Lab 1:
Development Tools and Lab Introduction

Luke Jones
ECE Department
University of California, Davis
Davis, CA, US
lpjones@ucdavis.edu

Jesus Delgado - Perez
ECE Department
University of California, Davis
Davis, CA, US
jadelgadoperez@ucdavis.edu

INTRODUCTION

Part 1

1. Open Code Composer Studio and create a new project
2. Import the blinky project from the cc3200sdk
3. Build the project and test that it works when debugging it on the board
4. Modify the blinky code to make the LEDs blink faster by decreasing the delay between LED changes
5. Build the modified project and test that the LEDs blink faster on the board

Part 2

1. Create a new project and import blinky
2. Open TI SysConfig and add the GPIO pins for the red, yellow, and green LEDs as well as the SW2, SW3, P18, and P2 headers
3. Then enable the RX and TX signals in the UART
4. Download the pin_mux_config.c and pin_mux_config.h files and import them into the project. Use them instead of the pinmux.c and pinmux.h files
5. Modify the blinky code to switch between two states. When SW3 is pressed the LEDs should count from 000-111 continuously and “SW3 pressed” should be printed to the console
6. When SW2 is pressed the LEDs should blink all LEDs on and off in unison and “SW2 pressed” should be printed to the console
7. When in the SW2 state the P18 pin should be set to high

Part 3

1. Open the UniFlash tool
2. Create a new configuration and set it up for the CC3200 on the right COM port
3. Put the path to the .bin file for the previous project in the Url section
4. Check the Verify and Update boxes
5. Put a jumper on the SOP2 spot on the TI board
6. Connect the board to the computer, click program
7. Once it finishes, disconnect the board and remove the jumper
8. Reconnect the board to the computer and the project should run

BACKGROUND

Hardware

- CC3200 LaunchPad - Board used to interact with the real world using code
- Micro-USB to USB-A cable - cable connecting the computer to the board

Software

- Code Composer Studio - Used to edit programs, build them, and upload them to the TI board
- TI SysConfig - Used to configure the pins on the TI board
- CCS UniFlash - Used to flash a binary file from a project to the board

Techniques

- Compiling - After writing code in C, CCS converts the code through compilation into a machine code executable that the TI board can understand and execute.
- Flashing - Loading the compiled firmware onto the hardware, usually in non-volatile memory

GOALS

The goals of this lab were to become familiar with all of the software and hardware needed to program on the CC3200 board. In the first part, we learned how to import projects and modify them to ensure that we understood how the code is interacting with the board by importing blinky and

modifying it to run faster. The second task had us write brand new code that taught us how to poll switches and go into different states on the board. Our task was to have two states controlled by switches where one state counted in binary and the other flashed all of the LEDs. The last task taught us how to flash projects onto the board so that programs don't need to be loaded onto the board after it loses power.

METHODS

Part 1

To decrease the delay between changing the state of the LEDs, we created a macro defining the new delay time as half of the previous delay time. Then we used the macro inside the previous delay statements effectively doubling the speed of blinky.

Part 2

In our implementation, we created 3 states instead of the required 2 for extra credit. Our 3rd state was blinky and it was accessible by pressing both SW2 and SW3. To implement this, we go into a while loop checking whether SW2 or SW3 are pressed indefinitely. Once either one is pressed, we poll the opposite switch for a set delay threshold. If the other switch is pressed, then it goes into state 3 which is blinky, If no other switch is pressed it goes into its corresponding state, counting for SW3, and unified blinking for SW2. Inside each state between changing the LEDs, instead of using a delay, we poll the switches as explained previously. We also had the pin18 set to high when going to the unified blinking state and set it to low when going to the counting state. This was verified using an oscilloscope shown below.

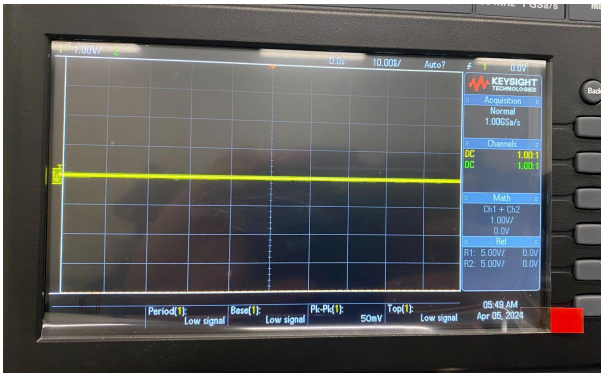

Figure 1. Image of Oscilloscope after SW3 is pressed.

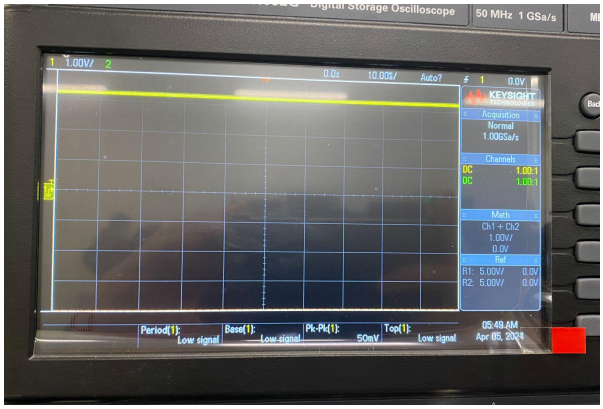

Figure 2. Image of Oscilloscope after SW2 is pressed.

Part 3

The flashing required no code and we used the UniFlash tool explained in the Introduction section.

DISCUSSION

The challenges we faced during the lab were all in part 2. We first struggled with figuring out the correct inputs to the write and read functions for GPIO in the documentation. Eventually we found them by looking up the manual for the board online. The other challenge was figuring out how to add a 3rd state to the program. This was solved using a buffer after pressing each button to give the user time to press the other button before instantly going into a particular state.

CONTRIBUTIONS

All of the work was done together on the same computer with both of us coming up with ideas to solutions for each part of the lab. The work done between us was relatively equal.