ECC 266 Lab #2, Wall Clock
Summer 2021

Lab location: Online only

Lab Time

- 8:00am – 10:40am, 6/29 & 7/1

Demo Due: The beginning of Lab 3 (7/6)

Submission Due: Tuesday, 7/6, 11:59pm

Introduction

The goal of this lab is for you to develop a simple, real-world embedded system application; and for you to collaborate with your partner(s) through GitHub. The learning objectives are:

- Develop a wall clock application
- Collaborate with your lab partner through GitHub
- Get familiar with the Grove base BoosterPack
- Get familiar with the Grove 4-digit 7-segment display

Note: You should have found a partner in Lab 1. Your TA will give your group a unique codename.

Prelab

- Read the following three parts of the GitHub Guide at https://guides.github.com: "Understand the GitHub flow," "Hello World", and "Git Handbook"
- Read the GitHub Desktop User Guides: https://help.github.com/desktop/guides/
- Read the Wiki page on Grove Base BoosterPack provided by Seeed, at this URL: http://wiki.seeedstudio.com/Grove_Base_BoosterPack/.
- Read about 7-segment display on Wikipedia: https://en.wikipedia.org/wiki/Seven-segment_display

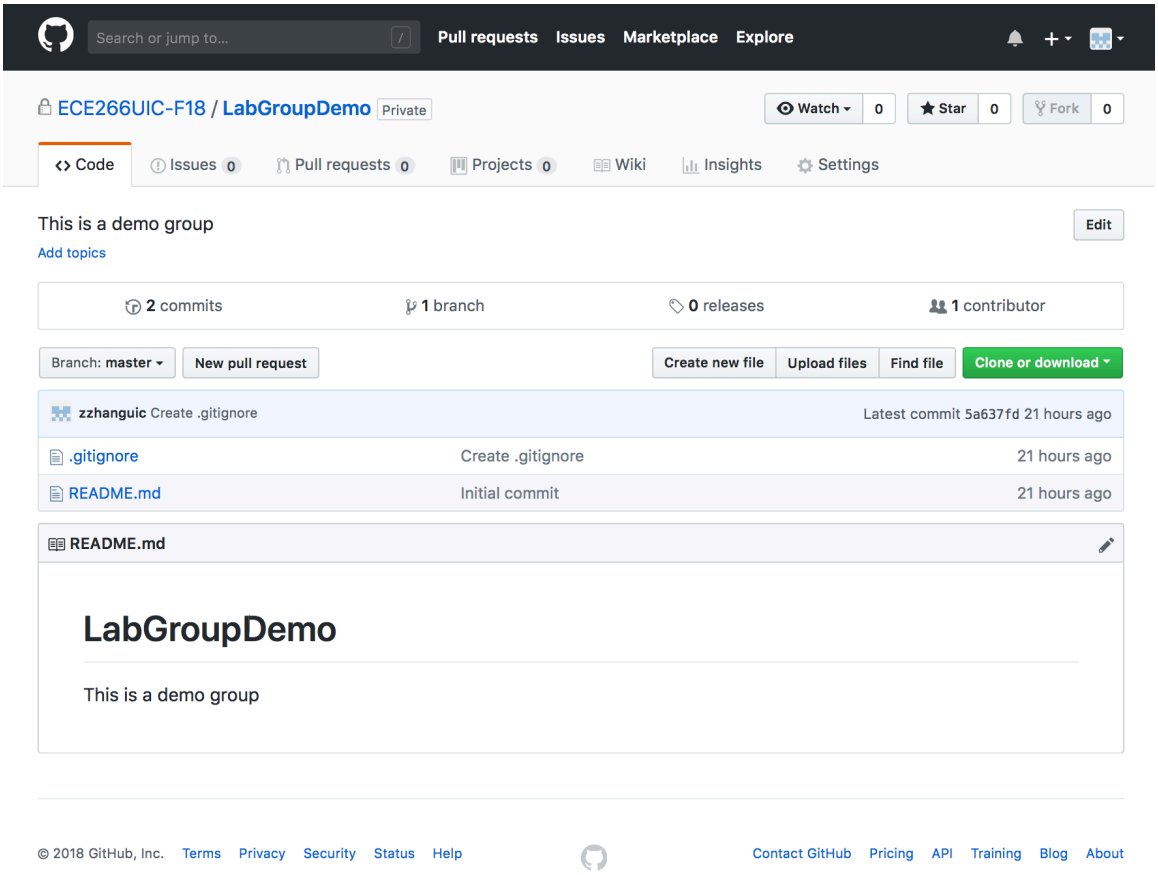
Part 1: Preparation

You will receive an invitation from your TA to join a GitHub group. That means to become a collaborator for a *GitHub repository* created for your group. The URL of the latter is in form of https://github.com/ECE266UIC-Su21/Group*X*.git, where *X* is your group number. Accept the invitation. You and your partner will use this repository for the rest of the semester. It is a repository private to your group, thus other groups won't be able to access your codes.

