EE 474: Lab Report 2

Michael Walsh, Josh Shackelford, Alex Finestead

TABLE OF CONTENTS

INTRODUCTION **3**

DESIGN SPECIFICATIONS **3**

Hardware Implementation 3

Software Implementation 5

TESTING **6**

JITTER ANALYSIS **7**

PRESENTATION OF RESULTS **8**

ANALYSIS OF ERRORS **8**

CONCLUSION **9**

REFERENCES **10**

Introduction:

In this second lab assignment we integrated a simple LCD display with our BeagleBone micro controller. The LCD was first mounted onto our breadboard, and then wired to the GPIO pins on the BeagleBone. These GPIO pins were accessed in much the same manner as our first lab project. Two different programs were written that interfaced with the display; the first is a program that allows the user to type on the computer keyboard and send this text to the LCD display; The second is a simple guessing game in which the user tries to guess a random number generated by the program. The LCD for this second program displays the current state of the game.

Design Specifications:

Hardware Implementation:

Before anything could be run, the LCD display had to first be connected to the BeagleBone correctly. To do this, GPIO pins on the BeagleBone were designated for each of the pins of the display. The following table (**Table 1**) was used to direct our wiring. Pins 14 and 15 are unlisted, but are used to control brightness. These were hooked up to ground to prevent any possible fluctuations.

Pin Number	Symbol
1	V_{ss}
2	V_{cc}
3	V_{ee}
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

Table 1: LCD Pin Assignments

A potentiometer was connected in series with the V_{ee} to give the user some control over the darkness of the LCD display. The potentiometer was set to a value of approximately 3kΩ; this gives the text on the display a dark enough texture to be visible without it being obscured by the background. Once all of the pins of the display were wired, the LCD could be initialized through software processes.

The BeagleBone Black's GPIO pins were hooked up in the following manner to the LCD module:

GPIO_48	RS
GPIO_49	R/W
GPIO_27	E
GPIO_115	DB0
GPIO_20	DB1
GPIO_60	DB2
GPIO_112	DB3
GPIO_66	DB4
GPIO_69	DB5
GPIO_45	DB6
GPIO_47	DB7

Table 2: GPIO to Pin Assignments

The following is a full wiring diagram of our BeagleBone Black, LCD module, and associated circuitry:

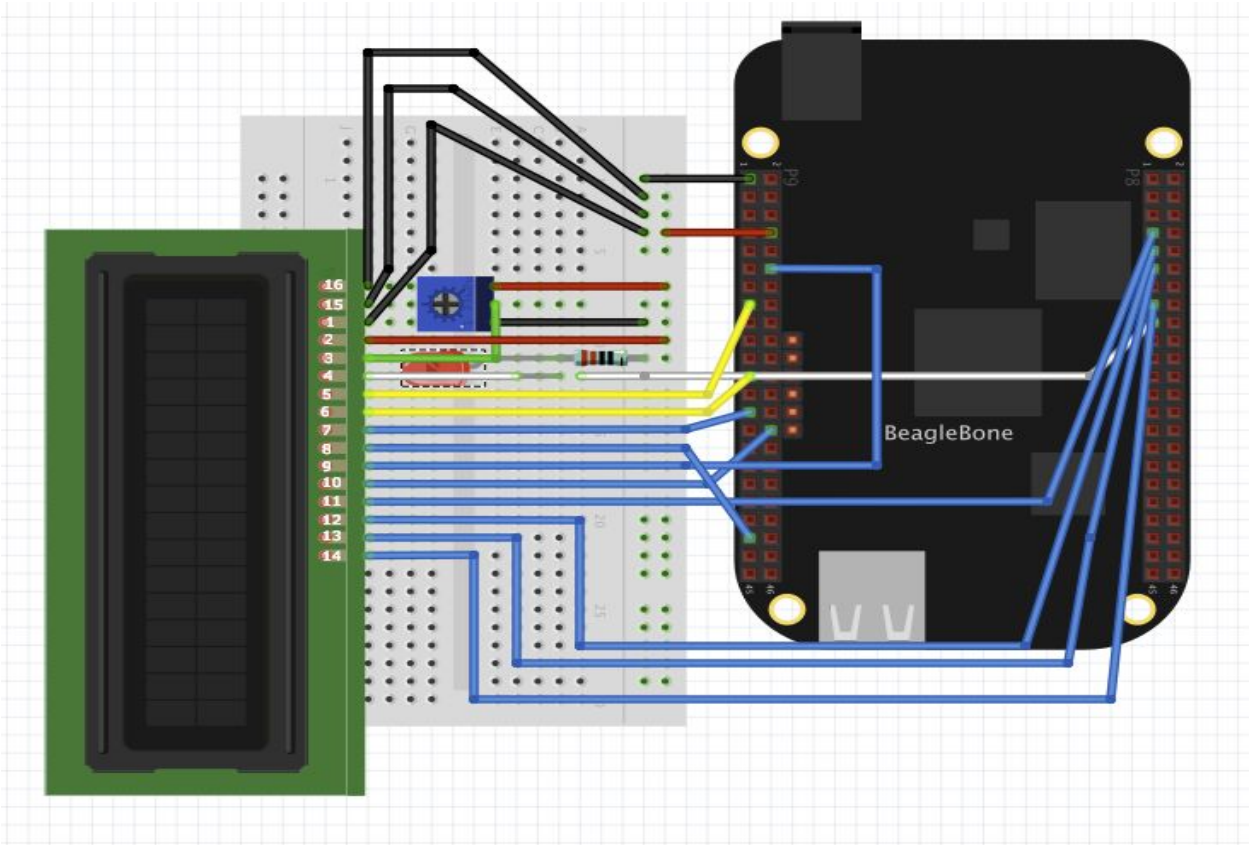

Figure 1: Full Wiring Diagram

As seen the LCD takes multiple different forms of input. The black lines represent ground, red represent 5V, green output of potentiometer, white line enable, yellow RS and R/W, and the blue lines representing the full data bus. An LED was placed in between the enable pins wiring to troubleshoot. This was also the same pin used to test jitter and an oscilloscope was placed where the LED is pictured in the diagram.

Software Implementation:

To get the LCD display up and running, a specific set of initialization instructions had to be sent to the display on boot up. In this initialization process, we specified our font size (5x7 pixel characters), the entry mode, and the cursor display status. Once this initialization was complete we could begin sending display data to the screen.

The next step was building a simple library to be easily able to perform screen functions simply. Such functions includes that of initial startup, set cursor, clear and return, write character, and write string. Our library also included simple wrapper functions to handle the beagle bones file I/O a bit easier by adding getStream and writeToStream functions to ensure no segmentation faults from I/O.

After this library was built, two simple programs were implemented with the LCD. The first was a simple guessing game where a random number was generated and took three user guesses from the terminal. This game had around five states that would be cycled on the screen. The block diagram is as follows:

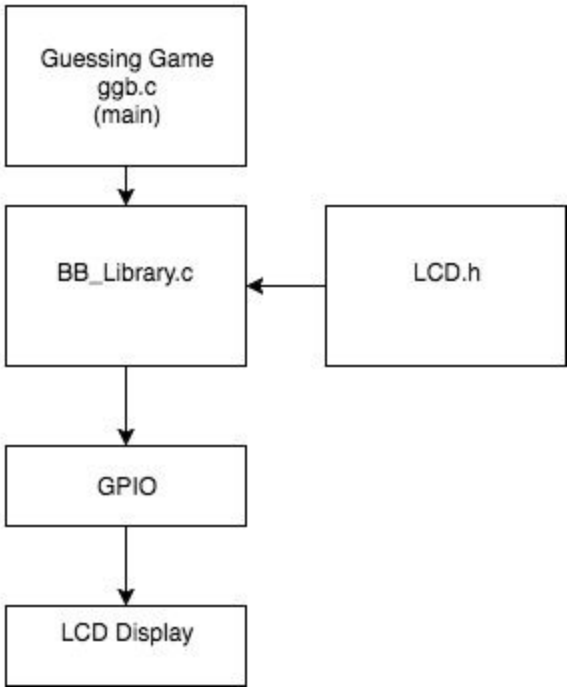

Figure 2: Guessing Game Block Diagram

The next part of the lab was to experiment with named pipes in the Linux file system and implement them within the C programming language. Our idea for a program to use this feature was a constrained user input to LCD write. To do this, one program wrote to the named pipe, and the other read from it. To do this two loops were used where one program was fed input from standard in and pushed it into the named pipe. The receiver would read the data, parse the data, and write to screen using the same library for guessing game. The block diagram for this program is as follows:

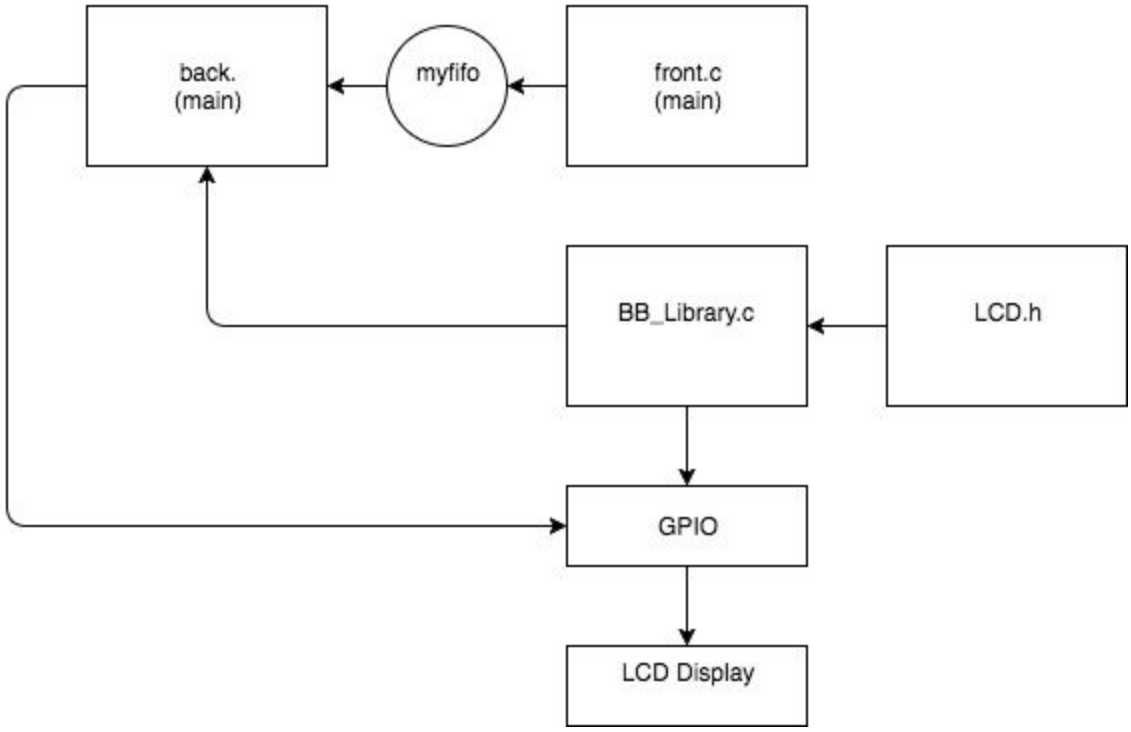

Figure 3: Pipe Project Diagram

Testing:

Testing of the LCD display was primarily done using the oscilloscope, or LEDs mounted on the solderless breadboard. Oscilloscope probes were used chiefly to ensure that outputs of the GPIO pins matched what was expected. Several discrepancies were caught this way that were vital to debugging the project. These errors are discussed in depth in the error analysis section. The oscilloscope gave us the ability to directly observe the signal being produced by the BeagleBone, making it easy to adjust hardware as necessary.

Infrequently, we encountered situations in which an oscilloscope was not readily accessible. In these cases, we made use of LEDs on the breadboard as a way of viewing the values coming out of the GPIO pins. A simple test was written in software to strobe outputs of 0 and 1 for all GPIO pins. Through this test, we expected the LED to blink predictably; if it did not, we knew that we had encountered an error. This test proved especially useful for finding burnt out GPIO pins.

Jitter Analysis:

In addition to running programs to interface with the LCD matrix. A jitter test was performed to see how accurately a GPIO pin could have been used for a clock. In this case GPIO 27 was used to generate a clock by sleeping 1 us via the usleep( ) function. Note that these measurements were done using an inexpensive oscilloscope probe at fast frequency. So some slopes are to be expected.

