Software Installation and Tutorials

BME554L (Fall 2025)

Invalid Date

Table of contents

Git (Version Control) 1
Installation & Configuration 1
Setup Duke GitLab Account 2
SSH Key Authentication 3
Git Tutorials 4
Visual Studio Code (IDE) 4
State Diagram Software 4
Zephyr SDK / nRF Connect / Nordic Dev Academy 5
Nordic Hardware to Buy 5
Technical Report Preparation 6
What to Submit 6

Git (Version Control)

Installation & Configuration

- Git - version control software
  - Note - MacOS usually comes with `git` pre-installed, so there is no need to install anything else.
- After installing `git`, please configure it: https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
- The setup steps above should create a `~/.gitconfig` file that has some barebones configuration options. We can expand that a bit by manually editing it with the following content:

```
[user]
    name = your first and last name
    email = yournetid@duke.edu
[alias]
    lg = log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C
[core]
    editor = nano
    autocrlf = input
    preloadindex = true
    fileMode = false
[color]
    status = auto
    branch = auto
    interactive = auto
    diff = auto
    ui = auto
[push]
    default = current
    autoSetupRemote = true
[pull]
    rebase = true
[init]
    defaultBranch = main
```

- The above configuration will:
  - Set your name and email address for commits
  - Set up a `git lg` alias that will show a nice graph of the commit history
  - Change the default editor for making commit messages from `vi` to `nano`. (Note, you can change this to something else.)
  - Set the default branch name to `main` (instead of the legacy `master` name). *It is important to do this for our grading scripts to work properly.*
- This is a good tutorial on using `git` within VS Code: Using Git source control in VS Code

Setup Duke GitLab Account

While GitHub is a very popular public git repository hosting site, we will be using Duke's GitLab server. Please make sure you can log into https://gitlab.oit.duke.edu using your Oath2 NetID authentication.

> ⚠ Warning
>
> This is **not** `gitlab.com`!

> 🔥 Caution
>
> Do **not** edit files in remote git repositories using the GitLab web interface!! Only work with your repository files using your local clone of the repository.

SSH Key Authentication

We will use SSH keys to authenticate us on GitLab to be able to `clone`/`push`/`pull` from GitLab without needing to always enter your username and password.

To setup an SSH key and add it to your GitLab profile:

> ⚠ Warning
>
> The guide below will reference `github.com`; you want to substitute `gitlab.oit.duke.edu`.

1. Generate a New SSH Key

    - While your private key is most secure by encrypting it with a passphrase, you will need to either enter that passphrase everytime the key is used, or you will need to add it to your local credential manager (i.e., keychain) or `ssh-agent` to keep it unlocked.

    - If you are just using this SSH key for this class, you can use an empty passphrase and not need to worry about needing to deal with the password management.

2. Add your SSH key to your GitLab Profile

> ⚠ Warning
>
> The guide above will reference `gitlab.com`; you want to substitute `gitlab.oit.duke.edu`.

Git Tutorials

We are going to cover `git` extensively in class, but it can really help to review some tutorials ahead of time to get familiar with the nomenclature and high-level overview of the workflow. If you have never used `git` before, I would recommend reviewing the following tutorials:

- Git Tutorial for Beginners: Learn Git in 1 Hour
- Backing Up and Sharing Code: Git

More comprehensive `git` documentation can be found at https://git-scm.com/doc.

Visual Studio Code (IDE)

We will be using Visual Studio Code as the IDE for all projects in this class. In addition to installing the base program, please install the following Extensions:

- `nRF Connect for VS Code (Extension Pack)`
- `GitLens` (help with git operations / visualization)
- `C/C++` (not the extension pack, since it include `CMake Tools` that has some incompatibilities with the Zephyr build tools)
- `Cmake`
- Microsoft `IntelliCode` (assistance with C syntax)
- State diagram software (see below)
- GitHub CoPilot (AI-assisted coding)
  - Sign up for free GitHub Education Student account to get access to this extension: https://github.com/education/students
  - You will need a document to verify your student status that includes your dates of enrollment, which your DukeCard does not contain. Instead, you can get an Enrollment Verification document through DukeHub (Academics Tab) and convert the downloaded PDF to a JPG or PNG file to upload to GitHub.
  - Install the GitHub CoPilot extension

State Diagram Software

We will be generating state diagrams all semester, which can be done with a variety of different software packages:

- PlantUML
  - Text-based diagramming tool that can be integrated into VS Code
  - A bit outdated, and a tad clunky in default layouts without some "nudging"
  - What will be used in lecture and lab assignments

- Mermaid Markdown
  - Markdown-based diagramming tool that can be integrated into VS Code
  - More modern and easier to use than PlantUML
  - Not as feature-rich as PlantUML
- draw.io
  - Web-based diagramming tool that can be integrated into VS Code (Draw.io Integration)
  - Very feature-rich and easy to use
  - Not text-based
- Lucidchart
  - Web-based diagramming tool
  - Very feature-rich and easy to use
  - Not text-based (less VCS friendly)

You can use whatever software package you prefer.

Zephyr SDK / nRF Connect / Nordic Dev Academy

Please complete the following tutorial to get your environment setup for using the Nordic nRF Connect SDK: Lesson 1 - nRF Connect SDK Introduction This will run you through the process of:

- Signing up for an account on the Nordic Semiconductor DevAcademy
- Installing the nRF Connect for VS Code extension pack.
- Installing all of the associated `nRF Connect` tools for your laptop, including the `nRF Connect SDK` and associated `toolchain`.
  - This should be done via VS Code, not the nRF Desktop application.
  - Be sure to use the `nRF Connect SDK v2.9.0` for this class.
- Flashing the `zephyr/samples/basic/blinky` to your `nRF52833DK`.
  - You will need a data-capable USB cable to connect your `nRF52833DK` to your laptop.
  - Make sure you can successfully do this, as it will test your USB cable connection, USB serial port device permissions and your build/flash environment.

Nordic Hardware to Buy

- **Order a nRF52833DK**.
- Make sure you have the appropriate USB cable to flash the device from your laptop.
  - This may require a USB adapter for some laptops.**

> **!** Important
>
> Make sure that your USB cable supports data transfer and is not just a charging cable.

Technical Report Preparation

Each lab exercise will have an associated technical report submitted. These reports will be prepared and submitted using Python-based Jupyter notebooks that will be included as part of your assignment git repositories.

You will need to be able to perform the following tasks in your Jupyter notebooks:

- Read CSV text data saved from an oscilloscope.
- Perform signal processing (e.g., FFT, filtering, etc.) on the data.
- Perform simple statistics (e.g., mean, standard deviation, 95% CI) on the data.
- Generate plots

An example technical report can be found here.

If you need to install a Python environment on your laptop, then this is a good starting point: Getting Started with Python in VS Code.

This is a good tutorial on getting started with Jupyter notebooks in VS Code: Jupyter Notebooks in Visual Studio Code.

What to Submit

Complete the online Gradescope "quiz" indicating completion of each of the main tasks above.