

IoT Bootcamp Style Guide

Brian Rashap, Ph.D.

September 30, 2021

1 IoT Style Guide

A programming style guide is an opinionated guide of programming conventions, style, and best practices for a team or project. Some teams call it their coding guidelines, coding standards, or coding conventions. While these each have their own meaning in programming, they generally refer to the same thing.

In IoT, we will use the below conventions:

- Start all code with the standard comment block

```
1  /*
2   * Project:
3   * Description:
4   * Author:
5   * Date:
6   */
```

- Provide meaningful names for variables, constants and functions. Variables and constants should be nouns and functions should be verbs.

- The one exception to variables being meaningful nouns is that integer variables i, j, and k may be used for indexing arrays or FOR loops.
- Constants should be in all capitals.

```
1  const int LEDPIN = 13;
```

- Variables and functions should be in camelCase¹, with the first letter lowercase: float elapsedTime;

```
1  float elapsedTime;
```

- Structs, ENUMs, and classes should also be in camelCase, but with the first letter capitalized.

```
1  enum TrafficState{
2      GREEN,
3      YELLOW,
4      RED,
5      REDYELLOW;
6  }
```

- Code should be organized as follows:

1. #include <>
2. #include ""
3. #define

¹Camel case is the practice of writing phrases without spaces or punctuation, indicating the separation of words with a single capitalized letter.

4. Data Types (e.g., structures and ENUMs)
5. Globals
6. void setup()
7. void loop()
8. user defined functions

NOTE: const variable should be used instead of #define.

- Avoid global variables where they are unnecessary.
 - Global Variables - The variables declared outside any function are called global variables. They are not limited to any function. Any function can access and modify global variables. Global variables are automatically initialized to 0 at the time of declaration. Global variables are generally written in the header-section of the code before void setup().
 - Local Variables - Local variables are declared inside of a function and only exist during that function call and are not accessible by other functions. Subsequent function calls reset the variable to its initial condition, which is 0 unless otherwise declared. Note: variables defined in void setup() or void loop() are local variables not accessible by other functions, and in the case of void loop() reinitialize during each iteration. Variables uses in void setup() and void loop() should therefore be declared as global variables.
 - Static Variables - A static variable is a local variable that is able to retain its value between different function calls. A static variable is only initialized once. If it is not initialized, then it is automatically initialized to 0.
- Opening braces should be on the same line as the conditional or function.

```
1 if(buttonState) {
2   digitalWrite(LEDPIN, HIGH);
3 }
4 else {
5   digitalWrite(LEDPIN, LOW);
6 }
```

- Code blocks should be indented.

```
1 for(i=0; i < 13; i++) {
2   value = i * 3;
3   if(num%2 == 0) {
4     Serial.printf("The number %i is even\n",value);
5     digitalWrite(LEDPIN, HIGH);
6   }
7   else {
8     Serial.printf("The number %i is odd",value);
9     digitalWrite(LEDPIN, LOW);
10  }
11 }
```

- Use braces; avoid loops and conditionals without them.

```
1 // Proper use of braces:
2 if(buttonState) {
3   digitalWrite(LEDPIN, HIGH);
4 }
5
6 // Avoid loops and conditions without braces. Do not do:
7 if(buttonState)
8   digitalWrite(LEDPIN, HIGH);
9 //or
10 if(buttonState) digitalWrite(LEDPIN, HIGH);
```

- Use parentheses for clarity, especially with bit operations.

```
1 pixelColor = (red << 16) | (green << 8) | (blue);
2 //or
3 if((currentTime - lastTime) < timerSetting) {
4   Serial.printf("Time is up \n");
5 }
```

- When "printing" text and values to the Serial Monitor, OLED, or UART devices, use `.printf()` instead of `.print()` or `.println()`

```
1 Serial.printf("We use .printf() to display value = %0.2f \n",value);
2 Serial.println("We do not use .print() or .println()");
```

