LAB 3

TEC 284

C PROGRAMMING FUNDAMENTALS II

OBJECTIVES:

* Download and install the Arduino IDE
* Familiarize yourself with the Arduino IDE
* Upload a program to an Arduino board

SUPPLIES:

1x Grove Beginner Board

INTRODUCTION:

*With an understanding of the serial monitor, we are going to write some code that relays information back to you for debugging purposes. Including simple lines like this, even as we get into more advanced programming, can be a helpful step in ensuring your program functions properly. It essentially acts as a simple “check” to verify your program is running the way that it should.*

GITHUB SETUP

1. Using either the web or desktop interface, create a new repo using GitHub. Be sure to add a descriptive README file!
2. Change your Arduino sketch location appropriately.

INTRODUCING THE LIGHT SENSOR

1. For the sake of simplicity, we are only going to be writing code in the setup() function for this lab!
2. There is a light sensor on your Grove Beginner Board located in the top right. It is internally connected to the Arduino’s A6 analog pin. Note that analog pins start with the letter “A” in the IDE, whereas digital pins are just the number. This can be confusing, because the Grove Beginner Board denotes digital pins as “D”.
   1. For example, the LED is written as “D4” on your board, but in the Arduino IDE, you would refer to it simply as “4”.
3. The Arduino Uno (what you’re working on now) has a 10-bit Analog to Digital Converter, meaning that it displays an analog value as a number between 0 – 1023. 1023 is the maximum value in this case.
4. Write the following code within your setup() loop:

A black screen with white text

Description automatically generated

1. Open the Serial Monitor by clicking on its icon in the top right of your IDE.
2. These lines of code should display a number between 0 – 1023 in your serial monitor. What is your number?

88

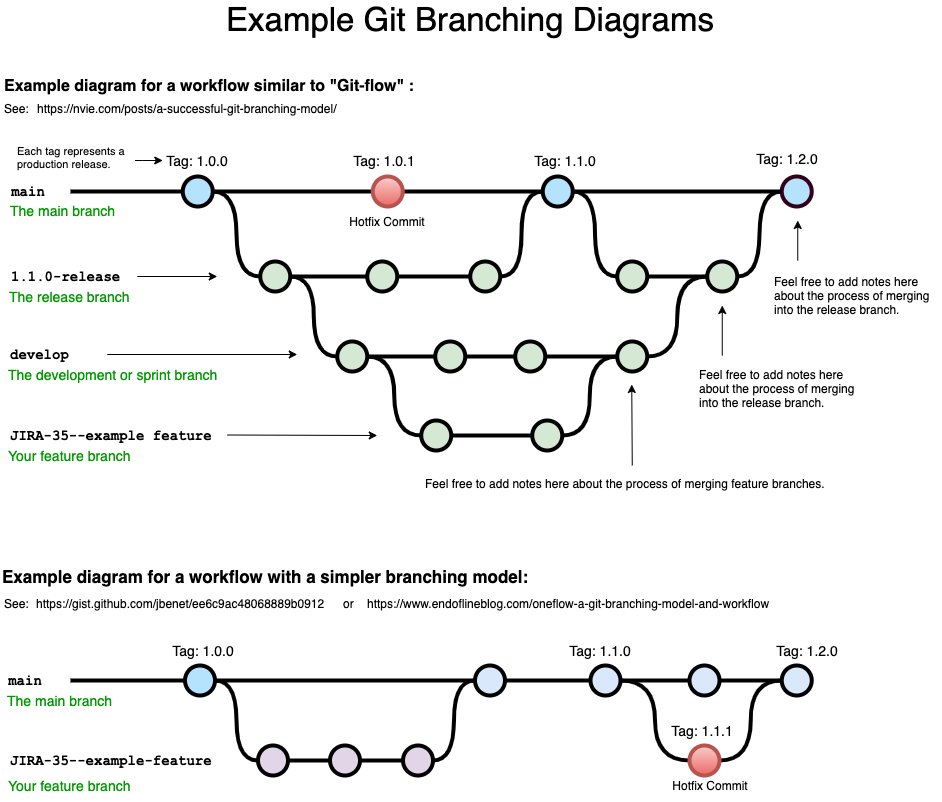
1. This number should correlate with the amount of light that is picked up by the little transparent bulb on your board. Try covering it with your hand, resetting the Arduino (there’s a reset button underneath the yellow header pins in the center-right of the board), and seeing how that affects your value.

CONDITIONAL STATEMENT

1. Now that you have a value, and you can see it in your serial monitor, let’s set up a conditional statement for the Arduino to say different things depending on the light level.
   1. If lightLevel is less than 100, have the Arduino write “It’s really dark!”
   2. If lightLevel is less than 200, have the Arduino write "It’s dim in here”
   3. If lightLevel is less than 700, have the Arduino write "It’s a little bright”
   4. If lightLevel is less than 1024, have the Arduino write "It’s really bright!”
2. Test your conditional statement by moving your hand over the light sensor.

BRANCHING OUT WITH GITHUB

1. So far, we have just been committing our changes to the *main* or *master* branch of your Git repository. This is almost never done in the real world, as the main branch is supposed to be for your “finished” product. This isn’t always the case, but it’s a common approach to development.
2. Git makes use of branches, which are basically isolated copies of the main branch. So, you might have a branch of this code here to, say, develop the lightLevel IF statement above. This allows you to develop this code and commit changes without affecting the “master copy” so to speak. When your branch code is bug-free and tested, you can merge the branch code into the main copy.
3. You can see an example of this in the diagram below:



1. Let’s try this out for ourselves. I’ll be showing this through the web version of GitHub, but this is possible with the desktop version as well.
2. Open GitHub to your repository’s summary page. Click on the “branch” button:



1. A new screen will pop up. Click on “New Branch” in the top right corner of the webpage. Name this branch “production” for now.
2. Notice that when you created this branch, you could choose which branch to copy from. You can make branches of branches, if you so desire! Back on the main page, we now have two branches to work from:

A screenshot of a computer

Description automatically generated

1. Notice that “production” is a direct replica of “main”. It has all of your code commits to it. If you make any changes to your “main” branch, you will see a message that your “production” branch is “X commits behind main”, where X is the number of commits.
   1. If you want to test this, you can actually edit your Arduino code right inside of GitHub. You won’t have the same functionality as in the Arduino IDE, but it’s possible!
2. Continue on with the lab for now, making sure that you are working within the “production” branch going forward!

DIGITAL OUTPUTS

1. The last part of our program will be to turn on the LED (a *digital* component) located at pin 4 *only when* the lightLevel is less than 100.
2. For this to work, you may need to set the pinMode of the LED to be an output. Remember that your LED is located at pin 4.
3. Make sure to turn the LED off when the program restarts!
4. When you are finished, commit your changes to the “production” branch.
5. Notice that when you commit your changes, you will be asked if you want to make a “pull request”. This is the process of merging changes in one branch to another. Go ahead and go through this process to merge your production changes with the main branch. You will need to comment on your changes. This typically gets sent to the project manager, but that’s you in this case!
6. While you certainly do not need to do this in this class, branches are a very powerful way to divvy up work to different collaborators or to work on code without fear of ruining the master copy. It’s a good thing to learn!
7. Once your main branch is all set and updated, submit this lab sheet to Canvas. Include a link to this GitHub repo as a comment on your submission.