Lab 1: Arduino µ-Controller: Generating & Observing Electrical Signals

1. Objective:

1. Putting circuits together and playing with them under the direction of Arduino.
2. Understanding the role of resistors in the circuit.
3. Understanding Arduino's digital output and analog input, and what you can do with them!
4. Learning how to measure what is going on in circuits: using multimeters, oscilloscopes, Light Emitting Diodes (LEDs) and speakers.

For the description of the components to be used in this lab, please refer to the *CastOfCharacters.pdf* that can be found on the course website under the lab. We assume that you read this file and familiarized yourself with the instruments and components covered there. If needed, we can/will give an overview/introduction accordingly.

Note: Arduino e-books posted on-line are excellent source of additional information.

2. Using Arduino to Generate DC Signals

In this part we will use Arduino's digital outputs to turn the LED on, and learn a few interesting things along the way. Assemble the LED circuit using schematic shown in Figure 1.

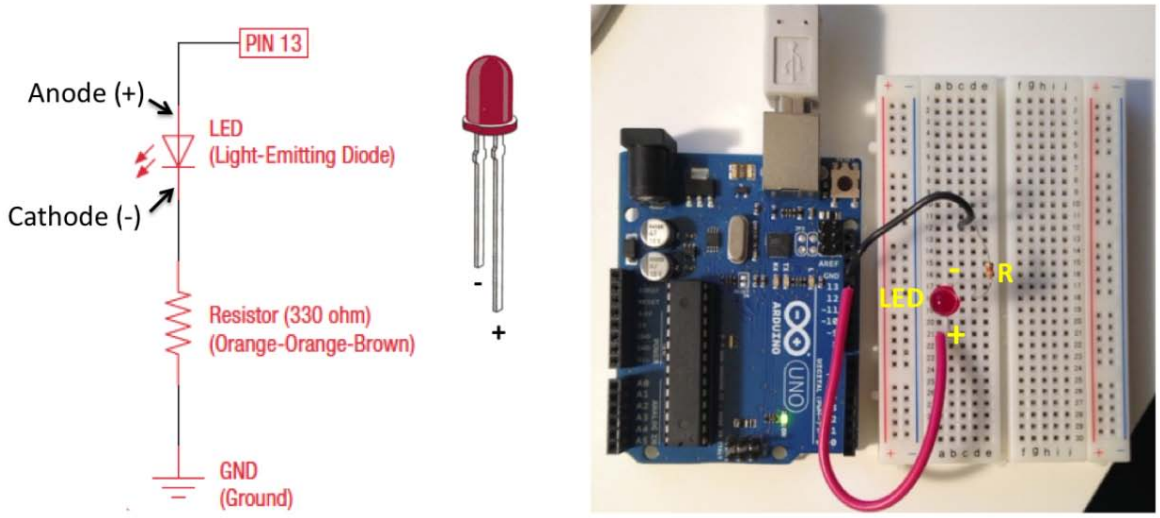

Figure 1. Schematic and photo of LED circuit. LED's Anode should be connected to PIN 13, whereas Cathode is connected to 330Ω resistor. The other end of the resistor is connected to Arduino's ground (GND) terminal. Note that Anode (the "+" terminal of LED), is longer than the Cathode (the "-" terminal).

Connect Arduino to the computer via USB cable, open the Arduino program and load up the file *BareMinimum*: (*File* > *Examples* > *01.Basics* > *BareMinimum*). This program is an

empty shell that we will add a few instructions to, and get Arduino to do cool stuff. Function *setup()* can be used to setup important parameters of the program, define variables etc. It is executed only once. Function *loop(),* on the other hand, can be used to include commands that you want Arduino to loop through over and over again (till the power is unplugged).

1. Add the following code to the setup function:

   ```
   pinMode(13, OUTPUT);
   digitalWrite(13, HIGH);
   ```

   Upload the updated code to the Arduino (by clicking on "->"). What happened to LED? Why?

2. Use multimeter to measure the voltage between PIN 13 and GND pin. You can do this by setting the multimeter as a DC voltmeter (turn the knob to V setting), and then connecting its probes (alligator clips come in handy!) to the points of interest. How high voltage did you measure? Now measure voltages across the resistor and across the LED. What voltages did you measure? What is the relationship between these three voltages? Why?

3. Now set PIN 13 to LOW and measure the voltage again. Make sure that you upload the new program to Arduino after you make the changes to it. What voltage did you measure now?