- Avoid code duplication. (i.e., appropriately use functions)
 - One of the main reasons to use a function is reusability. Once a function is defined, it can be used over and over and over again. You can invoke the same function many times in your program, which saves you work. Imagine what programming would be like if you had to teach the computer about sines every time you needed to find the sine of an angle! You'd never get your program finished!
 - Another aspect of reusability is that a single function can be used in several different (and separate) programs. When you need to write a new program, you can go back to your old programs, find the functions you need, and reuse those functions in your new program. You can also reuse functions that somebody else has written for you, such as the sine and cosine functions.
- Place a descriptive comment the line before all functions. Use comments in code as needed to clarify code logic.

```
1 // Get x-axis acceleration from MPU6050
2 float getAccX() {
3     byte accelHigh, accelLow;
4     int16_t accel;
5
6     // Set the "pointer" to the 0x3B memory location
7     // of the MPU and wait for data
8     Wire.beginTransmission(MPU_ADDR);
9     Wire.write(0x3B); // starting with register 0x3B
10    Wire.endTransmission(false); // keep active.
11
12    // Request and then read 2 bytes
13    Wire.requestFrom(MPU_ADDR, 2, true);
14    accelHigh = Wire.read(); // x accel MSB
15    accelLow = Wire.read(); // x accel LSB
16
17    accel = accelHigh << 8 | accelLow;
18    return accel;
19 }
```

2 Lab Notebook

It is critically important, and often underemphasized, to keep good documentation of your projects. The above style guide assists in the readability of code. We will also keep schematics of your circuits in a program called Fritzting. The third aspect of documentation is your lab notebook. In addition to keeping notes from class lectures, each assignment will be documented in your lab notebooks.

It is expected that your assignment documentation contains the following:

- Description of assignment objective.
 - You do not need to copy the assignment verbatim; rather, you should restate the assignment in your own words.
 - Be descriptive enough so that when you come back to your notes later you can recall what you were trying to accomplish.
- Hand drawn schematic of your circuit layout
 - This is an opportunity to think about the proper connections and interface types between the components and your microcontroller.

- Be sure to include the proper current limiting resistors, pull-down or pull-up circuits, and noise filters ²
- The schematic should be detailed enough that you can create your fritzing diagrams and/or wire your circuit directly.
- Flow charts and/or outlines of your code logic.
 - Flow charts are written in english³, not in C++ code.
 - Flow charts need not outline the entire code, but rather the conditions and flow control features that are integral to the assignment. For example, the flow chart in Figure 1 outlines how to publish data to the cloud at a set interval without using a delay.
 - Add comments (notes) to your flow chart to assist with readability and highlight important features, as necessary.
 - In addition to flow charts, use your lab notebook to outline any finite state machines and state transitions. An example is the traffic light states seen in Figure 2 and the flow chart below it showing the logic for state transitions.

