ECE 315 - Computer Interfacing Laboratory 4

Stepper Motor Control Using the L297 and L298 Integrated Circuits

Lab Section H21 Date of Lab: March 24th

Nicholas Serrano(1508361), Evan Timms(1504825)

**Abstract**

In this final lab for ECE 315, we used our knowledge of embedded microcontrollers to implement a web server that runs on the Netburner MCF board. This lab would combine the functionality of the previous four labs. To begin, we first used the markup language HTML to build a frontend that would use a browser to send POST and GET requests to our board. This static site would contain an HTML form that would send a POST request to our microcontroller server. To do this, we used a custom function call tag provided in the lab files. The user could submit data via the form and it would be parsed and stored in a FormData class on the backend. The server would process the data and perform the correct hardware interactions the user requested, before assembling and returning new HTML code to display to the user. The user would be able to rotate the stepper motor, display an onscreen message showing keypresses, and show data from the circuit's photocell.

**Design**

Exercise 1:

In this first exercise, we created a HTML form that would be displayed via the web server on the microcontroller that could be accessed through a browser. The HTML code was supplied to use via eClass, and we modified the contents to update the lab year. After downloading and creating a new Netburner project (along with importing the files), we could access the webpage by navigating to 10.0.0.102 on the computer. Below is the form as processed by a webpage. When reloading the page you could see that the counter would increment by one.

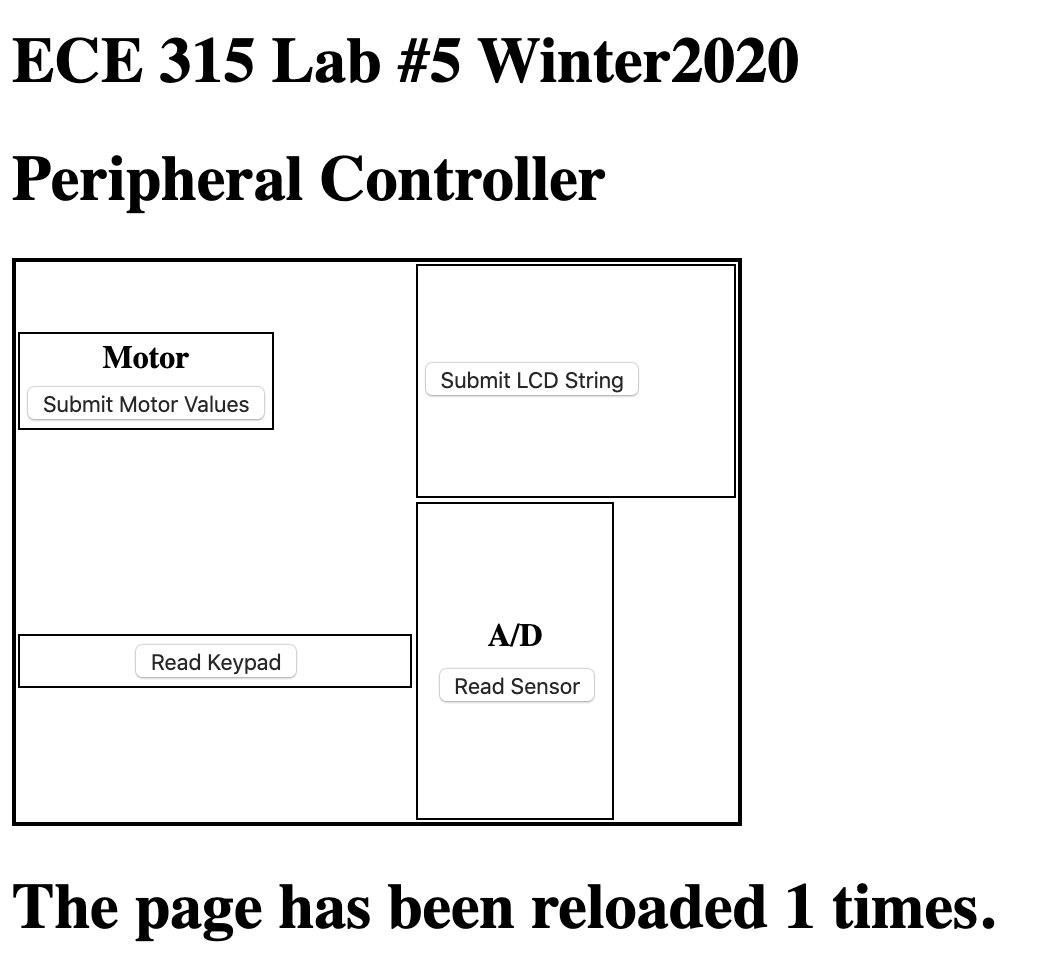


Figure 1: index.html

Exercise 2:

