ECE 4180 Lab 3 – Using the timer, interrupts, and the mbed RTOS

Section A Due Date: Feb 19 odd groups – Feb 20 even groups
Section B Due Date: Feb 21 odd groups – Feb 22 even groups

Names:_____Sect ____ _____Sect____

Item	Lab Demo	Extra Credit
Sonar with timers and interrupts		1%
Ticker blinking 4 LEDs at different rates		n/a
Mbed RTOS - LCD, LED, & Sound Effect Threads		n/a
Smart Phone App Control with Bluetooth		n/a
Hardware Breakpoints in Offline Compiler		n/a
EXTRA CREDIT OPTIONS		
Use an Android device with mbed		1%
Semaphore instead of LCD mutex lock		1%
5-way Tactile Switch Control		1%
Touch Switch Control		1%
RTOS thread scheduling using priorities & yield		1%
Light sensor to switch lighting modes		2%
Image or Video display on LCD		2-4%
Microphone Control Input		1%
High Quality YouTube video clip		1%
Flash Audio Player		1%
Thunderstorm Video		2%
Theremin using LIDAR sensor		2%

TA Signoff: _____ Score:_____

Part 1: Using a Timer with the Sonar sensor to measure distance (*15%)*

Lab Demo: Print the distance in feet to the nearest target on the PC using the second sonar example with the library code using interrupts and timers.

Additional Details After reading the Using a Hardware timer wiki, use the HC-SR04 sonar sensor in the kit to measure distance with the timer and interrupt option. A wiki page for the sonar used is available with wiring and code examples. The TA has extra Sonar sensors in the lab, but there is one in the kit. Read through the source code to understand how hardware timers are used in the examples provided. Make sure your Sonar is not tilting down at a slight angle towards the breadboard – It has a wide beam width and can pickup the breadboard!

Low Cost HC-SR04 Sonar module

(**Extra Credit 1%**) Build the Theremin using the Sonar and the Class D audio amp with the speaker. Feel free to change pitch scaling and offset to make it easier to play.

Part 2: Use Tickers to schedule code execution (***15%***)

Lab Demo: blink mbed's four built-in LEDs at different rates of 1, 2, 4, and 8 seconds using the Ticker function four times (no *wait()* allowed!). Ticker uses a hardware timer on mbed to periodically generate an interrupt that triggers a function to run. Ticker is similar to Timeout, but Timeout only works one time.

A periodic timer interrupt can be used to time slice the processors among various tasks and such hardware is used in an OS scheduler for time slicing among tasks. The mbed RTOS uses the timer to provide its 1 ms. time slice. With an RTOS, the scheduler controls the timer and makes it possible to run multiple threads. The RTOS also provides the needed synchronization primitives when threads share global variables or I/O devices. So once things get a bit more complex, an RTOS may be needed. The RTOS can make writing the code easier since tasks can be designed independently. Read the LED lighting effects wiki & code examples for an introduction to using the RTOS and threads using LED effects as examples.

Part 3 Use the RTOS to run multiple threads (*60%*)

Lab Demo: Use the RTOS to run multiple threads with each thread controlling different I/O devices or functions including LEDs, the speaker, and the LCD

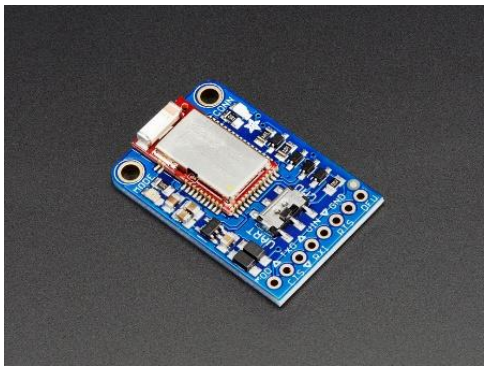
Additional Details: Use the mbed RTOS to run at least four threads. Make sure **not to use the new more complex Mbed 5 RTOS** (use the mbed 2.0 RTOS with links in this lab). Two threads put something on the uLCD display. Since the display requires a complex sequence of commands, mutual exclusion on the display is required and a mutex synchronization lock must be used in each thread before it can access the display. This avoids errors that could occur when the RTOS forces a switch between threads when they are in the middle of writing a command to the display. So, setup a single mutex that is always used in each LCD thread to lock and unlock access to the uLCD. Minimize the scope of the mutex lock by putting as many calculations as possible outside of the lock. Main always runs the first thread, so you only need to create three more threads. RAM is somewhat limited since each thread needs its own stack, so don't try more than perhaps 8 or so threads on the mbed LPC1768(only four at this point in the lab).

