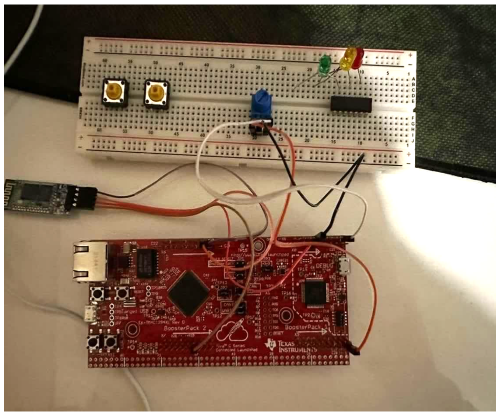
CSE 474: Lab 3 Report

Part 1: Procedure

Task 1: Analog to Digital Readings

1a: Controlling the LEDs using the potentiometer



To complete this task, it was needed to set up the Analog-to-Digital (ADC) module of TM4C1294NCPDT plus an internal temperature sensor on the board. The ADC used a Successive Approximation Register and also needed a Phase Locked Loop to successfully run. To initialize this component, it was necessary to enable the clocks, the multiplexer components, as well as the alternative select for some of the GPIO pins. Then the potentiometer was set up with the configuration in Figure 1. to give the internal temperature sensor a variable voltage divider value of 10kOhm that could be used in the equation for the potentiometer.

Figure 1: Potentiometer Set Up

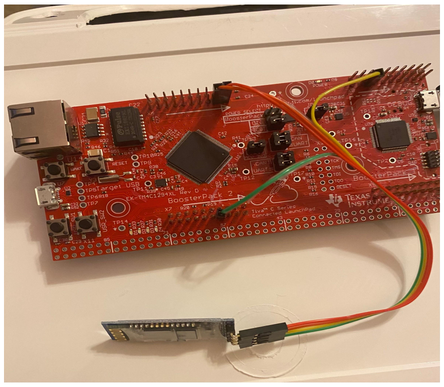
1b: Displaying board temperature

This task went a step further and made the ADC look for the value that uses the onboard temperature of the board. This requires the onboard temperature sensor to be configured and print the statements to the terminal. To do this, it was required to set up the PLL divider to make sure the values were reliable as well as two switches to alternate between 12MHz and 120MHz. This section simply required the initializations of the switches and the sensor and then the code to connect them all required the reading of the value and then printing it to the screen.

Task 2: UART and Bluetooth

2a: Sending temperature readings via UART

Instead of using the ADC to read the values, this method required the utilization of the Universal Asynchronous Receiver/Transmitter (UART) so that the reading were sent in a serial fashion, bit-by-bit, instead of through a different method. To accomplish this, we needed to initialize the UART in the same fashion that the ADC was set up but with different components. For instance, the UART also required the alternative function selection of GPIO pins as well as its own clock and system configurations. However, the UART required more transmitting and receiving units to make sure it was holding and sending the data correctly between the systems.



2b: “Return-to-Sender” function

This function worked very much like the temperature readings through the UART in 2a but instead of being hardwired, it was able to be accomplished through a bluetooth chip, set up in Figure 2. The data sent and received through this task was simply a keyboard input that would be read and returned on the terminal from the UART. This chip simply required that it was set up to the receiving and sending pins of the UART as well as grounded and power. With this, the bluetooth was acting as the correspondence between the UART instead of the cable.

Figure 2: Bluetooth Setup

Part 2: Results

Task 1: Analog to Digital Readings

1a: Controlling the LEDs using the potentiometer

We successfully completed this method by having the lights light up depending on the resistance displayed by the potentiometer. This meant that each range of values received from the potentiometer was to light up a certain amount of lights. One obstacle that was easily remedied was that not *all* the lights were lighting up even though the resistance was read correctly. This was due to the fact that the setting for the GPIODATA for the LEDs was not saving the previous LED (since for each range, more and more lights were supposed to light up, not just one).

1b: Displaying board temperature

This method worked similarly to the 1a and once the temperature module and the potentiometer were set up correctly, we were able to achieve the correct results. The temperature was in the 40s range and when we clicked the switches, a difference in the temperature was observed; the 12MHz switch caused there to be a cooling effect and the 120MHz switch caused it to heat up. In this sense, we were successfully able to receive and also adjust the temperature values that were being collected.

Task 2: UART and Bluetooth

2a: Sending temperature readings via UART

In this task we were trying to output the value of our temperature onto the display of Putty. The values of the temperature should be similar to the ones of 1b, which is what we saw in our demonstration. For this we had to check to see if the temperatures were being updated on the Putty and it was doing so. The only issue I ran into here was when my temperature display was too high but I found out one of my initializations was wrong.

2b: “Return-to-Sender” function

This function was easy to observe, although complicated to get correct. Once the bluetooth module was set up, which required the COM port to be found through the device manager and serially set up in Putty, then the device also had to have all its components perfectly aligned to read any input values from the keyboard. For instance, at once point, the code in IAR was giving us no errors but the putty terminal was not allowing any input. After thoroughly searching and adding a couple more initializations based on reading within the data sheet, there was still no response from the terminal. Then we tried changing the UART module from 0 to 2 and then also changed the ports from A1 and A0 to A7 and A6, which was advised by the teaching assistant but still did not fix the issue. It was also revealed that the wrong receiving flag was being checked with 0x20 but this was easy to fix with a 0x80, from reading the data sheet.

After more office hours, sessions, and observations, it was revealed that one of the TA’s had given bad advice about how to check the flag for if the UART receiver FIFO and the transmit FIFO were full. Rather than checking them the way they were, they needed to be checked for *not*, with a !, and this resolved both issues. After resolving the errors, the code worked successfully and was able to read a key input one-by-one into the Putty terminal, so when one key was pressed there was a one key press observed on the terminal.