BBM 434 – Embedded Systems Lab
Lab 3 Instructions

Based on Lab 8 of the EDX course UT.6.01x Embedded Systems - Shape the World.

Introduction

If you've been to any major airport at night, you may have noticed that there are a lot of different kinds of runway lights. Airport lighting is important for aircrafts operating at night. One of its major applications is to give pilots a visual guide during their descent to maintain a stabilized approach, which is known as Approach Lighting System. Airplanes can take off and land in either direction on a runway. The direction is determined based on the wind direction.

You may have also noticed a different hue and intensity of taxi and runway lights at some airports, and the reason for this are new LED lights. LED light prices have dropped dramatically in the past few years, so airports all over the world are changing to LEDs to lower electric bills and reduce maintenance costs. They're bright, and they look great!

You have been recruited to design a new runway lighting system for the new airport in Istanbul. LED lights will be placed along the runway to help pilots land in the proper direction.

Preparation

You will need a LaunchPad and access to TM4C123_LaunchPadUsersManual.pdf. In this lab, you will also need a switch, one 10kΩ resistor, three LEDs, and three 470Ω resistors.

Starter project

You can use the last week's lab file as the starter project.

Purpose

Lab 3 is our first lab requiring you to build circuits on the breadboard and connect them to the LaunchPad. The purpose of this lab is to learn how to interface a switch and an LED. You will perform explicit measurements on the circuits in order to verify they are operational and to improve your understanding of how they work.

System Requirements

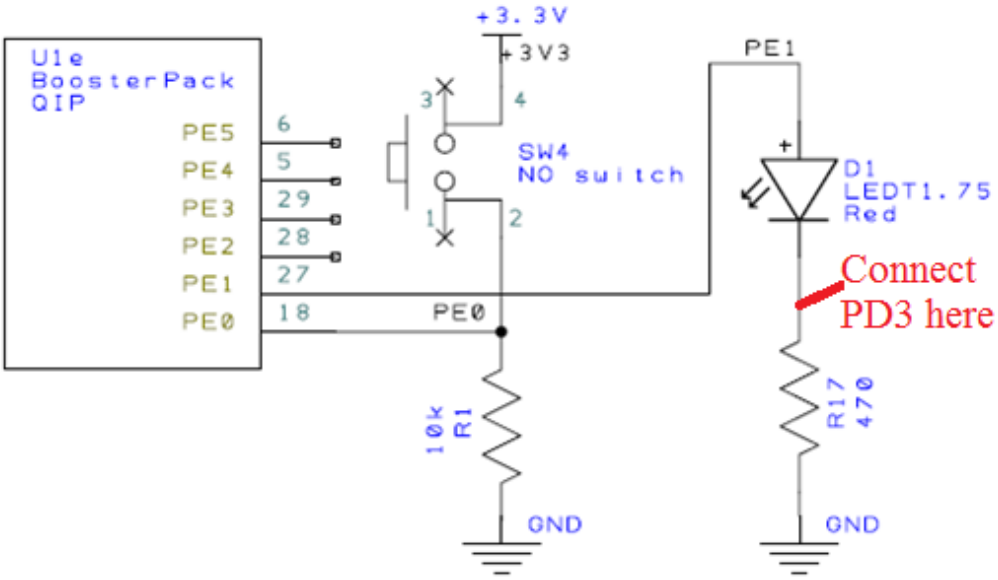
In this lab, you will implement an external circuitry, **but in this lab you will use SysTick timer to measure the time instead of loops**. On a breadboard, you will attach a switch and three LEDs (green, yellow, and red). Green will represent the start, yellow the middle, and red the end of the runway.