Important: You need a GitHub account. If you don't yet have one, create one at https://github.com using your UIC email address.

GitHub Repository Setup*: One in your group should do the following, with help of the other person*.

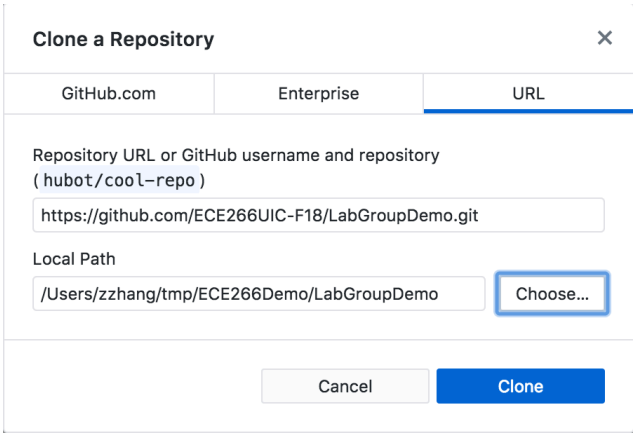
1. Check your access to your group's GitHub repository: Log into your GitHub account and navigate to your ECE 266 repository. You should see a window like the following (your group codename will be different).



Initially, your repository has two files, "README.md" and ".gitignore". The first file is a short description of your repository, which can you update. The second file is for

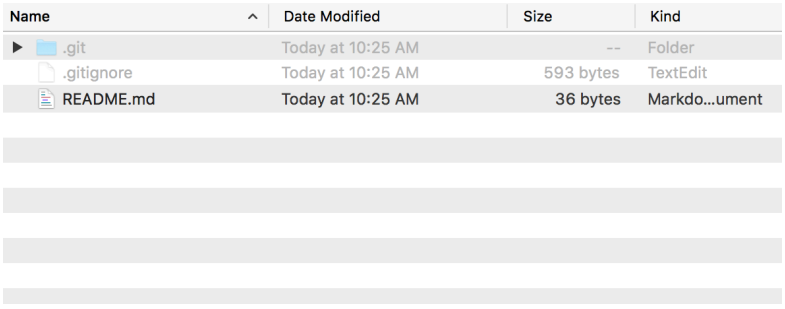
preventing unwanted files, for example temporary files, from being added into your git repository.

2. Click on "Clone or download", then choose "Open in Desktop". Confirm "Allow". You will see a dialogue window for cloning the repository. Choose your working directory for ECE 266 as the local path (see the picture below), then confirm "Clone".



You will see a new folder named Group*X*, where *X* is the codename for your group. This will be your working directory for Git. All your lab codes shall be put under this folder except Lab 1.

