Data Acquisition tasks

- Temperature measurement
  - Function to read resistance averaged over NTH measurements, with uncertainty estimate
  - Function to calculate temperature and uncertainty from resistance
- RH measurement
  - Function to read RH and temperature from Si7021
  - Function to calculate dew point and estimated uncertainty
- Pressure measurement
  - Function to read pressure and temperature from BMP180
  - Function to calculate equivalent sea-level pressure
- Clock
  - Function to read clock as numerical values
  - Function to format time as a string
  - Function to check clock
- SD card
  - Function to write to file and validate that the correct amount of data was written
  - Function to check that the SD card is present and working correctly
- Statistical function
  - Function to keep a record of first four moments of temperature, RH, and pressure
  - Function to reset the statistical storage variables
  - Functions to calculate and return the mean, standard deviation, skew, and kurtosis
- Formatting output
  - Use variables returned by other functions to assemble a single line to be added to the data file
  - Check that the variables make sense and substitute an error flag for bad values
- Timing/dispatch and error checking
  - Timing and decision making to call other functions
  - Blink LED with error status
- Setup/initialization
  - Initialize sensor and variables
  - Check for serial input enabled or disabled
  - Watchdog timer routine
- Calibration
  - Function to check for calibration file, load calibration, and validate the calibration constants

- Temperature measurement
  - Function to read resistance averaged over NTH measurements, with uncertainty estimate

    int readMeanResistance(float & Rth, float & Rth_unc)
    - Read the resistance NTH times from pin THERMISTOR_PIN
    - Return the mean resistance as Rth
    - Return the uncertainty as Rth_unc
    - Return 0 if successful, and some error code otherwise
  - Function to calculate temperature and uncertainty from resistance

    int calcTmp(float Rth, float & tmp, float & tmp_unc)
    - Calculate the temperature of the thermistor using Rth and return in variable tmp
    - Use calibration constants that exist as global variables called THERMISTOR_CAL_A, THERMISTOR_CAL_B, …
    - Calculate the uncertainty in the thermistor temperature and store in variable tmp_unc
    - Return 0 if successful, and some error code otherwise

- RH measurement
  - Function to read RH and temperature from Si7021

    int readRH(float & RH, float & RH_unc)
    - Read and return the relative humidity from the Si7021
    - Determine and return the relative humidity uncertainty from the Si7021
    - Return 0 if successful, and some error code otherwise

    int readRHTmp(float & temp, float & tmp_unc)
    - Read and return the temperature from the Si7021
    - Determine and return the temperature uncertainty from the Si7021
    - Return 0 if successful, and some error code otherwise
  - Function to calculate dew point and estimated uncertainty

    int calculateDewPoint(float tmp, float tmp_unc, float RH, float RH_unc, float & dewPoint, float & dewPoint_unc)
    - Calculate the dew point from a given temperature and RH and their uncertainties
    - Calculate the dew point uncertainty
    - Return 0 if successful, and some error code otherwise
  - Function to check the Si7021

    int checkRH()
    - Check that the Si7021is present and returns reasonable output
    - Return 0 if everything is good, some error code if not

- Pressure measurement
  - Functions to read pressure and temperature from BMP180

    int readPrs(float & prs, float & prs_unc)
    - Read and return the pressure from the BMP180
    - Determine and return the pressure uncertainty
    - Return 0 if successful, and some error code otherwise

    int readPrsTmp(float & tmp, float & tmp_unc)
    - Read and return the temperature from the BMP180
    - Determine and return the temperature uncertainty
    - Return 0 if successful, and some error code otherwise
  - Functions to calculate equivalent sea-level pressure

    int toSLP(float prs, float elevation, float tmp, float & SLP)
    - Take a local pressure, elevation, and air temperature, and calculate the equivalent sea-level pressure
    - Return 0 if the calculation was successful

    int toLocalPrs(float SLP, float elevation, float tmp, float & prs)
    - From equivalent sea-level pressure, elevation, and temperature, calculate local pressure
    - Return 0 if the calculation was successful

