Documentation of Weather code
June 12, 2018
Mahdi Mohamed

file names = blue , class/objects = brown , functions = purple, parameters = red

Bash (.sh) Scripts

- **startup.sh**
  - **Mounts usb flash drive, starts maintain-time.sh, runs main.py, sends email if main.py stops running.**
- **maintain-time.sh**
  - **Keeps the time accurate on the TS-4200 CPU on the Vor server**
- **end.sh**
  - **Sends a signal to main.py that will terminate at the rest cycle.**
- **maintain-time-wrapper.sh**
  - **Keeps the output of maintain-time looking pretty.**

Python (.py) Scripts

- **main.py**
  - **Schedules and executes various tasks.**
  - **WeatherScheduler class**: orders and schedules functions defined elsewhere.
  - **_init_ (self):** constructor that initializes DataReader and Dataoutputer, which are used by handleData.py . Initializes a scheduler.
  - **Functions:**
    - **run (self)**: gets data from PIC, runs main loop forever, and calls other functions.
    - **daily_checks (self)**: checks if PIC is missing strings if the board has enough space in memory, resets error flags, runs once every 24 hours.
    - **build_files_for_dash (self)**: reader/outputter adds data for GUI
    - **note_running (self)**: leaves timestamp in file so GUI knows that Freya (Vor's board) is running.
    - **maybe_end (self)**: checks if end signal has been received, and ends the program if it has.
    - **finish_cycle (self)**: called at the end of a cycle to wrap up.
    - **end (self, signum, frame)**: catches signal sent by end.sh, sets endToggle to true.
- **config.py**

- **Holds configuration information for main.py, errors.py, handleData.py. (configurable variables in these files should be changed here).**

➢ **errors.py**

- **Holds all of the functions for error logging/notifications.**
- **Functions:**
  - **debug (msg):** logs and emails debugging messages
  - **info (msg)**: logs and emails info messages
  - **error (msg)**: logs and emails error messages
  - **queueEmail (msg)**: adds messages to the holding folder, which is sent every 20 seconds.
  - **connect_and_send (msg string, n int)**: connects to SMTP (Simple Mail Transfer Protocol) server. Sends string msg, only when called by sentEmail.
  - **sendEmail (msg string, subject string)**: sends emails. If a message is specified it sends it. If no message is specified, it sends all pending email.
  - **initialize notification systems**: called when main.py imports errors.py, to set up logging. It will be called with every import.
  - **test( )**: test the functionality of errors.py to confirm that all features are working.

➢ **DataConverters.py**

- **Holds all of the equations for converting raw data, organized by block and number. Holds response logic.**
- **DataConverter class**: holds logic for data fields, one subclass per outgoing field. Gathers raw data, converts it, passing it to outputter. Checks if values are in range, and checks for errors. Format:

```
Class DataConverter (object):
        Command_error = False
        Reader = None
        Outputter = None
        Suppress_range_check = false
```

- **__init__(self)** : constructor that initializes DataConverter object.
- **Functions:**
  - **set_in_out (reader = None, outputter=None)**: updates saved reader and outputter for all DataConverter subclasses. To be used on WeatherScheduler initialization. Static method.
  - **process (self)**: converts raw data (as a string) into converted data (as a decimal). Must be overwritten in each subclass.

- **respond (self, converteddata)**: takes action. Sends notification in response to data, should be overwritten in subclass if a response is needed.
- **check_range (self, converteddata)**: confirms that converted value is within range defined. If not alerts and replaces with error value.
- **safe_process (self)**: function provides a framework that calls all user-written methods and handles errors. Probably should not be overwritten in subclass definitions. This needs to be called and all relevant things will be taken care of.
- **reset_flags (self)**: rests error flags so reming notifications will be sent, called daily.
- Rest of functions are for specific data fields, all are subclasses which are in this format:

```
class className (Dataconverter):
        def __init__(self):
                DataConverter.__init__(self)
                        self.lower_bound = some number
                        self.upper_bond = some number
                        self.dec_places_round = some number
        def process (self, rawdata):
                converted = some calculation on raw data
                        return converted
```

➢ **convert.py**

- **Initiates all DataConverter subclasses and calls on the instances of the safe_process method.**
- **all_converters: is a list that holds converter objects corresponding to every data field. When new fields are added, new subclasses of DataConverter should be defined and added here.**
- **Functions:**
  - **process_all ()**: loops through all of the converters and calls safe_process on them.
  - **reset_all_flags ()**: resets flags for all converters, called from main.py daily.