3. Check your working directory (i.e. the Group*X* folder): Switch to a file browser, navigate to your working directory. It should look like the following:

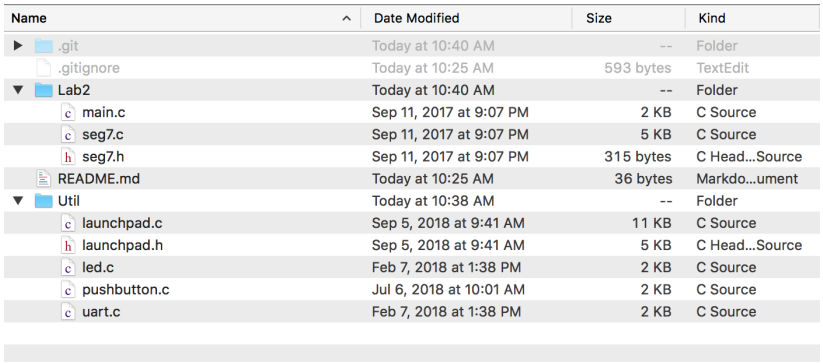


Your view can be different: If your file browser is configured not to show hidden files, then you will not see ".git" and ".gitignore".

Directory ".git" contains the *local repository*. It is maintained by git software (i.e. GitHub Desktop). Do NOT change anything in it.

4. Now you need to re-create the Util project so that it becomes part of the GitHub repository. In the file browser, download again "Util.zip" from Lab 1 assignment, unzip the file. You will see the "Util/" folder. You may then delete Util.zip.

Then, create another folder named “Lab2” in the working directory. Download “lab2_main.c”, “seg7.c”, “seg7.h” from Lab 2 attachments to this folder. At this time, your working directory may look as follows:



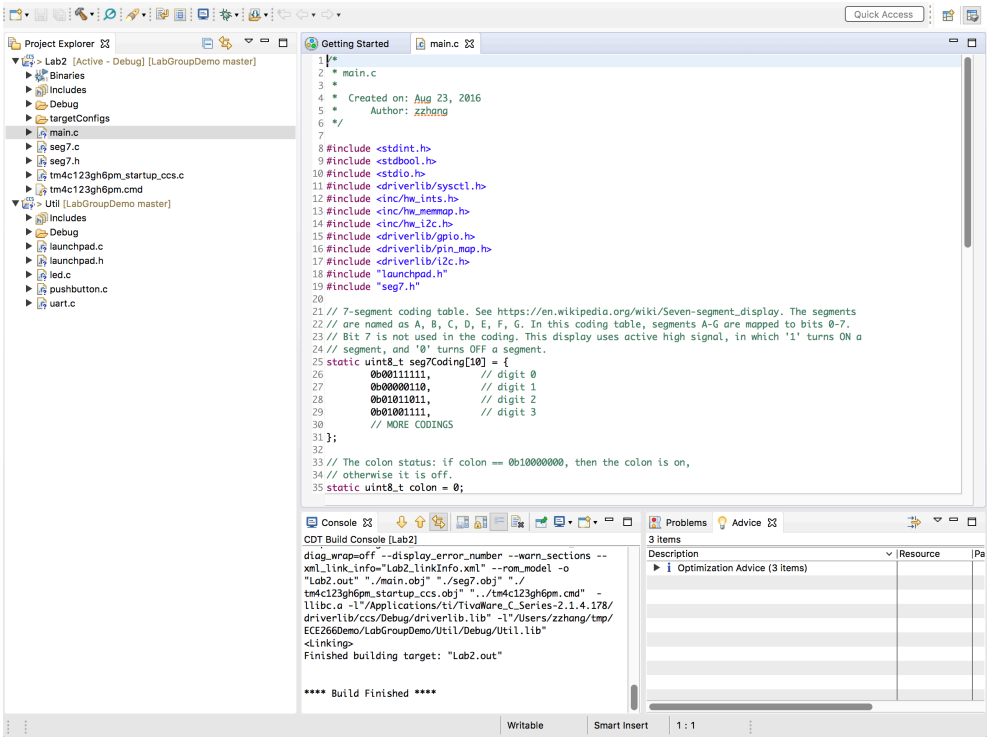
Name	Date Modified	Size	Kind
.git	Today at 10:40 AM	--	Folder
.gitignore	Today at 10:25 AM	593 bytes	TextEdit
Lab2	Today at 10:40 AM	--	Folder
main.c	Sep 11, 2017 at 9:07 PM	2 KB	C Source
seg7.c	Sep 11, 2017 at 9:07 PM	5 KB	C Source
seg7.h	Sep 11, 2017 at 9:07 PM	315 bytes	C Head...Source
README.md	Today at 10:25 AM	36 bytes	Markdo...ument
Util	Today at 10:38 AM	--	Folder
launchpad.c	Sep 5, 2018 at 9:41 AM	11 KB	C Source
launchpad.h	Sep 5, 2018 at 9:41 AM	5 KB	C Head...Source
led.c	Feb 7, 2018 at 1:38 PM	2 KB	C Source
pushbutton.c	Jul 6, 2018 at 10:01 AM	2 KB	C Source
uart.c	Feb 7, 2018 at 1:38 PM	2 KB	C Source

