Microcontroller Lab 1 (class 10)

Please bring your Arduino, USB cable and computer to class. You should have installed the software and run at least the **blinky** example before class. If you haven't please read this page to get started.

Revisions:
05 April 2013 -- add lab section numbers

Reading

Pleaser read the following *before* class:

- Introduction to Microcontrollers -- Google document by E. Hazen
- On the Arduino Website:
  - Foundations -- *Required (read each sub-section)*
  - Examples -- *Recommended.* (feel free to try some!)

After the reading, you should know how to write a sketch (program) and download it into your Arduino. You should understand the following terms:

sketch, digital input, digital output, analog input, analog output
variable, constant
function, operator
flash memory, RAM, EEPROM

Doing

Switches and LEDs

First we are going to warm up by connecting some switches and LEDs to your Arduino and programming it to emulate various types of digital logic.

Wire 4 switches to digital pins on your Arduino
Wire 4 LEDs to (different) digital pins
Connect the common side of the switches to Ground
Connect the common side of the LEDs to Ground
Connect the Arduino GND to Ground

- [ ] **Lab 1.1**

Write a sketch which reads the switches and displays the value on the LEDs
You will need to enable the “pull-up” function as described in the reading.
Trouble? Make sure you use **pinMode** to set the LED pins as outputs.
Use **digitalRead()** on each switch and **digitalWrite()** on each LED.
Does a switch “on” result in a logic ‘1’ or logic ‘0’. Why?

▢ **Lab 1.2**

Write a sketch which performs the following logic functions on the switch inputs and displays the result on the outputs. You need two switches and one LED for each (if you like you can display all four functions at once). Show the results to your lab partner and double-check the truth tables.

AND	hint: C language “&”
OR	hint: C language “|”
NOR	
XOR	hint: C language “^”

▢ **Lab 1.3**

Connect a push-button to a digital pin on your Arduino.
Write a sketch which counts inputs from the button in binary and displays the count on the LEDs. Modify the sketch so that the switches set the maximum count in binary, i.e. if you set the switches to “1010” (off-on-off-on) the counter will wrap around at 10 (decimal).

▢ **Lab 1.4**

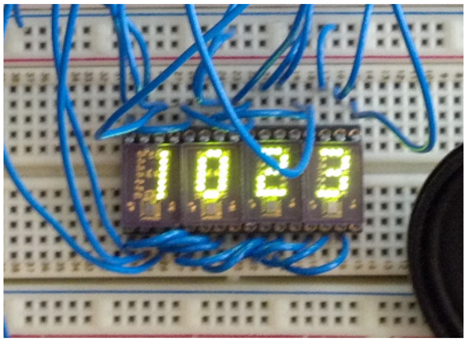
Modify your sketch to count seconds, starting and stopping when the button is pressed. Display the seconds in binary on the LEDs.

Project - Digital Voltmeter

For your first “serious” project you are going to turn your Arduino into a digital voltmeter, which displays an unknown voltage on a 3-digit display.

Display

First we need to figure out how to display information in a more convenient way than in binary on the LEDs. You are going to wire up a four-digit display like the one shown below. See page 348 in your lab manual or the HP5082-7340 data sheet for wiring details.



▢ **Lab 1.5**

Wire up one digit of the display. Connect four digital Arduino pins to pins 8, 1, 2, 3 of the display. Connect GND to pin 6 and 5V (from the breadboard) to pin 7. Tie pin 4 and 5 to GND.

Write a sketch which outputs various values to the four digital outputs. Figure out how to display the values "0" through "9". This can be done cleverly in about 3 lines of C code.

Notice how the value displayed changes as soon as the output from the Arduino is updated. For a multi-digit display, this would take a lot of wires (16, for four digits). There is a better way called *multiplexing*.

▢ **Lab 1.6**

Disconnect pin 5 of your display from GND and wire to an Arduino digital pin. This is the LE* (latch enable, active low) input. When it is at logic '0', the display track the inputs. When it is at logic '1', the display is latched (frozen).
Modify your sketch so that it updates the display as follows:

Output desired binary value on pins 8, 1, 2, 3
Output a logic '0' to pin 5
Output a logic '1' to pin 5

Now the value is latched by the display.