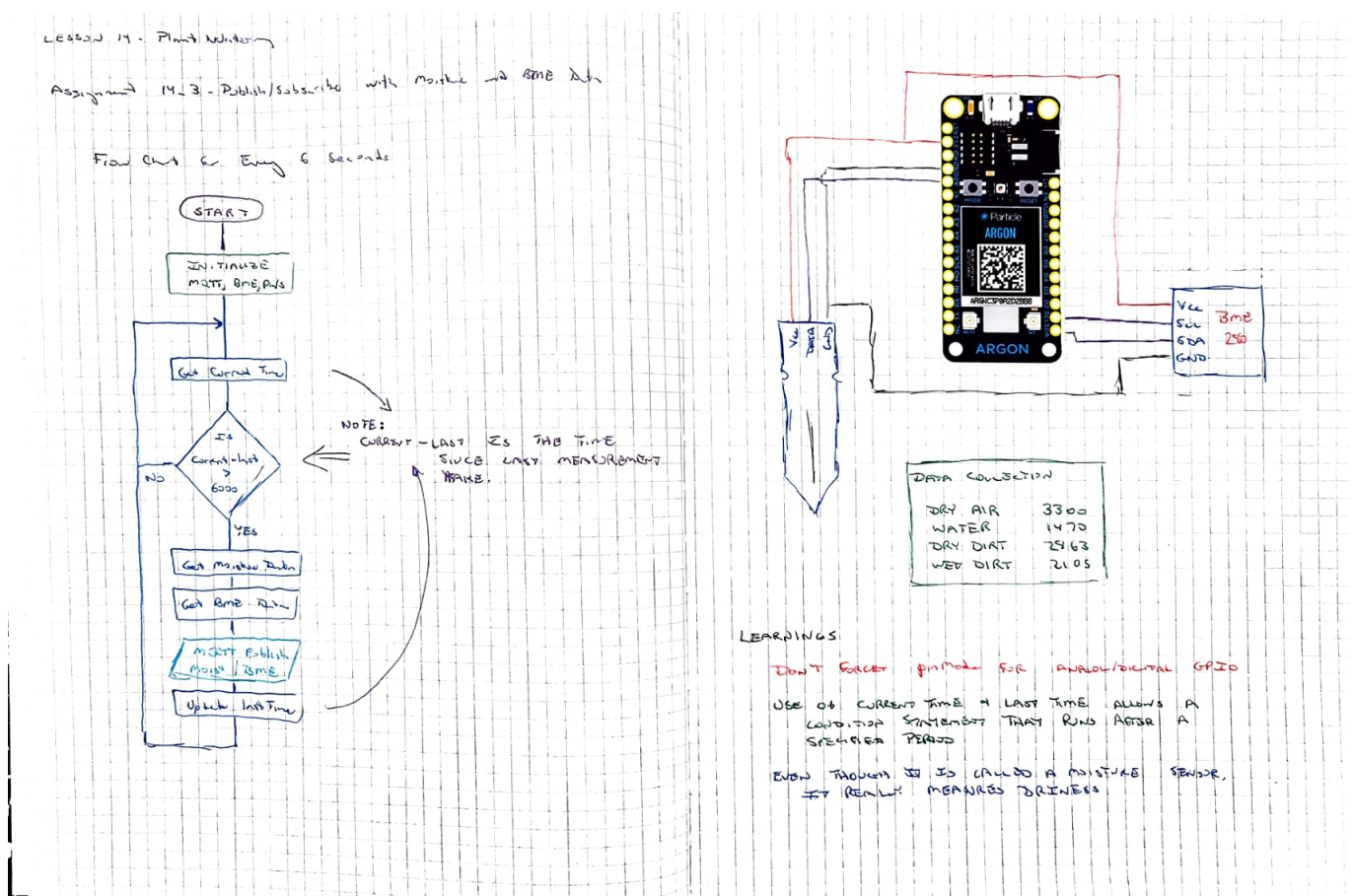


Figure 1: Lab Notebook example: Soil Moisture

- Any results obtained from the assignment.
 - Most assignments will not explicitly ask for documenting results. However, where appropriate you notebook to document readings obtained by your IoT device (i.e., the box on the right hand side of Figure 1).
- A lessons learned conclusion where you document learnings that you obtain from debugging your code, circuit layout, etc.

²All of these will be introduced as we progress through the class lessons

³English, or the human language of your choosing

- ## Assignment: British Traffic Light

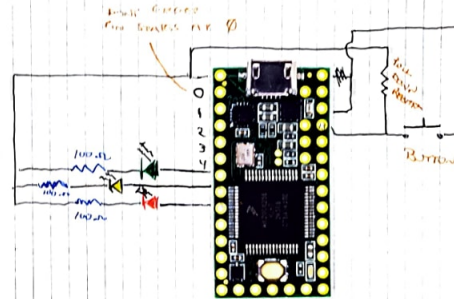


Figure 2: Lab Notebook example: Soil Moisture

5