**Our Journey**

During our first week of implementing our project. We decided to use the sample codes given to us to observe the results. This is because we wanted to understand the how to code the different sensors we use in our board and test whether the code is working as expected. We managed to test the temperature and humidity sensor and the brightness sensor. When we were working on the temperature and humidity sensor, we ran into a problem. Though the code seems fine the temperature and humidity displayed on the serial monitor was “nan” for both. Then we realize could it be because of the sensor. So, we played around with the position of the DHT11. Then we saw actual values. For the brightness sensor we had to find the right limit for it to turn on, so we had to do a little trial and error. That week we learnt that it is always better to start from the fundamentals, so errors could be identified more easily than trying to do everything at once. We also learnt to be more flexible and adaptive, finding out that there is nothing wrong with our code just that the sensor is a little loose.

In our second week, we worked on the moisture sensor and the lcd display. Overall, it was quite simple, and we did not run into any problem. However, since we never actually use a moisture sensor before we were confused to how to apply water on it without pouring water on it as the board could get damaged. So, we came up with the idea to use a wet tissue on the sensor so water could be applied only on the sensor.

In our third week, we worked on how to combine the codes of the different sensors and try out the SD card module. Combining the codes was not very difficult as we already worked out the codes for the individual sensors, so we only needed to combine them. However, when we tried using the SD card module code we were confused as to how to obtain the txt file. Then we realized that we only needed to remove the SD card from the Arduino board and put it in a SD card reader to read from our computer. We also worked on how to display all the data on the lcd display. Although we did not run into any problems with the code, we had to edit the code a few times to make sure that everything was displayed correctly on the lcd. In this week we learnt that we cannot expect everything to be given to us, things like how to use a SD card should be common sense so we just need to refer to our experiences to know how to use it.

In our fourth week, we worked on thingspeak. We ran into lots of problems while trying to implement it. The main problem was that the board couldn’t connect to our hotspot. But we don’t know what the problem is. The code we used was like the code we used during our lab lessons, and it was working fine then when we used it. So, we tried searching on the internet for possible solutions. Some suggested changing the baudrate, but it still doesn’t work. So, we decided to use the sketch test IO ESP01 to see whether it works. It still doesn’t work. Then we thought to ourselves that how could it not work it was a code that we used before in a lab lesson, and we also searched in the internet that is what most people use to test AT commands for the wifi module. So, we approached teacher with this problem and our suspicions were right, our wifi module was faulty that was why it could not work even though the codes we tested were correct. We also tried combining our main program with the thingspeak, however we couldn’t get it to work due to lack of time.

In our fifth week, after our combination of the code, we ran into a serious error – running low on RAM. At first when we searched online, it seemed like it wouldn’t affect the program running. However, the program seems unable to send the signal to ThingSpeak still. We read the serial monitor to attempt debugging and noticed some garble in the output. Some of the AT commands also seem to be cut halfway. For example, an AT command to establish wifi connection, “AT+CWJAP=”iPhone”,”konopowa”,80”, gets cut off halfway, and the output becomes “AT+CWJAP=”iPhone”,”konopow” and proceeds straight to the next command. After hours of debugging, our conclusion is that it is caused by the running low of RAM. We searched online for methods to reduce RAM usage and came across this thing called “The hidden Arduino F() Macro”. It did fix the problem, but it does not end here. Because we were unfamiliar with AT commands, we did not know how to modify the school’s code to suit our program, needing it to restart the entire circuit after every loop. After more hours of research, we found out that we should be adding “AT+RST” at the end of the loop “as a good habit”, and this fixes the problem.

**What we learnt**

We learnt that time management is very important in such projects. As there are many parts to the project, we must try to complete each part on time, and juggle between our other modules. We also learnt that when implementing the code for such projects which is relatively complex, require us to get the fundamentals right before we try to combine everything.

**Some new skills we learnt from researching**

**The F() Macro** - *Search the entire Arduino Reference Page, and you won’t find a single mention of the F() macro. Which is unfortunate because it is one of the most powerful functions, which was added with the 1.0 release of the IDE. (*[*https://www.baldengineer.com/arduino-f-macro.html*](https://www.baldengineer.com/arduino-f-macro.html)*)*

The F() macro tells the compiler to leave this particular array in PROGMEM. Then when it is time to access it, one byte of the data is copied to RAM at a time.

**The use of AT-RST after every loop**

It is a good practice to reset the Module before/after every loop