**Lab 2: Using the Software Development Kit (SDK)**

Alana Ordonez

UIN: 428008825

ECEN 449 506

September 23, 2022

**I. Introduction**

This lab was to acclimate us with not just more capabilities of Vivado, but also using its software development kit. Instead of using simply Verilog, we wrote C code to control the LEDs in FPGA hardware. This is important because it reinforces our knowledge of building/simulating our modules, and it teaches us how to implement C to program our Zybo.

**II. Procedure**

A block design named “led\_sw” was created and configured with a processor and general purpose IO blocks to connect to the LEDs, switches, and buttons on the Zybo. The reset ports were redirected so that it corresponds to our already dedicated components on our hardware, then a constraints file called “led.xdc” was created to set our LEDs, switches, and buttons to the desired function. After generating the bitstream, the SDK was launched so that the C code could be written for the software application. Once it was completed, the FPGA was programmed in order to display the functionality written in the code.

**III. Results**

When “lab2a.c” was programmed, the LEDs counted upwards from LED0 to LED3, displaying every 4 bits. The console displayed which LED was lit during the process. With “lab2b.c,” the LEDs counted downwards whenever push button 0 was held, while they counted upwards whenever push button 2 was held. Holding down push button 2 showed on the console which switches were toggled, and pressing push button 4 displayed the count on the LEDs.

I was struggling a lot with lab2b.c in the sense that defining the device IDs was giving me so much error when they were passed into their respective initialization functions. I talked to a TA for help and they helped me with the correct syntax for both. It was also difficult to work out how to check which buttons were pressed and released, etc., but thinking through how to connect them as well as referencing our previous 4-bit counter really helped.

**IV. Conclusion**

This lab was really helpful in understanding a different way to program hardware, which is using a purely software implementation. It reinforced our abilities with Vivado and introducing us to using its SDK, and now we know a way to bridge the gap between software and hardware programming.

**V. Questions**

1. *In the first part of the lab, we created a delay function by implementing a counter. The goal was to update the LEDs approximately every second as we did in the previous lab. Compare the count value in this lab to the count value you used as a delay in the previous lab. If they are different, explain why? Can you determine approximately how many clock cycles are required to execute one iteration of the delay for-loop? If so, how many?*

The first lab used a clock rate of 125 MHz, while this week’s lab uses about a 100 MHz clock. To have executed one iteration of the while-loop delay, approximately 6-7 clock cycles were required.

1. *Why is the count variable in our software delay declared as volatile?*

The count variable can change throughout the duration of the process without our knowing. This ensures that the compiler does not alter the code in hopes of optimization, which would affect the order in which commands in our program are executed.

1. *What does the while(1) expression in our code do?*

while(1) creates an infinite loop that can only be terminated by the user. Because it is constantly running, we can check the console for the active changes to ensure it’s operating correctly.

1. *Compare and contrast this lab with the previous lab. Which implementation do you feel is easier? What are the advantages and disadvantages associated with a purely software implementation such as this when compared to a purely hardware implementation such as the previous lab?*