5. Switch to CCS. Delete the old “Util” project and recreate the “Util” project as you have done in Lab 1, with an external reference to the “Util” folder in the working directory. Recall that the output type should be “Static Library”, and you have to configure the include path correctly. Build the project and check “Util.a”.

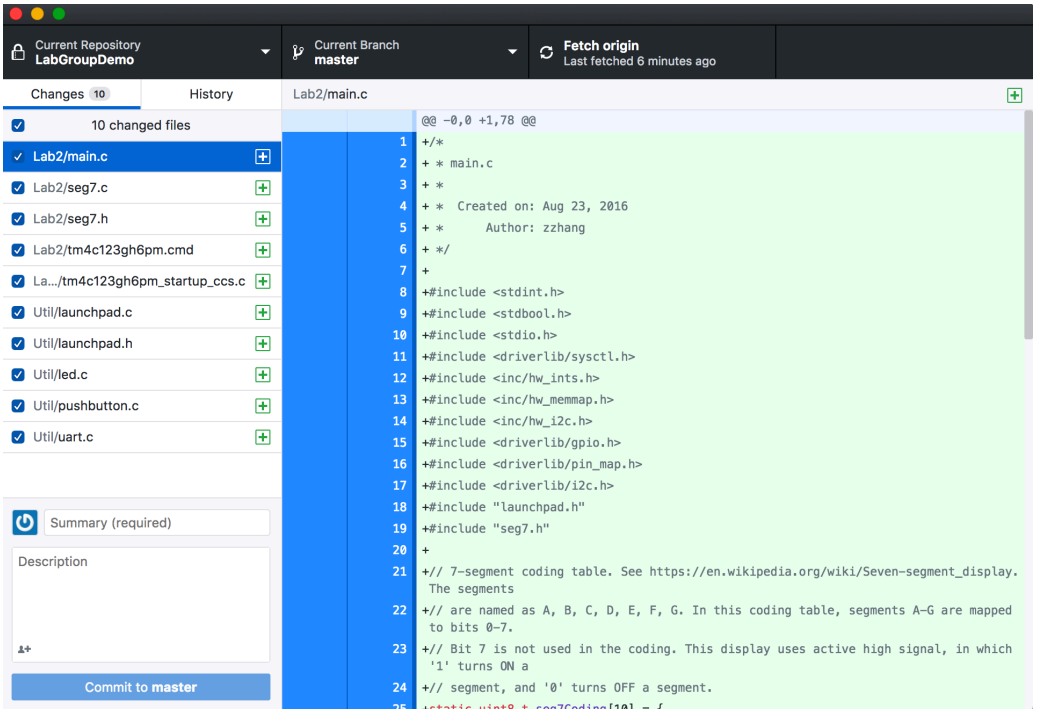
   Note: Do this step after you demo Lab 1. Otherwise, you may have to change the include and library paths for Util.lib in your Lab1 project.

