COSC 4P78 – Lab 2

Preliminary Task:
Before doing anything else, you'll need to finish your task from last week. Particularly important to remember is that you need a male header for connecting the serial cable.

Primary Task:
This one shouldn't require any soldering (that's what the solderless breadboard's for, right?)
All you need is a setup that satisfies the following requirements:

- Two LEDs
- Two momentary switches (buttons)
- Each LED should have its behaviour changed according to the switches:
    - Include some aspect of toggleability
    - Include selectable brightnesses (other than on/off)

Note that you're welcome to try something else a little different, so long you clear it with your lab demonstrator.

If you soldered in a light on GPIO 13, it's okay to use that as your non-dimming LED.

Serial cable
Each lab station has an FTDI cable. These add a serial COM port to the computer, that communicates with the microcontroller over TTL serial. As a reminder, pins 2 and 3 on the ATmega (GPIO pins 0 and 1 in the IDE) are used for serial. It's advisable to not dual-purpose them for anything else.

How you use them for programming is pretty interesting:
The ATmega has been preflashed with the *Arduino bootloader*. It's a very simple bit of software that, upon starting, checks the serial line to see if there's a program waiting to be loaded up into the flash memory. If so, that sketch is loaded in, and then executed. If no program is waiting to send, it just moves on to executing the last program that was loaded into its flash storage.

- Be very careful when handling the connector for the serial cable. Grip the plastic block; not the wires! (Trust me, it's easier to grip them wrong than you'd think)
- If you've connected the cable's +5V to your VCC rail, then **disconnect the 9V**!
- When you plug the cable into the computer, it should add a new COM port. You'll be selecting that COM port in the Arduino IDE
- To upload a sketch, you'll be clicking on the 'upload' button, and resetting the microcontroller just as the progress bar fills

Arduino IDE
An Arduino program is called a *sketch*. All of the programs you write will be stored in your *sketchbook*. It should be in a folder within your user profile.
When you first start the IDE, it should automatically create a skeleton sketch.

Refer here for a helpful guide:
https://www.arduino.cc/en/Reference/HomePage

The skeleton will include two default procedures:

- `setup` – code to run once upon booting
    - A common task here is to set the *pin-mode* of the pins you're using
    - This is also where you'd start the Serial connection, if you were going to actively communicate between the microcontroller and computer for the task (which, this week, you won't)
- `loop` – code to repeat as long as it's running
    - It's highly advisable to use this, rather than to use an indefinite `while` loop of your own

  including sketches, sketchbook

You may wish to also create your own procedures/functions, and also add your own defined constants.

You'll likely be starting off by loading up a premade example that comes with the Arduino IDE. Still save the program to your own sketchbook before using it, though.

Pinouts
Don't forget that you have a pinout 'cheat sheet'
(reminder for the link: http://tinkerlog.com/2009/06/18/microcontroller-cheat-sheet/)

Take special note of which pins offer PWM, as you'll be using it for dimming your light.

Note: From here on out, you should probably work out how to proceed on your own. You'll have to get used to 'reading the manual', as it were.
To start, look here:
https://www.arduino.cc/en/Reference/HomePage
In particular, you'll want to read the references for `setup`; `loop`; `HIGH` and `LOW`; `INPUT`, `OUTPUT`, and `INPUT_PULLUP`; `pinMode`; `digitalRead` and `digitalWrite`; `analogWrite`; and `delay`.

Furthermore, the Arduino tutorials are pretty good:
https://www.arduino.cc/en/Tutorial/HomePage in general
https://www.arduino.cc/en/Tutorial/Foundations is useful for starting out

And for today in particular:
https://www.arduino.cc/en/Tutorial/DigitalPins
https://www.arduino.cc/en/Tutorial/PWM
https://www.arduino.cc/en/Tutorial/Button

That said, you can still keep reading for some more general tips.

Connecting inputs and outputs
We'll have to decide whether we want our buttons to use pullups or pulldowns. Let's try both!
For my own example, I was using Arduino pins 3 and 4 (ATmega pins 5 and 6).

- I want 3 to use a pulldown and control the toggle light (I'll call it Button1 or A)
- I want 4 to use a pullup and control the dimmer (I'll call it Button2 or B)

My toggleable LED was on Arduino pin 13; my dimmable on Arduino pin 9 (ATmega 15)

- Note that you need to pick one that supports PWM for the dimmable one!

So, how to wire them up?
For A and the regular LED:

- Use a 10K Ohm resistor to connect the pin directly to ground
- Also use a button to (sometimes) connect it to 5V/VCC

Normally, the button will be 'LOW', but when you press it, it will go 'HIGH'.

For the LED, you know the drill.

- A 330 Ohm resistor to connect the LED to the pin and to ground
- This is literally the same as you did last week
    - If you soldered it in, you're already done

For B and the dimmer LED:
Pullups are easier in Arduino than pulldowns. Why? Because the ATmega includes *internal* pullups that can be enabled.

- Assuming we'll be doing this, just connect the pin (through a button) to ground
    - This is like your reset pin from last week, except even easier

For the LED, you connect it the same way (330 Ohm resistor), except to a PWM-capable pin.

How to begin with a sketch
Honestly, I'd suggest first loading one of the provided examples.

- In particular, the *Blink* example under *Basics* should seem very familiar…

Then, just start adding extra code. I've included my own at the end of this document (but I can't stress enough how you should probably write your own).

Once you're done, first 'verify' (compile) your sketch. Then, just click on the 'upload' button and reset your hackduino just as the progress bar fills.

```
#define LED 13
#define DIMBULB 9
#define BUTTON1 3
#define BUTTON2 4

int buttonA=LOW;
int buttonB=HIGH;
int lastB=buttonB;
int dimState=0; //one of 0, 1, 2, or 3

void setup() {
  pinMode(LED,OUTPUT);
  pinMode(DIMBULB,OUTPUT);
  pinMode(BUTTON1,INPUT); //I'll assume this is normally ground
  pinMode(BUTTON2,INPUT_PULLUP); //This is usually better for buttons
  //Note: I also could have had:
  //pinMode(BUTTON2,INPUT);
  //digitalWrite(BUTTON2,HIGH);

  updateDimmer();
}

void loop() {
  buttonA=digitalRead(BUTTON1);
  buttonB=digitalRead(BUTTON2);

  //For ha-ha's, I want A to just directly control the LED
  digitalWrite(LED,buttonA);

  if ((buttonB)&(!lastB)) {
    dimState=(dimState+1)&3;
    updateDimmer();
  }

  lastB=buttonB;
  delay(50); //measured in ms (there's also a microseconds version)
}

void updateDimmer() {
  analogWrite(DIMBULB,dimState*85); //analogWrite is how we do PWM
}
```