Introduction

The following provides a description of a “Web of Things” inspired, do-it-yourself, solar powered, wireless weather station perfect for the hobbyist or home investigator. The implementation described below, though certainly not unique, represents a balanced tradeoff between cost and function. A high priority has been placed on using all "open-source" hardware and software. Avoiding proprietary, closed source software, as well as proprietary “cloud” services, allows for much greater flexibility. This flexibility facilitates ease of deployment and scale-out on a large variety of open source platforms, and for a large variety of end user applications.

For example, the weather station might be in a location which only provides wireless connectivity to the Internet. In this case, the weather station would be configured to send data via the Internet to a server remotely located somewhere else. Alternatively, the weather station might be connected via a local area network to a web server. In this case, the local network router may be configured to allow Internet access to the weather station web page hosted on the server. Many deployment options are possible.   This project plan uses the later model to provide an example of a “turn-key” solution that includes both the weather station, and the implementation of “back-end” services.

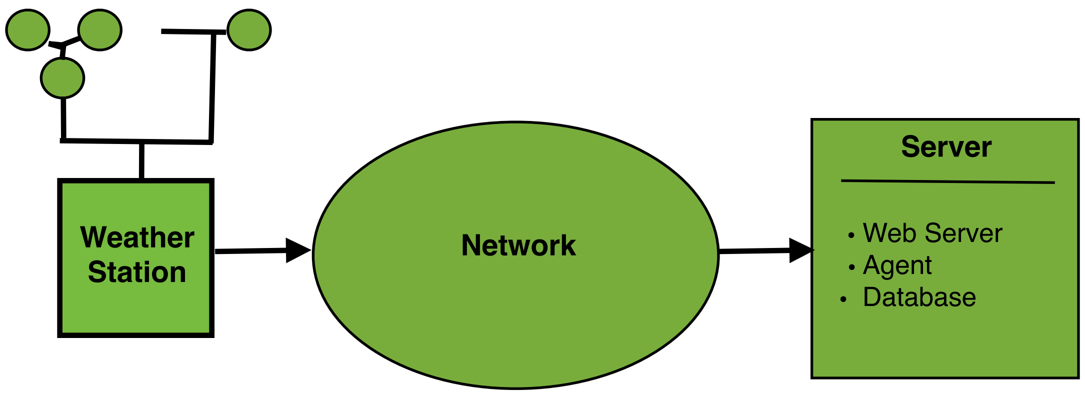

Figure 1. Overall conceptual view showing weather station, network, and server components.

Prerequisites for building and deploying the DIY Weather station include: soldering, ability to follow electrical hookup guides, knowledge of Arduino technology, Linux system administration, and some knowledge of the C and Python computer languages. The open source software provided in the Github repository should be sufficient, with only minor modifications, for most applications. The server side software has been designed around the file structure and system configuration found in most common Linux distributions. In most cases, the majority of modification will be in the HTML, customizing the look and appearance of the web page presentation to the end user.

DIY Weather System Overview

The DIY Weather system comprises three main components. Referring to figure 1, the weather station hardware, the network connecting the weather station to the server, and the server, itself. The following project description will focus on the weather station hardware and the server software. The network connecting the two, including wireless access points, switches, routers, etc., will be considered background infrastructure beyond the scope of this document.

The weather station consists of a number of hardware components, along with a software component that runs on the weather station micro-controller. The server hardware will be considered background infrastructure and not discussed, other than regarding software dependencies. The server hardware can be implemented in anything from a Raspberry Pi, running Raspbian, to a full up Internet cloud server running Linux, to a wifi enabled laptop. The server software consists of two main components: an agent for processing data received from the weather station, and a web server along with HTML documents for displaying the data.

The agent component, acting as client, periodically sends an HTTP request through the network to the weather station. The weather station responds with a short text string containing the latest weather information gathered by the weather station sensors. The agent, a Python script, runs as a process on the server, and periodically sends the HTTP request to the weather station. The agent processes the response from the weather station, writes data to the rrdtool database, and updates the weather data file for use by HTML documents. The weather station HTML document gets the weather information from the weather data output file and displays it in the user's web browser. The agent also sends to the weather station a reset command every midnight, which resets the day's weather statistics back to zero.