   Optional: You may recreate your Lab 1 project so that it’s part of the GitHub repository.

   Create another project named “Lab2” with an external reference to the “Lab2” folder in your working directory, as you have done in CCS project “Lab1” in Lab 1 assignment. Again, recall that you should NOT use the default location suggested by the CCS, the connection type should be “Stellaris In-Circuit Debug Interface”, and the output type should be “Executable”. Configure the include and library file path correctly, as you did in Lab 1 assignment. Build the project. Your CCS view will be like the following:



6. Switch to GitHub Desktop and check. You should see all the new source code files in Tab “Changes”.

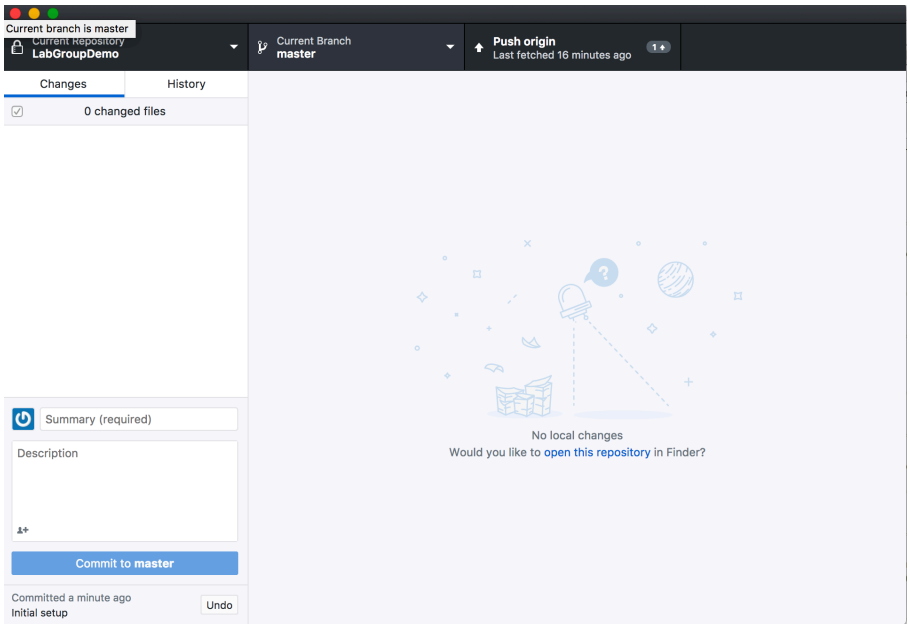


Note that there are two files generated by CCS, namely “tm4c123gh6pm.cmd” and

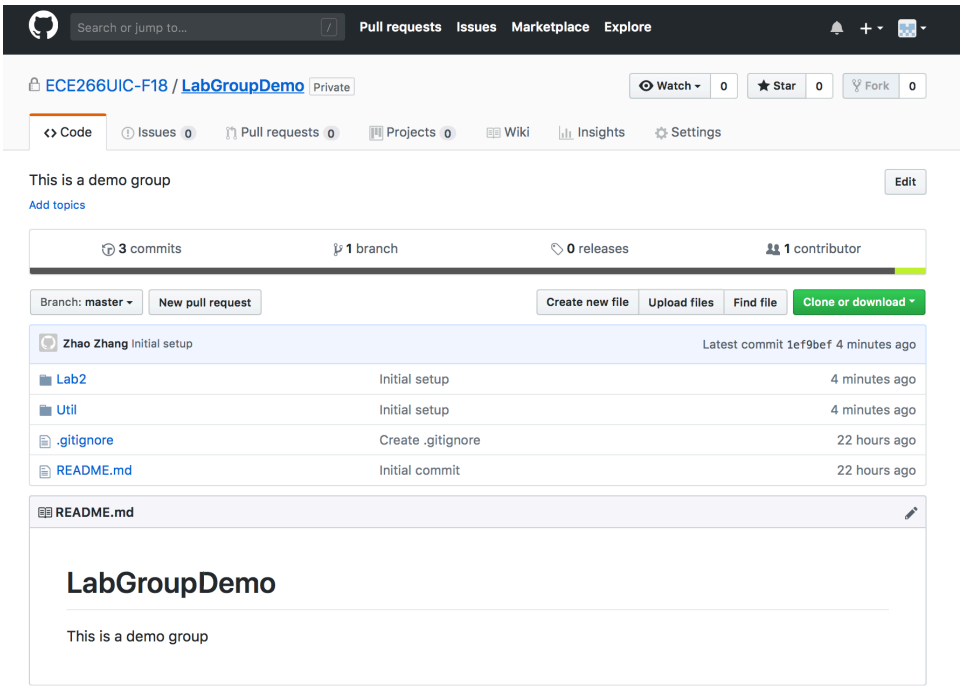
"tm4c123gh6pm_startup_ccs.c". They should be included in the GitHub repository. The first one is the Link Commander file that tells the CCS linker how to link all objective files in project Lab2. The second one is the startup code for project Lab2, which includes the default interrupt handlers and other C routines.

7. Type a message in the "Summary" box, for example "Initial setup". You may also put a description in the "Description" box. Then click on "Commit to master". It will trigger a local commit, i.e. to commit the changes to the local repository in the ".git" folder. Then, all the changes will disappear from the "Changes" Tab.

8. Click on “Push origin” in the GitHub Desktop. This will push all the committed changes in the local repository to the *remote repository* on GitHub.



9. Check your repository at https://github.com again from a web browser. The new files should appear in your group’s GitHub repository. Click on “Lab2” and “Util” and check the contents.



