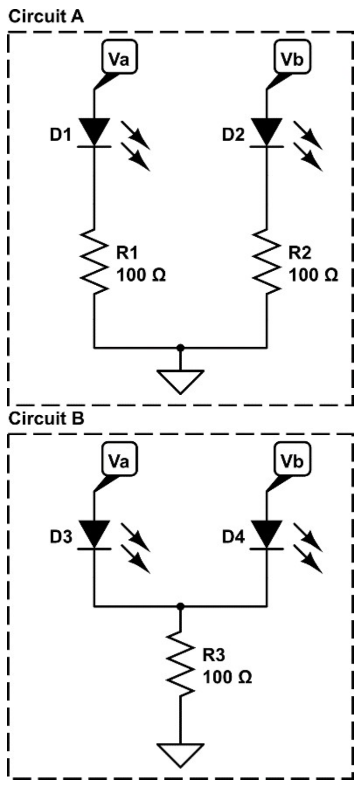
ECE 206: STEMTera and LEDs

Challenge: STEMTera Programming and LEDs

In this lab, we'll be writing some code for the STEMTera. The STEMTera is an Arduino compatible microcontroller which has a development platform based on the Processing language, which is similar to C/Java. We will introduce the concept of real-time programming.

Prelab Deliverables:

NOTE: Responses to Pre-lab questions are submitted in class



- Consider circuits A and B on the right. Fill in the following table for LED currents ID1 – ID4. Assume LEDs are modeled as diodes with Vf = 2V, assume LEDs are identical (e.g. current will split in half going through parallel LEDs).
  (Note: "V_f" refers to the forward voltage across a diode)

Va	Vb	ID1	ID2	ID3	ID4
0V	0V				
0V	5V				
5V	0V				
5V	5V				

- Which circuit yields more consistent LED current (therefore light output) in all situations?

Challenge:

Part 1:

This challenge has two parts, for the first, you will use the STEMTera to blink a LED (simply use 1 segment of the 7-segment display as the LED). You probably already did this to verify your install was working correctly, but now you will hook up your own LED. Start with the following code outline:

```
// one time run setup function
void setup() {
  // initialize digital pin 13 to drive LED
  pinMode(13, OUTPUT);
}

// infinite loop for real-time operation
void loop() {
  digitalWrite(13, HIGH);    // LED should go on
  for (int i = 0; i < insertyourdelay here; i++); //manual delay, calibrate to 500 ms
  digitalWrite(13, LOW);     // LED should go off
  for (int i = 0; i < insertyourdelay here; i++); //manual delay, calibrate to 500 ms
}
```

Connect a LED to pin 13 of the STEMTera through a resistor (100 Ω is sufficient; 1kΩ is also acceptable). Calibrate the delay such that the LED blinks for a period of 1 second (use a stopwatch to estimate this). This gives you some insight into how fast the STEMTera can execute your code.

Possible Problem: In the code above, the variable ***i*** is stored as an integer data type. This means that the largest value ***i*** can reach on your Arduino is $2^{16-1} - 1 = 32767$. Attempts to use delay values larger than that value will fail.

Solution: Delay values larger than 32767 are not necessary for this lab.

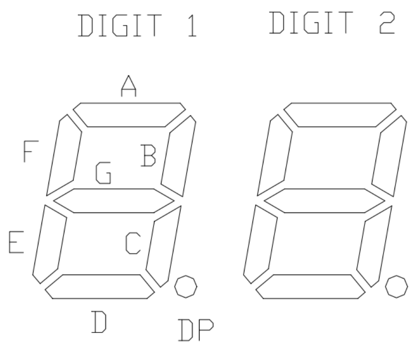
Possible Problem: Some groups will find that changing the delay value in their code does not make a difference. The *compiler* is the computer program that converts the C code you write into 0's and 1's that the Arduino understands. Sometimes, a compiler will apply optimizations to speed up code execution and because our code has long *for* loops that do nothing but count upward, the compiler may decide to completely ignore the *for* loops in your code.

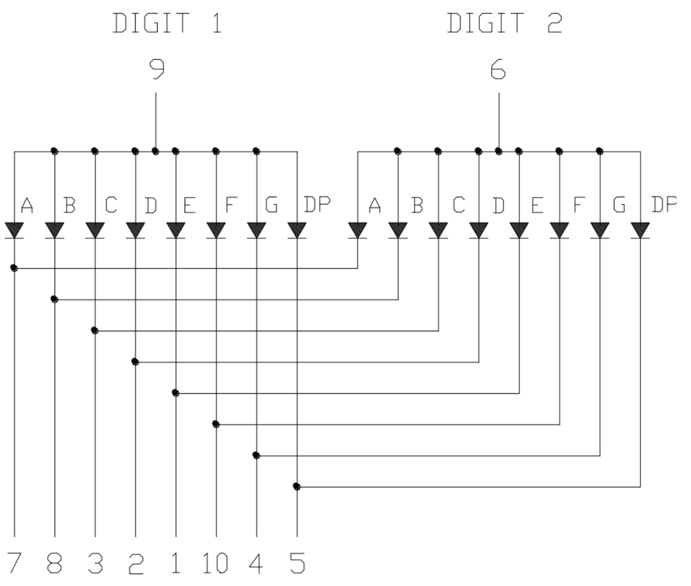
Solution: Every group should attempt Part 1 of the lab. If you believe that your compiler is optimizing your code to ignore the *for* loops, ask your TA to look at your code to confirm. Your TA will ask you to show them the oscilloscope traces for two substantially different delay times. If the traces do not change, and the TA determines your code is functioning properly, then the TA will tell you to proceed to part 2. When you write your report, document what happened, and include a measurement of what delay the Arduino gave you with the optimized code. Also document two delay values that you attempted to use, and the number of your lab bench. These special instructions only apply to groups that have the possible problem described above.

Important note: In practice, it would be inconvenient to manually calibrate a *for* loop every time a delay is needed. In this class, starting with Part 2 of this lab, you will instead use Arduino's built-in *delay()* function. Read the reference here before using it:
https://www.arduino.cc/reference/en/language/functions/time/delay/

Part 2:

The second part of the challenge is to hook up the 7 segment display and count from numbers 0-9 taking 10 seconds to do so (1 second per digit). The schematic for a 7-segment display is shown below. You will only need to use a single digit for this lab. You should use the equivalent of Circuit A, having a separate resistor for each segment (why?) along with 7 digital outputs on the STEMTera (remember to configure them as outputs in the setup function).





Hints: The anode (triangle side of the diode) terminal of the LED is shared. Use what you learned in the prelab to ensure consistent current when lighting up LED segments. Also, the pins are labeled from pin 1 (bottom left) to pin 5 (bottom right) and then pin 6 (top right) to pin 10 (top left).

You will only need to light up one digit (e.g. set terminals 6 or 9 to 5V) for this lab. Is it possible for the two digits to display different numbers? How might you get around this limitation (to display a 2-digit number) in a future lab?

Report Deliverables:

- Include a diagram/sketch of you circuit (Fritzing) for part 2.
- Include commented code for part 2, as well as the delay values you determined
- Obtain your TA signoff on the circuit in operation

Report Element	**Value**
Documentation of lab completion	40
Circuit diagrams for Part 1 and Part 2	15
Commented code	15
Part 1 delay values OR discussion on inability to find delay values	15
Report quality and all other elements	15
Total	**100**