CMPE1250
Winter 2025
v1.1

ICA 0
Github Introduction

1 Introduction

This assignment is intended to introduce you to the basic functions of Git and GitHub.

Git is a version control system that is commonly used in software development. It will track changes made to the files in your software projects. This allows you to see differential changes and revert to a previous version if an update introduces a catastrophic bug. While Git offers many powerful features, we will focus on it's basic functionality.

GitHub is a very popular web service that uses git to provide a way for teams to collaborate on software projects. You will be using GitHub to submit your assignments in this class. Your instructor will also use it to provide feedback on your code.

By the end of this assignment, you should be able to:

- Clone an existing repository from GitHub to your computer.
- Create a new branch on a repository
- Commit changes to git
- Push changes to a remote repository
- Merge a development branch into a main branch
- Resolve simple merge conflicts
- Revert to previous commit

Git and GitHub are required tools for CMPE1250 but are not focuses of the class. As such, little class time will be devoted to teaching them. Online resources have been provided below to aid in this independent learning activity.

To complete this assignment, you will need an account on GitHub. You will also need to have accepted the assignment link that was sent in class after you have created a GitHub account.

2 Git and GitHub Resources

You may be able to follow along with the instructions in this document, but they do not explain how git is working. It is **strongly recommended** that you take the time to work through the Codecademy Git courses. The other resources below are provided in case you get stuck or have issues with git in the future.
Codecademy: Git Introduction
Codecademy: Git Branching and Collaboration
VS Code Git Documentation
Git Documentation
GitHub Cheat Sheet
GitHub Training Manual
Google

3 Assignment

Once you have gained some familiarity with branching and merging in git, work through the assignment below. You will be branching and merging repositories throughout the course, so it is important to understand these steps. You may use the command line, git tools within VS Code, or GitHub desktop to perform these tasks. The goal is for you to demonstrate that you are able to perform these tasks with a tool you understand.

3.1 Clone your repository

After you have accepted the GitHub assignment that your instructor provided to you, you will have a GitHub repo that matches the pattern:
`https://github.com/CMPE1250/home-1242-[SECTION]-[USERNAME]`
Clone this repository to your local computer. This is where you will do most of your work for the course, so make sure you save it somewhere convenient. All of your assignments will be done in this repository.

3.2 Create a new branch

When you start working on a new assignment, you will create a new branch on your repository. This will allow you to experiment with all of your code and libraries without needing to worry about affecting any of your existing code. Nothing you do in the new branch will change the main branch that your instructor is marking. Create a new branch now. Be sure to give it a name that describes what it is being used for. Something simple like ICA0 will be fine. If you are unsure how to create a branch, refer to the Git and GitHub Resources.

3.3 Modify, commit, and push the development branch

Now that you have a development branch, you can start working on the ICA. Create a new project in the \ICA directory called ICA0. Your instructor will walk you through starting a new project in Code Warrior, but the key options to watch out for are as follows.

Listing 1: CodeWarrior Project Settings

```
            Processor = HCS12X - MC9S12XDP512
            Connections = USBDM
            Memory Model = Small
            Floating Point = float is IEEE32, double is IEEE32
```

There are resources on Brightspace to show you how to start a new project if you need to review the process.

There is an example project in your InClassDemos repository. If you do not see it, update that repository with the directions provided by your instructor. Copy the main.c file from the demo project to the one you have just created. Don't worry about understanding all of the code at this point. If you were going to make your own project, you would want to update the title block at this point.

You should now have a CodeWarrior project with a main function that looks like the following.

