CPSC334 Creative Embedded Systems

Observations: Lab 4 ESP32

Below are observations and tech notes from today's lab.

*** Please read the tech notes carefully as there are a lot of important details that should inform your work the rest of the week and into the next. ***

If there are notes, solutions, or anything you think would be helpful to your classmates (that you discovered in the lab) please let us know and we will include them in these notes.

Team performance:

1. In general, there was good teamwork and good inter-team camaraderie (cooperation.)
2. Progress was somewhat stifled by inexplicably slow network speeds downloading the Arduino ESP32 library. We will try to be more proactive regarding downloads (if necessary) in the future to avoid this.

Technical Notes:

1. Pin conflicts exist, some of which are detailed in the datasheet for the ESP32 DevkitC. If any pins are flipping when they shouldn't (when other buttons or switches are activated) switch to a different GPIO or use a pulldown resistor to preclude noise (see below no. 7).
2. Be sure of your GPIO pin numbers and whether you are using the "wiring" number or the physical pin. Arduino uses the physical pin number. Python (depending on the library used) uses a different number. Several groups used inconsistent pin numbering.
3. Be sure of your cables. At least three groups used their own microUSB-->USB(TypeA) cables and did not realize the cables were "charging" cables -- these lack data+ and data- and thus, obviously, will not work for programming the ESP32 (or anything else).
4. When troubleshooting:
    a. simplify your hardware circuit, check all connections and/or remove all but one connection.
    b. simplify your software output. Focus on one data type at a time and post/print one value at a time.
5. The best strategy for circuit design is to connect and test one physical sensor at a time. This is a surefire way to know when a pin conflict or other problem arises.
6. Remember momentary switches like the button we used is one of two configurations, momentary open or momentary closed. You can "switch" this behavior in software using pullup or pulldown statements in python (or other languages/libraries.) In Arduino, only PULLUP exists as a possibility (because the poorly-designed Arduino only has pullup resistors internally) so you should use a 10k resistor to ground if you want the pin pulled low. [see below]

7. If you are getting "noise" on digital pins (lots of 0s and 1s even when the switch or button is in a resting state) you should pull the resistor down with a resistor (10k is fine) to GND. This keeps any voltage noise on the pin from tripping the rising edge flipping the bit high.