Lab 6 Course Project

Timeline & Grading

- **Pre-proposal**: Saturday, April 13th at 9:00 PM on CMS, **groups of 2**
- **Meeting with a TA**: Discussion section, April 15th – 19th
- **Proposal**: Monday, April 22 at 9:00 PM on CMS
- **Code, Video Demonstration, and Project report**: Friday, May 17th at 4:30 PM on CMS
- **Grading**: 11% of total grade

Section I: Overview & Grading

In the final course project, your group will take an idea that we explored in the course (but haven't yet done as a lab) and extend it. You are free to choose your project topic, but your project must include significant components implemented in C or assembly by your group on the FRDM K64F board.

Note that you only have 2-3 weeks to work on the project implementation. Please choose the project scope carefully. Organize your project around a baseline that you are confident that your group can finish on time with potential extensions that you would like to have if time permits.

Grading:

Your project will have a maximum grade set based on its difficulty.

Difficulty is broken down into two categories:
- Easy (80% score cap)
- Hard (100% score cap)

Evaluating the difficulty of your proposed project is partially subjective, so it is difficult to provide a complete rubric of how it will be judged. In general, you will be evaluated according to this standard:
- A project that does not go beyond the topics covered in the labs will either be rejected by the TA's or be considered "easy" difficulty.
- A project that goes beyond the material covered in the labs and implements some non-trivial concept covered in lecture (e.g. implementing a SPI interface with a peripheral device) or some topic not covered in the course at all will be considered "hard" difficulty.

Note that the difficulty of a project may change depending upon how you make use of outside resources. For example, if you make use of a peripheral device but take advantage of an existing library for communicating with that device, this is unlikely to be considered difficult. If you make use of peripheral device and have to rewrite pieces of an existing library to make it work on the FRDM board, this may be considered difficult.

Extra Credit:

The top 5 video demonstrations will be awarded up to 5 bonus points. The top 10 projects in the class will be awarded up to 10 bonus points. It is possible to win both.

The following list shows some of the top projects from the previous year as inspiration. ***Under no circumstances should to copy work done in previous years. You must complete your own project from scratch.***

- Internet-synchronized digital clock
  - https://cornell.box.com/s/0y69gyxzmbo7f5tj0va1e5l0miku7ovi
- Capacitive touch KEYboard
  - https://cornell.box.com/s/8ah4jc0qaj539fs8ajkmq4c823u169m4
- PENbot: Drawing robot
  - https://cornell.box.com/s/r8hatuxbjvug7pcjnjlsc20rv478e9i5
- Rhythm game
  - https://cornell.box.com/s/geabow85onu3sx5uul9hyfj65r44c5se
- VR zombie game
  - https://cornell.box.com/s/cdexhmu4tdzu3h60o0jtsk3g1a260ffg
- Internet-connected weather display & alarm
  - https://cornell.box.com/s/gmbepqm4mhf018n44ku2f97stm4y3kns
- Digital multi-tool using NeoPixel LEDs
  - https://cornell.box.com/s/ewz9qatjyitw5s3569pdg4hqukxrpqc5

Section II: Pre-proposal

Submit a pre-proposal on CMS (0.5 pages, max). Provide a brief description of your project idea and mention any extra hardware you will need to buy.

Note that any extra hardware will be purchased and paid for by you. **You are not permitted to solder on the FRDM board under any circumstances.** You are welcome to limit your project to use only what is available to you on the FRDM board.

Section III: Proposal

After discussing your project idea with a TA, your group needs to write and submit a full proposal. The proposal should be submitted in PDF format, and use 11pt font and single line spacing. The proposal needs to explain your group's project idea in enough detail for us to evaluate if the difficulty is appropriate.

All proposals should include the following:

1) **Title**
2) **Project group** – Names and NetIDs of the team members
3) **Overview** – Description of the project. What are you planning to do? Provide the necessary background information.
4) **System Description** – An explanation of how the system will work.
5) **Major technical tasks and testing** – How do you plan to implement your system and test it?
6) **Development timeline with milestones** – How do you plan to finish the project in time?
7) **Work distribution plan** – How do you plan to split the work on the project?

