USC Viterbi School of Engineering

Ming Hsieh Department of Electrical Engineering

EE 109L - Introduction to Embedded Systems

Spring 2015 Lab Assignment #3
Buttons and LEDs

1 Introduction

In this lab exercise you will build a signaling device that makes an LED flash the Morse code signal for a letter of the alphabet. Rather than have 26 input buttons it will only have three, each assigned to one of three letters of the alphabet. When one of the buttons is pressed, an LED will flash the Morse code signal for the corresponding letter.

2 Morse Code

Morse code is a method of signaling letters and numbers by sending on and off patterns of light or audible tones. It was first developed in the 1830's and for many years was the standard way of communicating via telegraph and radio. It is still used today by amateur radio operators, the military, and sometimes for emergency communication. Characters in Morse code consist of a unique set of short or long signals, known as "dots" and "dashes". The codes for letters A through Z and 0 through 9 are shown in Fig. 3. The dots and dashes of a character are separated by a short gap, and a longer gap separates characters. The most widely known Morse code sequence is the distress call "SOS" of three dots, three dashes, three dots.

Figure 1: Telegraph key for sending Morse code by electricity

Figure 2: Navy signalman sending Morse code by light

The length in time of a dot or dash in Morse code is not specified directly, only the relationship between the dot and dash length:

- **Dashes are three times longer than a dot.**
- **Gaps between the dots and dashes of a single character are the same length as the dot.**
- **The gap between characters is the length of the dash.**

The speed of the transmission of the characters can cover a wide range, but the above ratios should be adhered to in order to make the transmission understandable by the receiving party.

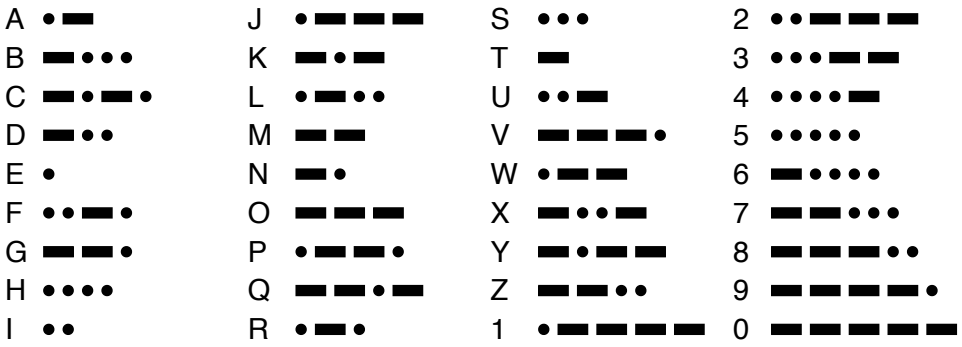

Figure 3: Morse code patterns for A-Z and 0-9

3 Buttons

The buttons you will be using are classified as momentary action, single pole, normally open buttons. These terms define how the button works.

Momentary action - The button is in the ON state only when it is held down. Once you release pressure on the button, it returns to the OFF state on its own.

Single pole - The button has one on-off circuit inside it. Buttons can be found with 2, 3, or more circuits that all turn on and off together when the button is operated.

Normally open - When the button is NOT being pressed the switch contacts are OPEN so the two terminals of the switch are not connected. Pressing the switch closes the contacts. Switches are also available as "normally closed" models meaning that the contacts are connected (closed) until the switch is pressed.

The buttons for this lab (Fig. 4) have four pins on the bottom that are spaced so they will fit in the holes in the breadboard. One side of the button has a flat part that is used to determine how the four pins should be connected. On each side of the flat spot, the two pins are connected together. When the switch operates, it opens and closes the connection between the two pairs of pins as shown in Fig. 5.

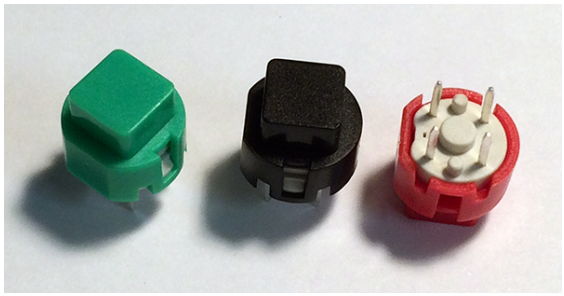

Figure 4: Push buttons

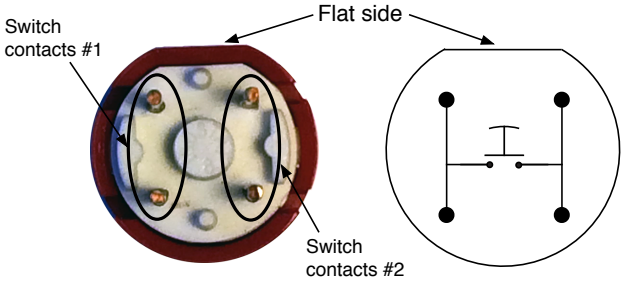

Figure 5: Bottom view, showing how pins are connect to switch

When installing the button on the breadboard, it should be oriented so switch contacts #1 are in one of the 5-hole rows on the breadboard, and switch contacts #2 are in another 5-hole row as shown in Fig. 6. The two pairs of contacts must be in **different** rows of five hole breadboard holes. When done that way, the switch opens and closes the connection between the two rows of the breadboard holes.