Listing 2: Project main() function

```
void main(void)
{

  // main entry point
  _DISABLE_COP();
  //EnableInterrupts;

/*****************************************************/
  // one-time initializations
/*****************************************************/

PT1AD1 &= 0b00011111;   //Turn off the LEDs
DDR01AD1 = 0b11100000;  //Set LED pins to be outputs
ATD1DIEN = 0b00011111;  //Enable digital inputs on switches

/*****************************************************/
  // main program loop
/*****************************************************/

  for (;;)
  {
    static unsigned counter = 0;

    while (++counter); // Example of a blocking delay
    PT1AD1 ^= 0b10000000;

  }
}
```

Commit your changes in git and push the changes to GitHub. Again, if you are unsure of the commands to do this, refer to the Git and GitHub Resources. You can also use the CodeWarrior debugger to verify that the code runs on your MC9S12 board. When you're done, close CodeWarrior and commit your changes if there are any. **The CodeWarrior project files sometimes change as the project is closed.**

3.4 Merge branches

Switch back to the main branch of your project and merge in the ICA0 branch that you have been working on. You will need to close CodeWarrior before you can do this. Git cannot modify the .mcp project file to the state it was in the main branch if CodeWarrior has it open. You should not have any conflicts or issues at this point. Your ICA0 project is now part of the main branch on git.

3.5 Modify and commit on the main branch

You wouldn't normally make a commit directly to the main branch, but we're going to intentionally cause a conflict so you can try resolving one. Add a comment to the line `PT1AD1 ^= 0b10000000;`. Make your best guess about what that line of text does. Commit the change when you're done.

3.6 Modify ICA0 branch

Move back to the ICA0 branch. This time, modify the same line of code to be `PT1AD1 ^= 0b10100000;`. Commit the change with git.

3.7 Merge branches again

Move back to the main branch and try to merge again. The main branch now has a change that wasn't present in the ICA0 branch. That isn't good, but git would be able to deal with it if ICA0 didn't have a change on the same line. Git doesn't know which version of that line of code is the correct one. So it creates a merge conflict and makes us pick.

3.8 Resolve conflicts

Your main.c file should now have a chunk of code that looks something like the following.

Listing 3: A git merge conflict

```
<<<<<<< HEAD
PT1AD1 ^= 0b10000000; //What does this line do?
=======
PT1AD1 ^= 0b10100000;
>>>>>>> ICA0
```

The code between `<<<<<<< HEAD` and `=======` is the code on the current branch. The code between `=======` and `>>>>>>> ICA0` is the incoming code. To decide which code should be kept, simply remove the other. Make sure to delete all of the tags inserted by git as well. These are all of the `<, >, =, HEAD,` and `ICA0`. If working in Visual Studio Code, there may be an option to resolve the conflict in the conflict manager. This is a helpful tool that simplifies the process by providing several likely merge options to choose from.

3.9 Merge and delete development branch

Once the merge is complete, you can commit the changes on the main branch and push to GitHub. If you want to store the ICA0 branch on GitHub, you can switch to the ICA0 branch use the `git push origin ICA0` command. However, we are done with the ICA0 brach at this point. You can delete it as described in the Git and GitHub Resources. To remove the ICA0 branch from GitHub, you can either delete it through the website or with the `git push origin --delete ICA0` command.

3.10 Revert the repository to the original state

As a final exercise, we'll revert the main repository to a previous commit. You will understand the value of this if you have ever had a working programming assignment and then found that it was no longer working after attempting to add a new feature. Git allows you to simply revert the project to that previously working state. Use `git revert` to return to a previous state in this class. There is another command called `git reset` that has a similar function. However, git reset deletes commit history. This can cause conflicts and other issues that are very difficult to fix. It can also make evaluating your work very difficult as your instructors use the commit history to grade your work.

Revert the main branch to the point where you had just added the ICA0 to the main branch.

4 Conclusion

You probably had to look up some commands to get to this point and that's okay. Reference documents exist so you don't need to memorize every command of every tool that you use. You will end up memorizing the most common commands from use. If you know where to find the commands you haven't memorized, this assignment has served its purpose. You should now be well equipped to use git and GitHub in CMPE1250, your other courses, and possibly a workplace. Branching, merging, and resolving conflicts are some of the core operations within git.