DORDT UNIVERSITY

Engineering Department

EGR 304 LAB #2 GETTING STARTED WITH ARDUINO SPRING 2020

OBJECTIVES

- To install the Arduino IDE on the lab's computers (optionally on your own too) and get it working.
- To be able to properly connect LEDs and switches to the Arduino's digital I/O pins.
- To understand and the code needed to configure digital I/O pins and to read or write data to or from the pins.

LAB EQUIPMENT

- Lab desktop-style computer running Windows 10
- Power supply (9 V)
- Solderless breadboard
- General purpose lab equipment such as a DMM, VOM, patch cords, etc.

REFERENCES

- Joe Pardue's "Arduino Classroom," First serialized in *Nuts and Volts* Magazine beginning with the January 2014 issue and now available in several other formats.
  We will be using Chapters 1 through 4 of "Arduino 101." The entire "Arduino Classroom" is available at
  https://web.archive.org/web/20181216185738/http://arduinoclassroom.com/index.php/arduino-101
  Chapters 1 through 4 are available as PDFs from Canvas. (Maybe these are more convenient.)
  Chapter 1 Getting Started Chapter 2 LEDs Chapter 3 Programming in C Chapter 4 Pushbuttons

- The official Arduino web site: http://www.arduino.cc

- A web search engine is your friend. Some questions on the topic of this lab can be answered via a search.

DELIVERABLES

After you get your version of "Cylon Eyes" working please invite the instructor to your workbench and demonstrate your version. Then save your code (in the IDE) as a *.ino file and upload it to Canvas using the link provided in Canvas for this lab.

ACTIVITIES

Read the Arduinoclassrrom.com tutorials, Chapters 1 through 3 as needed to get up-to-speed again on driving LEDs with an Arduino. See especially Chapter 2, Figures 28 and 29 regarding the polarization of the LED. Note that a resistor must always be used in series with the LED to limit the current flow. If the resistor is missing (or if the resistor is of too low of a resistance) then the output pin of the Arduino Uno will of necessity limit the current flow. That is bad because it will cause heating in the microcontroller that should not be there. To calculate the correct resistance you need to know three things.

V_S is the power supply voltage, 5 V in our case.
V_f is the forward voltage drop of the LED, which varies according to the color of the diode's light.
I_f is the nominal forward operating current for the LED.
This could be anything from 100 uA to tens of amperes, depending on the model of LED used.

For real accuracy you should find your LEDs forward voltage and nominal operating current from the datasheet of the LED and then calculate the needed series resistance from this formula: $R = (V_S - V_f) / I_f$.

However, for this lab such accuracy is not needed. You may just assume that V_f is 1.2 V for infrared LED's, 1.8 V for red and yellow LEDs, 2.0 V for ordinary green, and 3.2 V for high-efficiency green and for any blue LED. (Notice how the forward voltage drop increases as the wavelength of light decreases.) Most small LEDs such as are stocked in our lab supply have nominal operating currents of 10 to 30 mA. The Arduino Uno is specified to allow 40 mA per output pin maximum. Typically, you should design for quite a bit less. So, as a practical matter we must limit I_f to about 30 mA no matter what the LED's specification is.

As a simplification of the situation, notice that 5 V applied across a 1 KΩ resistor (no LED in series) results in 5 mA. Thus to save time and effort "designing" the correct resistance value, you can just choose a 1 kΩ resistor and be 99.9% sure the LED will light reasonably brightly and not overload the Arduino Uno. Or more aggressively, choose $R = 330$ Ω and the current will still be less than 15 mA.

THE GOAL OF THIS LAB

Today's main goal is to design a Cylon Eyes display using your Arduino Uno board. Use about eight LEDs. (Or use exactly eight!) Your version of Cylon Eyes must have a way to change the speed.

There are many hobby web pages giving instructions on how to build this. A typical video of the resulting project is this one: https://www.youtube.com/watch?v=Mx0e8GRNg5c Many of these web pages are sort of like a paint-by-numbers project. They give detailed step-by-step directions, but you do not necessarily learn why the steps work—they are not very artistic overall. Often the code presented in hobby web sites is super-simplified and thus not very representative of industrial-grade code. You are welcome to consult these hobby web sites and probably you will be able to pick up some good ideas from them if you do not follow them too literally. Evaluate several of them. However, instead of simply doing the steps from one web site, find some amalgamation of good ideas and create your own version of Cylon Eyes. (BTW this project is also known on the Web as "Knight Rider." The names, "Cylon Eyes" and, "Knight Rider" relate to old TV shows.)

Once you have your Cylon Eyes display working, take a look at the Arduino Classroom 101 Chapter 4. Give special attention to Figures 12 and 13 in that Chapter. (Note that in most renderings of Figure 13 ground symbols near the bottom of the figure got cropped off. The wires that "just end" are supposed to be grounded.

You may use a "DIP" switch, toggle switch, slide switch, or other switch instead of a pushbutton if that is more fitting for your need.

Note that Chapter 4 Figure 13 shows the *positive-true* style of making a digital logic signal with a switch. In this case when the switch is open the logic value generated is logic-0, false, or 0 V, whichever way you want to label it. This method works fine with CMOS logic such as is used on the Arduino Uno. It does not work well with some other families of logic, especially TTL, although it can be made to work by lowering the value of the resistor (and increasing power consumption).

Figure 1 in this handout, shows another way to generate a digital logic signal with a switch. This is the negative true style when the switch is open the logic value generated is logic 1, true, or 5 V. This configuration works well with all logic families. It has a minor disadvantage in that when the switch is closed or "on" then logic-0 is generated, which is counter-intuitive for some people. Just mount the switch "upside-down" to flip its physical action and then invert the logic with code in your software and then you have a switch that does everything as expected.

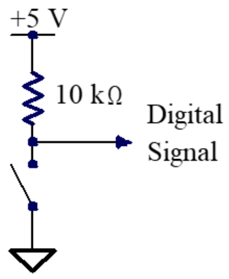

Figure 1, Generating a logic level with a pull-up resistor and a switch.

Explore connecting a switch to control one LED via the method as described in Arduinoclassroom Chapter 4. Then try the circuit shown in Figure 1 in this handout and write some code to test it.

After you understand how switches can generate a digital logic signal, use two switches to generate two digital signals. Let your code read these signals and based on them, operate your Cylon eyes at four different speeds: If the digital signal is "00" then the display should be frozen—speed is zero. If the signal is "01," then slow. If the signal is "10" then medium, and if "11" then fast.