As part of your project, you may create an embedded system with expanded sensing and/or actuation by adding extra hardware. **However**, you're on your own in terms of getting it done; you are responsible for purchasing hardware components and implementing the extension. If your project requires an additional hardware component, the proposal should also include information on the hardware extension.

8) A complete **Bill of Materials (BoM)** – include specific part numbers (e.g. "ADXL345 3-Axis digital accelerometer", not "a motor") and a catalog number from the source (Digitkey, Mouser, etc.) with quantities and pricing.

9) **A System Schematic Diagram** – Check that parts will work with the FRDM board (e.g. correct voltage/current ranges, compatible communication protocols, etc.), and ensure that there are adequate pins for your system (not every pin is brought off the board).

Section IV: Project Report & Demonstration Video Guideline

Demonstration Video:
You will submit a link to a private **2 minute** YouTube video in which you demonstrate your project and its features. Include this link in your project report.

A good demo will include the following elements:
1. Discussion of whether your group achieved what you set out to do in the proposal

2. Explain what libraries and code you obtained from outside sources and what code was written by your group. Also, discuss which part of the lab was most challenging.
3. Show that the project works as described. Your code should not be significantly different from what you submitted on CMS. If necessary, describe any last-minute changes you made.
4. The project works reliably, even if somebody with no prior background (e.g. TA) tries to make it work
5. Both group members understand what is going on and a significant portion of the code was done by the group instead of copying it from other sources
6. It is clear that both partners contributed to the work

Project Report:
PDF format. 8 pages maximum, 11pt font and single line spacing.

This is a suggested outline for the report. Whatever format you decide to use, you should cover all the points below.

1. Introduction
Give brief background information about your project. Describe what your final project does. What does it achieve?

2. System diagram
Include a block diagram of your software system. Briefly describe how the different components interact. Explain and justify any major architectural choices. Also explain and justify the use of interrupts, scheduling, real-time, etc.

3. Hardware description (if applicable)
Provide a bill of materials for your project. Include a schematic showing how any external hardware is connected.

4. Detailed software description
Describe the main routines that your code uses. Describe the main data structures that are used throughout your design. Describe the protocol used to communicate with all the actuators and/or sensors. Describe which software components were written by your group and which ones were pulled from outside sources.

5. Testing
Explain your testing strategy and how you convinced yourself your implementation is correct. Include a brief description of any test cases you wrote and what they are intended to test.

6. Results and challenges

Did you achieve what you proposed? What was the most complicated part of your design? Describe how your final implementation differs from your project proposal (if it does). What would you do differently next time?

7. Work distribution

How did you carry out the project? Identify the major components that constituted the project. How did you collaborate? In your own words, explain how you divided the work, how you communicated with each other, and whether/how everyone on the team had an opportunity to play an active role in all the major tasks.

8. References, software reuse

You must cite the source of any code you do not personally write. It is ok and expected to build from small sections of example code (including what we gave you), you just need to acknowledge the source. Please list any manuals, data sheets and other documents that you used for your project.

Section V: Pre-packaged Projects

Easy Difficulty: Reaction-time Game

Using two buttons and several LEDs, build a game to test the reaction time of two players. With an LED, signal the players when to hit their button. Show with an LED which player had the fastest reaction time.

Hard Difficulty Project: Temperature & Humidity Display

Interface the board with an Adafruit 931 OLED display and an Adafruit DHT11 temperature-humidity sensor. Output the current temperature and humidity on the display.

Hard Difficulty Project: Joystick & Buzzer Game

Interface the board with an Adafruit 16x32 RGB LED panel, an Adafruit 245 Joystick, and an Adafruit 160 Piezo Buzzer. Use the joystick to pilot an agent around a virtual space, showing the agent as an LED on the panel. Activate the buzzer if the user drives the dot into the edge of the screen.

