Code Documentation

Overview

Code commenting is a crucial habit to develop when writing code as it greatly enhances readability, maintainability, and collaboration. To take this class you will have taken other programming courses that also likely had some level of requirements around properly documenting your code, but due to the complexity of low-level hardware/software interaction, the requirements for regulatory compliance, and the collaborative and multidisciplinary nature of embedded systems, commenting your code is perhaps even more essential.

At some point in the quarter you may come to find documentation for a part, a tool, a library, etc that is either lacking, difficult to parse, or frustratingly disorganized, so to avoid contributing to that problem we will have guidelines that we'd like you to follow for all code you turn in.

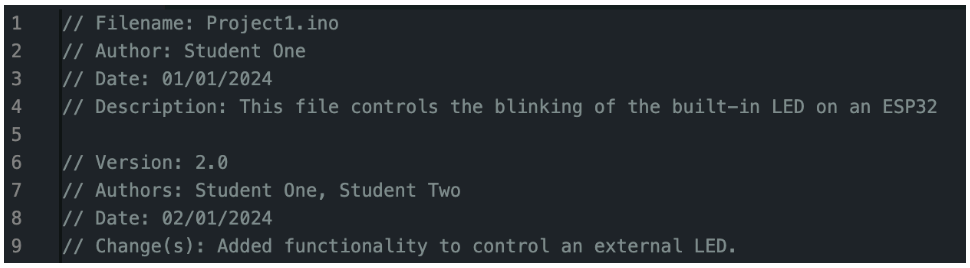
General Rules

- Comments, variable names, or function names should improve readability and clarity, not introduce confusion or clutter.
- Comments should *add value*, not restate the obvious.
- If you had to struggle to find the solution, thoroughly explain it so others won't have to.
- We *strongly recommend* that all code you turn in is developed and written by you, but it is essential that you learn to utilize all resources available to you as an engineer (Google, GitHub, StackOverflow, ChatGPT, etc).
    - If you use or are inspired by code someone (or something) else wrote, you ***must*** link to the source and acknowledge the author. Academic and professional honesty applies to code just as much as any other written work.
    - Even if you use someone else's code, you must still indicate you thoroughly understand what that code is doing through comments.

Requirements

Header Comments

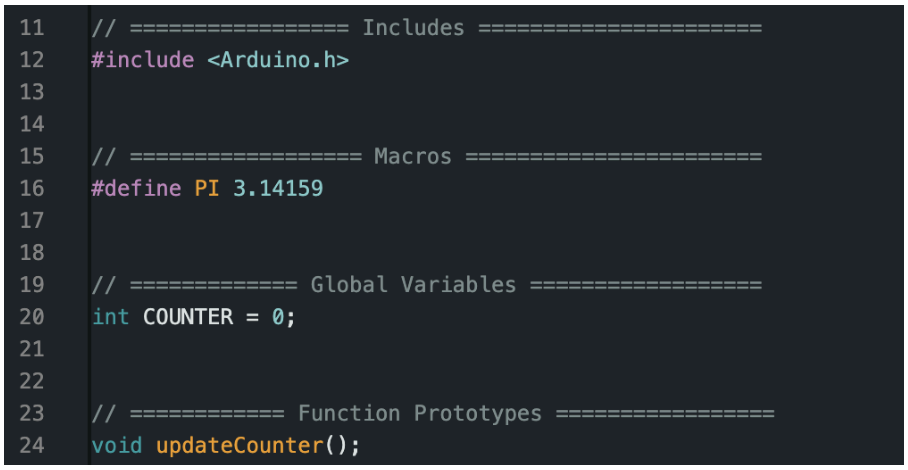
- At the top of every file please include the file name, authors(s), date, and a brief description of the file.
- Any additional iterations on a file should include a version number, author(s), date of update, and brief description of changes made.



```
// Filename: Project1.ino
// Author: Student One
// Date: 01/01/2024
// Description: This file controls the blinking of the built-in LED on an ESP32

// Version: 2.0
// Authors: Student One, Student Two
// Date: 02/01/2024
// Change(s): Added functionality to control an external LED.
```

Divide Sections

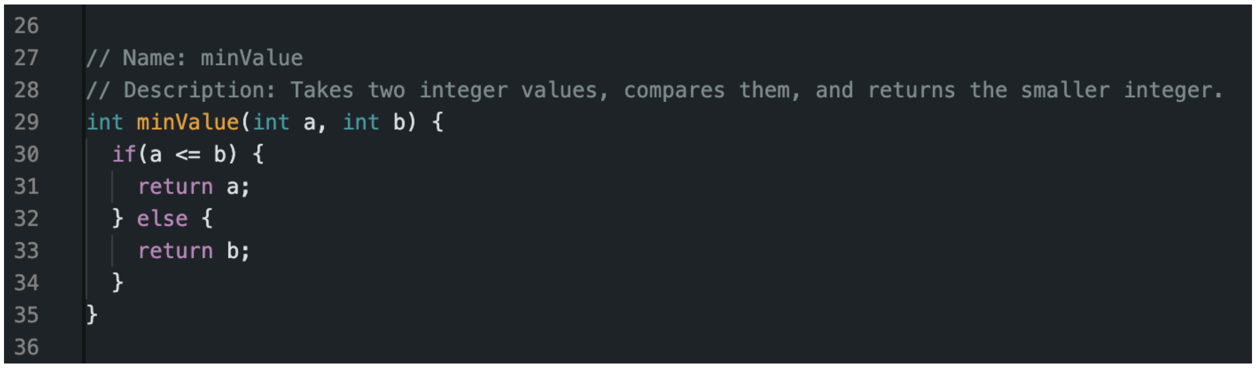
- Separate out the includes, macros, global variables/constants, function prototypes, and function implementations.



```
// ================= Includes =======================
#include <Arduino.h>


// ================== Macros ========================
#define PI 3.14159


// ============= Global Variables ===================
int COUNTER = 0;


// ============ Function Prototypes =================
void updateCounter();
```

Describe Functions

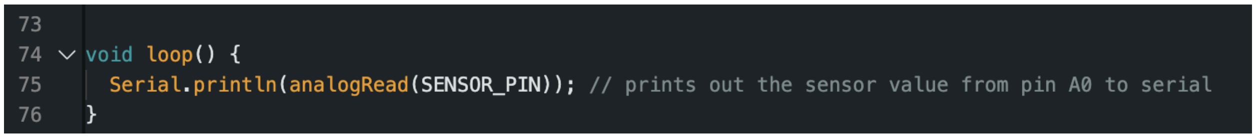
- Every function (outside of setup and loop) should have a description of use, parameters, and return values.



```

// Name: minValue
// Description: Takes two integer values, compares them, and returns the smaller integer.
int minValue(int a, int b) {
  if(a <= b) {
    return a;
  } else {
    return b;
  }
}

```

Inline comments

- If it’s anything more complex than “incredibly obvious”, explain it.



```

void loop() {
  Serial.println(analogRead(SENSOR_PIN)); // prints out the sensor value from pin A0 to serial
}
```