Component Descriptions

Weather Station Hardware

Referring to figure 2, the weather station hardware consists of components for generating and managing power to the station electronics: a micro-controller, a weather shield, a wifi shield, and weather meters. A solar cell provides power, through a charge controller, to both the LiPo backup battery and the micro-controller.

The wind meters provide signals which the micro-controller interprets to determine wind speed and direction. The weather shield provides an electrical interface from the wind and rain fall sensors, to the micro-controller. The weather shield also contains the pressure and humidity sensors and an electrical interface for these two sensors. The pressure sensor measures both barometric pressure and temperature.

The micro-controller, the heart of the weather station, consists of a single, Arduino "Uno" type micro-controller. Arduino "shields" are various Arduino compatible circuit boards that plug directly into the Arduino circuit board. The wifi shield is plugged into (or stacked on top) of the Arduino board. The weather shield is plugged into (or stacked on top) of the wifi shield. This arrangement or "stack" requires that board and shields all be compatible with each other, both electrically and at the data layer.

The solar cell generates power that must be conditioned to both charge the backup, LiPo battery, and power the station micro-controller and attached shields. Solar cell current and voltage varies with the amount of solar energy falling on the solar cell. Clouds reduce the amount of solar energy falling on the cell, and, obviously, at night the solar cell generates no power. The charge controller provides conditioned voltage and current for both charging the backup battery and powering the micro-controller stack. The LiPo battery provides power during times when the solar panel is not generating enough power to run the micro-controller stack.

