**Breakdown of each Weather Station file**

**Wind Vane**

The file firstly lists the physical wiring that will need to be done in order to use the file. This is the same as the wiring given in the training manual. The file then sets the basic parameters required for the test to take place. The file then sets the parameters for where to take a voltage reading from, including which port on the Raspberry Pi to test from. The file then sets up various print statements relating to the voltage received and which direction the given voltage corresponds to. It also has a statement saying that if the voltage received is out with the parameters for any of the wind directions then print “unknown”. The file then directs the data to be stored within a database in SQLite 3. The file will repeat until a user input is made to stop it.

**Anemometer**

The file sets the basic parameters required for the test to take place. The file then defines that the date and time will be added next to each recorded wind speed value. The address from which the readings will be taken in then defined, before the connection to SQLite 3 is made. The file then runs for 10 seconds and counts the amount of clicks from the anemometer during the 10 second period. 1 click is equivalent to 0.1492mph, therefore the file has a formula that multiplies the amount of clicks within a 10 second period by 0.1492, in order to calculate the wind speed for that 10 second period. The wind speed is then rounded to 2 decimal places and printed with the units of mph. Another 10 second period then starts, and so on until a user input is made to stop the process. All the while, a collection thread is running to store the data within SQLite 3.

**Rain Gauge**

Various imports of all the required files are carried out firstly. The file sets the basic parameters required for the test to take place. The file then defines that the date and time will be added next to each recorded amount of rain. The address from which the readings will be taken in then defined, before the connection to SQLite 3 is made. The file defines the parameters it will measure and that the number of inches of rainfall per click is 0.011 inches. The file then runs for 1 minute and counts the amount of clicks from the rain gauge during the 1 minute period. The file has a formula that multiplies the amount of clicks within a 1 minute period by 0.011, in order to calculate the rainfall for that minute. The rainfall is then rounded to 2 decimal places and printed with the units of inches. An average rainfall per minute is also calculated and added into the results. Another minute period then starts, and so on until a user input is made to stop the process. Each time a result is recorded, the average rainfall updates accordingly. All the while, a collection thread is running to store the data within SQLite 3.

**Temperature and Pressure**

The file sets the basic parameters required for the test to take place. The file then defines that the date and time will be added next to each recorded temperature value, as well as each recorded pressure value. The address from which the readings will be taken in then defined, before the connection to SQLite 3 is made. A formula is put in place to convert the pressure reading, which would be received in Pascals, to millibar. The file provides a constant temperature and pressure reading until a user input is made to stop the process. All the while, a collection thread is running to store the archived temperature and pressure readings within SQLite 3.

**LDR (Light Dependant Resistor)**

The file sets the basic parameters required for the test to take place. The file then defines that the date and time will be added next to each recorded light value. The address from which the readings will be taken in then defined, before the connection to SQLite 3 is made. The file then sets the parameters for where to take a voltage reading from, including which port on the Raspberry Pi to test from. The file provides a constant reading of the light levels until a user input is made to stop the process. All the while, a collection thread is running to store the archived readings of light levels within SQLite 3.

**Humidity**

The file sets the basic parameters required for the test to take place. The temperature is then recorded. A formula is put in place to use the temperature reading to find the humidity. The file provides a constant temperature and pressure reading until a user input is made to stop the process.

**Weather Station File**

The file sets the basic parameters required for each of the tests to take place. The file then starts the script for each of the sensors in use, as well as where to test each sensor from. There is an error message built in to be displayed if any of the files do not start up correctly or are not recording any data. Each file will run exactly as identified previously, the key difference being that they will all run simultaneously.

**Adafruit Files**

Adafruit provides device driver files for various chips used on the Weather Station, such as the BMP and I2C chips. This code will never change so it does not need to be learnt. However it is essential to have it added into each program that it will be used in, as has already been achieved.

**.git ignore**

Due to all the files installed in order to create the weather station, there are a lot of files which will be compiled are not necessary. In order to stop them from being compiled, a .git ignore file has been created. Anything that should not be in use should be included within the .git ignore file and subsequently the unwanted file will not be used.

**.git**

In order to store your files securely, as well as to make it available to the internet as a whole, you will wish to upload your files to Git Hub. Git Hub is a large internet repository used for storing folders, sub folders and files online for both storage purposes and for general public viewing. In order to achieve this, a git init must be performed, before performing a git remote add origin, in order to set the directory on Git Hub. A git add should then be performed, followed by the name of the folder / sub folder / file looking to be added. If you are sure that the correct file has been chosen, a git commit should be performed in order to commit the information formally to git hub. This step can be confusing and there can be consequences to getting it wrong and accidentally uploading the wrong information. If you are unsure with this step it is wise to seek assistance.