Programming Assignment 1 – Upgraded Weather Station

Assigned/Due: See online for exact due date (11:59pm)

YOU MAY WORK WITH OR WRITE CODE TOGETHER WITH YOUR PARTNER ON THIS PROJECT.

Functional Requirements

In this project, you will push the boundaries of your knowledge to add new features based on some functionality we haven't quite covered in class (but should be within reach with a bit of research).

The below requirements must be met to receive full credit (see **Grading Breakdown** section at the end), but you are given considerable freedom and flexibility to implement these features however you see fit; that is, you don't need to write code in a certain way and are generally free to design your screens as you feel best, with minimal exceptions. Extend the Weather Station program from class to include ALL of the following:

1) *(EASY)* The ability to **convert back and forth between F and C** on the main display with the press of a button.
   a) Use a simple math equation to convert F=>C and back to F
2) *(MEDIUM)* A **timestamp on the main weather screen (Ex: 05:23:45PM)** of the screen showing the time of the last sync. Time can be obtained/synced by doing the following:
   a) Make call over WiFi (for example, by using a service/library like NTPClient) to get the current time from a time syncing server (just returns the current time to you)
      i) *NOTE: You can search "NtpClient" in the PlatformIO library search and add it to your project that way*
3) *(HARD)* An additional **screen that you can go to to manually select each digit of a 5-digit zip-code**, such that when you go back to the weather screen, it will fetch and display weather for the newly-selected zip-code.
   a) You will need to update the Weather API to use the zipcode API instead of the city name API we used in the LAB
      i) NOTE: You can hardcode the country to "us":

```
api.openweathermap.org/data/2.5/weather?zip=94040,us&appid={API
key}
```

   b) Your zip-code edit screen must:
      i) show all 5 digits (you can hardcode a default zip-code, which should then be editable and remembered when you return to the zip-code edit screen)
      ii) have a touch button above and below each zip-code digit such that each digit can be incremented/decremented with its own touch screen button
         (1) HINT: There are classes to make this much easier: https://github.com/m5stack/m5-docs/blob/master/docs/en/api/touch.md
4) *(MEDIUM)* Weather display screen should be redesigned to be more appealing and should better-handle the display of short AND long cities without looking awkward

a) *NOTE: You are free to re-design any of the layout and/or change color schemes, so long as all the required info is there (points may be given for creativity)*

Grading Breakdown

Grading will be broken down as follows:

- **10% - Turned in all source code**
- **90% - Video demonstration**
    - **30% - Weather Display Screen**
        - 10% - Demo conversion from F➔C➔F w/ the push of a button(5% each way)
        - 10% - Demo timestamp on weather display screen that shows time of last weather update (i.e., HTTP call)
        - 10% - Demo new weather screen layout redesign that elegantly displays long and short cities (e.g., Chino & Washington, DC) without awkward overlaps
            - *NOTE: Overlap is not necessarily a bad thing (e.g., you may wish to try placing the logo behind the text)*
    - **30% - Zip-code Edit Screen**
        - 10% - Zip-code screen uses 10 touch buttons to edit zip-code displayed on screen
        - 10% - Upon user editing of zip-code, user presses button to return to Weather Display screen, which shows new location's weather
        - 10% - Program remembers last zip-code entered by showing it on zip-code edit screen when returning from Weather Display Screen
    - **30% - Showed and discussed key code portions, explaining how you implemented each of the functional requirements**

NOTE: Points can and WILL be deducted at any point for performance issues (e.g., response is too slow, there is a noticeable lag between pressing the button and the screen updating with the result).

Submission Instructions

1.) Submit to Blackboard the following:
   a. **Video recording** (preferably YouTube link as a submission comment) demonstrating the functionality of your program and source code explanation:
      i. Include **at least 2 tests of the program for the following two zip codes (clearly showing the device change back-and-forth between these cases without resetting the device):**
         1. **20001 (Washington, DC)**
         2. **91708 (Chino)**
      ii. Make sure to cover ALL of the areas receiving points in the Grading Breakdown section above
      iii. Your video should be a 1-3 minute demo with 3-5 minutes of code highlights (no need to go into excessive detail, although you won't be penalized if you do)
      iv. Make sure there is sound
      v. Phone recordings are fine, just make sure the video clearly shows the LCD screen and what is happening with button presses
         1. If the instructor cannot see your screen clearly, you will not get any points for demoing features
   b. **Source files**, including:
      i. All **.cpp files**
      ii. All **.h files**
      iii. Your **platform.ini file**