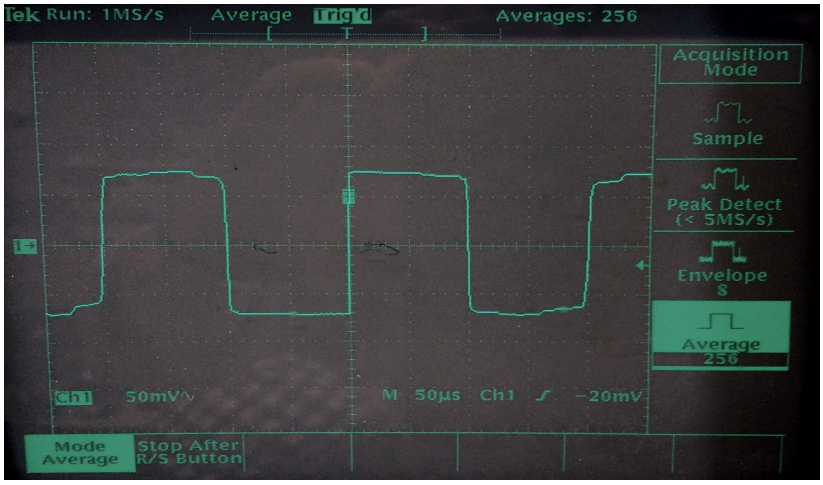

Figure 4: Average Sample of GPIO as a Clock

As shown, the BeagleBone Black produces a fairly accurate clock, but has some definite skew between clock edges.

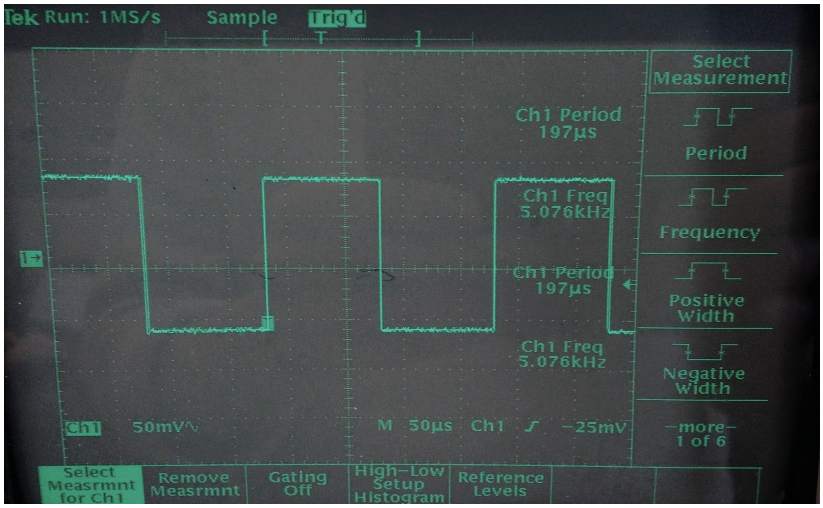

Figure 5: Period, Frequency Measurements

Though period was supposed to be rated at 2us via program. That was scaled up by a factor of 100 by the beaglebone. Conversely, that also means 100x slowdown as to what was expected.