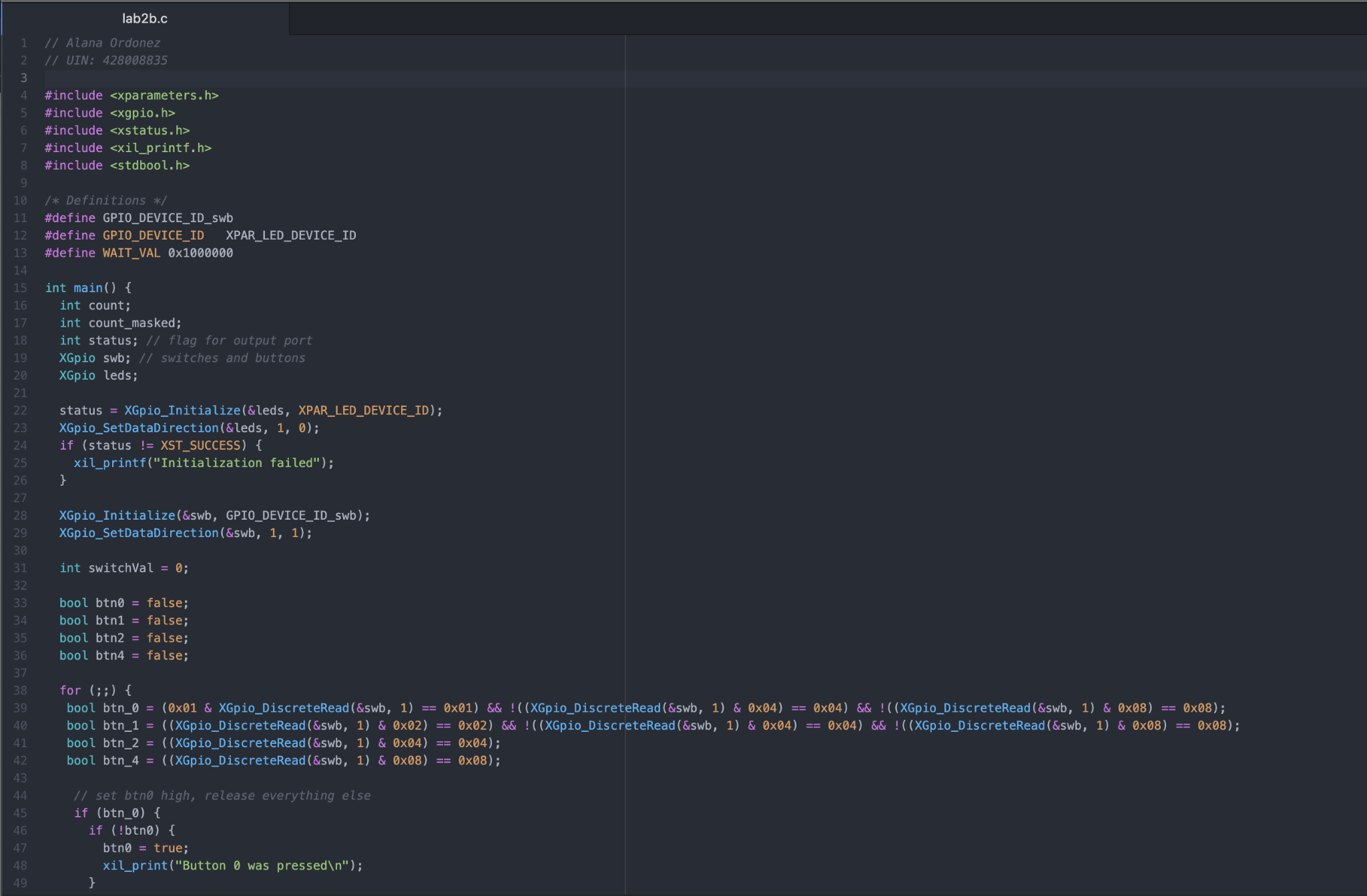
The full Verilog implementation from the last lab was much easier, in my opinion, mainly because it was much faster and more straightforward about the connections and logic. Doing the purely software implementation took much more syntax knowledge, as well as knowing how to visualize the connections and implementing it with C. However, this implementation can be much more versatile in terms of environments, and in this case, it was written in a ubiquitous language that is recognized by most, if not all, programmers.

