California State University, Channel Islands (CSUCI)
Department of Computer Science

COMP-462: Embedded Systems
Lab Report
Fall 2019

Lab Number: Lab 3

Lab Topic:

Breathing LED Interfaces with Switches and PWM

Student Name: Marc Julian Jamerlan
Student ID: 001956436
Student Major: B.S. Computer Science

I. Objectives

This lab aims to give students practice with programming and building LED interfaces, as well as introducing students to the concept of using pulse-width modulation (PWM) and logic switches in order to change the properties of an LED circuit. The goal of this lab is to be able to create an LED interface that can increase its pulse using a button input, and to create a "breathing" LED interface using another button input and an automatic PWM algorithm.

II. Introduction

Pulse-width modulation is the act of varying the amount of time a signal is high or low when driven through a circuit. By changing the ratio between when the signal is high (or "on") versus when the signal is low ("off") for a certain time period (which is repeated for the entire operation of the circuit), we can effectively modify the amount of power driven through certain components of the circuit. For example, by varying this ratio, known as a "duty cycle", we can change the amount of power passing through an LED, and thus either increase or decrease its brightness.

The task for this lab was to use PWM in order to create an LED interface that blinked at 2 Hz and can be toggled to vary its duty cycle. The system starts with the LED blinking at 2 Hz, roughly 2 times per second, and with a 30% duty cycle, meaning that the LED would be ON for 150 ms and OFF for 350 ms. A button powered by the microcontroller is read by PE2 as input, and when toggled would stop the oscillations and increase the duty cycle by 20%. Once the button is released, the system would continue blinking at the modified duty cycle. If the duty cycle would increase beyond 100%, the system would loop back to a 10% duty cycle and increase from there.

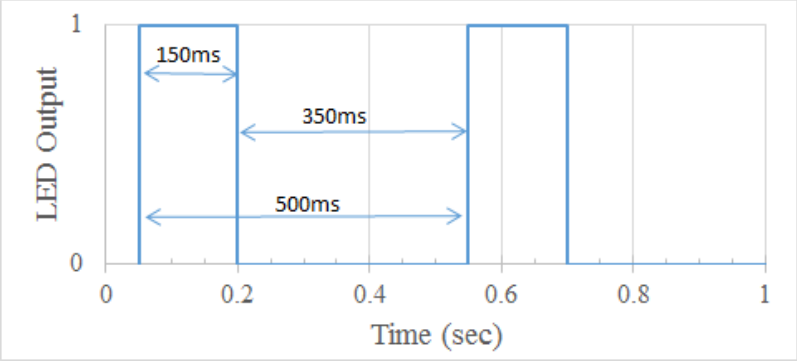


Figure 1-1: PWM signal of an LED blinking at 2 Hz w/ 30% duty cycle

A second task for this lab was to create an algorithm that would make the LED "breathe", which can be best described as changing the LED's brightness with automatic PWM over a period of time. First, the LED slowly turns brighter, "breathing up", and then when it reaches its maximum brightness would slowly turn darker, "breathing down". The system was to go into breathing mode when PF4, a button on the microcontroller itself, is held down, and when released the system would go back to blinking at 2 Hz with a 30% duty cycle.

III. Procedure

1. Implemented an algorithm that blinked an LED at 2 Hz and with a duty cycle of 30%.

   a) Pseudocode:

```
Start
 Turn on Port E clock
 Make PE2 and PE3 output
 Enable interrupts for measuring devices
 Initialize values for frequency, delays and duty cycle
Loop1
 Check if PE2 button is pressed
 If so, B ButtonPressed
 If not, B Loop2
Loop2
 Set PE3 high
 Delay for 150ms (or depending on current duty cycle)
 Clear PE3 low
 Delay for 350ms (or depending on current duty cycle)
 B Loop1
ButtonPressed
 Do nothing until PE2 button is released
 If so, B ModDutyCycle
ModDutyCycle
 DutyCycle = DutyCycle + 20% (modulo 100%)
 Delay1 = Frequency * DutyCycle
 Delay2 = Frequency - Delay1
 B Loop1
```