- Clock
  - Use RTClib (see file RTClib.h)
  - Function to format time as a string

    int getTimeString(char timeStr[])
    - Get the time, and format it as a string (“HH:mm:ss”) in timeStr
    - The character array time is 9 characters long (char time[9];)
    - Return 0 if successful, error code if not

    int getDateString(char dateStr[])
    - Get the date and format it as a string (“DD/MM/YYYY”) in dateStr
    - The character array date is 11 characters long (char date[11];)
    - Return 0 if successful, error code if not
  - Function to set clock

    int checkClock()
    - Check the status of the clock
    - Return an integer reflecting the clock status (0 if everything is good)

- SD card
  - Function to write to file and validate that the correct amount of data was written

    int writeDataToSDFile(String datastring)
    - Using the function getSDFileName(), open the file on the SD card
    - Check the file size
    - Write datastring to the file
    - Close the file
    - Check the file size and return 0 if it has increased by the correct size
  - Function to check that the SD card is present and working correctly

    int checkSD()
    - Check the SD card to make sure that it is present and working
  - Function to determine the current SD filename

    int getSDFileName(char filename[])
    - Using the function getTimeString, format the file name (“YYYYMMDD.csv”) and return it as a string
  - Function to initialize the SD card

    int initSDCard()
    - Initialize the SD card

- Statistical function
  - Function to keep a record of first four moments of temperature, RH, and pressure

    void updateStats(float x, float storageArray[])
    - Update statistics will calculate the mean, standard deviation, skew, and kurtosis of each variable. The data will be passed in one value at a time, and storageArray will hold the data necessary to calculate the statistical values.
    - Read refer to Wikipedia for background on Standard deviation, Skewness, Kurtosis, and read the page titled "Algorithms for calculating variance". This function should implement the algorithm outlined in the section titled "Higher-order statistics".
    - storageArray will hold the values of n, mean, M2, M3, and M4 from the sample Python function online_kurtosis on the Wikipedia page.
    - In the documentation (comments), cite an *academic* source (there are several articles cited on the Wikipedia page).
  - Function to initialize the statistical storage variables

    void initStats(float storageArray[])
    - Initialize storageArray to make it ready for calculations using updateStatistics (all array entries should equal zero).
  - Functions to calculate and return the mean, standard deviation, skew, and kurtosis

    float calcMean(float storageArray[])
    - Use the values in the storageArray to calculate and return the mean

    float calcStd(float storageArray[])
    - Use the values in the storageArray to calculate and return the standard deviation

    float calcSkewness(float storageArray[])
    - Use the values in the storageArray to calculate and return the skewness

    float calcKurtosis(float storageArray[])
    - Use the values in the storageArray to calculate and return the kurtosis

- Formatting output
  - Use variables returned by other functions to assemble a single line to be added to the data file

    int formatValues(char output[], int outputSize, float values[], int arraySize)
    - Validate the variables and replace them with error flags if necessary
    - Format the variables and assemble them as comma-separated values in the string line
    - The format should be fixed so that the string is always a constant length
  - Check that the variables make sense and substitute an error flag for bad values

    float validateTmp(float tmp)
    - Return tmp if the temperature is within a reasonable atmospheric range
    - If the temperature is not valid, return -99.0

    float validateRH(float RH)
    - Return RH if the relative humidity is within a reasonable atmospheric range
    - If the relative humidity is not valid, return -1.0

    float validatePrs(float prs)
    - Return 0 if the pressure is within a reasonable atmospheric range
    - If the pressure is not valid, return -1.0

- Timing/dispatch and error checking
  - Timing and decision making to call other functions
  - Blink LED with error status
  - See the instructor/TA for more information. You should communicate with the group working on the setup code to ensure compatibility.

- Setup/initialization
  - Initialize sensor and variables
  - Check for serial input enabled or disabled?
  - Watchdog timer routine

    int resetWatchdogTimer()
    - Get the time from the clock chip
    - Set the timer on the clock chip to trigger after a determined interval
    - Return 0 if successful
  - See the instructor/TA for more information. You should communicate with the group working on the loop (timing/dispatch) code to ensure compatibility.

- Calibration
  - Function to check for calibration file, load calibration, and validate the calibration constants

    int loadCalibrationFile()
    - Check for a calibration file on the SD card
    - Open the calibration file and read in the calibration constants
    - Assign the constants to the appropriate global variables
    - Return 0 if all constants were read successfully
    - If any calibration constants are missing, assign appropriate estimate values to the global variables
    - Return an error code indicating which constants are missing