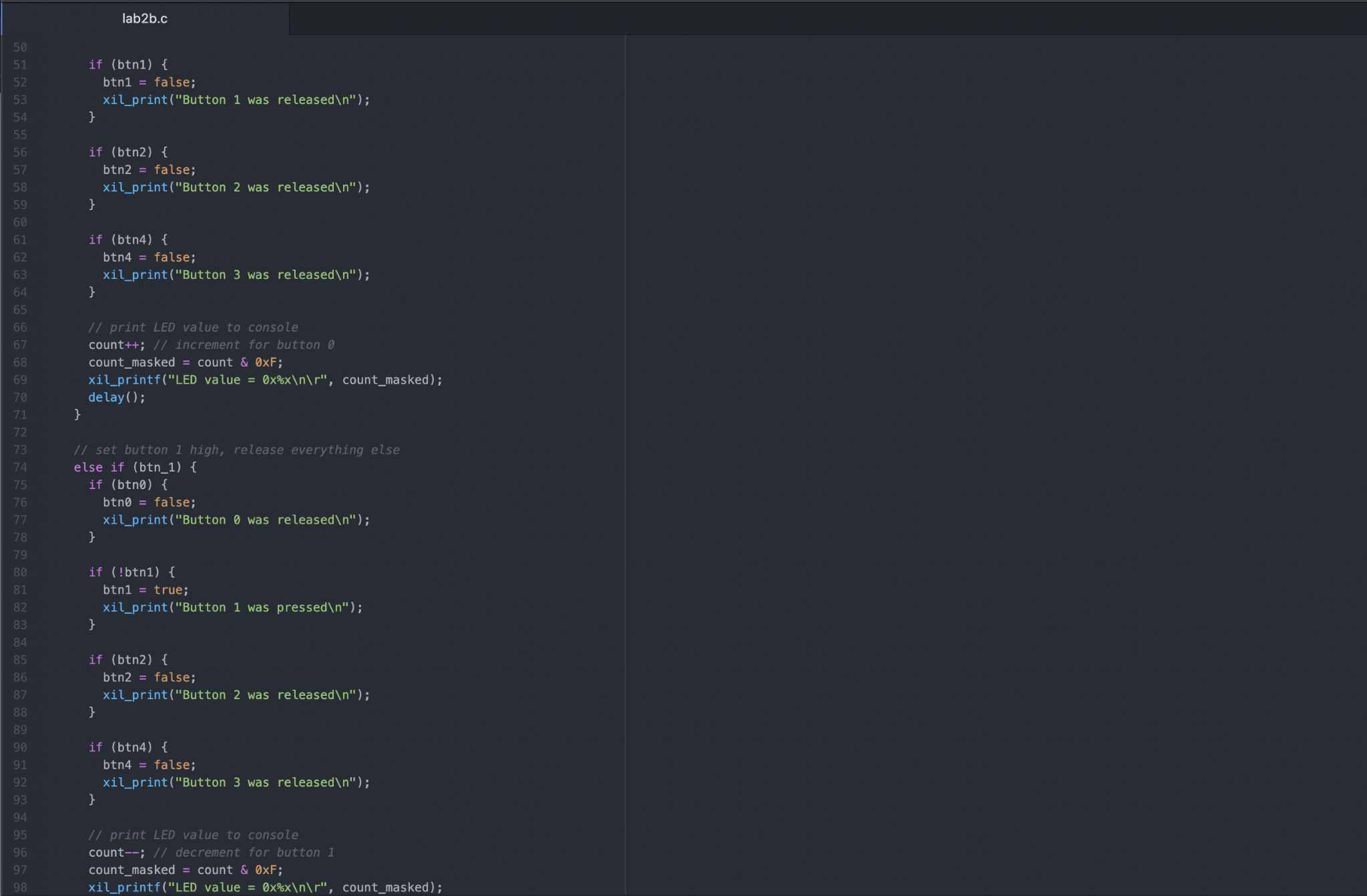
**VI. Appendix**