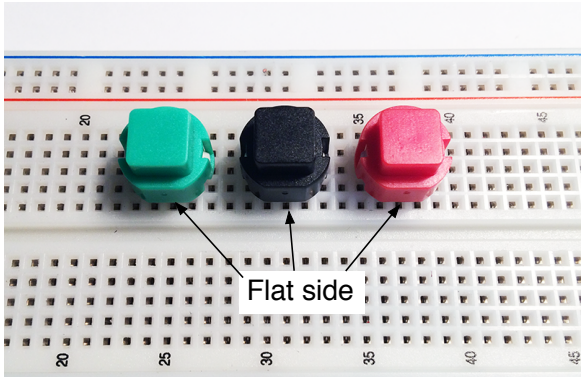

Figure 6: Orient the buttons so the flat spot goes across multiple rows of holes

4 LEDs

We used LEDs in Lab 1 on electronic circuits. In this lab you will be connecting one to a port bit of the Arduino configured to be an output and your program will determine whether it is on or off. As in Lab 1, the LED will need a resistor in series with it to limit the amount of current flowing through the LED. Recall that in Lab 1, the current through the LED and resistor was given by $I = (V_s - V_{LED})/R$ or $R = (V_s - V_{LED})/I$ where V_s is the source voltage, V_{LED} is the voltage drop across the LED and R is the resistance. When an output bit is in the high state the voltage on the pin is close to $5V$ so we can assume $V_s = 5V$. If $V_{LED} = 1.7V$ and we want about $15ma$ through the LED, this works out $R = (5 - 1.7)/0.015 = 220\Omega$.

The LED must be oriented so the anode (positive end) is towards the Arduino and the cathode (negative end) is towards ground. The resistor can be in series with the LED either between the Arduino and the LED, or between the LED and ground.

5 The Circuit

Use your Arduino Uno and breadboard to build the circuit shown in Fig. 7. The three switch inputs should be wired to the Arduino inputs D2, D3 and D4, which are connected to Port D, bits 2, 3 and 4 of the microcontroller. The LED and current limiting resistor should be connected to pin D8 of the Arduino, which is Port B, bit 0 of the microcontroller.

The ground wire from the Arduino to the breadboard should go to one of the columns of holes that forms the bus that runs the length of the breadboard. Use one of the buses by the blue line for ground. The ground connections to the switch and the resistor should all be made to that bus column.

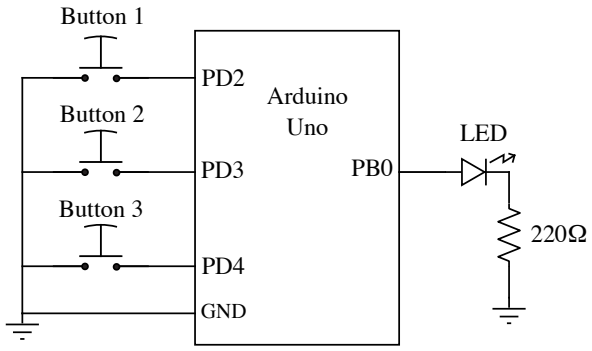

Figure 7: Schematic for three-button Morse code typewriter

6 The Program

In your `ee109` folder create a `lab3` subfolder. Copy the `ee109.c` file into the `lab3` folder and call it `lab3.c`. Also copy `Makefile` into the `lab3` folder. At this point your `lab3` folder should contain two files: `Makefile` and `lab3.c`. Do a "`cd lab3`" to get into the Lab 3 folder and then edit the Makefile to change the name of the project object file on the `OBJECTS` line to be "`lab3.o`". You're now ready to starting adding code to `lab3.c`.

To create the dots and dashes of certain lengths, the program will have to execute a number of delays during execution. To implement delays in the program put this line at the top of the program.

```
#include <util/delay.h>
```

When you need to have a delay in the program, use the `_delay_ms` function. For example the following line would cause a $300msec$ delay.

```
_delay_ms(300);
```

The argument to the `_delay_ms` function must be an integer value. In addition, it should be a constant value, not a variable. The avr-gcc compiler allows variables on some types of systems but on other they can not be used.

In your program you should choose what delay amount to use for the length of the Morse code dot. All the other timing amount (dashes, gaps, etc.) are based on the length of the dot. Select a dot length that makes the output code readable, not too fast and not too slow. If your first choice for a delay doesn't seem right, experiment a bit and come up with one you prefer.

Your program should do the following operations.

- Configure the appropriate Port B line as an output.
- Configure the appropriate Port D input lines to have their pull-up resistors enabled.
- Start a loop that does the following:
  - Examine the state of button 1 to see if it has been pressed. If so send the pattern of dots and dashes for the character 'U' by making Port B, bit 0 go high and low at the correct times.
  - Do the same thing for button 2 to send an 'S'.
  - Do the same thing for button 3 to send a 'C'.

When generating the patterns of dots and dashes make sure to use the proper lengths as defined above.

7 Testing

In the process of getting your program working it's recommended that you first confirm that you can simply control the LED from the switches. Before adding the code to generate the Morse code dots and dashes, first just have the program turn the LED on if button 1 is pressed and turn it off if the button 1 is not pressed. If this works, change the code to have it work with button 2, and then try it with button 3.

Your goal is to eliminate possible sources of problems by testing parts of the whole thing before they all have to work together. If you just build the final circuit and try it with the full program, what do you do if nothing works? Is it a problem with the buttons, with the LEDs, with the program, or what?

8 Results

Once you have the assignment working demonstrate it to one of the instructors. Turn in a copy of your source code (see website for details.)