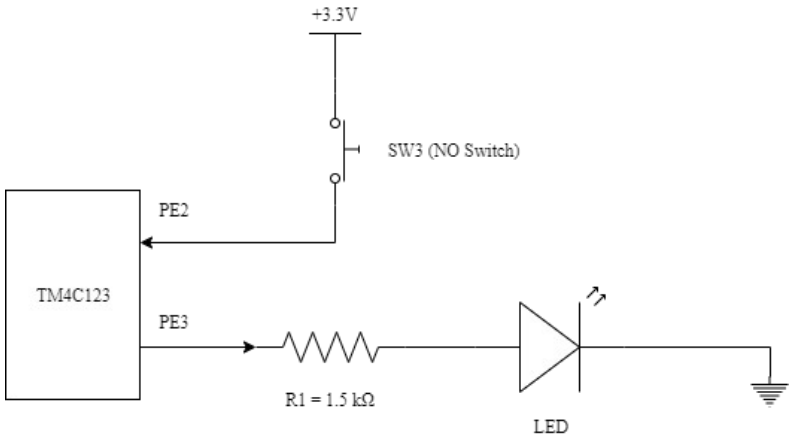


Figure 3-1: Circuit Diagram for a simple LED driver circuit

2. After testing Step 1 using the simulator, built the circuit described in Figure 3-1 to test the code on hardware.

3. Implemented the algorithm for the “breathing” LED.

    a) PF4 was used as a negative logic switch that when held, starts the breathing algorithm. When released, the operation of the system should continue as normal.

    b) Essentially, the breathing algorithm is very similar to the algorithm for changing the duty cycle, except this time it was done automatically rather than requiring the press-and-release of a button for every duty cycle change. Duty cycle was increased by increments of 10%, and the frequency of the system was increased to 100 Hz so as to make the blinking of the LED imperceptible.

    c) Rather than jump back to minimum duty cycle upon reaching the maximum, a counter was implemented so that when the maximum duty cycle was reached, the system decrements the duty cycle by 10% increments until it reaches the minimum.

    d) See the `main.s` included with the lab report for details on the implementation.

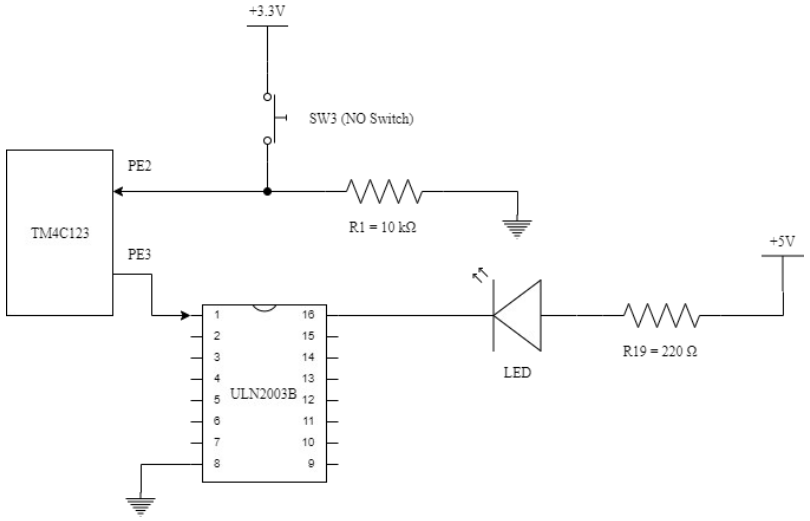


Figure 3-2: Circuit Diagram for the breathing LED

4. After testing Step 1 using the simulator, built the circuit described in Figure 3-1 to test the code on hardware.

    a) A ULN2003B transistor IC was used to amplify the current passing into the LED so changes to brightness are more noticeable.

    b) A 5V power supply was used to drive voltage through the anode side of the LED.

5. Took measurements of voltage, resistance, and current of both the switch circuit and the LED circuit described in Figure 3-2. All measurements are described in Figures 5-2 and 5-3.

IV. Problems

Because assembly language does not support timed delays, delays had to be implemented in an alternative manner. Delays were instead programmed by taking advantage of how the microcontroller takes a set amount of time to complete instructions. A single instruction takes around 4 cycles to complete, and a cycle takes 12.5 nanoseconds. The following calculation was used to find the number of instructions needed to simulate a certain amount of time:

4 * number of instructions * 12.5 ns = time needed

For example, to find the number of instructions needed to delay for 150 ms:

number of instructions = 150,000,000 ns / (4 * 12.5 ns) = 3,000,000 instructions needed

Other than the usual design questions during the implementation of the breathing LED algorithm, another problem encountered was attempting to load very large hexadecimal numbers using `mov` instructions. The solution was instead to load these numbers using `ldr`, initializing the number as an address rather than a constant (e.g. `ldr r0, =2DC6C0`). Once these numbers were loaded into registers as such, they can be moved to other registers normally using `mov`.

