CSCE 236 Embedded Systems, Spring 2015
Homework 3

Started: Thursday, Feb 5th, 2015
Due: Tuesday, Feb 17th, 2015 (start of class)
Remember to plan ahead for the checkoffs

Instructions: This homework is an individual assignment, collaboration is not allowed. If you discuss any problems with others, please note this on the assignment as described in the syllabus. Also note any materials outside of lecture notes, course textbooks, and datasheets that you used. Show your work and describe your reasoning to get partial credit if your solution is incorrect. This homework is due on the date listed above before the start of class.

Name:

Problem 1 (5pts)**.** *(To be completed at end of assignment) Approximately how much time did the total assignment take? Which problem took longest and how much time did it take?*

Problem 2. *Timing operations. In homework 2, problem 3 you estimated how many iterations of a loop it took to delay for a particular amount of time with different bit-length counters by using a stopwatch. In this problem, you will determine more precisely how long it takes to subtract two numbers from each other for different variable types.*

The first approach you should take is to look at the assembly code generated when you subtract two numbers. To view the assembly code you first need to find the directory where the compiler places the compiled code. To do this, configure the verbose compile output in the Arduino Sketch setup, which will cause it to print the full path when compiling. Then open a terminal/command prompt and go to that directory. Run the command:

```
avr-objdump -d -t -h -S file.cpp.elf
```

but you will need to replace file.cpp.elf with the right filename for your project. The command `avr-objdump` *is located in different places depending on your system and you probably need to specify the full path to execute it. On OS X it is most likely located at:*

```
/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-objdump
```

in GNU/Linux it should already be in your path so you can just type `avr-objdump`*. In Windows it is probably located at:*

```
\Program Files\arduino-1.0.3-windows\arduino-1.0.3\hardware\tools\avr\avr\bin\objdump.exe
```

There are also more details about how to do this for windows here:

http://rcarduino.blogspot.com/2012/09/how-to-view-arduino-assembly.html

and other google searches will give you results for other platforms.

a) (15pts)**.** *Determine exactly how many clock cycles it takes to subtract together two numbers from each other for types* `int8_t`*,* `int16_t`*, and* `int32_t` *(give me three different answers). When you are doing these subtractions, make sure that the numbers are large enough so that the compiler doesn't optimize the results and you may also need to use the results in other parts of your code so the compiler doesn't optimize it out. Include the relevant lines of C code you used* **and** *the assembly the compiler generated for each type (make sure to also include loads and stores). Hint: you should lookup how many clock cycles each assembly operation takes. I would also recommend indicating in the assembly code how long each instruction takes.*

b) (5pts)**.** *How many microseconds does each addition take for each of the three types (*`uint8_t`*,* `uint16_t`*, and* `uint32_t`*)?*

c) (5pts)**.** *Now instead of doing a subtraction, multiply the three different types (*`uint8_t`*,* `uint16_t`*, and* `uint32_t`*). What assembly operations are used and how long do they take?*

d) (5pts)**.** *How many microseconds does it take to subtract and multiple two floating point numbers (e.g.* 3.141*)? Explain how you determiend this for full credit. Hint: You probably do* **not** *want to to count assembly instructions. Note: I want two separate answers, one for subtraction and one for multiplication.*

Problem 3. *Schematics.*

a) (5pts)**.** *In Lab 1, Figure 1, the circuit for connecting the LEDs is shown. The LEDs "drop" a fixed voltage and the resistors serve to limit the amount of current that flows through the LEDs. The drops for each of these are RED 2.0V, GREEN 3.2V, and BLUE 3.2V*[1]*. How much current flows through each of these resistors (and therefore the same amount flows through the corresponding LED, which determines the brightness)?*

b) (5pts)**.** *If nothing else is using any current, how many LEDs could be controlled by processor? Hint: Look at how much overall current the processor can supply/sink.*

Problem 4. *Input*

a) (10pts)**.** *In Lab 1, you configured the button by using a pullup resistor. However, as you learned in class, the input pins also have the ability to activate an internal pullup resistor. Write the* `C` *code (setting registers* `DDRx`*, etc) below to activate the internal pullup resistor of pin* `PC4`*. Also give the "Arduino" version using functions we discussed in class to set pins.* **Instructor sign off required:** *You should also implement this and show the instructor the functionality of using your button in Problem 5 without the resistor.*

[1] As an example, for the BLUE LED, which drops 3.2V the voltage across the corresponding resistor will be 1.8V (since $1.8V + 3.2V = 5.0V$).

Problem 5. *For this problem you should complete the sections in* `morse.c` *where STUDENT CODE is indicated. To do this problem, you will need to include* `morse.c` *and* `morse.h` *in your sketch (use the menu* `Sketch->Add File`*. To call functions from* `morse.c`*, you will need to put* `#include "morse.h"` *in your main sketch file.*

Complete the `morse.c` *code so you will be able to send Morse code blinks. You should make sure that you are able to specify any of* `LED_RED`*,* `LED_GREEN`*, or* `LED_BLUE` *as the LED to output blinks or any combination of them (e.g.* `LED_BLUE | LED_RED`*). I would recommend creating helper functions that turn on or off LEDs so if you switch the pin that controls the LEDs, you only have to change code in one or two places. All of the code described here should be in a single program that runs at the same time.*

You must also turn in your code for all parts of this problem by visiting `http://cse.unl.edu/~cse236/handin/`**.** *Failing to electronically turn in your code will result in a 10 point penalty on this assignment. Points may also be deducted for coding errors, poor style, or poor commenting.* Note, you do not need to turn in a printout of your code for this assignment.

a) (5pts)**.** *In* `morse.c`*, the Morse blink pattern (dots and dashes) for each character are stored in a single byte. Read the code and describe how this is done and what the meaning of each bit is.*

b) (10pts)**.** *Now write the* `C` *code to turn on and off the LEDs by setting the registers (e.g. DDRx, etc.). What pins did you connect the LEDs to? How did you configure these pins as output? How do you turn the LED on and off? Make sure to include the relevant code here.*

c) (10pts)**.** *Write code that will blink "Hi W" when the board starts with "Hi" blinked with the green LED and "W" blinked using the red and blue together.* **Instructor sign off required, no written answer needed.**

d) (10pts)**.** *Now implement code that will output "d" (for dot), "D" (for dash), or "s" (for a long pause) over the serial port depending on how long the button is pressed. Use fixed times for your dot, dash, and long pauses (e.g. anything press under a second is a dot and anything over a second is a dash and any pause over a second is a long pause). Note that bouncing buttons can cause problems, so beware.* **Describe** *how you implemented this.* **Instructor sign off and written answer required.**

e) (10pts)**.** *Finally, implement code that will turn on the Red, Green, or Blue led if the Morse code for 'r', 'g', or 'b' is entered, respectively. Keep the LED on for approximately 1 second and then turn it back off.* **Instructor sign off required, no written answer needed.**

Do not forget to fill in the amount of time you spent on this assignment in Question 1.