1) Decide on the ports that you will connect your switches and LEDs. **You will not use PortF**.
2) Configure the port and pins to which you will connect the switch as input, and LEDs as output pins.
3) When the system starts, LEDs should flash from green, through yellow, to red in a continuous loop, as to signal which way an aircraft should land on the runway. There should be no pauses between the turning off of a previous LED and turning on of the next one (e.g. green LED will turn on and as it turns off, yellow LED should turn on at the same time, etc.). You can choose the length of ON periods as you wish, but make sure you do not exceed 1 second (or make it too short to be useful), and that all LEDs stay ON for the same amount of time. You must use SysTick timer in this experiment.
4) When the wind direction changes, an operator will press the switch SW2. Each time the switch SW2 is pressed, the color order should reverse, i.e. from red, through yellow, to green and so on, signaling pilots that they should land in the opposite direction.

Note: As it was the case in Lab 2, if the switch is pressed in the middle of one period, your system should wait till the end of that period, and then it should switch the direction of the lighting system.

Implementation

Part a. Decide on which ports to use as input or output. First, draw a circuit diagram similar to the figure below. Note that there will be three LEDs and a single switch in your experiment. You can choose the switch and LEDs to be positive or negative logic as you prefer (refer to Week 3 lecture notes for details), **but you must implement both in your circuit** (e.g. if switch is positive, LEDs should be negative logic). State clearly in your report which parts of your circuit are positive and which are negative logic. Your external circuit should be compatible with your circuit diagram.



An example circuit diagram that you need to draw for your system (you need to include all parts).

Notice in the figure above that the PE0 and PE1 wires cross over each other in the circuit diagram, but they are NOT electrically connected, because there is NO dot at the point of crossing. Conversely, the PE0 signal to the microcontroller, pin2 of the switch and one end of the 10kΩ R1 resistor ARE electrically connected, because there IS a dot at the point of crossing. Be careful about these details.

Part b. Write the software that satisfies the requirements for this lab. You already have a code from Lab 2 to get you started, but you need to make some changes regarding the input/output ports. **You will not use Port F this time**. Since you need to use a different port, you need to replace the constant declarations to access port registers using symbolic names. You can include the following file for the register definitions:

http://users.ece.utexas.edu/~valvano/Volume1/tm4c123gh6pm.h

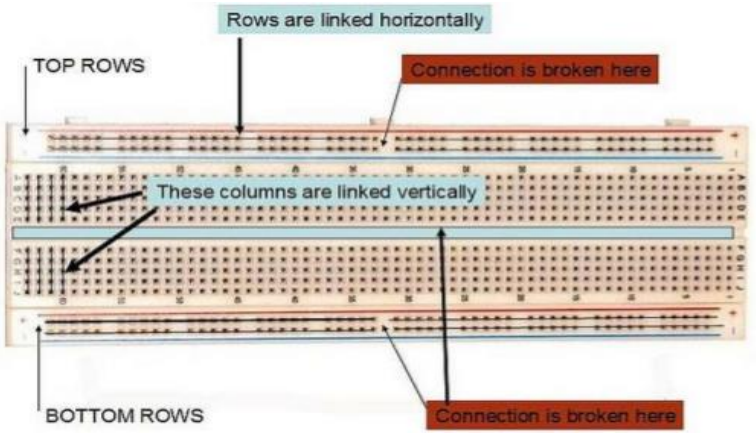
In this lab, you have to change your code so that you use SysTick timer to measure the time. You have to configure the timer as we have discussed in class. You can refer to 10.1 and 10.2 from the following link:
http://users.ece.utexas.edu/%7Evalvano/Volume1/E-Book/C10_FiniteStateMachines.htm

You will still check if the timer has expired inside a loop using the following code:

```
while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait for count flag
}
```

Note that this is still not a good practice. Interrupts are a better method.

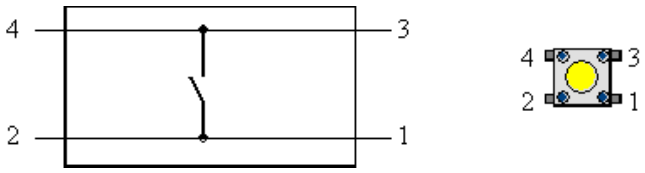
Part c. After the software has been debugged on the simulator, you will build the hardware on the real board. To build circuits, we'll use a solderless breadboard, also referred to as a protoboard. The holes in the protoboard are internally connected in a systematic manner. The long rows of holes along the outer sides of the protoboard are electrically connected. Some protoboards have four long rows (two on each side), while others have just two long rows (one on each side). We refer to the long rows as power buses. If your protoboard has only two long rows (one on each side, we will connect one row to +3.3V and another row to ground. If your protoboard has two long rows on each side, then two rows will be ground, and one row will be +3.3V. You can label the voltage on each row.