V. Results

The result was an LED interface that blinked at 2 Hz and can have its duty cycle increased using a button input. Furthermore, a second button input can toggle "breathing" mode on the LED, and allowed it continue breathing until the button was released.

Parameter	**Value**	**Units**	**Conditions**
Resistance of the 10kΩ resistor, R1	9.89	Kilo-ohms	With power off and disconnected from the circuit (measured w/ ohmmeter)
Supply Voltage, $V_{+3.3}$	3.284	Volts	Powered (measured w/ voltmeter)
Input Voltage, V_{PE2}	0	Volts	Powered, but switch not pressed (measured w/ voltmeter)
Resistor current	0	mA	Powered, but switch not pressed, $I = V_{PE2}/R1$ (calculated and measured with ammeter)
Input Voltage, V_{PE2}	3.284	Volts	Powered and switch pressed (measured w/ voltmeter)
Resistor current	0.332	mA	Powered, and switch pressed, $I = V_{PE2}/R1$ (calculated and measured with ammeter)

Figure 5-2: Switch measurements

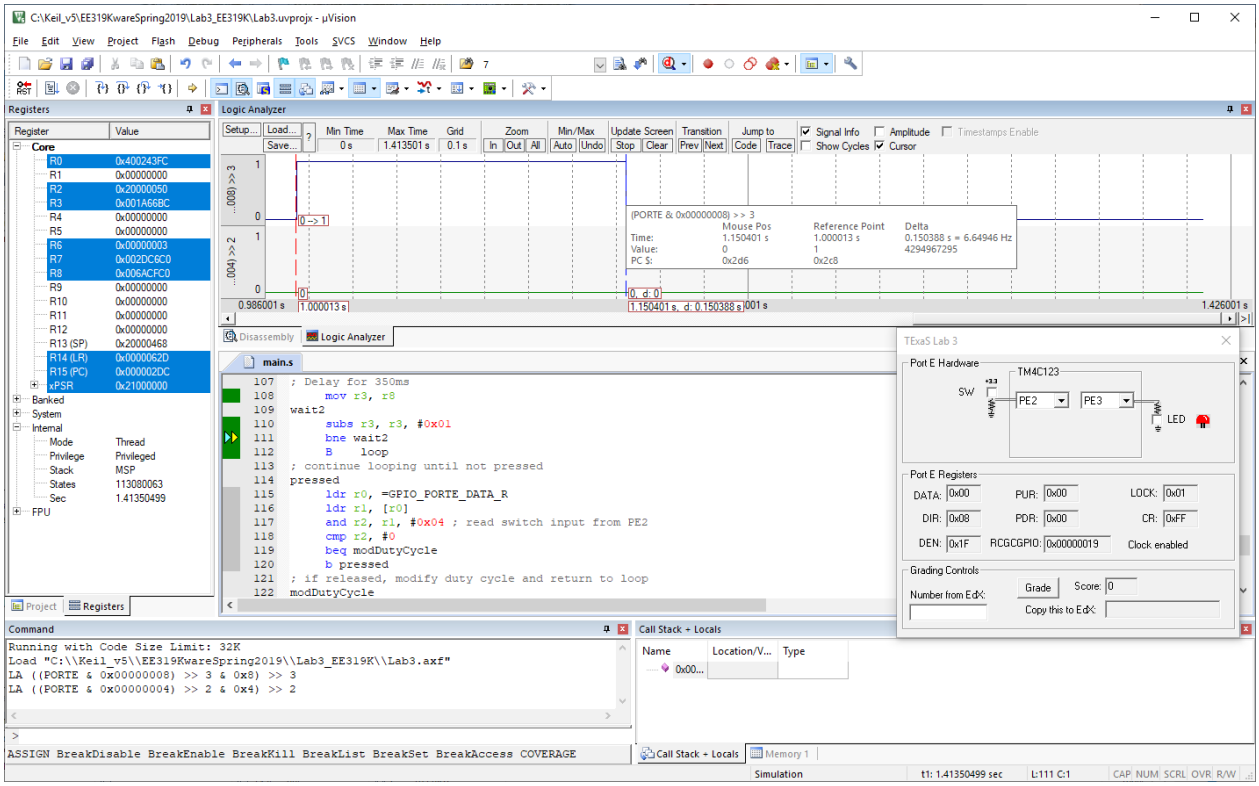
Parameter	**Value**	**Units**	**Conditions**
Resistance of 220Ω resistor, R19	216	Ohms	With power off and disconnected from the circuit (measured w/ ohmmeter)
+5 V power supply, V_{+5}	4.98	Volts	(measured w/ voltmeter relative to ground)
TM4C123 Output, V_{PE3}, input to ULN2003B	0.0094	Volts	With PE3 = 0 (measured w/ voltmeter relative to ground). This is the V_{OL} of the TM4C123.
ULN2003B Output, pin 16, V_{k-}, LED k-	0.001	Volts	With PE3 = 0 (measured w/ voltmeter relative to ground). Taken from floating measurement.
LED a+, V_{a+}, bottom side of R19 (anode side of LED)	4.98	Volts	With PE3 = 0 (measured w/ voltmeter relative to ground). Taken from floating measurement.
LED Voltage	4.98	Volts	Calculated as: $(V_{a+} - V_{k-})$
LED current (off)	22.6	mA	Calculated as: $(V_{+5} - V_{a+}) / R19$ (measured w/ ammeter)
TM4C123 Output, V_{PE3}, input to ULN2003B	3.225	Volts	With PE3 = 1 (measured w/ voltmeter relative to ground). This is the V_{OH} of the TM4C123.
ULN2003B Output, pin 16, V_{k-}, LED k-	0.721	Volts	With PE3 = 1 (measured w/ voltmeter relative to ground). This is the V_{OL} or $V_{CE(sat)}$ of the ULN2003B.

