**Description and Purpose**

This lab introduces you to Arduino programming and design, as well as source control. Basically, this is the full cycle you will use if you want to develop an Arduino solution in a professional way and even it over YouTube if you want.

You will use the Fritzing application to design your application before even starting. You will develop a small fritzing technical diagram as a base.

Then, you will use the standard “Arduino” application which gives you an integrated programming environment for the Arduino platform (also made by Arduino).

Finally, you will save your Arduino code to GIT, using classical command-line commands.

This lab will hold more weight than the other labs you have done. Basically, this lab counts for the value of two labs due to the amount of work needed. Also, you will have more time than usual to complete this lab.

**Learning Objective(s)**

1. Use Fritzing to make a small diagram
2. Implement an interesting project using sensors and/or actuators
3. Implement GIT to track your code changes.

**Procedure**

**PART 1 – Decide on what you will build**

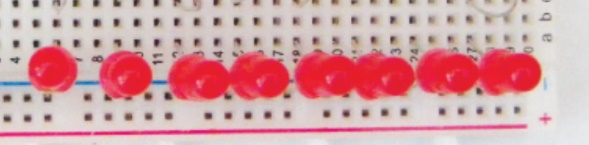
In class, we saw a way to read analog data from different devices with the Arduino, also how to light LED’s. Also, by the time you read this, you should see at least one way to interact with things like DIGITS (arrays of LED’s).

The application should read in values from a sensor (of your choice) in order to do something with this data (read on). The data being read should be ANALOG, meaning that it comes in as a value from 0 to 1023 (Arduino Uno).

Common analog choices are: Variable resistor, Photoresistor, Resistor(s), Thermistor, Sound level meter

The application should then do something with the data involving something more complex than just 1 LED (RGB LEDs, arrays of LEDs, a digit, a LED 2D array).

As a minimum, here are some suggestions.

1. A series of LED’s which react to sounds. (eg: a sound level meter)
2. An RGB LED which changes colour depending on the level of light or sound. Perhaps switching to RED when the noise level is too high.
3. Anything else **OF YOUR CHOICE** that demonstrates analog INPUT and output consisting of MORE than just one LED. If you choose this, I would appreciate you get my OK for this (during CLASS not over email).
4. Anything built which shows extreme creativity above and beyond – WILL gain 1-2 bonus points. This bonus allocation is at the discretion of the instructor and not arguable 😊. Creative projects generally are projects you can’t easily find as “tutorials” already on the web.

**PART 2 – Fritzing design**

Now that you’ve decided on what you want to build, use Fritzing to illustrate it. If you’re not comfortable drawing the project before building it – you can build it first and debug it – then you can diagram it later – your choice.

Use Fritzing to build the closest representation you can of your project. Make it so that someone else can easily build the project from scratch with your prototype’s design. Try to illustrate it as much as you can and show notes and comments on it (your instructor needs to learn also how to make better diagrams). Paste your diagram in the area below and paste your own comments and notes about the diagram under it – anything you found interesting about it, or any special things I need to know that were “strange” in the build, etc. If you feel your project merits that “above and beyond” special quality, here’s where you mention that also.

Paste the diagram here.

|  |
| --- |
|  |

Your notes can be included here.

|  |
| --- |
| I find it interesting how the value of the photoresistor changes rapidly as the light gets dimmer/brighter. Also only one analog input is needed for the whole project. |

**PART 3: Setting up the Arduino hardware project**

Set up the Arduino project yourself and implement it (build it). I would like to see a video of it once it’s done – which should be uploaded in the ZIP file you create (see submission instructions).

**PART 4: Using GIT**

Since the Arduino environment does not include a built-in GIT management system, you’ll have to manage it manually. We will use the command-line to control our GIT repositories. It sounds more complicated than it is, it’s a very simple process. Note: This works for Windows and Linux, but I cannot confirm it will work via the Mac terminal (but I can’t see why not).

**Step 1**

Choose how you’d like to save your projects (the structure of your repository). Your repository structure is up to you.

Choose a folder on your computer where you store your code – which you probably already have.

**Step 2**

In a windows CMD or Terminal: Find your code folder created above. The easiest way to do this, is via the Arduino IDE by selecting File-Preferences and observing the “Sketchbook Location” textbox value.

Make note of the location, because you’ll use it in the following steps.

On a typical Windows machine, your location might look something like: C:\Users\{yourname}\Documents\Arduino.

If you want to change the location to somewhere else (like a D: drive) now is the time to do it. I personally keep everything on a drive separate from the operating system in case the OS is damaged.

Create a directory for your project and save your .ino file in that directory.

Add any files you might want to include there (it might be nice to include a fritzing file, a howto, or anything else you might think helps.

