**EE 474 Lab 1:**

**Meeting the Beaglebone Black and the Linux Operating System**



**University of Washington**

**by Brad Marquez, Joseph Rothlin, Aigerim Shintemirova**

**EE 474**

**April 14, 2016**

Table of Contents

1. **Abstract2**
2. **Development2**

**2.1 Connecting to the Board2**

**2.2 Transferring Files to the Board2**

**2.3 Accessing the GPIO Pins2**

**2.4 Counter Programming2**

**2.1 Pulse Width Modulation Signals2**

**3. Discussion4**

**4. Conclusion4**

1. **Abstract**

The purpose of this lab is to introduce the Beaglebone and the Linux OS. In the first part of the lab we wrote a C program to display an 8-bit counter between 3 LEDs on the Beaglebone. In part 2, we used PWM pins on the Beaglebone to play an analog sound. The code was added to the original code from part 1. As a result, the PWM played a different sound for each count value.

1. **Development:**

**1. Connecting to the Board**

Establishing the proper connection to the board is essential to accomplish the goal of this lab. It is done using SSH (secure shell) networking protocol commands. For instance, ssh [root@192.168.7.2](mailto:root@192.168.7.2) is a command that connects the PC to the Beaglebone board. Once, the connection is established, the program can be loaded to the board.

**2. Transferring files to the board**

Files are transferred directly to the board using SSH protocol. Our makefile cross-compiles the C code into the board for faster compilation.

**3. Accessing the GPIO pins**

The first part of this lab consists of designing a counter and displaying it on 3 LEDs. This is done by connecting each LED to a GPIO pin on a board in order to output the code through the pin.

Each GPIO pin is configured by locating its directory in /sys/class/gpio. We chose pins 45, 47, and 27 for each LED light that were each connected to the pin with 1 kOhm resistors in series to prevent LED burnout. In order to access each pin, the command echo PIN\_NUMBER > /sys/class/gpio/export is executed. This provides access to the device files where we were able to configure them.

**4. Counter programming**

Per lab spec we created a 0...7 down counter using K-maps. The following logic expressions were used to implement the design.

counter[0] = ~temp0;

counter[1] = temp0 ^ temp1;