4. Now measure the voltage on one of the digital pins that isn't being used. What voltage did you measure and how does that compare to the previous two voltages?

5. Now let's see what is the effect of the resistor on the current flowing through the LED and LED's brightness. Include a 10kΩ potentiometer (variable resistor) in series with 330Ω resistor, and insert Amp-meter to measure the current that flows from PIN 13 to GND. Draw the circuit diagram that illustrates the changes that you need to make to accomplish this, and show it to your TF. Once TF approves it, build the circuit that you synthesized, and set PIN 13 to HIGH. How and why do the current and LED brightness change as you are turning the potentiometer knob?

   NOTE: Remember, that unlike with measuring voltage where you can observe a working circuit without modifying it, to measure current the multimeter needs to be part of the circuit: circuit needs to be broken and multimeter inserted in it in "series" with other elements. Also, we have to configure the multimeter as an Amp-meter in order to measure currents: move the red probe from the port labeled 'V, Ω' to the one labeled A (for "Amper"). Also, turn the multimeter knob to A position.

3. Using Arduino to Generate AC Signals

Turning the LED on is fun. But getting it to blink on and off is even more fun! You can use either the original circuit (shown in Figure 1) or the version with the potentiometer. You can take Amp-meter out of the circuit if you wish, though it does notmatter

1. Open the *Blink* program (*File > Examples > 01.Basics > Blink*) in the Arduino code editor. Upload the code. What is going on?

2. Change both delays in the loop function of the code to 100 milliseconds instead of 1000 milliseconds, and load the code to Arduino. What is going on? Why?

3. Connect the oscilloscope to measure the voltage between PIN 13 and GND. Make sure the oscilloscope is set to DC coupling with the probe multiplier set to the same value as the switch on the probe. Adjust the "SEC/DIV" knob to 50 ms. Also adjust the Channel 1 "Position" and "VOLTS/DIV" knob to make the signal appear as large as possible on the display.
4. What is the frequency of the signal? (Hint: count the number of horizontal divisions in a period and take note of the scale set by the "SEC/DIV" knob. Also try using the "Measure" button.) Is the signal a perfect square wave?
5. Change the delays to 10 milliseconds and upload the new code. Can you see the LED blinking? Use oscilloscope to confirm that periodic signal is still supplied to the LED and measure the frequency of this signal.

   NOTE: if a light were to blink on and off at a rate (frequency) higher than so called *flicker fusion threshold*, the light appears to be continuously on. The threshold rate has its origin in limitations of our visual physiology and is approximately 16 Hz (btw, how does this compare to the frame rate used in videos?). Fortunately, we have instrumentation that can respond more rapidly than our eye (or ears.) can.
6. Comment out the delays (by starting the lines of code with "//") and upload the code again. What is the frequency of the signal on the scope now? (you may have to play with SEC/DIV knob) If there are no delays in the code why do you think the frequency isn't higher? Is the signal a perfect square wave now? Is it a better or worse square wave than when we had a delay of 100 ms?
7. Now that you convinced yourself that "seeing is believing" let's try "hearing" instead of "seeing". Replace LED with small speaker (you can remove all resistors) and run the same code. Still, use the scope to measure the voltage between PIN 13 and GND. Can you hear anything? Put delays back in and try changing them in the range of 1 - 5ms. Does what you hear correspond to what you see on the scope?
8. And now the real deal. Start new Arduino sketh (program), and copy paste following set of instructions into it: *ardx.org/src/circ/CIRC06-code.txt*. Make sure that you modify the code so that it says

   int speakerPin = 13;

   Upload the code and enjoy :). Look through the code and try to understand what each line does. IF you have questions, do not hesitate to ask your TF.

   Finally, if you are musically inclined, try to compose something yourself (or you can search the net for some other songs….)

4. Sensing and Controlling the Physical World with Arduino

Arduino is adept at handling inputs and using its internal instructions (which you can program) to control the outputs. If we connect sensors, which give us some information about the environment (such as light, temperature, pressure, sounds), then we can control an output signal (like an LED) through information that the sensors give us. In this part, you will be using a photo resistor, which changes resistance based on how much light the sensor receives. In fact, as discussed in Lecture 2, many of the sensors you will end up using in ES 50

(potentiometers, photoresistors, pressure sensors, etc.) are resistors in disguise: their resistance changes in proportion to whatever they're sensing (light level, temperature, sound, etc.). Arduino's analog input pins measure voltage, not resistance. But we can easily use resistive sensors by including them as part of a "voltage divider" that we studied in class.