LED a+, V_{a+}, bottom side of R19 (anode side of LED)	2.538	Volts	With PE3 = 1 (measured w/ voltmeter relative to ground).
LED Voltage	1.817	Volts	Calculated as: $(V_{a+} - V_{k-})$
LED current (on)	8.259	mA	Calculated as: $(V_{+5} - V_{a+}) / R19$ (measured w/ ammeter)

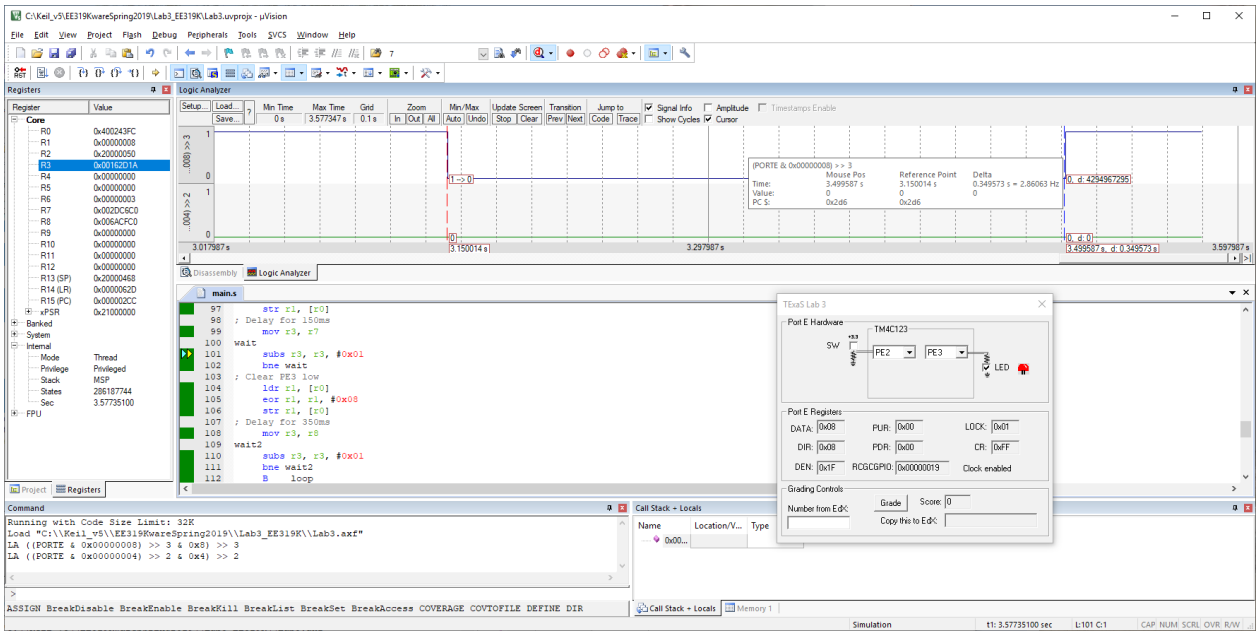
Figure 5-3: LED measurements w/ 220Ω resistor

Measurements were taken using a combination of a voltmeter and the Stellaris ICDI debug environment in Keil, which was used to take measurements for fluctuating values during the real operation of the system.