Now you’re done with the GitHub setup.

GitHub Repository Check-out: ***The other partner*** should do steps 1-3 to checkout the repository, and then step 5 to create and build projects "Util" and "Lab2".

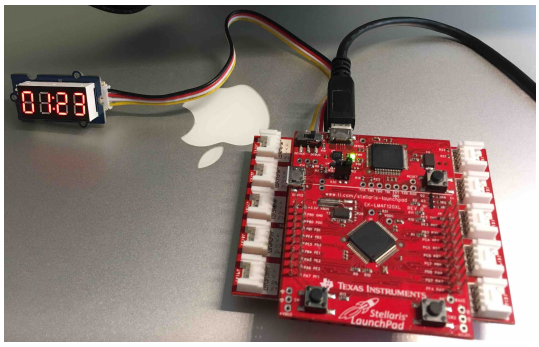
What's Next with GitHub: In the future, if any partner makes new changes to the source code files in CCS, the changes will appear automatically in GitHub Desktop. He/she may commit the changes to the local repository, and then push the committed changes to the remote repository at GitHub. The other partner(s) may start GitHub Desktop, pull the changes into his/her local repository, and then the changes will appear automatically in CCS.

Note: The GitHub flow (see https://guides.github.com) recommends that you create a new branch (of the code), work on the branch, create a pull request, ask your partner to review the pull request, and then merge the new branch into the master branch. You don't have to do that. Both you and your partner can work on the master branch without creating the new branch.

The suggestion here is for simplicity; the recommended GitHub flow is good for a real-world project of a fair amount of complexity.

Now you may continue to the hardware setup for Lab 2.

Attach the LaunchPad with the Grove base BoosterPack, as shown in the following picture. Be careful with the orientation of the boards, and make sure you put the LaunchPad on top of the Grove base. Attach the Grove 4-digit 7-segment display to the **J10** jumper using a jump cable. Double check that the correct jumper is used.



Follow the steps in Lab1 to build and run the provided Lab2 program on the board. You will see the display shows "01:23". The colon ':' will flash on and off every 0.5 second.