**Step 3a**

If you don’t have it installed, install the “GIT” version control system on your computer. To check if you have GIT installed, type

git --version

If you get something back something roughly similar to this, you are OK and you can skip the next step.

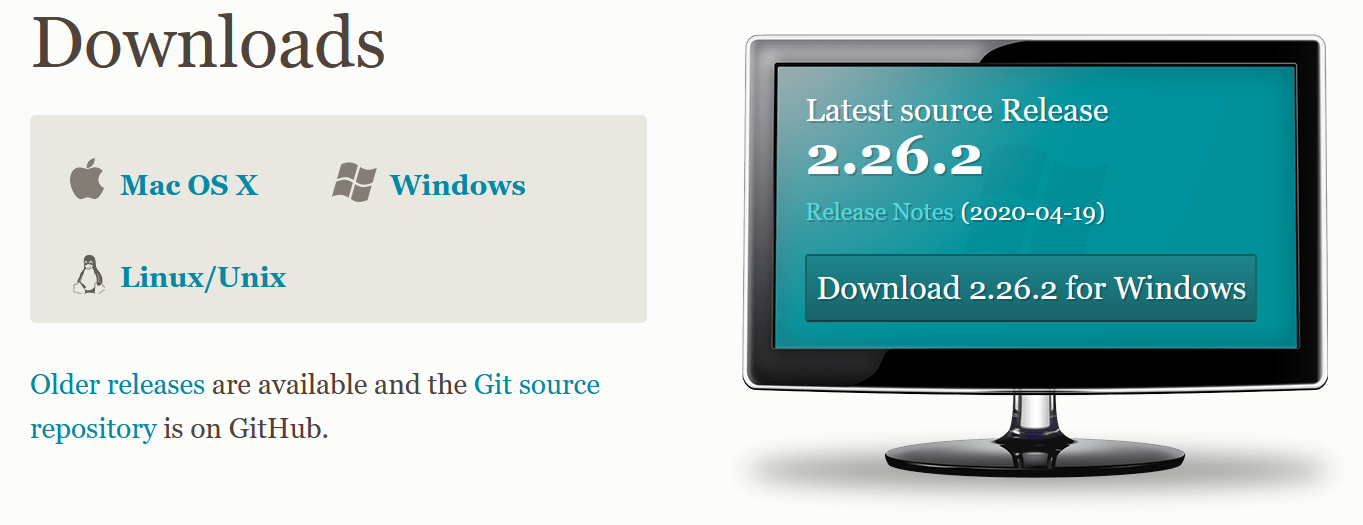
Git version 2.26.2.windows.1

To install GIT for **Linux**, you need to just do:

* sudo apt update
* sudo apt install git

For **WINDOWS**, read on:

To install GIT, you must download it from: <https://git-scm.com/downloads>



The installer will run and ask many questions (this installer is the most complicated installer I’ve ever seen). But the good news is you can click “next, next, next” for ALL the questions.

**Step 3b**

To create a git repository of your project:

Open a command line and navigate inside your Arduino directory – then inside your project director. Look at the example below to make sure you’ve navigated to the right place.

Type: git init

C:\Users\bwood\Documents\Arduino\Lab4>git init

Initialized empty Git repository in C:/Users/bwood/Documents/Arduino/Lab4/.get/

Type: **git add .** (don’t forget the period here, meaning “add the current directory to git”.

Don’t worry if you get a whole bunch of errors saying something like “LF will be replaced by CRLF”. GIT is just trying to make the code more standard.

Go into GitHub (assuming you have your account created already), and **click on New to create a new repository**. Give it a meaningful name (think about if you see this in 10 years from now, will the name make sense?). Also try not to be funny or crass with the name, because maybe you can show this to interviewers in case you need to show your projects in an interview.

Create a reference to your remote github repository: Change the git link below to the one YOU made.