In this exercise, we wrote a number of C handler functions that would handle the form data from each different area of the HTML table. We created a number of handler functions that were called using the FUNCTIONCALL tag. Each of these functions would use our global FormData object to read the data sent from the client, parse it and then return the result to the client. We wrote four functions in total, each to deal with the keypad, motor, A/D converter and LCD string. In this exercise, it was not required we implement the functions, so we left them as method stubs for now. Below is a picture of the HTML form along with an example table section.

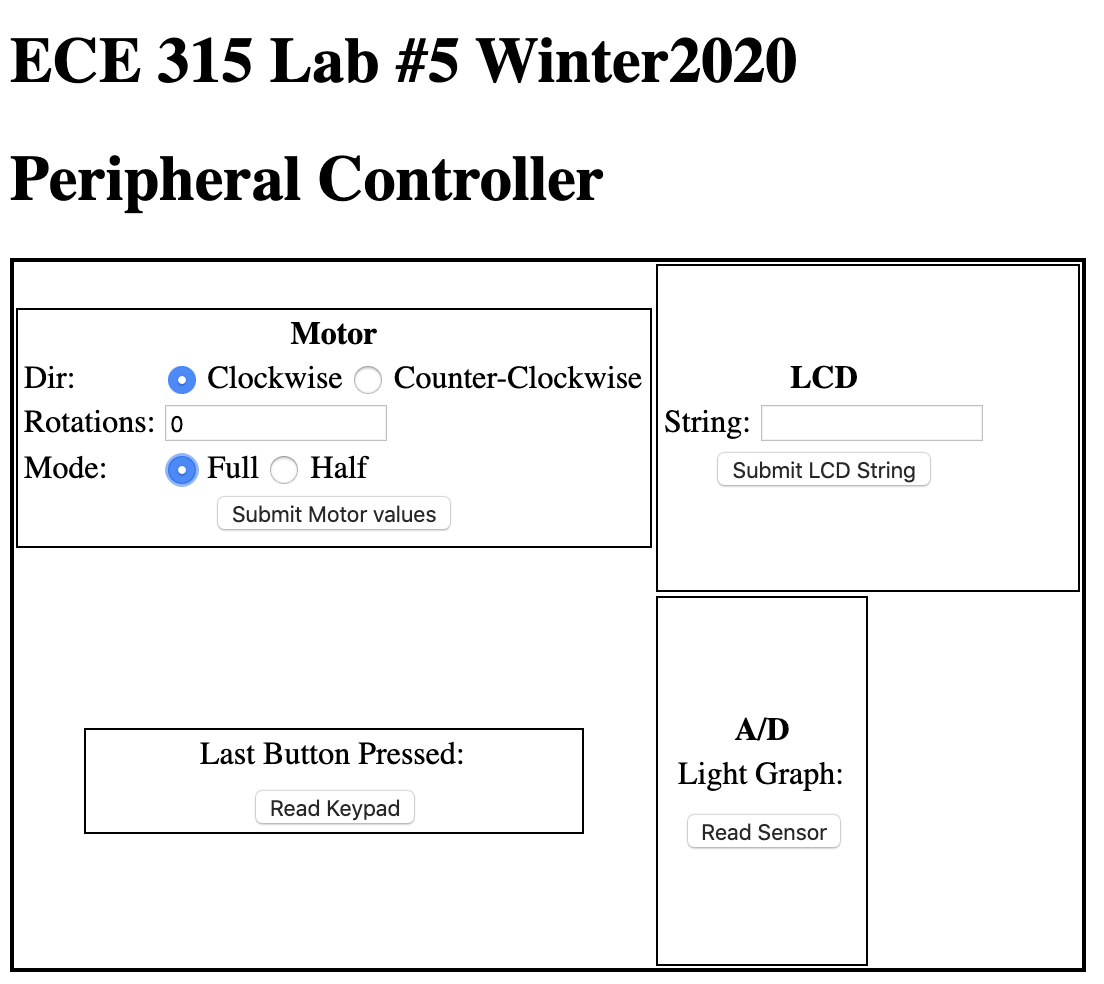


Figure 2: Modified index.html (without implemented handler functions)