Section VI: Resources

Part 1: Accelerometer Tutorial

1) Install the full Kinetis SDK

The files that you installed in Lab 0 only included the bare minimum for running code. Keil provides a much richer set of drivers for interfacing with everything from I2C devices and SPI devices to USB, Ethernet, and the SD card, as well as many useful libraries (eg a TCP/IP stack, and a FAT filesystem for the SD card slot). We need to install those additional libraries.

1. From uVision, open the Pack Installer (highlighted above)
2. In the left pane, select All Devices » Freescale » K60 Series » MK64FN1M0xx12 (as in Lab 0). Now, in the right pane, install the following components (these will take some time):
    * Keil::Kinetis_SDK_DFP
    * Keil::MDK-Middleware
    * Keil::ARM_Compiler
3. Close the Pack Installer. If you see a dialog asking you reload packs, click yes.

2) Create a new project

When creating the project, do not select the device you normally do. Instead, choose Freescale » K Series » K6x Ethernet Crypto MCUs » MK64FN1M0VLL12.

In the dialog that pops up next, you need to select two items:
    * Board Support » Accelerometer (API) » Accelerometer
    * Device » Startup

Once you've chosen those, click the Resolve button in the bottom left. This will automatically select all of the other required libraries. You should now see something similar to the screenshot on the right.
Note: You may want to explore the options here, particularly under File System, Network, and USB. These contain extra libraries and drivers which may be useful to some projects.
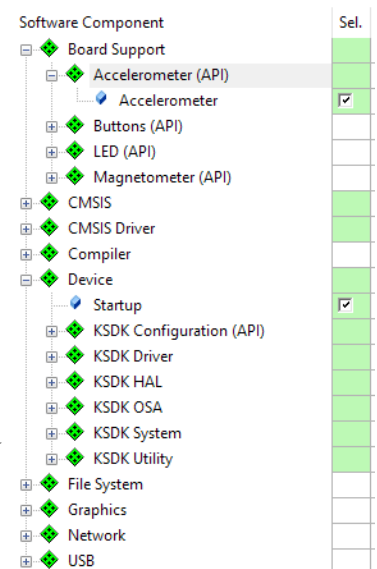
3) Configure the linker

When the project is created, the compiler's linker (the program which decides how to place code and variables in memory) is not fully configured. Open the "Options for Target" dialog (the one with the debugger settings), and go to the Linker tab. Click the button with 3 dots next to the Scatter File field. Starting from the directory you created your project in, browse to RTE\Device\MK64FN1M0VLL12 and select MK64FN1M0xxx12_flash.

While the dialog is open, make sure you go to the Debug tab and select the CMSIS-DAP debugger. Unlike previous labs, however, you should not check the "Use MicroLIB" box or change the floating point hardware.

4) Configure the I/O peripherals

Table 5. Accelerometer and magnetometer signals connection

FXOS8700CQ	K64
SCL	PTE24/UART4_TX/**I2C0_SCL**/EWM_OUT_b
SDA	PTE25/UART4_RX/**I2C0_SDA**/EWM_IN
INT1	**PTC6**/SPI0_SOUT/PDB0_EXTRG/I2S0_RX_BCLK/FB_AD9/I2S0_MCLK/LLWU_P10
INT2	**PTC13**/UART4_CTS_b/FB_AD26

The libraries enabled for this project need to know what peripherals will be used, and how they should be configured. In the project browser, expand the Device tree, and open *RTE_Device.h* (it should be near the bottom). Click the Configuration Wizard tab at the bottom of the editor. From the Configuration Wizard, expand the I2C0 entry. Check the box next to I2C0, then within I2C0 change I2C0_SDA to PTE25, and I2C0_SCL to PTE24. These settings were found in the FRDM-K64F User Manual (the relevant table is shown above). The settings should look like this when you're done:

Option	Value
⊞ ENET (10/100-Mbps Ethernet MAC) [Driver_ETH_MAC0]	☐
⊟ I2C0 (Inter-Integrated Circuit Interface 0) [Driver_I2C0]	☑
I2C0_SDA	PTE25
I2C0_SCL	PTE24
DMA Rx	Disable
DMA Tx	Disable
⊞ I2C1 (Inter-Integrated Circuit Interface 1) [Driver_I2C1]	☐

