School of Advanced Technology

CST8227 Lab 3: Simple Series Circuits, Analog Outputs

Due date: the beginning of your lab period, Week 4

What you will do:

1. Confirm maximum safe current values for the ARMCortex-M4
2. Research and determine correct operating parameters for the “tri-colour LED”
3. Correctly prototype a simple circuit using a breadboard
4. Use analog outputs to obtain variable brightness from a LED
5. Use appropriate delay functions to cycle through a range of values
6. Use a polling loop to read a momentary contact switch

What you need to submit and when:

1. The pre-lab for this lab, as per the due date indicated on Blackboard.
2. Complete the demo of: [2 marks each item]
   (1) automatic colour-cycling with smoothly varying colours (task 4);
   (2) push-button controlled colour-cycling between fixed colours (task 5).
3. The post-lab online quiz.

Marks:

Each of the demos identified above are weighted equally, and combined to generate your mark for the in-lab portion of this week's lab.

Required Equipment:

- Computer with Arduino IDE & Teensy extensions installed and working
- Teensy board and USB cable
- Resistors of different values, based on calculations you have designed for (eg. 220Ω, 330Ω, etc..)
- Tricolour LED
- Push-button switch

References and Resources:

- You will need to memorize the "resistor colour code" (BBROYGBVGW, silver, gold)
- Course textbooks: “Beginning Arduino” and “Getting Started with Arduino”
  (For examples and explanations of how to set pin modes and reliably read a switch)
- Layout of breadboard: electrical connectivity among rows & columns
- Spec sheet for ARMCortex-M4 and the tricolour LED.
- PartsList-Explanations file with part numbers, for referencing data sheets
- Arduino API references for digitalRead(), digitalWrite(), and analogWrite() functions.

Task 1: Determine Sink/Source Limits for the ARMCortex-M4

Step 1: Locate the data sheet for the processor on the Teensy 3.1 board.

Go to www.freescale.com and do a Search keyword/part number for the processor. The PJRC website specifies the part number for the processor. Alternatively use a magnifying glass to see what part number is printed on the chip - this will also confirm what the manufacturer is saying about their product – don't forget – ***you*** are the engineer. Hint: there are a few documents to choose from – a suitable one would be the pdf with the title "K20P64M72SF1.pdf".

Step 2. There is a lot of information in a data sheet – you are going to have to decide what information is important for the task. There are at least two sections of interest for now. Navigate to "Ratings", then to "Voltage and current operating ratings" (Section 4.4). The table of data lists absolute maximum operating parameters. The parameter that you are interested in is I_D.

Then navigate to section 5.2.3 Voltage and current operating behaviors, Table 4. Voltage and current operating behaviors. There is a current listing for the situations when there is a high output voltage (V_{OH}) and a low output voltage (V_{OL}). This is the current that we will design for when calculating a suitable size resistor to install into the circuit.

Task 2: Obtain Operating Parameters for Tri-colour LED

Step 1. You'll need to start with the part number for the tri-colour LED from the file "Parts List-Explanations" posted on the course web site.

Step 2. Obtain the part spec sheet using the manufacturer's part number & web site: http://www.hebeiltd.com.cn/led.datasheet/599R2GBC-CC.pdf

Step 3. The operating current and voltage (and brightness!) is different for each of the three LEDs. Write these parameters for each colour.

Step 4. Calculate a suitable resistance value for each colour, assuming a perfect 3.3V source. (Note that the Teensy 3.1 gives access to both a 5V supply and a 3.3 volt supply).

Step 5. Determine a suitable combination of resistors to approximate the required resistance.

Step 6. Take care to note how to identify the common connection for the three LEDs.

Task 3: Verify Basic Operation

This will be the first circuit wired up for the course, so we'd like to start simple and verify the ability to successfully wire a basic circuit using a breadboard.

Step 1. If you haven't already done so, carefully insert your Teensy in one end of the breadboard. I would suggest putting it at the end with rows numbered ~55-60.

Step 2. Connect the 3.3V and GND connections to the rails of the breadboard, so that 3.3V/GND rails on both sides are connected.

Step 3. Using power directly from the 3.3V rail, wire the resistors and the LED to light each of

the colours. Get help from the lab Professor or a classmate if you have problems.

Step 4. One by one, transfer the wiring so that Teensy pins supply 3.3V for each of the LED colours (one Teensy pin per colour).

Step 5. Modify your basic LED blinking program from last lab to illuminate each of the colours.

Step 6. Write a basic program to create different colours by combining red, green, and blue. Cycle through the combinations, keeping each colour lit for about a second.

Task 4: Use Analog Outputs for Variable Brightness (Demo)

Many pins on the Teensy have multiple operating modes. Some pins are capable of generating "analog" outputs. These can be used in combination with delays to generate smoothly varying colours from the tri-colour LED.

Step 1. Consult the pinout card you received with the Teensy to locate the pins marked "PWM". These are the pins capable of generating an "analog" output.

Step 2. Rewire your circuit, if necessary, so that the three colours are driven by analog pins. (Did you hard-code your pins in the previous task? If you did, that was a bad plan!)

Step 3. Modify your program to use the "analogWrite() function with values from 0-255 to control each LED. Verify your understanding of this function by activating one LED at a time.

Step 4. Modify your program to create different colours by combining red, green, and blue. Use the analogWrite() function to create a smooth transition (over some significant fraction of a second) between each colour combination. What length of delay should you use between each change in the analogWrite() value? Should you be varying only one value at a time or multiple values at a time?

Step 5. Throw out your lava lamp and show off your creation to your friends.

Task 5: Reading from a Switch to Control Colour-cycling (Demo)

The goal is to start with automatic colour-cycling, and then change to a "manual" mode where there is a single change (jumping) from one colour to another on the first and each subsequent press of a button.

Step 1. Check the textbooks for examples of using a push-button switch
– Beginning Arduino, page 38, Project 4 – Interactive Traffic lights
– Getting Started with Arduino, page 42, using a pushbutton to control the LED

Step 2. Note the use in both examples of a "pull-down" resistor to make sure the input pin is **not** affected by stray voltage (eg. static electricity). It is also possible to arrange the resistor as a "pull-up".

Step 3. Modify your code to check the state of the push-button. The first time the button is pressed, freeze the current colour. On the second and subsequent presses, jump to a new (random) colour and freeze on the new colour.