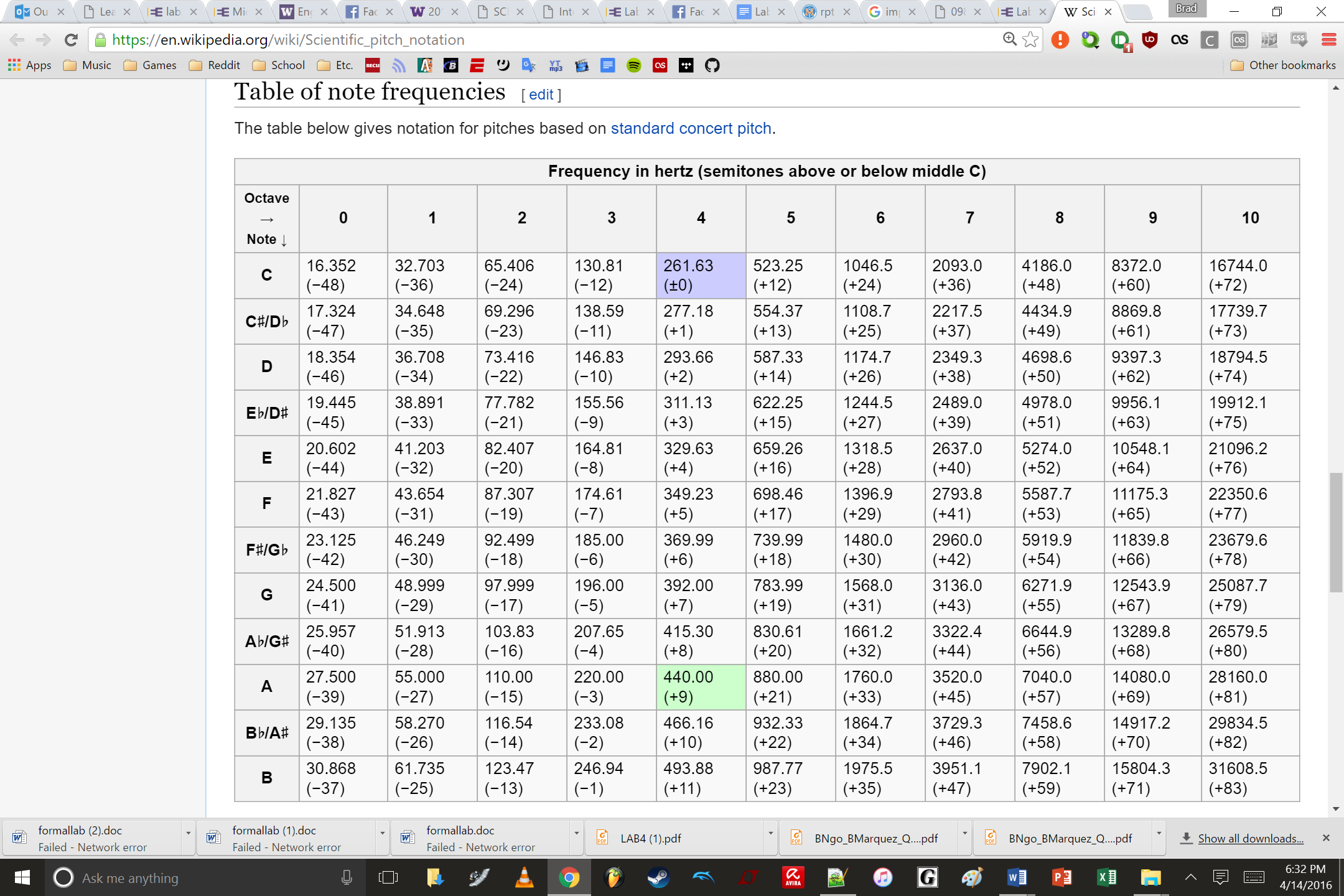
counter[2] = (~temp0 & temp2) | (~temp1 & temp2) | (temp0 & temp1 & ~temp2);

**5. Pulse Width Modulation Signals**

For the second part of the lab, we used PWM pins to play a different sound for each value of the counter. Essentially, we assigned a unique sound/note to each value of the counter based on the lab's specs. The frequencies were picked to be to emulate the Imperial March sound based on the note sheet shown in Figure 1.



**Figure 1.** The sheet music used to code the sound into music.c. Note that this song is in common time which means that each counter stage is a quarter beat.



**Figure 2:** The table used to convert notes to period which the pulse width modulation pin takes as an input. Note that period (nanoseconds) is the inverse of frequency, which is given in this table. The notes used were all in the 4th and 5th octave.

1. **Discussion**

One of the major difficulties that was faced during the lab was finding and playing the right frequencies in order to emulate the sound correctly. It was a lot of manual labor to find the appropriate frequency on the table and convert it to the appropriate period in nanoseconds. It’d be possible to write a script for this, but we would then have to consider whether the time invested in that script would outweigh the time saved. Another difficulty was figuring out how to play non-quarter notes. Our solution was to provide the sleep function individually for each case, as long as it summed up to one whole ‘sleeptime’ constant. This let us play notes that were shorter than a quarter beat.

1. **Conclusion**

The main purpose of this lab was to explore the Linux OS and the Beaglebone board. We were able to examine and configure GPIO and PWM pins on the board. For the first part of the lab, an 8-bit counter that was displayed on 3 LEDs connected to GPIO pins. Then, the code was appended to the counter code in order to assign a note to each value of the counter and emulate the song. The result is a 3-bit binary representation of a down counter on 3 LEDs and different notes played on a mounted piezo buzzer. The buzzer follows the change in states of the counter by having one quarter beat for every counter stage.