git remote add origin **https://github.com/bwood-crc/arduino-example.git**

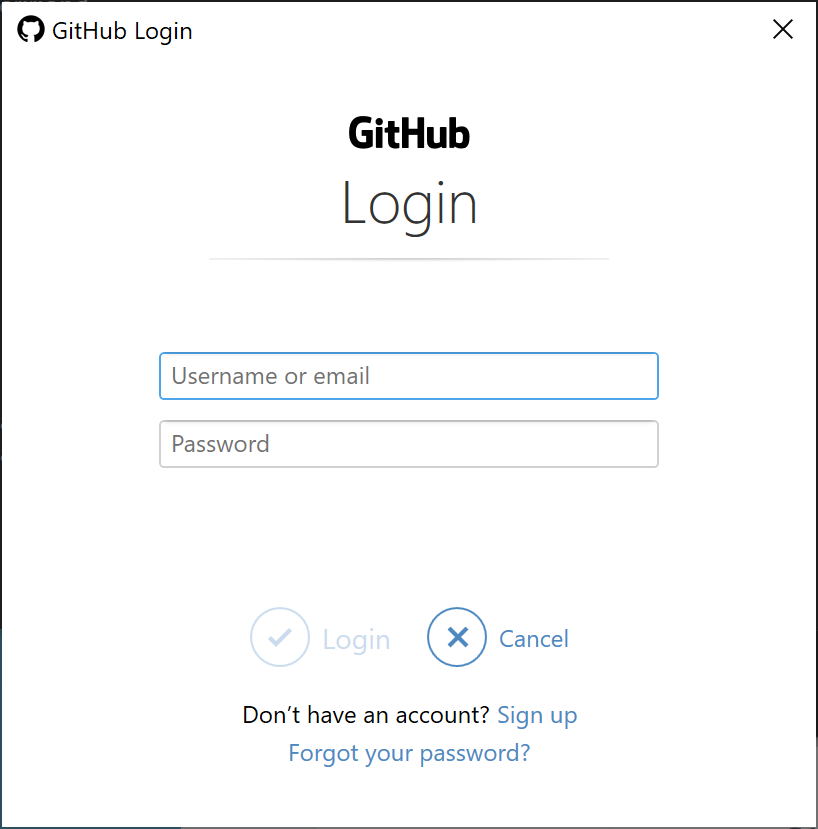
Commit your code

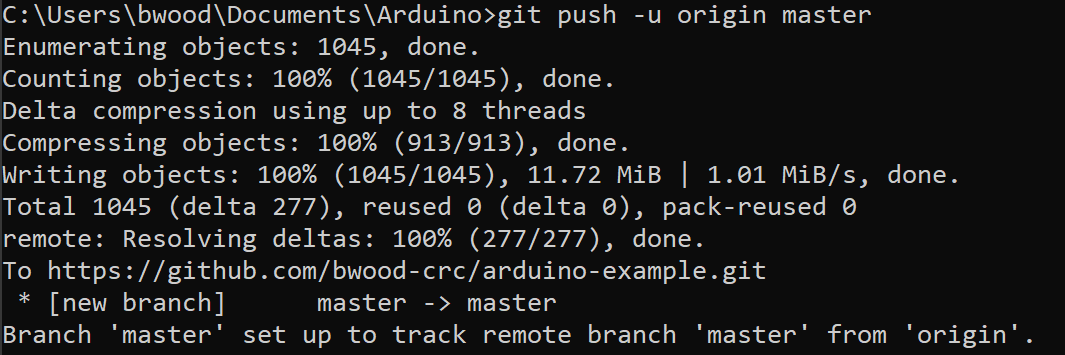
git commit -m “Initial Commit”

Finally, push your local repository to the remote.

git push -u origin master

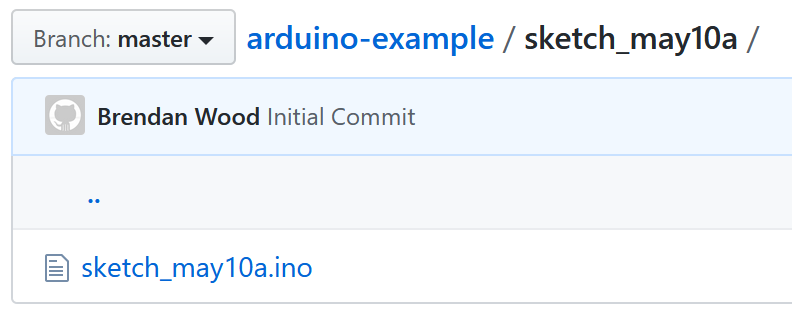
Fill in username, password if asked. (On linux it will just ask you via the console without any windows like below). You might get a login window, fill in our github details.





Go to Github to make sure the code shows up there!

Your sketches should show “ino” as their extensions.



Making a Change to your Code and Updating GIT

1. Make a change to a file in one of the sketches you have (create a comment if necessary).
2. To show if any changes are detected, type: git status
3. To commit changes, you need to re-add the changes again: git add .
4. To commit your changes, type: git commit -m “change description here”
5. Then finally, push your changes to the remote: push -u origin master

Then repeat this process every time you make changes, at the end of your programming sessions. If you plan to continue from home to school – *don’t forget to commit and push changes because you will not have them available.*

