Lab 2: Wireless robots and writing LCD drivers

In the first lab you gained familiarity with the Arduino environment, learned how to use the XBee as a wireless serial link (they are capable of a lot more than that by the way), wired a motor up to an H-bridge and got basic controls working for a simple robot. While the exact tools and interfaces were likely new to you, it was basically an introduction to a new environment.

In this lab, we'll mostly build on what we've done so far, but shift to thinking about interface design. The first thing you will design is a C++ class to control a LCD where your goal is to create an interface that is easy for *programmers* to use. The second thing you will do is create a *user* interface for your robot. You're goal here is to create an interface which makes it easy for the user to control the robot. Your goal here is to focus on designing with the focus on the *user* rather than focusing on what is convenient for *you*.

After that, you will get a very brief introduction to another microcontroller platform: the MSP430 Launchpad. The goal of this section is to expose you to another programming environment. The reason we've chosen the Launchpad is that it is an extremely cheap platform ($4.30) which is quite capable of doing useful things.

1. Pre-lab

As you'd expect, this pre-lab is mainly about getting you a background needed to be successful with the lab, but it also involves doing some interface (both user interface and software interface) work before you make it to lab. ***All pre-lab answers must be typed.***

Part 1: LCD and interface design

In this lab you will be designing a software interface for a liquid crystal character display (character LCD or just LCD) for the Arduino. You are going to need to design the C++ *interface* as well as write the code to *implement* that interface. The basic idea is that you want to create a class that allows for easy interface to the LCD.

Q1. Consider the standard a standard 16x2 HD44780 character LCD with an 8-bit interface[1].

a. Describe each of the interface pins on such a board.
b. Describe the command you'd need to turn on the display with the cursor underline off. Exactly what would have to be sent on each pin?

Implementation issues, such as those above, are certainly important. But let us also worry about the C++ interface. We want something that is easy for a beginning programmer to use, but allows for a full and flexible interface.

[1] http://www.sparkfun.com/datasheets/LCD/GDM1602K-Extended.pdf is a nice reference to use.

Q2. Design and document a class interface that allows for a simple, clean, interface to the 16x2 LCD. That is, define all the members (data and function) of your class. Think carefully about your interface: what might a user want to do? How might they wish to do it? What if they just want to print a message? What if they wish to just write a single character? What about non-ASCII characters? Make your interface as flexible as you can without overly complicating things. **The quality of your *design* (and its later implementation) will make up a large part of your lab score for this lab.** One interesting note: Do not rely on constructors when using Arduino—you are better off calling a specific initialization functions (at least with respect to using I/O pins). This interface design should be one that you could hand off to another group and they could code it. ***You'll want to have access to an electronic copy of this during the in-lab as you may be modifying it during the lab.***

Q3. Consider a different type of interface: controlling a robot with a keyboard. Describe an interface that would nicely allow for both direction and speed control. Specifically, the user should be able to navigate tight spots with a fairly high degree of precision while also easily moving rapidly across fairly open spaces. Think about an interface that would be ideal for winning a race that involves a fairly open space with a few turns as well as a tight space where you aren't allowed to touch the walls. Specifically explain how your interface would work in enough detail that if you gave your specification to someone else and they implemented it, you could drive the robot without first asking questions. (Later we're going to ask you to add other features to the interface like a way of sending arbitrary strings for display on an LCD. You may want to consider that though it's not required as a part of this pre-lab question.)

Part 2: Python

Q4. Consider the following Python code. Briefly describe what this code does. Specifically, what do you suppose `ser.write()` and `msvcrt.getch()` do? [2]

```
while 1:
    # Poll keyboard
    if msvcrt.kbhit():
        key = msvcrt.getch()
        if key == 'f':
            ser.write('C21FE')
        elif key == 's':
            ser.write('C21SE')
```

Part 3: MSP430 Launchpad

Q5. Do a bit of looking into the MSP430 Launchpad.

a. How much does it cost?
b. How does its specifications (RAM, Flash, number of GPIOs, number of LEDs, etc.) compare to the Arduino you have been using in lab 1?
c. Take a look at http://www.energia.nu/ and in your own words, describe its purpose.
d. What do the Launchpad folks call the equivalent of an Arduino shield?

[2] We expect answering this question will take only a few minutes. We just want you to have looked at the code and be aware that Python will be showing up.

2. In-lab

We'll continue working with the robot you started on in lab 1 and then move on to playing a bit with the MSP430 Launchpad. Be aware that Part 1 is quite time consuming. The other two parts aren't as bad.

Part 1: LCD interfacing

In the pre-lab, each student designed a C++ interface for the LCD (and each was asked to keep a copy). Now as a group the two of you need to decide how best to define that interface.

Q1. Redo the pre-lab question and come to a consensus about your LCD interface. This interface design should be one that you could hand off to another group and they could code it. You are to attach this interface to your lab write-up.

Implement your interface for the LCD as an Arduino sketch. We recommend you have the GSI look your design over before you start on your implementation—poorer designs will result in fewer points even if they are implemented well.

G1. Use your interface to write "Umich/AF" on line 1 and "31-25!" on line 2. And have line 2 change to back and forth to "But not 41-7!" every 2 seconds. Show your GSI.

Q2. Print a copy of your C++ class and attach to your answers.

Part 2: Python control

Now you are going to combine three different things: your working robot, the Python code written above, and your proposed interface for your robot. Agree with your partner what that interface should look be. Modify your robot's code so that you can implement your proposed interface. At the least this should involve being able to change speed. Use the same basic format for instructions to the robot.

G2. Show your GSI that your interface works and generally allows for solid control of your robot. Show you can navigate tight turns. Your GSI might set up a course and let folks compete. Maybe we'll give out a few small prizes (candy bar?). We'll see.

Q3. Print a copy of your Python code and attach it to your answers.

Now modify both your Python code and your Arduino code so that you can request that relatively arbitrary messages be displayed on your LCD.

G3. Demonstrate this to your GSI that you can display arbitrary messages via the Python interface while still being able to drive around.

Part 3: MSP430 Basics

Get an MSP430 Launchpad from the GSI. Connect it to your PC and walk through labs 2 and 3 of http://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/MSP430_LaunchPad_Workshop/LaunchPad.pdf. You will need to run

http://softwaredl.ti.com/trainingTTO/trainingTTO_public_sw/MSP430_LaunchPad_Workshop/MSP430_LaunchPad_Workshop.exe and put the created files in a reasonable place on your computer. *You should skip steps that involve soldering the crystal onto the board.*

G4. Show the fast blinking light that you should get at step 37 of lab 3 to your GSI.

Having walked through those labs, write a program which causes one of the I/O pins to generate a square wave with a period of about 1KHz if button P1.3 is pressed and 2KHz if it is not pressed.

Q4. Use your scope to measure the square wave. What is the average period and standard deviation (measured over at least 1000 samples)? (If you take the right measurement with the scope you'll get a LOT of data. It's a cool scope!)

Q5. Print a copy of your code which generates the square wave and attach it to your answers.

G5. Show your GSI your square wave on the scope.

3. Post-lab

Q6. The XBee's are capable of a (nearly?) full-fledged version of the ZigBee standard. Explain what ZigBee is and what features such functionality brings beyond the transparent serial links we are using them as. Describe a project where the ZigBee networking would be useful. We expect this answer to be about a page typed out.

Q7. Compare MSP430 environment and the Arduino environment. What are the advantages of the MSP430 environment? The Arduino environment?

Q8. What are the main topics of TI's Launchpad labs 4-7?