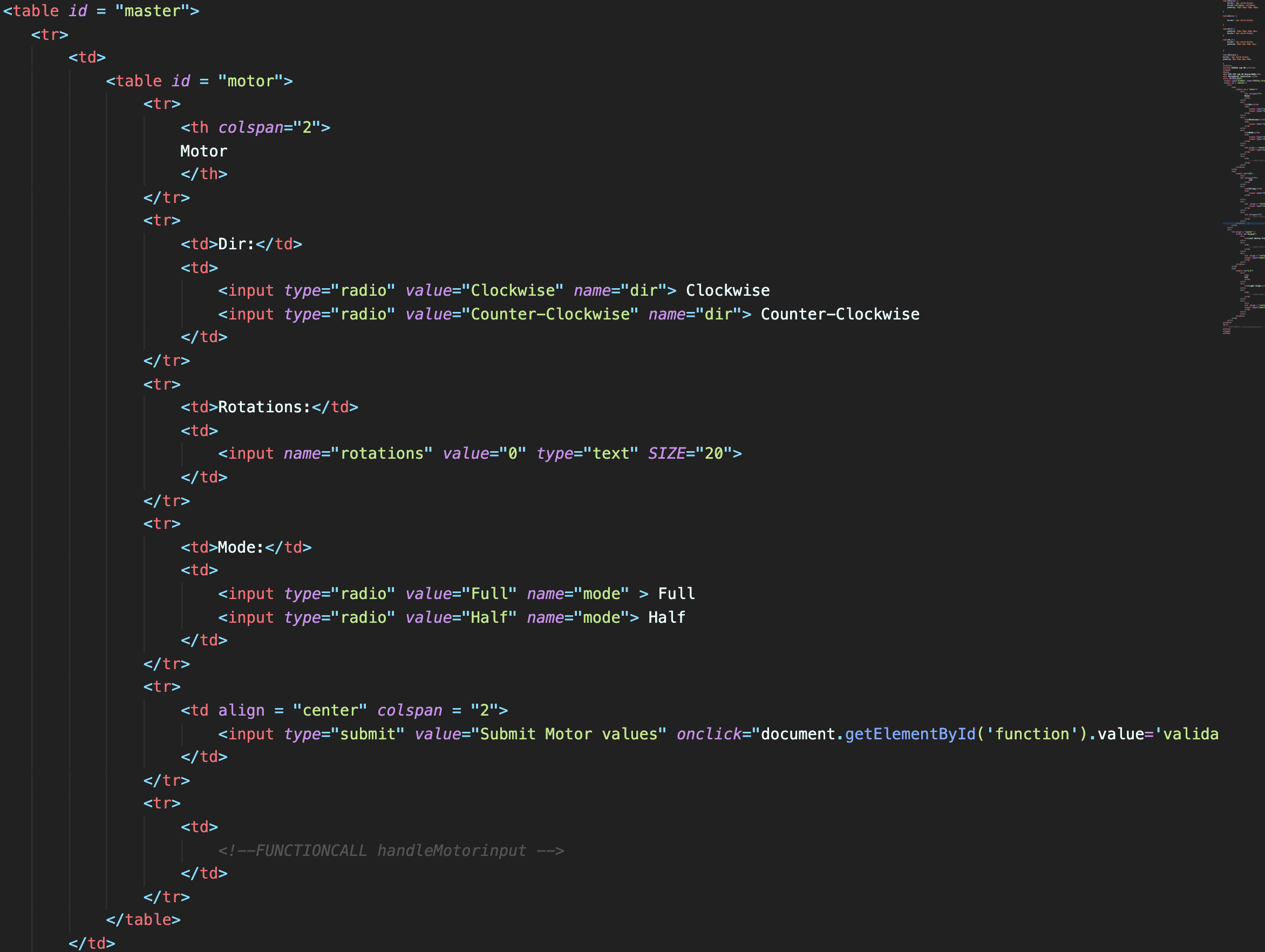


Figure 3: Motor Control HTML Form

Exercise 3:

This exercise involved us implementing the C code to properly extra incoming POST data and returning data to the client dynamically. To do this, we first parsed and validated the incoming data in the MyDoPost function. Parsing the data involved performing string conversions to ensure the numbers inserted by the user are converted to ASCII. Once this conversion was done, we validated the data server side. The steps to complete the implementation of this function was the following:

1. Check the form was extracted properly by calling ExtractPostData. If this returns -1, redirect and return.
2. Check with function was called. Perform validation and update data.
3. Set the post buffer as our local buffer.