5) Modify the hardware setup code

The default code does not enable the clock to PORTE, since the default settings don't use PORTE at all. Open *hardware_init.c* (it should be directly below *RTE_Device.h* in the project browser), and add CLOCK_SYS_EnablePortClock(PORTE_IDX); to hardware_init(). The code should look like the following when you're done:

```
35
36 ⊟void hardware_init(void) {
37
38     /* enable clock for PORTs */
39     CLOCK_SYS_EnablePortClock(PORTA_IDX);
40     CLOCK_SYS_EnablePortClock(PORTB_IDX);
41     CLOCK_SYS_EnablePortClock(PORTE_IDX);
42
43     /* Init board clock */
44     BOARD_ClockInit();
45     dbg_uart_init();
46 }
```

6) Write the main program

Create a new .c file (you can name it anything you want, main.c is a good choice), and put the following code in it:

```
1   #include <Board_Accelerometer.h>
2   #include <Board_Magnetometer.h>
3   #include <fsl_debug_console.h>
4   #include <board.h>
5
6   ACCELEROMETER_STATE state;
7   MAGNETOMETER_STATE mstate;
8
9 ⊟int main() {
10     hardware_init();
11     Accelerometer_Initialize();
12     Magnetometer_Initialize();
13
14 ⊟  while (1) {
15       Accelerometer_GetState(&state);
16       Magnetometer_GetState(&mstate);
17       debug_printf("%5d %5d %5d %5d %5d %5d\r\n", state.x, state.y, state.z, mstate.x, mstate.y, mstate.z);
18     }
19 }
20
```

Board_Accelerometer.h and *Board_Magnetometer.h* contain the interface for the accelerometer/ magnetometer chip. *fsl_debug_console.h* provides an interface for printing debug information to a computer over the UART, and *board.h* provides high level functions for the board.

`ACCELEROMETER_STATE` and `MAGNETOMETER_STATE` are structures that the accelerometer driver uses to pass data back to your code. Each has an x, y, and z field, for the 3 axes that each sensor measures.

The `hardware_init` function configures some of the lower level parts of the board (sets up clocks, enables the GPIO ports, etc.). The I2C peripheral is automatically initialized by `Accelerometer_Initialize` and `Magnetometer_Initialize`.

