Tutorial 3: Commenting Your Code

When programming, it is always good practice to write your code in a way that is easy to understand.

Use descriptive variable names and keep the program as simple as possible

For example, use variable names that make sense. When you use names like:

`var1`

It doesnt help anyone understand what that variable represents. Instead, name is something like:

`LEDPinVal1`

Thats much more descriptive. It tells us that it controls some sort of LED, and probably represents a pin value. Likewise, if you want to write a function that calculates the square root of a number, call it:

`int squareRoot(int num)`

Instead of something generic like `function1(int num)`

You should also keep the program as simple as possible. Dont use complicated logic to achieve a simple task. Anyone who has to look at the code and/or modify it later might have a difficult time understanding. If you find that a piece of code is starting to get too complicated, see if you can break it down into smaller pieces - create functions for small, simple tasks that you are using more than once.

Sometimes though, to help any future readers (yourself, your project partners, your TAs) understand exactly what your code is doing, you will want to add comments.

How to add comments?

To add a comment in the Arduino language, simply type:

```
// your comment goes here
```

Anything that appears after the `//`will be a comment and won't be run by the Arduino. You can also add large blocks of comments using:

```
/* your comment goes here
* more comments
* even more comments
*/
```

What are comments used for?

Comments help explain bits and pieces of your code. Unless something is extremely obvious, you should add a comment to explain what each line of code is doing. It is helpful to think in terms of your audience: what will help someone who has never seen my code understand what is going on. Other people cant read your mind, and might not be able to understand your logic. When in doubt, always add a comment.

This project will probably last a couple weeks. When you go back to look at your code you wrote 2 weeks ago, there is a good chance you wont remember why you multiplied that one variable by 3. When you have to debug code (and you probably will), comments are immensely helpful.

Another thing comments are good for is to leave notes for yourself or future readers. Perhaps you have yet to implement part of a function; a comment is a good way to tell someone something is missing. Or you can use it to tell your project partners about any bugs that havent been resolved, certain standard to follow, etc. You can use headers like TODO:, BUGS:, so others can quickly identify what each comment is roughly going to say.

You might also want to comment out chunks of your code at times to see how it runs with something missing. This could come in handy when trying to locate a problem in your code.

What makes a good comment?

Just like your variable names, your comments should be descriptive. You should be explaining at a higher level of abstraction than the code. For example, you have the bit of code below:

`currTemp`= (adcVal/1024)*TempScale;

Instead of saying `// multiply`by tempScale and divide by 1024, which doesnt give us any more information than what is already apparent in the code, say something like // convert 10-bit ADC reading to Celsius. This tells us what the code is doing on a conceptual value: it explains why we are dividing by 1024, and tells us the units of the final temperature.

Comments should be as short as possible while still giving all the necessary information. Remember, you dont have to comment every line, but things that you suspect might be any less than obvious should definitely be commented.

When you are writing functions, you should give a brief summary of what inputs the function takes, what it does to those inputs, and what outputs it gives.

We wont grade you on the quality of your comments, but having good comments will make it easier for you to work on your project, allow the TAs (who will be grading your projects) to more easily help you, and is good practice in general.