Once this was completed, we then finished implementing the FormData class that would allow us to take the parsed POST data and properly interface with our hardware. We added getters and setters for motor mode and direction, which read the value passed from the client.

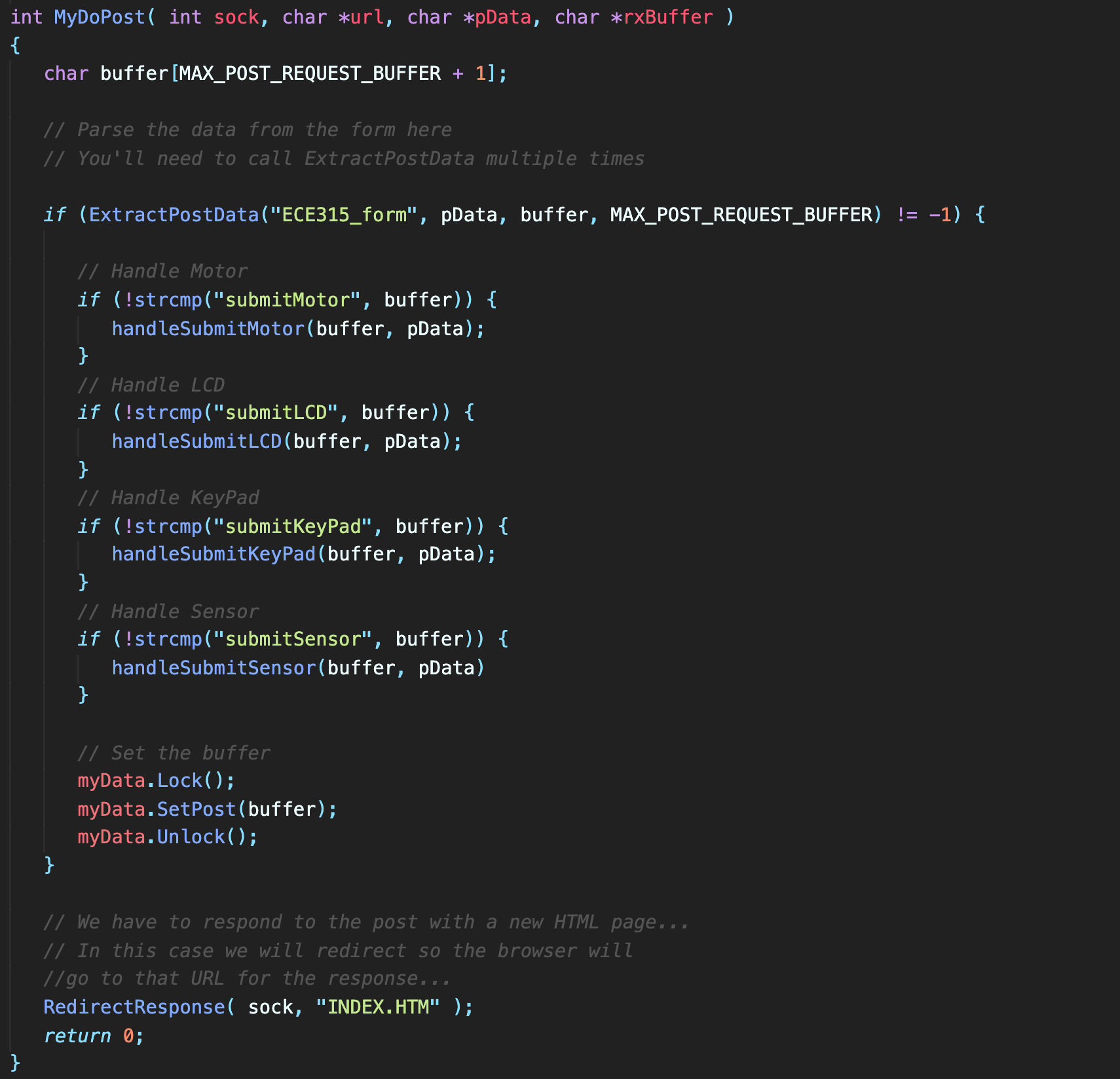


Figure 4: MyDoPost Function

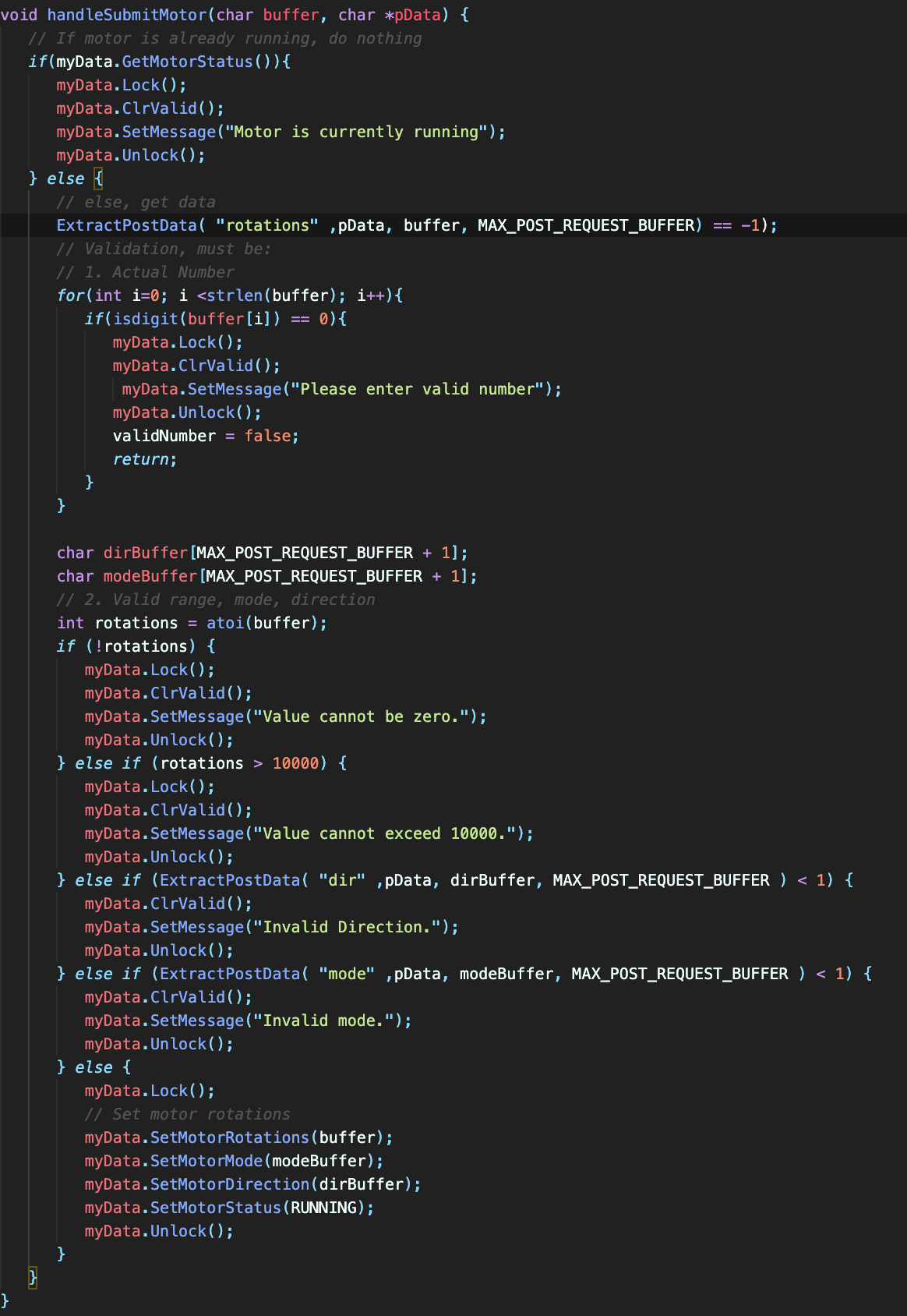


Figure 5: MyDoPost Helper Functions (Motor Handler)

Finally, we implemented the code in lab5.cpp that would handle dynamically updating the response to the POST request.

**Testing**

|  |  |  |
| --- | --- | --- |
| Test Description | Expected | Result |
|  |  | Results matched expected. |
|  |  | Results matched expected. |
|  |  | Results matched expected. |

**Conclusion**