The other two threads should do the following:

1. Use the RGB LED or Shiftbrite to display a new lighting effect. Some ideas are one of the multicolor airport beacons in the LED lighting effect wiki or perhaps a thunderstorm, RGB Bar/Disco, welding, or fire effect that changes both intensity and color. Another idea is a Sonic screwdriver (any Doctor that has a flashing one with sound).
2. Play a wave file from an SD card or USB flash drive on the speaker Class D amp setup with sound effects appropriate for the LED lighting effect. Sound effect web sites have sound clips to use. Audacity can convert clips to *.wav files. The cookbook's waveplayer code will run in the RTOS without changes per LED effects wiki page.
3. The two LCD threads should display something related to the lighting and sound effects with two different time update rates in different areas of the LCD. Could be time, status, graphics, image, or video.

The easy way to set this up in the RTOS is by using Thread::wait(ms); in each thread at the end. There is a similar RTOS thread idea for a different LCD hardware setup on the mbed application board at http://mbed.org/users/4180_1/notebook/mbed-application-board-hands-on-demos/ if you need to see more RTOS code examples with threads, waits, and mutexes. There are similar graphics member functions for the uLCD display as in this example, but they do not have exactly the same names.

Add Bluetooth for remote control (*15 of the total of Part 3s 60***) -** Use an iOS or Android Smart Phone or Tablet for the control input to your Part 3 demo to control the RGB effect, sound effect, and LCD from Part 3 in a useful way. An Adafruit Bluefruit BLE UART friend module must be used (in kit and the TA has a few in the lab available for checkout). Details on the Bluetooth module can be found at https://developer.mbed.org/users/4180_1/notebook/adafruit-bluefruit-le-uart-friend---bluetooth-low-/. Other Bluetooth modules require more a lot more complex software than this one. The color picker control would be an interesting way to change color, and/or use the more basic control pad.



Adafruit Bluetooth LE UART Friend

When using getc() in an RTOS thread for the Bluetooth module, it needs the same mutex lock as printf (even on different serial port I/O hardware) or things can lock up. Printf() calls putc() and Scanf calls getc() as virtual functions so they are all related. Most of the functions in the C STDIO library are not reentrant (have global or static variables that need the same STDIO mutex lock). If you add the lock around getc(), don't grab the lock until you first check if it is *readable()* (i.e., *readable()* means a character is available for *getc()* in the UART buffer) or things could lock up until a new character shows up from the Bluetooth module. If you switch the Bluetooth device to use "*RawSerial*" instead of "*Serial*" this also seems to fix it. "*RawSerial*" does not use interrupt driven I/O on the serial port with buffering. The danger with "*RawSerial*" is that you could drop characters at high baud rates, if they are not read out fast enough. It likely is OK at just 9600 baud since each Bluetooth command only sends a few characters. A third alternative would be to use the I/O device hardware register pointers to write your own getc() routine to directly read a character from the serial port hardware, but you would need to be careful about using other mbed APIs on the same serial port and not let it setup interrupt buffers for the serial port.

Part 4- Using the offline ARM/Keil MDK C++ IDE (*10%*)

Lab Demo: Compile and run the Cylon LED demo using the offline compiler and debugger. Set breakpoints in the C++ code using the IDE.

Additional Details: For software development, the easy-to-use mbed cloud compiler is typically used. Projects from the cloud compiler can also be exported to the offline compiler (this link has the setup instructions needed). The ARM/Keil MDK full C++ compiler is widely used in industry and will compile and debug code for just about any microprocessor that exists. Many compilers used on microcontrollers do not support the full C++ standard or even the full C standard, but this one does. The cloud compiler is based on this tool with added modifications to automatically work using the cloud and the microprocessor and memory setup on mbed board.

Breakpoints will only work in the offline compiler via the USB cable with a firmware update for the mbed module. Microsoft web browsers just started renaming the file extension on downloaded **.if* files to **.htm* and this messes up the automatic mbed firmware install after the power on reset. So if you use them, be sure to rename the file extension on the mbed drive back to **.if* to fix it, or the new firmware does not install when you power cycle. Breakpoints will not work, if the mbed has older firmware. Microsoft must be using **.if* files for some strange new web stuff!

The offline compiler takes a bit more time and effort, but it supports breakpoints and full I/O emulation for debugging complex programs. For the offline compiler, download and install the free demo version (<32K code size) of the ARM/Keil MDK compiler. Note that this full C++ compiler supports just about any microprocessor that exists! The install packages are now split up into several options, since installing support for every microprocessor requires a huge download and a huge amount of disk space.

If you install your own version of the compiler, install "Ulink" if asked and install legacy Cortex M device NXP LPC1768 support if asked when you first open the project. To enable the offline compiler's full features, you must start the compiler and then connect over the network to a license server to enable the full version. For those installing on your own PC, the first time you open an mbed project after installing the compiler or are trying to connect to the license server the first time, be sure to start the compiler using right click "**Run as Administrator**" or the compiler might not be able to create and save new configuration and processor support files that it downloads after install. Some versions of Windows (such as OITs) seem to be locked down with some security settings and do not install the flash config file needed to download the mbed's flash with new code. If you get an error like this when downloading code to mbed, manually copy the file *LPC_IAP_512.FLM* available here to C:/Kiel_v5/ARM/Flash (or whatever path was used for compiler install).