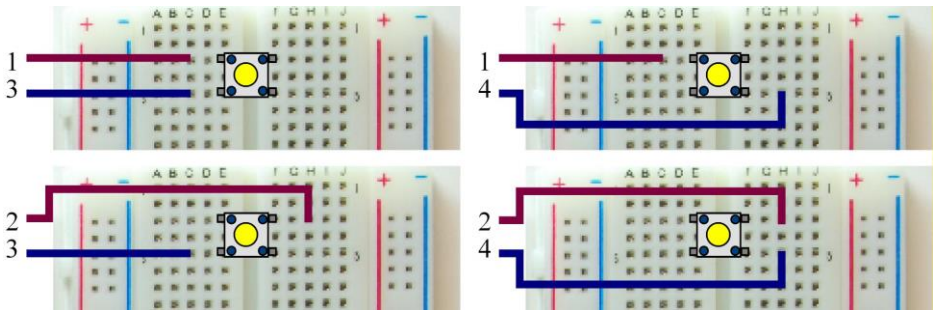
In the middle of the protoboard, you'll find two groups of holes placed in a 0.1 inch grid. Each adjacent row of five holes is electrically connected. We usually insert components into these holes. IC chips are placed on the protoboard, such that the two rows of pins straddle the center valley. To make connections to the TM4C123 we can run male-male solid wire from the bottom of the microcontroller board to the protoboard.

You should watch the green power LED when you first power up a new circuit. If the green power LED on the LaunchPad does not illuminate, you should quickly disconnect the USB cable.

Notice the switch has 4 pins in a rectangular shape, as shown in the figure below. Each button is a single-pole single-throw normally-open switch. When you press the switch pins 1-4 and 2-3 are connected.



There are four possible connections that will work. In the figure below you see the switch circuit can be built across pins {1,3} {1,4} {2,3} or {2,4}. Another scheme for figuring it out is to use an ohmmeter to test across which two pins the switch exists.



Do not place or remove wires on the protoboard while the power is on.

The next step is to build the LED output circuit. Using the data sheet, hold an LED and identify which pin is the anode and which is the cathode. LEDs emit light when an electric current passes through them, as shown in Figure 3.5. LEDs have polarity, meaning current must pass from anode to cathode to activate. The anode is labelled **a** or **+**, and cathode is labelled **k** or **-**. The cathode is the short lead and there may be a slight flat spot on the body of round LEDs. Thus, the anode is the longer lead. Furthermore, LEDs will not be damaged if you plug it in backwards. LEDs however won't work plugged in backwards of course.

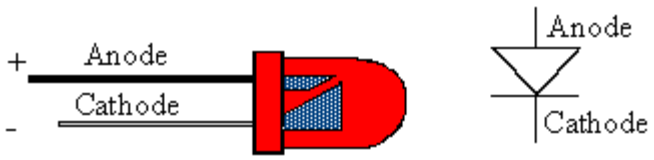


Figure 3.5. A drawing and the circuit symbol for an LED. "Big voltage is on the big wire."

