ESE 519 Real-Time and Embedded Systems
Lab 1: Morse Code Encoder/Decoder

-- --- .-. ... .   -.-. --- -.. .   -.. . -.-. --- -.. . .-.

Background:
http://en.wikipedia.org/wiki/Morse_code
https://developer.mbed.org/handbook/Creating-a-program
http://developer.mbed.org/handbook/Debugging

Your Mission: Danny Ocean has hired you to be a part of his trusted 11 (Later Matt Damon). His next heist has Basher using "The Pinch" to wipe out the entire casino's communication with an electromagnetic pulse. While this allows Ocean's 11 to slip past security without being detected, they've also managed to wipe out their own methods of communication with home base. Luckily, Rusty has an antiquated morse code transmitter, but you can't make sense of the signals he's sending!

Your mission, should you choose to accept it, is to build a Morse Code decoder/encoder using polling and interrupts on an input switch. Output the results on LEDs connected to output pins, and write diagnostic information to the serial port.

Make sense of Rusty's distress signal and get them out of there!

Part 0: Simple Setup

In this step you will load a simple polling program onto your mbed board and get it to run. You will use the 4 LEDs on the board to run an 18 second pattern of lights of your choice. The pattern itself should be random every time the board starts (but repeat itself for 18 seconds).
The idea is to make sure you can get a simple program to load up and display on the board.

Part 1: Polling for

Hook up a provided keypad. Determine the pins for the "#" key, and use a 1 kΩ resistor to make the input connection. Test your configuration by writing a simple input polling program to light the first LED whenever the "#" key is pressed and second LED when the "2" is pressed. What happens when you press both ? how fast are you polling ? can you type faster than the poll ?

Part 2: Did I Interrupt You?

Now, instead of polling, write an interrupt-based program that will respond by lighting the LED appropriately to input capture events (i.e. it should provide the same functionality as the previous step, except it will use interrupts rather than polling).

The microcontroller will need to respond to both the low-to-high ("#" depressed) and high-to-low ("#" released) transitions in order to turn on and off the LED at appropriate times.

See for example:

http://developer.mbed.org/users/yoonghm/code/keypad/docs/e48ba5b4c497/classKeypad.html

http://developer.mbed.org/cookbook/Keypad

EYES ONLY

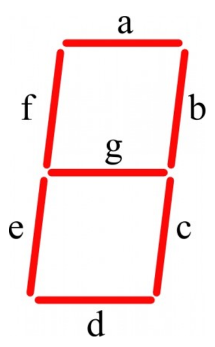
Part 3: Dash or Dot

Now that the microcontroller responds to input transitions, it will need to time the transitions in order to determine whether the input is a dot (.), dash (-), or space ( ). For this lab, a dot is defined as the key continuously depressed for approximately 30ms to 200ms, a dash is defined as the key being depressed for longer than 200ms, and a space is defined as the key not being pressed for longer than about 400ms.

See: http://developer.mbed.org/handbook/Timer

You can debug the code by having the microcontroller print out the timer variables as well as the transition times on each press and release of the key. Notice that sometimes there are several events of very short duration-this is due to mechanical bouncing of the contacts on the switch. By ignoring these short transition times, the program should be able to handle debouncing robustly.

Part 4: LEDs



Now we want to add some kind of indication of dot/dash. For example we can wire up a horizontal dash LED from the 7-segment display to indicate a dash, and a decimal dot LED to indicate a dot. Otherwise you can use 2 leds of one color for dash and single led of another color as a dot. Have the program light these LEDs in response to your Morse Code keypresses. These LEDs should only be lit for a brief period of time after the keypress.

Part 5: Morse Code → ASCII

This is fun ☺. Now program the microcontroller to decode the dots and dashes into actual characters (A, B, C, D, …). A listing for the Morse Code characters is provided at the end of this document. Have the program output the correct characters through the serial port. You can also try to output the characters (as many as you can design) on a 7-segment LED display.

Bit Extra credit 1: Speakers output

Add a speaker to be able to output a dot and dash sound.

Extra Extra credit 2/3/4:

Add a microphone and or light detector to be able to take more code input from another project group and be able to pass it on. So if you are hearing a more code sound, it should be able to turn it into led flashing lights, and if it detects light more code it should be able to turn it into sound morse code

Can you translate this too: . -- -… . -.. -.. . -.. / … -.-- … - . -- … / .- .-. . / ..-. ..- -.

Report:

You will need to have a TA verify that you have a working setup. Please make sure one of them signs your program listings and include it in your (very brief) lab report. Each group must submit a single lab report. The report should include:

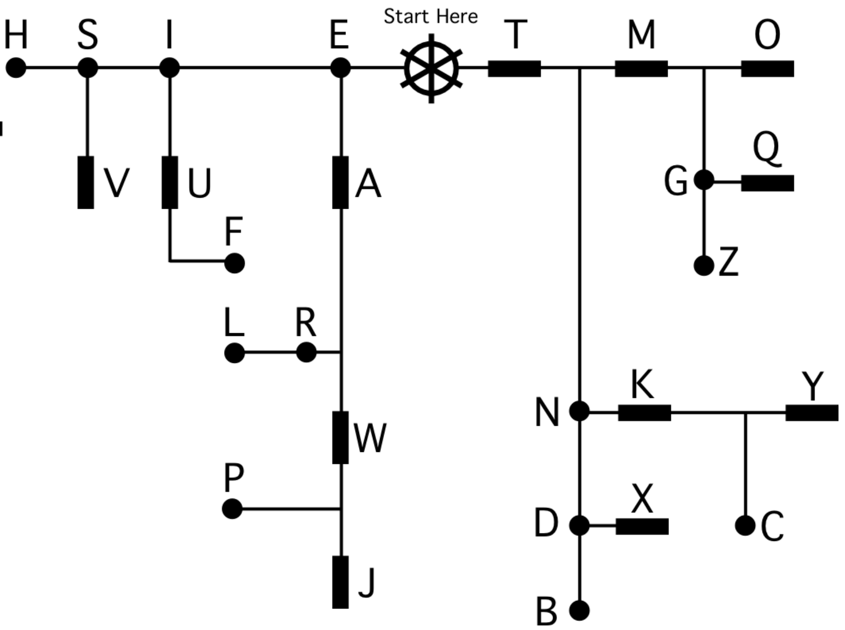
(a) List of lab partners
(b) Sketch and description of the hardware configuration (see fritzing)
(c) Completely documented listing of the code with an explanation for each module of the program. In addition the comments in the code will be a part of your grade.
(d) Discussion of differences between polling and interrupt-driven programs and how applicable they are for a task such as this.
(e) Any extra work you decided to add to the project to make it cooler, please document
(f) If something did not work, please also document
(g) Any issues you had to solve in order to get it working please document.
(h) If you use code/ideas from online sources, please cite.

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

Morse Code Receive Decoder Chart (a-z)



TOP SECRET
CONFIDENTIAL
TOP SECRET