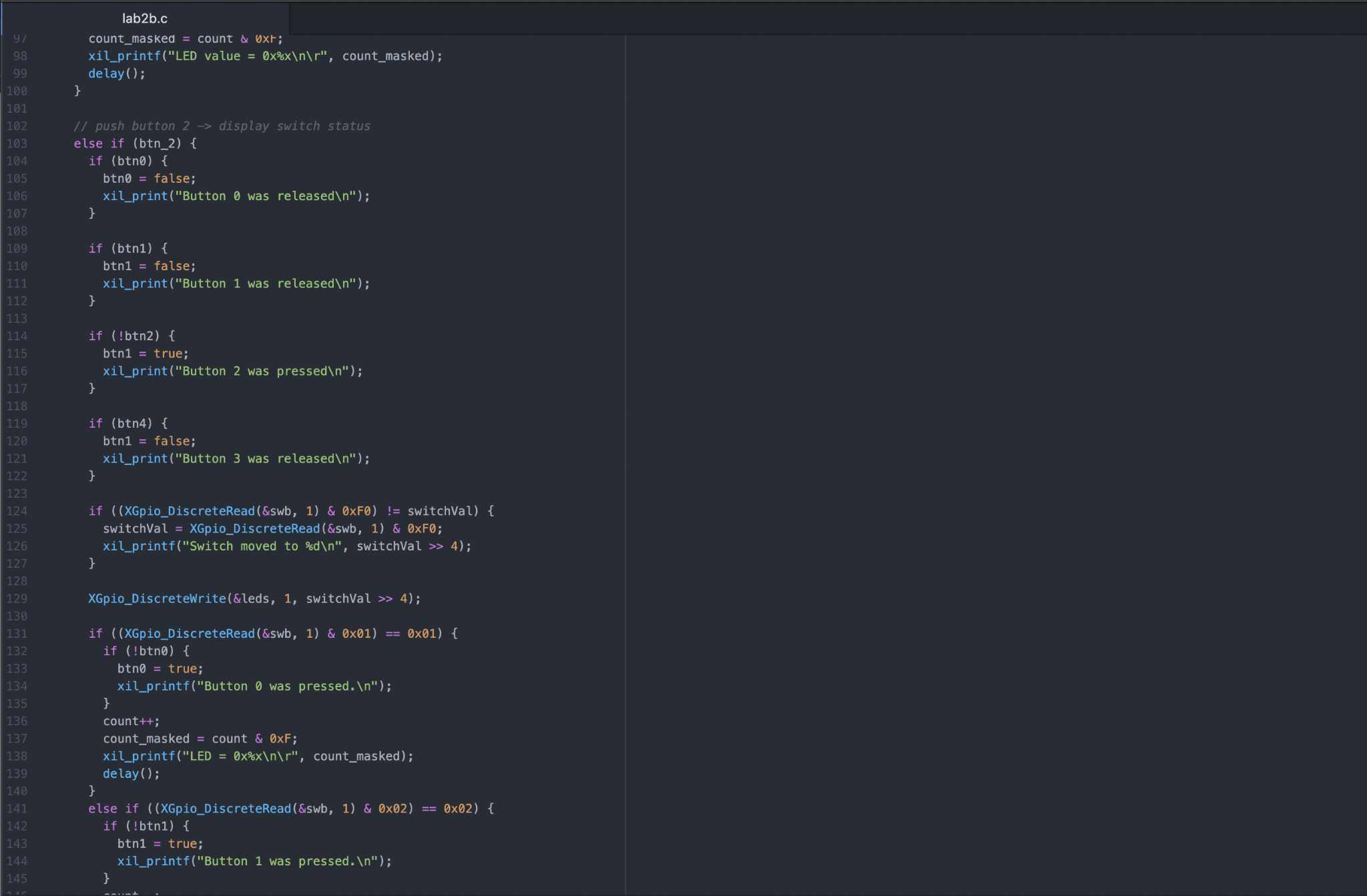
lab2a.c

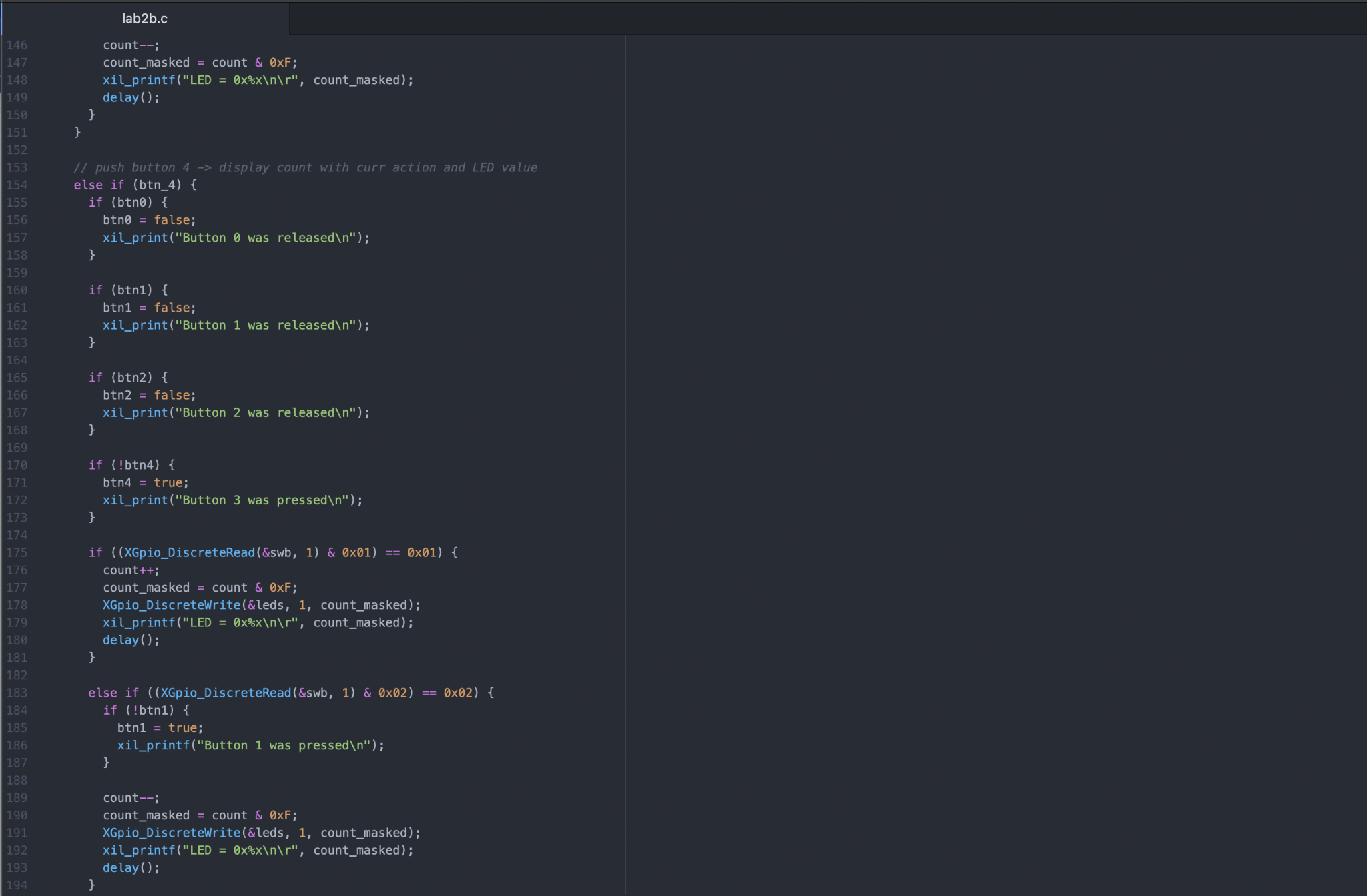


