EEC172 Lab 1

ABSTRACT

This lab was completed on January 10, 2020. The overall task was to write a simple program in C and run the code on the CC3200 LaunchPad. In the first section, we got familiar with the IDE Code Composer Studio (CCS) and ran a simply "blinky" tutorial program on the board. This program caused the LED on the board to blink at a specific frequency. In the second section, we programmed the LEDs to blink in response to input, namely pressing the SW2 and SW3 switches. In the last section, we programmed a flask memory to ensure that the program we wrote in section two remained in memory after the board was disconnected from power. We verified this by disconnecting the board from power and ensuring the LEDs still blinked in response to the switches being pressed, accordingly.

ACM Reference Format:

. 2020. EEC172 Lab 1. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SOFTWARE AND HARDWARE TOOLS USED

- CC3200 Launchpad
- CCS Uniflash
- Our Wonderful Brains

2 GOALS COMPLETED

We used the debugger to run "Blinky", "uart-demo", and "Lab1", and used the pin mux tool to generate a specific configuration for our use case. Finally, we flashed "Blinky" and "Lab1" to the device.

3 METHODOLOGY

3.1 Experimental Setup

As explained in the data sheet, we had two switches programmed as input, 3 LEDs programmed as output, and a pin (P18) programmed as low-high output. The board therefore had GPIO functionality, but there was also UART functionality, since at button press a message would be displayed on the terminal. P18 was connected to an oscilloscope as well, in order to detect the change in low/high voltage.

3.2 Software Setup

Our code was structured with 3 main components: The main loop, the LED subroutines, and the switch status. The switch status was stored using GPIOPinRead() , which we would use in the main loop to switch between subroutines. The key to this portion is keeping track of the value that the switch takes as well as whether the switch has already been pressed, since we want certain things to happen only on the first press or when pressing the other switch. Therefore, on the first press (or switching the input pressed) we set a flag that will ensure that "SW2/3 pressed" is not logged to output multiple times. We also put the code to set P18 as low or high in that same logic. In any case, a routine will always be invoked: countingRoutine() or blinkAll(). This is ensured by having two branches that check the value of the switch and invoke their relative functionality at each pass of the loop.

4 SKILLS LEARNT

First, we gained an understanding of the CC3200 LaunchPad and the Code Composer Studio (CCS). With respect to the board, we learnt where diferent parts of the board, including GPIO pins, flask memory, switches, and LED lights were located. In regards to the CCS, we gained an understanding of the interface and how to debug, compile, and run C programs.
We gained familiarity with the GPIO and UART libraries, flashed a routine to the launchpad, and wrote C code for a hardware system. We learnt to use an oscilloscope to measure the output signal of a pin.

5 CHALLENGES

Flashing the program to memory was somewhat difficult, and we're still not sure what the issue was when we finally got the software to flash. There was a slight learning curve in understanding what GPIOPinWrite() did.

6 CONCLUSION

Overall, we learned a lot from this introductory lab. We gained an understanding of the software and hardware components of the board we will use over the course of this quarter.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn