

# **EEC 172 SQ23 Lab One Report**

Bradley Manzo Thomas Ke⠀⠀

EEC 172 Student EEC 172 Student

916953788 917513004

brmanzo@ucdavis.edu thke@ucdavis.edu

# 

# 

# **INTRODUCTION**

This lab was essentially an introduction and tutorial to the tools that we will be using throughout the quarter. These tools include development software such as Code Composer Studio, UniFlash, and Pin Mux; as well as hardware such as TI’s CC3200 Launchpad.

# **Goals/Tasks**

# The objective of the lab is to familiarize ourselves with this quarter’s software such as Code Composer Studio and Uniflash and gain some experience using the tools to complete simpler tasks such as composing a binary number counter before continuing on to more difficult objectives in later labs.

Divided into three parts, Lab One tasked us with three objectives. The first part tasked us with programming the blinky and uart\_demo example programs onto the CC3200 board. The second part tasked us with creating a new CSS project and enabling the specified functionality through interfacing between the board and its peripherals, as well as establishing serial asynchronous communication between the board and a terminal. Finally the third part tasked us with flashing the program to the board’s nonvolatile memory using CSS UniFlash.

# **What we Should be Learning**

Part I

In part one, we should learn how to utilize the basic functions of CCS, such as creating a workspace, importing example programs, and then testing those example programs using CCS’s debug functionality and terminal emulator software to output the example program to our CC3200 Launchpad. We must also learn how to make simple modifications to the example code such as changing the timing of various functions.

Part II

In order to program the specified functionality to the board, we had to develop a strong understanding of UART communication and how to implement said communication through the “uart\_if.h” library functions. Universal Asynchronous Receiver/Transmitter Protocol allows for devices to exchange serial data with each other, in this case between the board and our terminal. Furthermore, because I/O devices such as the LEDs and Switches exist separate from and peripheral to the ARM Core, they are interfaced using Inter-IC Communication (I2C) protocol. In order to achieve this, we learned how to use the GPIOPinRead functions from the “gpio.h” library in order to address each pin and either read from or write to it.

Part III

In the process of programming the external flash memory of the CC3200 Launchpad, we learned the difference between volatile and non-volatile memory, being that the former is erased when powered off while the latter is retained.

# what we should be learning by doing these tasks

# Methods

- how you did it (**methods**/software architecture used to implement state-machine, etc.),

# Discussion of Challenges

and a **discussion of challenges** that were met/overcome.

One of the challenges that we overcame was lacking the precise knowledge on how to manipulate the pin mux configuration/relations, such as not understanding how to poll the SW2 and SW3 switches using the code.

# Contribution Breakdown

high level description of tasks done by each team member, and how they contributed to the team's work. It is understood that some tasks may be completed collaboratively by both team members.

Both of us worked together on Parts I, II, and III. Although, due to technical difficulties, Thomas was unable to set up the project’s software environment until later. As a result, we worked in the lab using Bradley’s computer to run the required software. We collaborated together in all of the parts, but once we experienced an issue in Part II that we were unable to solve in Tuesday’s timeframe regarding our lack of knowledge on how to poll some parts of the board, namely SW2 and SW3, Bradley spent some time out of class to research CCS and the CC3200 Board and found the method to do so. We were then able to spend time in the lab together to complete Part II.