In the loop, the code requests an update from the accelerometer and magnetometer using the GetState functions. The values are then sent to the computer using debug_printf ("\r\n" in the format string tell the computer to start a new line after the printf)
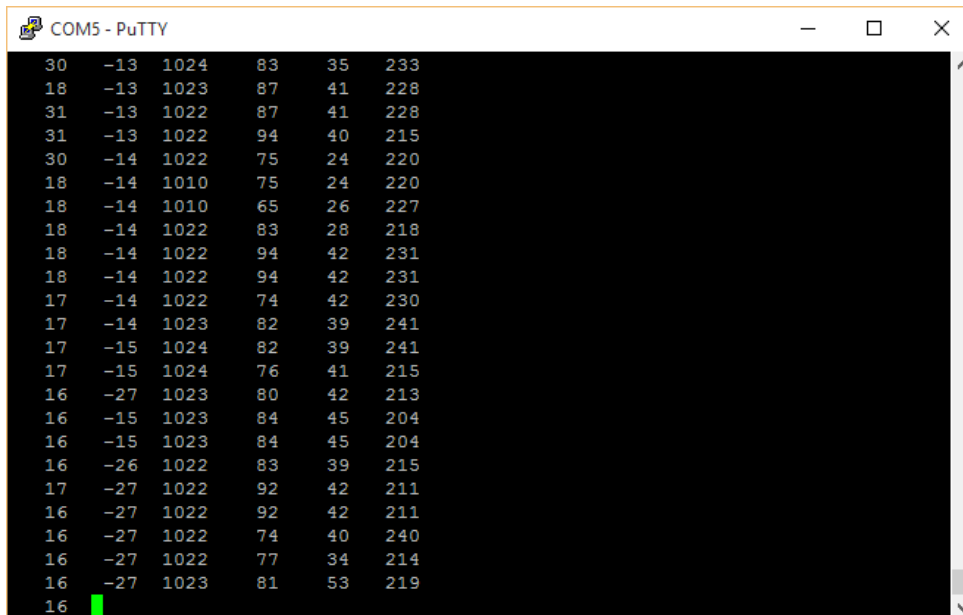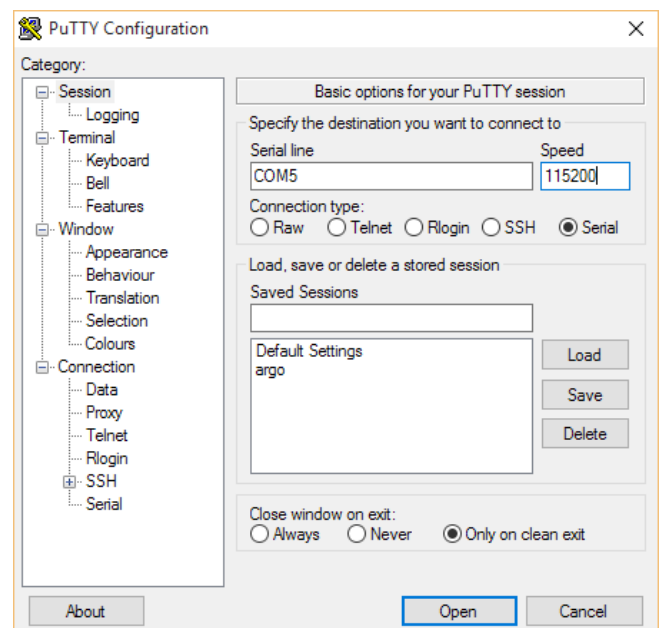
7) Test it

You can upload the program to your board as usual. To view the output, you'll need to use a
serial port monitor such as PuTTY. You'll also need to know the name of the serial port exposed
by the board. To do this, open Device Manager (guide here), and expand the Ports section. You
should see an "mbed Serial Port" with a port name, as shown below.
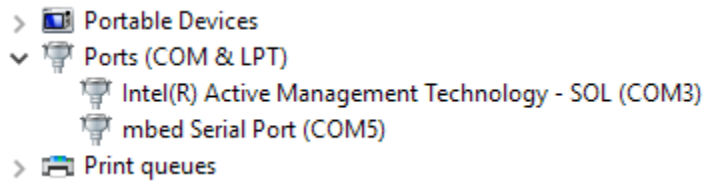
From PuTTY, select the Serial radio button, and put the
port name you found in Device Manager in the Serial
line box (COM5 in this example). Set Speed to 115200
(this is often called Baud Rate). Your setting should look
something like what is shown to the right:

Click Open, and if everything worked correctly, you
should see the accelerometer data in the console!

Part 2: Additional Documentation

After doing step 1 in the tutorial, a number of other libraries that may be useful for your projects will be installed onto your computer. Here are some links to their documentation:

Kinetis SDK: [Click Here]
This is a low level library that provides a relatively simple abstraction over each of the peripherals on the microcontroller. It mostly just provides functions that make accessing registers in components such as the PIT simpler.

CMSIS-Driver: http://www.keil.com/pack/doc/CMSIS/Driver/html/index.html
This is a higher level abstraction for the communication interfaces (I2C, SPI, SD card, USB, Ethernet), designed to be a simpler interface that is shared across chips designed by different manufacturers. If your project involves interacting with another device over I2C or SPI, you probably want to use this library. The accelerometer library is built on top of this one.

MDK-Middleware: http://www.keil.com/pack/doc/mw/General/html/index.html
These are a series of high-level libraries built on top of CMSIS-Driver, which provide a very rich set of tools for your projects. Features include filesystems for the SD card, TCP/UDP stacks, HTTP/FTP/email servers/clients, and APIs for building USB devices, to name a few. The documentation also provides starter code examples for most of these features, which could be a good place to start for a lot of projects.