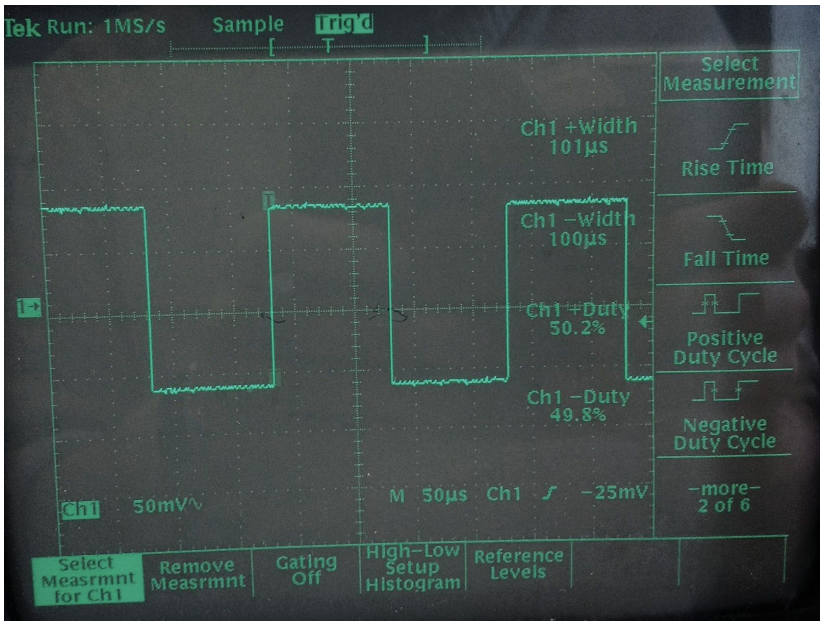

Figure 6: Width, Duty Measurements

For the brief analysis of jitter, there was a noticeable percentage difference between the positive and the negative duty cycle. The positive duty cycle tended to stay between 50-53% while the negative duty from %47-50. However, it was noticeable that this would jump at times to a breakup of 70% and 30% (high and low). This could be due to a multitude of factors, from properties of electromagnetics to the order the GPIO was set in the program.

Presentation of Results:

All programs functioned as expected, and to specification. The first program allows the user to write text in a terminal window and send it to the LCD display. This program limits the user to displaying only 16 characters per line, as per the size constraints of the display. The program correctly handles and interprets all ASCII symbols, letters, and numbers, as well as spaces, giving the user extensive display capability.

The second program also functioned correctly. The program starts by creating a random integer between 1 and 10, and prompts the user to guess it. The user is then given 3 opportunities to guess correctly. If the user gets the answer right, the LCD display will present the user with a “You Win!” screen; otherwise, the display will show “You LOSE”.

Analysis of Errors:

There was one major hardware issue that was encountered throughout this lab project, and several software bugs. All errors, however, were eventually resolved. The hardware issue involved malfunction of the GPIO pins, and was one of the first errors we had to address. Upon testing all pins with the oscilloscope, it was realized that GPIO 117 produced a consistent

triangle wave regardless of values written to the pin. This error was one that appeared in other lab groups as well, and so after trying to find a work around, we ended up moving to a different GPIO and not using 117.

Piping also presented many debugging issues. Our inexperience with pointers led to printing the entire contents of memory onto the LCD via improper loop conditions. The biggest issue however was presented from scanf and fgets while retrieving user input. scaf consistently cause many headaches as the string would contain many different kinds of null characters in it. This was specific to the BeagleBoard and output varied on different kinds of unix systems.

Conclusion:

In this second lab project, we integrated an LCD display with our BeagleBone Black development board. The display was wired onto a solderless breadboard and connected to the GPIO pins of the BeagleBone. Two programs were written in C to interface with the LCD display. The first was a program that gives the user the ability to write text in a terminal window and send it to the LCD for display. The display allows for two lines of 16 characters each. The second program is a random number guessing game. The user is prompted to guess a value between 1 and 10, and is given 3 chances to do so. The state of the game is shown on the LCD display.

This lab gave us the opportunity to explore in-depth the concept of adding peripherals to our BeagleBone. It also introduced us to more complicated software techniques that did not appear in the previous lab. For future developments, it would make sense to try to reduce the number of GPIO pins used by the LCD display as much as possible. Currently, the display takes up a whopping 14 pins on the BeagleBone, but this could be reduced significantly by making use of a shift register to send data to the pins of the LCD instead. This would free up several pins that can then be used for additional peripherals.

References:

[1] DOT MATRIX CHARACTER LCD MODULE USER'S MANUAL
https://class.ee.washington.edu/474/2017spr/lab_specs/Lab2/19988_Optrex_DmcSeries.pdf,
accessed 18, April, 2017

[2] NEWHAVEN DISPLAY CHARACTER LIQUID CRYSTAL DISPLAY MODULE USER'S MANUAL http://www.newhavendisplay.com/specs/NHD-0216BZ-FL-YBW.pdf
accessed 18, April, 2017

Block Diagrams created using draw.io
Wiring Diagrams created using fritzing

Peckol is watching

Always watching