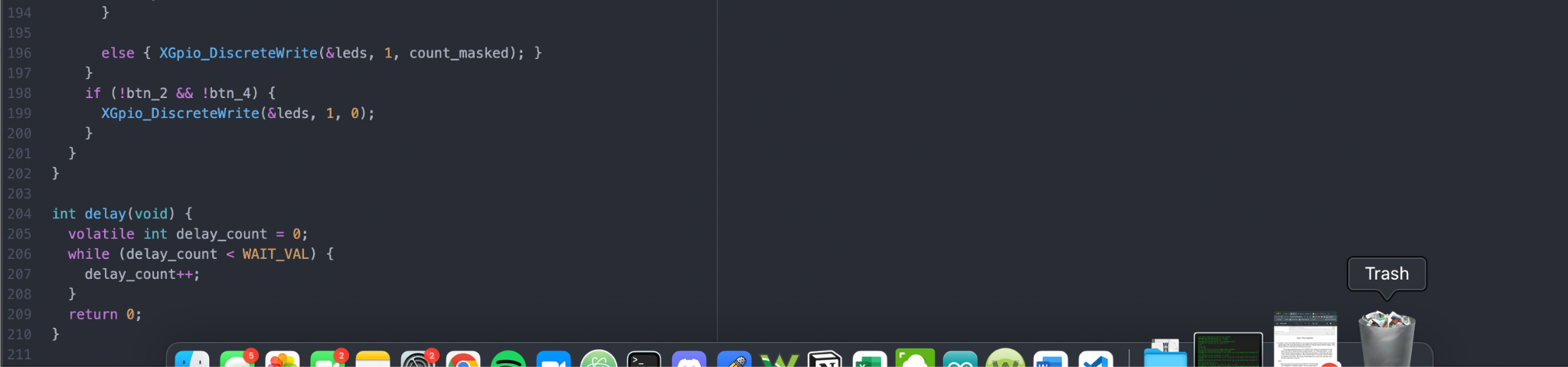
lab2b.c











led.xdc