Part 2: Wall Clock

Revise the code in "lab2_main.c" such that the program shows a running clock on the display, with the following requirements:

1. The clock shows two digits of minutes to the left of the colon, and two digits of seconds to the right of the colon. The clock advances every second.

2. The colon (':') flashes on and off every half second.

3. The user may push pushbutton SW1 to fast forward the minutes of the clock with wrap-around. Hint: You may use the pushbutton function in lab 1 as template.

4. The user may push pushbutton SW2 to fast forward the seconds of the block with wrap-around.

Read the source code in lab2_main.c carefully before you start to revise the code. Also read through seg7.c and seg7.h.

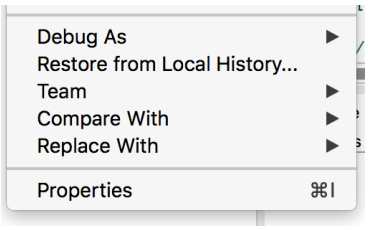
You and your partner may work concurrently to revise different lines of "lab2_main.c". You are suggested to divide the work as follows:

- One partner may revise the clockUpdate() function.
- The other partner may create a checkPushButton() function. Use the checkPushButton() function from Lab 1 as a template.
- Hint: You may use global variables shared by the two functions, so that checkPushButton() may change the status of the clock. In embedded system programing, it is common to use shared global variables for this purpose. You still have to be careful not to use global variables unnecessarily.

Use GitHub Desktop to push your changes to your group's GitHub repository, and to pull the changes made by your partner. Both partners should do so. Review the changes made by your partner.

If any conflict happens, i.e. if you and your partner(s) happen to change the same code lines, resolve the conflict – see https://help.github.com/articles/resolving-a-merge-conflict-on-github/.

Note: CCS has built-in support for git and GitHub. The functionalities can be accessed through the right-click menu of a project, as follows:



The menu items "Restore from Local History", "Team", "Compare With", and "Replace With" are related to git and GitHub. However, in the beginning, you may find GitHub Desktop is easier to use.

Lab Demo

First show your GitHub repository at https://github.com to your TA. Then, demo your wall clock program.

At the time of demo, each of you may have to answer questions from your TA, which test your understanding of the working details. Your answer may affect your lab grade.

Other Activities

You may explore the following documents:

- The User's Guide of Tiva C Series TM4C123G LaunchPad Evaluation Board
- TM4C123GH6PM Microcontroller Datasheet
- The Wiki page of Grove base boosterpack: http://wiki.seeedstudio.com/wiki/Grove_Base_BoosterPack#Us

You may find the first two documents under the "Resources" tab on BlackBoard.

Lab Submission

Before the submission, make sure that your source code has a good C programming style. For your reference, here are advices from the GNU Coding Standards: https://www.gnu.org/prep/standards/html_node/Writing-C.html.

Push the latest version of your code into your group's repository on GitHub. *You must do it before you may receive a grade for this lab*.

Your group should write a lab report that includes the following:

1. Describe the responsibility of each person in your group, and estimate the breakdown of contribution in percentage (e.g. 50% and 50% for a two-person group, or 33.3%, 33.3% and 33.3% for a three-person group).
   Note: If one contributes 40% or higher in a two-person group or 25% or higher in a three-person group, he or she may receive the same grade as other member(s) in the group. Otherwise, there will be a penalty, which will be decided by the teaching staff case by case.
2. Describe how you have revised the code in lab2_main.c to make the clock work.
3. How much flash memory does your program take? How much SRAM memory? You may find out this information in CCS under menu "View" => "Memory Allocation".

Submit your 1) revised lab2_**main.c**, 2) revised **seg7.c**, and 2) **lab report** on Blackboard under Lab 2 assignment.

GitHub Update and Extra Backup

Make sure that you have pushed the latest version of your code into GitHub. You will need to reuse the code in future labs. **As safeguard, make an extra copy of your code in a separate place, e.g. a thumb drive or an external hard drive.**