At Georgia Tech connect using VPN (if not on campus) with your GT password to Georgia Tech's FlexLM license server. VPN should not be needed on an ECE lab PC. In the compiler, use **File->License Mangement ->FlexLM**. Then set the FLEX server to **8224@ece-winlic.ece.gatech.edu** and the full version will be enabled. **Note:** The checkoff can be done using the demo version since it is less than 32K of code.

Demo breakpoints using this Cylon LED project code which can be imported to the cloud compiler and exported to the offline compiler.

If you have problems with compile errors and/or breakpoints (As of 9/28 it looks like this is an issue!) in the MDK offline compiler, it appears that the current mbed cloud compiler project export tool does not always set up the project's target file info correctly right now for the offline compiler. It is likely a recent change in the newest version 5 of the offline compiler, and the cloud compiler export has not been fixed yet to handle it for all 150+ different mbed boards. It is a lot of work to fix all of the settings needed in 5 GUI windows in the compiler manually, so you can just run this offline compiler Cylon LED project file zip that has already been exported for the offline compiler on the PC with the newest compiler version and it should work. The settings were fixed using an older project file to compare them. There are two zip files available, one for version MDK 4 and a newer one for 5.

Extra Credit Options

Extra Credit: (1%) Send data from mbed to an Android device via USB. Read the Using mbed with an Android Phone or Tablet wiki. Print the names on the team members on the Android display for checkoff. Using the Android device to compile and download is not required, but feel free to try it! OTG cables and adapters are available in the lab for checkout. (an **additional 2%** for the first group to get this same idea working on an iPhone or iPad and write a similar mbed wiki notebook page for it – it may not be possible!)

Extra Credit: (1%) Use a Semaphore to provide mutual exclusion instead of the mutex lock in Part 3. Semaphores are more powerful, but also take a bit more execution time.

Extra Credit: (1%) Use the 5-way tactile switch (joystick) in the parts kit to control the RGB effect, sound effect, and LCD from Part 3 in a useful way. Use 4-5 of the switch inputs. Some functions to control might be dim, speed, volume, start/stop etc…

Extra Credit (1%) Instead of the 5-way tactile switch in the previous option, use the Touch switch keypad module in the parts kit to control your Part 3 demo. See this touch keypad wiki page for info and example code. If interrupts are used, make sure they do not use wait or printf and any global variables should be type "volatile" in code.

Extra Credit: (2-4%) Display an image (2%) or video clip (4%) on the LCD from the LCD's SD card in your Part 3 solution. Can only do one of the two options and it must be related to the LED lighting effect selected. See uLCD wiki page for examples and more instructions on image and video file conversions for the raw format uLCD's SD card.

Extra Credit: (1%) Experiment a bit with different RTOS thread scheduling. Use yield and set_priority. Give higher priority to threads that need to run more often.

Extra Credit: (2%) Add a light sensor to your Part 3 demo. Many bright LED displays use a light sensor to automatically dim at night or even switch to a completely different lighting scheme at night. Many radio and TV towers go from white to red at night. The parts kit now has an ambient light sensor (looks like a small clear LED). They should hookup just like the phototransistor circuit example seen on the wiki page at https://developer.mbed.org/users/4180_1/notebook/using-a-photocell-to-determine-light-levels/.

The TA should also have some CdS photocells that can be used instead of a photo transistor. (the older CdS photocells are now banned in Europe since Cadmium is toxic! – so new designs should not use them) There is a wiki page available on it at **https://developer.mbed.org/users/4180_1/notebook/using-a-photocell-to-determine-light-levels/**

Extra Credit: (1%) For LED effects that should change based on the audio output, use the microphone to pickup the speaker output and keep the lighting effect in rhythm with the sounds. The bar/disco/night club effect is one that should work well with this.

Extra Credit: (1%) Produce a high quality YouTube video clip of your LED effect demo (i.e., in focus, no shaky camera, clear audio without external noise, with a large image of the displays in Part 3). Don't forget to hold your phone **rotated 90 degrees** during video recording to get the correct aspect ratio needed for nice large YouTube videos.

Extra Credit: (1%) Redo your Part 3 demo using on-chip Flash memory instead of the SD card to store the audio data. Read the Using Flash to play Audio Clips wiki page for details.

Extra Credit: (2%) Develop a random thunderstorm mbed-based audio video effect and record it in a YouTube video. There are commercial products like this for model train layouts and even aquarium lights. This video has a demo of the system for model trains. And this one for an aquarium LED light thunderstorm system.

Extra Credit: (2%) Develop a Theremin using the LIDAR sensor instead of the SONAR sensor. It has the potential to be easier to play since it supports a faster update rate and has a narrower detection beam width.

Early Bird Bonus (2%) extra credit to first two groups to demo the regular lab 1 assignment including all the extra credit options (except the new iPhone wiki). The first couple of lab teams to finish it will likely wind up doing a bit of extra work blazing a path forward for everyone else. So they will get extra credit.