▢ **Lab 1.7**

Add three more digits to your display. Wire them just like the first one, except that the connections to pins 8, 1, 2, 3 should be *shared* by all the digits (parallel). Each digit should have a separate connection from a digital pin to the LE* input. Write a sketch which displays a four-digit number. This is a bit of tricky coding, and you may need to ask for some help. That's fine. Here are some hints:

- Update one digit at a time
- Use the "/" (division) and "%" (modulo) operators to get each digit's value

- [ ] **Lab 1.8**

Make a simple stopwatch which counts in 10ms increments up to 99.99 seconds (use the **delay()** function to delay 10ms). How fast can you start and stop the watch?

- [ ] **Lab 1.9**

Connect a potentiometer to provide an adjustable DC voltage from 0 to 5V and wire it to the A0 (analog input 0) pin of your Arduino.
Use the **analogRead** function to measure the voltage on the potentiometer, and use the code you wrote above to display the value on the digital display.

- [ ] **Lab 1.10**

Modify your sketch to calibrate the display so it reads in volts,
i.e. from 0.000 to 5.000. Compare the results with a VOM. Is it accurate? If not, how could you improve it?

You're Done!

Here are some "extra credit" ideas in case you want to go further with this:

- [ ] **Lab 1.11**

Replace the potentiometer with a thermistor (temperature-sensitive resistor) connected between GND and A0 on the Arduino. Connect a 10k resistor from A0 to +5V. Run your "voltmeter" sketch... you should see a voltage of about 2.5V for a nominal 25 deg C lab temperature. The voltage should change if you warm up the thermistor by putting your finger on it.

Now let's make a calibrated thermometer.

The thermistors in your kit are type **NTCLE100E3103JB0** (link to datasheet).
This data sheet will tell you how to convert the ADC value to degrees. First, look at page 2. Note that the part number **NTCLE100E3<u>103</u>...** corresponds to the line in the table below starting with "10 000", which is the resistance of the thermistor at 25 degrees C.

ELECTRICAL DATA AND ORDERING INFORMATION								
R_{25}	$B_{25/85}$-VALUE		UL APPROVED	SAP MATERIAL NUMBER	OLD 12NC CODE	COLOR CODE [3]		
(Ω)	(K)	(± %)	(Y/N)	NTCLE100E3....B0/T1/T2 [2]	2381 640 3/4/6.... [1]	I	II	III
470	3560	1.5	Y	471*B0	*471	Yellow	Violet	Brown
680	3560	1.5	Y	681*B0	*681	Blue	Grey	Brown
1000	3528	0.5	Y	102*B0	*102	Brown	Black	Red
1500	3528	0.5	Y	152*B0	*152	Brown	Green	Red
2000	3528	0.5	Y	202*B0	*202	Red	Black	Red
2200	3977	0.75	Y	222*B0	*222	Red	Red	Red
2700	3977	0.75	Y	272*B0	*272	Red	violet	Red
3300	3977	0.75	Y	332*B0	*332	Orange	Orange	Red
4700	3977	0.75	Y	472*B0	*472	Yellow	Violet	Red
5000	3977	0.75	Y	502*B0	*502	Green	Black	Red
6800	3977	0.75	Y	682*B0	*682	Blue	Grey	Red
10 000	3977	0.75	Y	103*B0	*103	Brown	Black	Orange
12 000	3740	2	Y	123*B0	*123	Brown	Red	Orange

Also note the value "K" which is 3977. Now look at page 4. Our thermistor corresponds to line 9 in the table (K=3977). From this you can read the constants A-D and A_1-D_1 which are used in formula (2) on page 4 to convert from resistance to degrees C.

Modify your sketch to use formula (2) and the constants from the table to convert to degrees C. If you store any intermediate values in variables (recommended!), use the "float" data type since the values are floating point.