➢ **handleData.py**

- **This file contains two classes: DataReader and DataOutputter.**
- **DataReader is responsible for reading and storing raw data from the PIC. Its methods are scheduled in main.py by the WeatherCollector and used in convert.py by DataConverters that need raw data.**

- **DataReader class**: Essentially a wrapper for the serial port that allows data to be requested by label.
- **__init__(self)** : constructor for DataReader. Responsible for initializing locations, a dictionary keying (string, index) by field label, from picdata.conf. If this fails, method will raise SystemExit. Also opens serial connection as configured in config.py. If connection fails at first, method will continue attempting indefinitely.
- **Functions:**
  - **get (self, label)**: given a label, returns that label's stored data as a string. If no such label is found in the given format or if the corresponding data point is not found in received data, registers an error message. Errors are stored in a list as the name (label, stringID, or ID+index) of the missing location. If a location is already in the list, email will not be resent. These errors are expected to handle cases of typos in picdata.conf or data converting code, missing strings, and so on. This method does not check type -- calling methods get either the raw data exactly as received (as a string) or Config.ERROR_VAL. **Be sure NOT to use this method to request the name of a string (eg, B or T)**
  - **numStrings (self)**: Returns the number of PIC strings in the current format.
  - **numBadStrings (self, reset=False)**: Returns the number of bad (empty or misformatted strings) received since last reset or start up. Also resets counters and flags if reset is True.
  - **send (self, cmd)**: Sends string **cmd** to PIC via serial connection.
  - **build_file_for_dash (self)**: overwrites the file for dash with raw data.
  - **read (self, header, header_1, toggle)**: Reads strings of raw data from serial port and checks them for errors before saving them in a dictionary. The strings that make it through this method are guaranteed to have the right ID and length, but any given piece of data could conceivably be missing (""). Checking that data exist is the job of get().
  - **send_commands (self)**: Reads files holding all commands for the PIC and sends them. Command file should be formatted as <cmd><tab><source>, with one command per line.
- **DataOutputter is responsible for collecting, formatting, and saving converted data. Its methods are also scheduled in main.py.**

DataConverters give DataOutputter converted data as they finish converting it.

- **DataOutputter class:** handles outputting data.
- **__init__ (self)**: DataOutputter constructor: parses outdata.conf into a list, makes a document holding format of data to dashboard, and initializes error flags.
- **Functions:**
  - **receive (self, label, data)**: Pass data (in any format) for field <label> to this outputter. Called by DataConverters once they have finished converting data. Outputter will store the data and ultimately format/save it.
  - **get_converted (self, label)**: Recover converted data from outputter. Used for second-level conversions.
  - **save (self, header, toggle)**: Formats all received converted data and saves it to the appropriate configuration of files. Clears all data so outputter is available to receive a new batch.
  - **reset (self)**: Resets error flags and lists so notifications will be re-sent.
  - **build_file_for_dash (self)**: writes data for the dashboard. this file will be read using pysftp by the dashboard application whenever it loads. It opens format file and data file. And for each line in the format file, if the line is a string tag if copies it over, and if it's a label, it is copied over with data appended.

Configuration (.conf) Files

➢ **outdata.conf**

- A text file that holds the format of data outputted by the TS-4200 CPU on the Vor Server. (**If any new fields are to be added they should be added here so the board knows in what order to output them.**)

➢ **picdata.conf**

- This file holds the format of the data received from the PIC as the TS-4200 expects it. (**If new fields or strings are to be sent by the PIC, the should be added here.**)