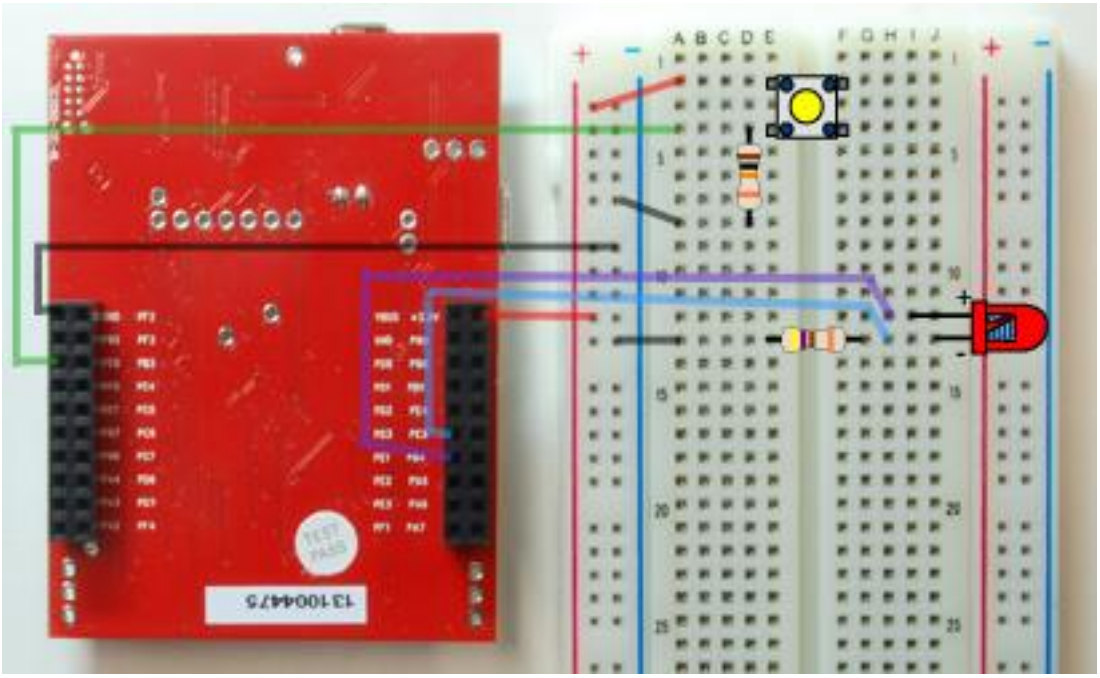


Figure 3.6. A photo showing an example layout of the circuit. The black wires are ground. The red wires are +3.3V. The purple wire is PE1. The green wire is PE0. The blue wire connects PD3 to the signal between the LED and the resistor; it will be used to measure LED current in Part d). Brown-black-orange resistor is 10kΩ. Yellow-purple-brown resistor is 470Ω. Again, it doesn't matter what color the wires are, the colors are used to identify which wires are which signals. **For the switches you have, probably pins 1-4 and 2-3 are connected when the switch is pressed. So your circuit will be different from this one.**

Part d) Debug your combined hardware/software system on the actual TM4C123 board. First, we will use the debugger to observe the input pin to verify the proper operation of the switch interface. You will have to single step through your code that initializes your selected port. You then execute the **Peripherals->SystemViewer->GPIO->GPIOE** command.

As you single step you should see the actual input as controlled by the switch you have interfaced.

We can verify all aspects of the LED interface are proper by beginning in the **Peripherals->SystemViewer->GPIO->PORTE** debugger window (depending on your choice of ports). In the debugging window you should see the pins of the port you chose. You can calculate the current across the LED using Ohm's Law. The LED current will equal the current in the resistor, which is

$I_d = V_{PD3}/470\Omega$

Assuming the port output voltage, V_{OH} is 3.0V, the voltage across the LED is about

$V_d = (3.0\text{-}V_{PD3})$

The LED voltage is only an approximation because we are guessing what the voltage on the pin will be.

You should include all calculations in your report along with all the steps of your experiment.

Bonus Part (Optional)

If you want to get a chance to win a **surprise award**, you should extend your project to make a more realistic runway prototype with at least 5 LEDs on both sides of the runway.

How to Submit a Written Report

The deadline for the submission is **Friday 29 March 2019 until 13:45** (no further extensions will be allowed!). Submit your work through http://submit.cs.hacettepe.edu.tr/ (submission via e-mail will not be accepted!) in the following format (**one submission per group**):

- **b<studentID>.zip**
    - **report.pdf**