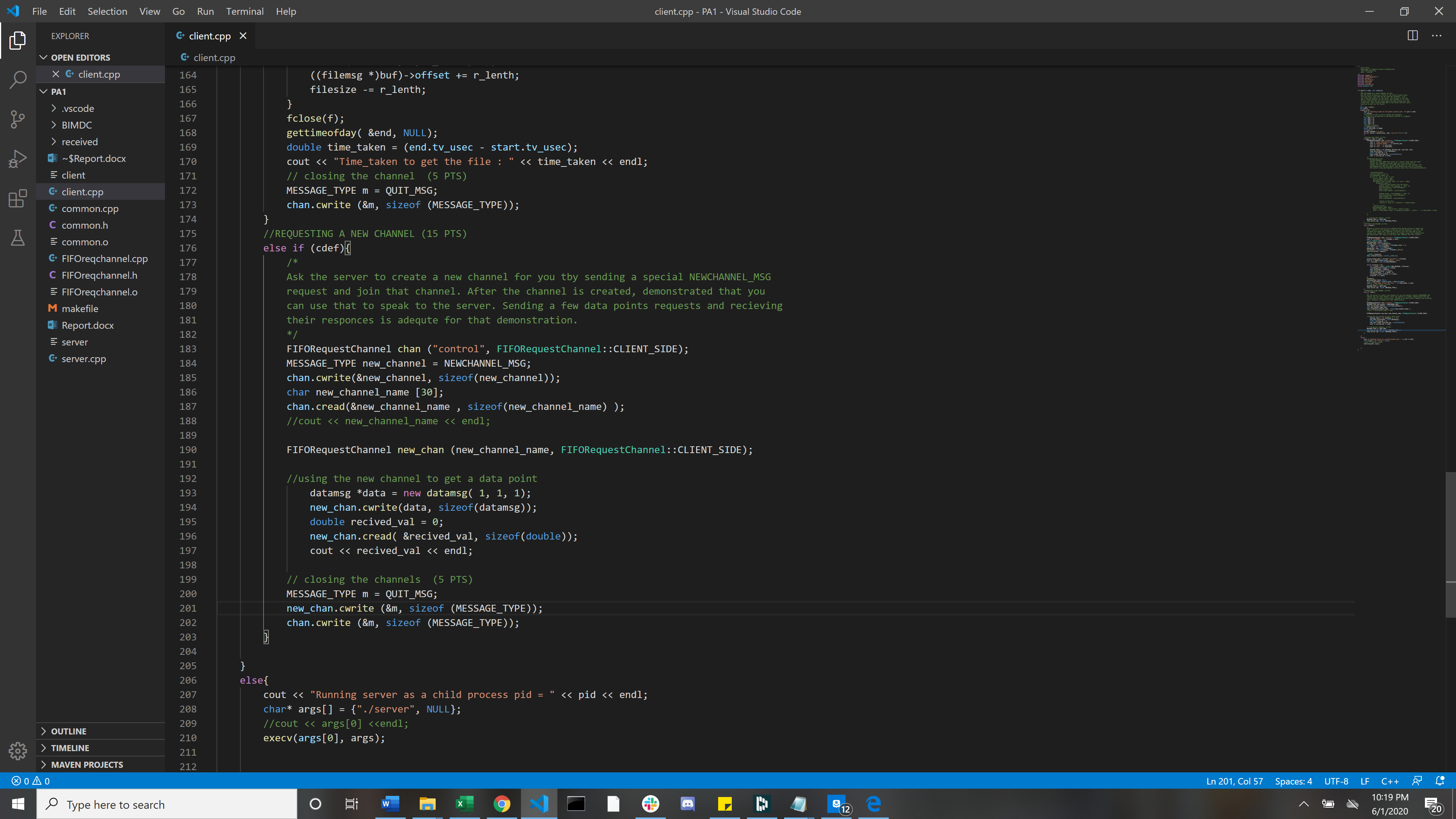
Requesting Files (35pts)

Requesting a file was done in a simiar manner. We wrote to the server and then created a location to store the data we recived from the server. From there we wrote to the new file 64 bits at a time. I recored the time of transfereing files. The graph is attached below.



Requesting a New Channel (15pts)

We had to use the control channel first to create a new side channel. This took me a while to figure out because I was trying to store the new channel name as a string. I then realized I had to be using a char array to store. Once I created a new side channel I requested a single data point then closed both channels.



Run the server as a child process (15 pts)

Running the server as a child process was quite easy once I looked up a video on creating a fork. I created an a fork at the beginning of the program and then made an if/else statement asking for the PID. If the PID was 0 then the process was the parent process. Else it was child and I would run the server. I did not need a wait() because the server would stay open until a quit message was sent.