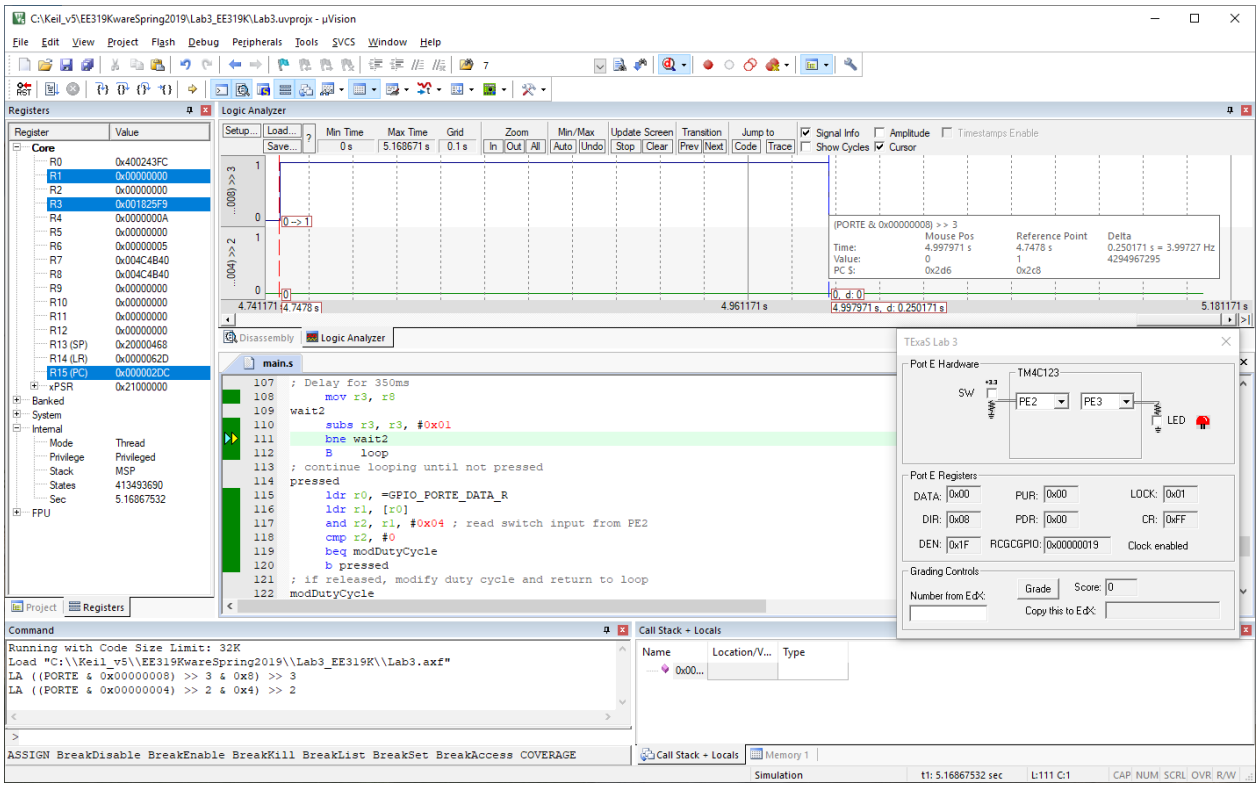
The following are screenshots of the results of running the duty cycle modification algorithm through the Logic Analyzer:



Screenshot 1-1: ON range of the blinking LED, showing duty cycle at 30% (150 ms)