**Resources, Reference, Please See**

**Arduino IDE**

<https://www.arduino.cc/en/Main/Software>

**GIT**

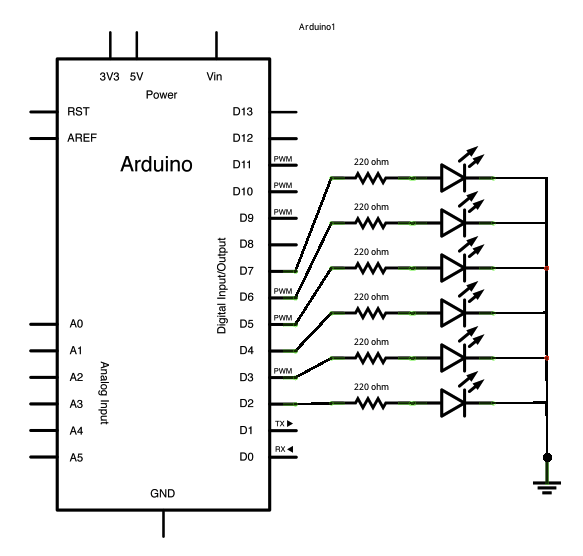
<https://git-scm.com/downloads>

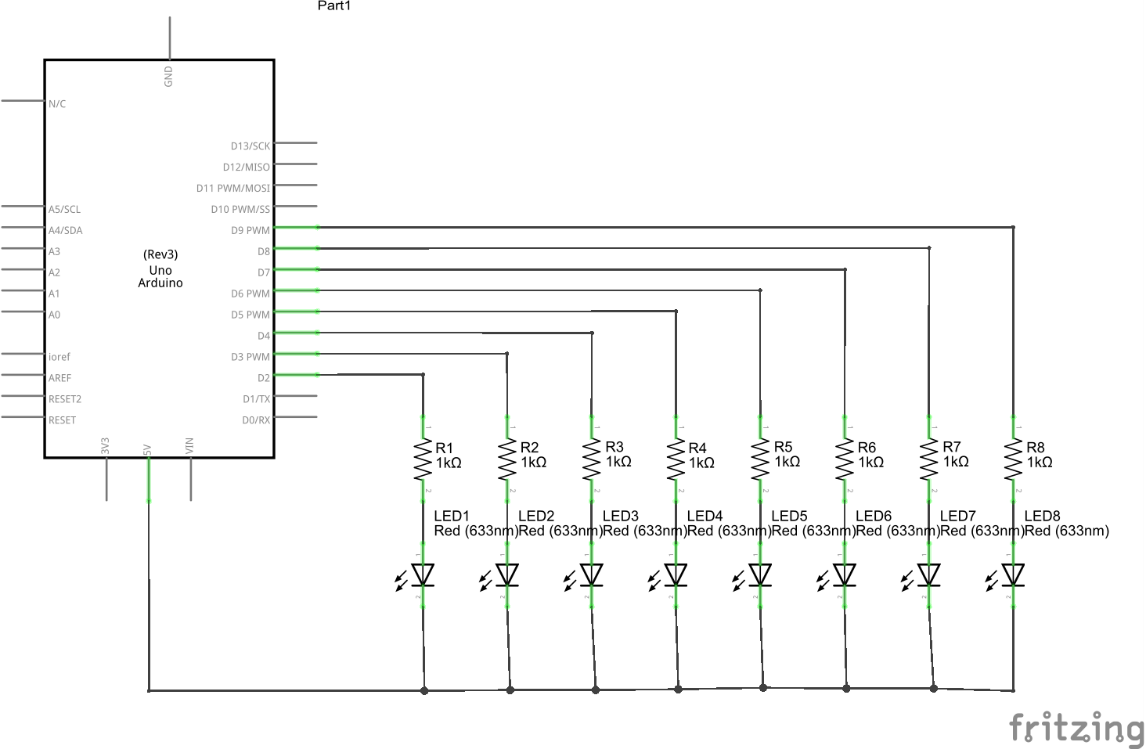
**GIT commands overview PDF**

<https://training.github.com/downloads/github-git-cheat-sheet.pdf>

**Multi-LED Setup**

How to connect a series of LED’s to Arduino (Schematics) – Experiment with your resistor values to get the best brightness desired, use the formula if necessary based on 5V.





**Submission Procedure**

To submit your work, ZIP the following items, and upload the zipped file to Moodle before the deadline.

1. A screenshot (using snipping tool is fine) of your Fritzing project.
2. Your fritzing file.
3. A SMALL video of your project working (remember there is a limit of 500MB for the zip file. It should show all the basic operation of your project (like a small movie trailer – to the point).
4. A GIT link showing the code that drives this project. Make sure the project is public, not private.

Submissions will only be accepted via Moodle (eg: not Mio). Moodle is giving us a hard time this semester, so you may have to try to submit a few times. If you really find it’s impossible, please let me know, and also, submit a ticket to [infocenter@crcmail.net](mailto:infocenter@crcmail.net) to encourage them to fix Moodle.

**Grading Scheme / Rubric**

The project will be graded on 10+2.

Submitting everything, done as instructed guarantees you 6 points.

2 points are given for quality of coding.

2 points are given for the creativity of the project.

2 bonus points are given for going above and beyond in your project and including interesting things that can be shown to the class.