1. Measure and write down the resistance of photocell when: i) it is pointing to the light source (R_{light}) , and ii) when it is covered (R_{dark}).
2. Build the circuit illustrated in Figure 2. Connect one side of the photoresistor to 5 Volts (5V) output of Arduino. Connect the other side of the photoresistor to ANALOG pin 0 (PIN A0). Connect a 10kΩ resistor between PIN A0 and GND.

   This simple arrangement creates a voltage divider, with the photoresistor one of the two resistors. The output of the voltage divider (connected to A0) will vary with the light level. Using the resistance values you measured in step 1, calculate the voltage on PIN A0 with light on and light off. You can use the equations shown in lecture.
3. Hook up LED and 330Ω resistor as before, BUT make sure that LED Anode is connected to PIN 9 this time, as shown in Figure 2.

   IMPORTANT: PIN 9 has capability of *pulse-width modulation* (PWM) and PIN 13 does not – hence this pin switch. We will talk more about PWM later, but for now you can check *http://arduino.cc/en/Tutorial/PWM*
4. Download code *Lab1_Part4.ino* from the course website, and upload it into Arduino. How does the brightness of LED change as you cover the photoresistor?

   Look through this program carefully and try to understand every line - there are many useful tricks there. If stuck, talk to TF or check http://arduino.cc/en/Reference/HomePage.

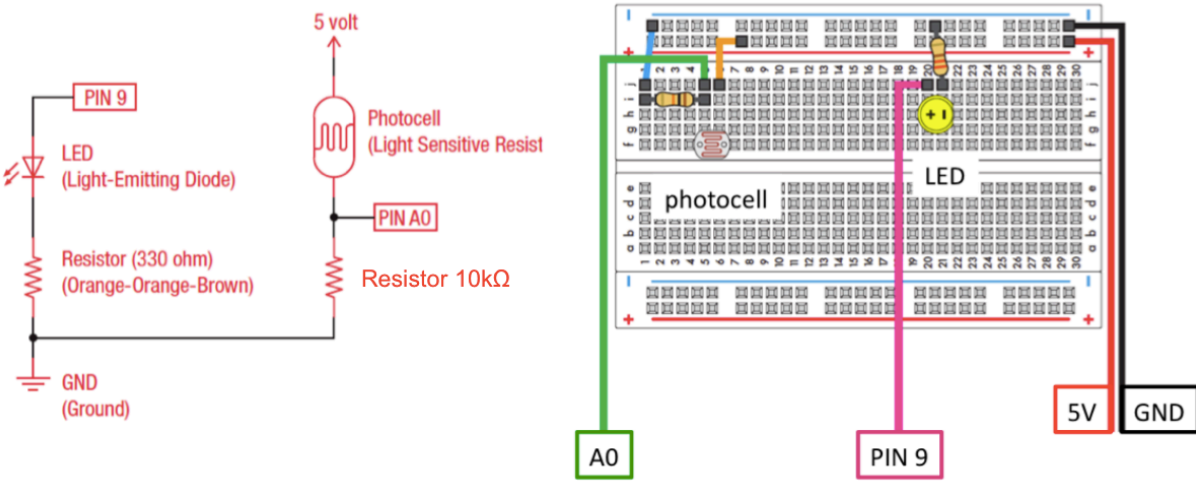

Figure 2. Schematic and layout of photosensor circuit. The brightness of LED depends on the amount of light that hits photoresistor.

5. Now lets try pressure sensors - they are resistors too! Simply replace the photocell with pressure sensor and try the same code. Does LED brightness depend on the pressure you put on the sensor (you may need to use two fingers btw ☺)? You may need to modify the code a bit and adjust the values of the resistor in the voltage

divider (in series with your sensor of choice). You may also want to measure the resistance of pressure sensor with and without pressure applied.

6. If step 5 fails, simply stick in 10kΩ potentiometer instead of the photo-resistor. LED brightness should change as you are turning the potentiometer knob.

5. Fun

You are almost done! Now.... go wild! You can try doing another circuit from the SparkFun kit (fun circuits are 3, 5, 7, 9 and 10) or try building something yourself!