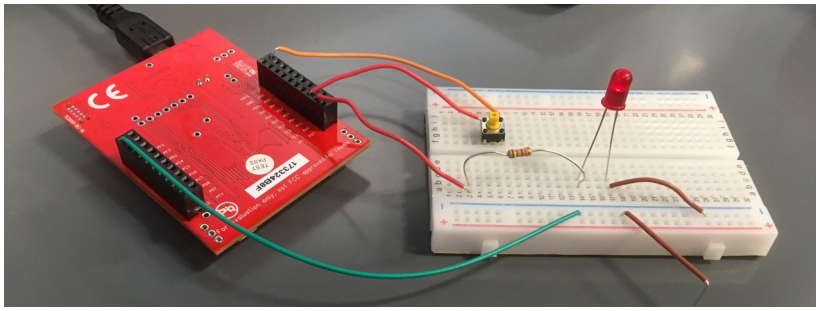


Screenshot 1-2: OFF range of the blinking LED, showing the remaining 350 ms of the 500 ms period

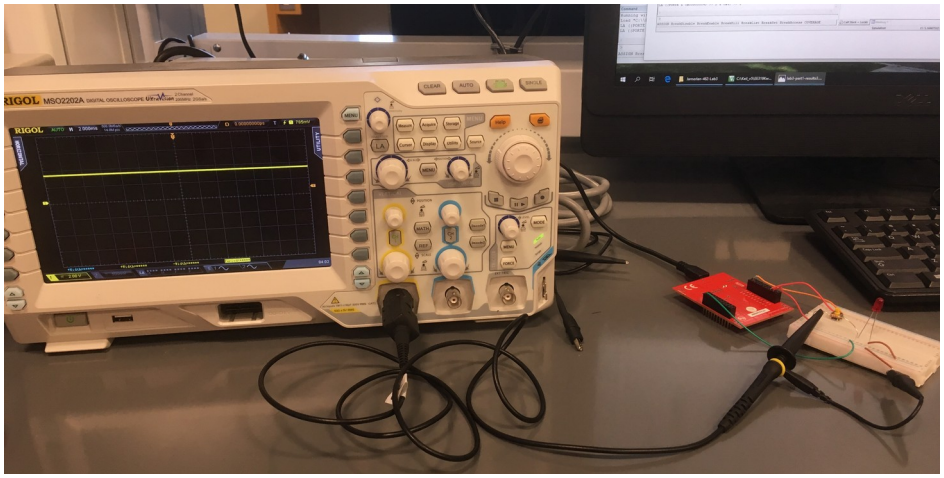


Screenshot 1-3: ON range after PE2 button input was pressed-and-released once, showing a 50% duty cycle (250 ms)

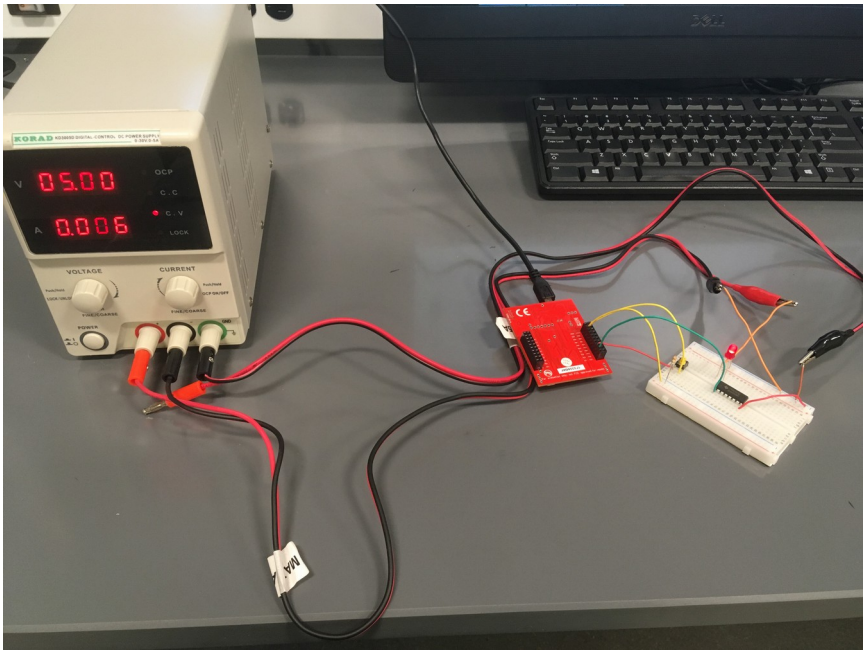
The following are screenshots of the actual operation of the LED interface:



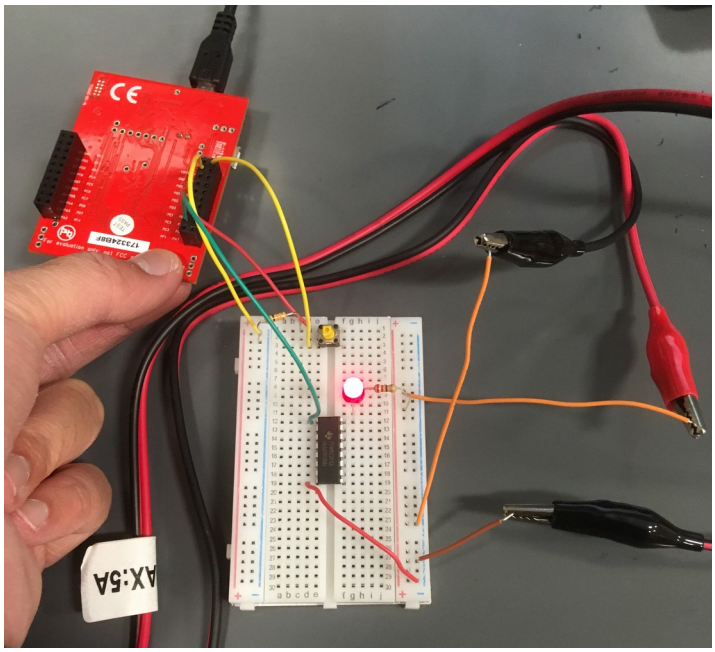
Screenshot 2-1: Physical circuit for Figure 3-1



Screenshot 2-2: Circuit in Screenshot 2-1, measuring signal with an oscilloscope



Screenshot 2-3: Physical circuit for Figure 3-2



Screenshot 2-4: Operation of the breathing LED circuit, with PF4 being held

The source code for both the duty cycle modification and the breathing LED algorithm is included alongside the lab report in `main.s`.

VI. Q&A

1.) Why was the ULN2003B used to interface with the LED? (i.e. Why did we not connect the LED directly to the TM4C123?)

The ULN2003B is used as a safe way to amplify the current being driven through the circuit, thus allowing us to give more power to the LED. Without the ULN2003B, the LED would be much dimmer.

2.) What would the flashing LED “look” like if the delay were 1ms?

The LED will look as if it’s either blinking incredibly fast or even imperceptibly fast at higher frequencies.

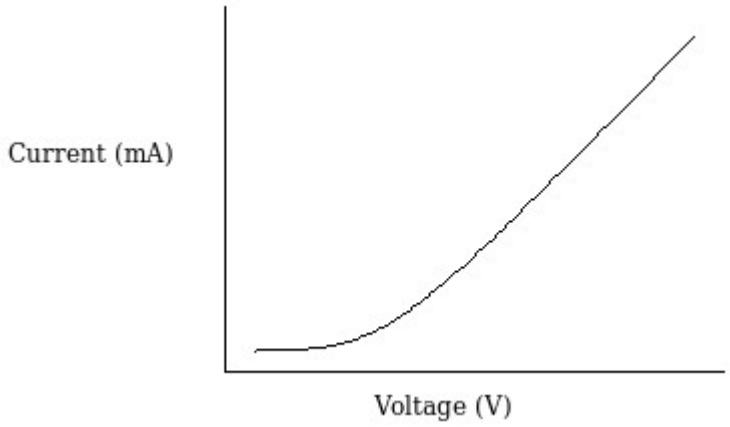
3.) How would you modify the software to change the rate at which LED flickers?

You can change the rate of the LED flicker by increasing or decreasing the frequency of the system. A higher frequency means that the LED will flicker more rapidly, and vice versa.

4.) What operating point (voltage, current) exists when the LED is on?

The LED’s operating point is at 1.8V and 8 mA.

5.) Sketch the approximate current versus voltage curve of the LED. Explain how you use the resistor value to select the operating point.



Resistor values can be used to limit the amount of voltage that can be driven into an LED, allowing us to stay within the LED's operating point.

6.) What is the difference between a positive logic and negative logic interface for the switch or the LED?

A positive logic interface means that 1 is an active "high" signal and 0 is an inactive "low" signal. Conversely, a negative logic interface uses 1 as a "low" signal and 0 as a "high" signal.