Assignment #2 – A Wunderboard Binary Clock & Implement \texttt{itoa(int x, char *str, int base)}
Due: Sunday, 04/21/13, 11:59pm

Part 1

You will take what you have working from Lab #2 and adapt it to make a binary clock. Your clock will use binary coded decimal, BCD, which means it will have two places for the hour, minutes, and seconds on the LED array:

\begin{tabular}{cccccc}
H & H & M & M & S & S \\
0-2 & 0-9 & 0-5 & 0-9 & 0-5 & 0-9 \\
\end{tabular}

To learn more about BCD, please read:

http://en.wikipedia.org/wiki/Binary-coded_decimal#BCD_in_Electronics

You can choose the color and whether you want to code a 12 H or 24 H clock. You will code each decimal number in the placeholder by lighting the corresponding binary number in the rows for that placeholder. For an example, see: http://www.thinkgeek.com/product/59e0/

Remember, the Wunderboard has to be programmed in C, not C++!!! You will need to make use of the \_delay\_ms() and \_delay\_us() function to have all the columns lit at one time, as well as getting your clock to keep time.

---

Part 2

For the second part of this assignment, you will write a C program that contains a function to convert an integer value to a string, without using any predefined string functions (\textbf{in other words, no, you can’t use sprintf, snprintf, itoa, or similar}). Since, \texttt{itoa()} is not a standard function that is only supported by some compilers, you cannot be guaranteed that it exists. Therefore, you will write you own \texttt{itoa()} function, \texttt{myitoa()}, which takes an integer, a pointer to a C-style string, and a base. For more detail, see http://www.cplusplus.com/reference/cstdlib/itoa/

Since gcc supports \texttt{itoa()}, then I suggest you call your function \texttt{myitoa()}.

- Your program must dynamically create a character array the exact size to hold the integer value as a string in a specified base. This is not done inside your \texttt{myitoa()} function, but needs to be done prior to calling your \texttt{myitoa()} function. Your \texttt{myitoa()} function expects
a string large enough to hold the integer in the specified base, and stores the result in the array.

- In addition, your program must handle converting N integers to strings, and your program MUST NOT dynamically create N character arrays on the heap!!! You must have only one character array on the heap at any given time.

In order to submit the files, you will be creating a bzipped tar ball. In order to do this, you will use the following command, adding all the source files to the end of the command:

```
tar -cjvf cs162_hwx_username.tar.bz2 file1 file2 file3...
```

This tar ball (replacing username with your ENGR username), and only this tar ball, will be submitted via TEACH.