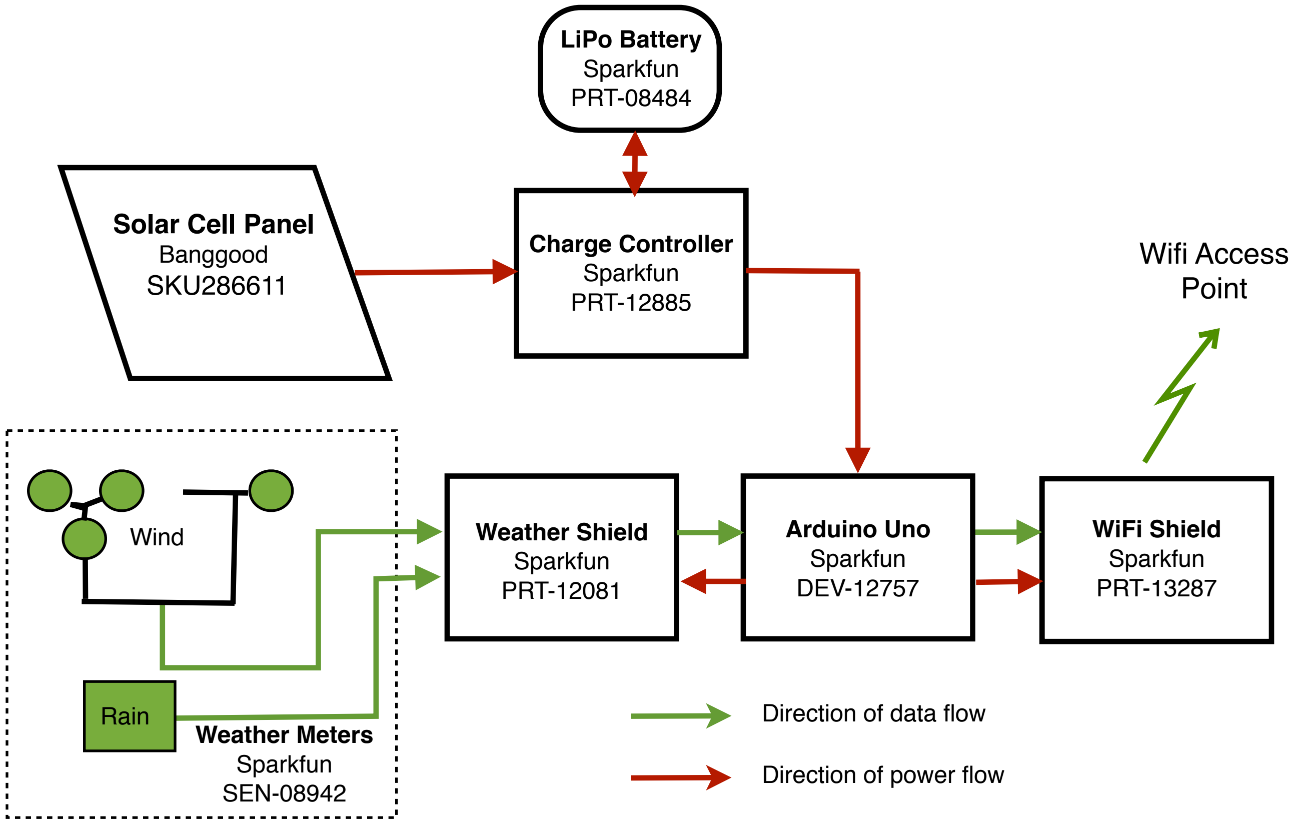

Figure 2. Block diagram of weather station hardware showing individual electronic components.

Weather Station Software

The software for the weather station micro-controller - the heart of the weather station - consists of an Arduino "sketch", along with several ancillary libraries that must be included at compile time. Referring to figure 3, the main module, or sketch, may be viewed as a software "stack" made up of three layers. The top most includes the setup routine that runs once at boot up time, and the main loop that contains all the routines that run forever after initial setup.

Initial setup runs routines that initialize the weather sensors and the wifi interface. The main loop runs the second level routines referred to in figure 3 as *Do Every Second*, *Do Every Minute,* and *Listen for HTTP Clients*. *Do Every Second* triggers third layer events that happen at intervals determined by one-second marks, which includes *Update 1 Second Wind Gust* and *Update 2 Minute Wind Average*. Similarly, *Do Every Minute* triggers events that happen at one minute marks, which includes *Update 10 Minute Wind Gust* and *Update 1 Hour Rain Fall*.

Update 1 Second Wind Gust runs every second and compares the current wind speed with the wind speed measured one second earlier. If the current wind speed is greater, then both the wind gust speed and direction values get updated with the current values. The wind gust speed (and direction) is stored in a ten element array. After a minute elapses, the array pointer gets incremented, thus each element of the array contains the highest wind speed measured over a one minute interval. Taking the largest value in the array gives the speed of the highest wind gust in the last ten minutes.

Update 2 Minute Wind Average runs every second and stores the instantaneous wind speed and direction in the 2 minute average wind array. When *Calculate and Report Weather* gets called, the 2 minute average wind array is used to calculate average wind speed and direction for the previous 120 second period.

Listen for HTTP Clients runs every pass through the main loop and responds to HTTP requests from network clients. Upon receiving a request, the request gets checked for configuration management commands or a reset command. Configuration management commands change station password, SSID, and WPA passphrase. If no commands are found, then weather information gets sent back to the requesting client. Commands are embedded in the URL for the weather station.

Calculate and Report Weather, called by *Listen for HTTP Clients*, calculates wind and rain, and retrieves current readings from sensors. The calculated data includes the highest wind gust for the day, the highest wind gust in the last ten minutes, the average wind for the last 120 seconds, and the total rainfall for the last 60 minutes. The sensor data reported includes the humidity (percent), barometric pressure (Pascals), and temperature (Fahrenheit).

Update 10 Minute Wind Gust, triggered every minute, increments a pointer to the ten minute wind gust array. Each element of the array contains the highest wind speed measured over a one minute period, the entire array covering a ten minute period. When the pointer reaches the bottom of the array, it gets “rolled over” back to the top of the array, thus implementing a “round robin” data structure. This method works since only the maximum of the array is needed, not the value of any individual element.

Update 1 Hour Rain Fall, triggered every minute, stores the amount of rain fall measured during that minute in the rain fall array and then increments the array pointer. Similar to the above, when the array pointer reaches the bottom of the array, it gets rolled over to the top of the array. Again, this method works since only the total amount of rain over the last sixty minutes is needed.

The above description provides an outline summary of the micro-controller program. The program, itself, involves quite a bit more: interrupt handlers for wind speed and rain fall sensors, functions for calculating instantaneous wind speed and direction, and a function for processing maintenance commands from the server.

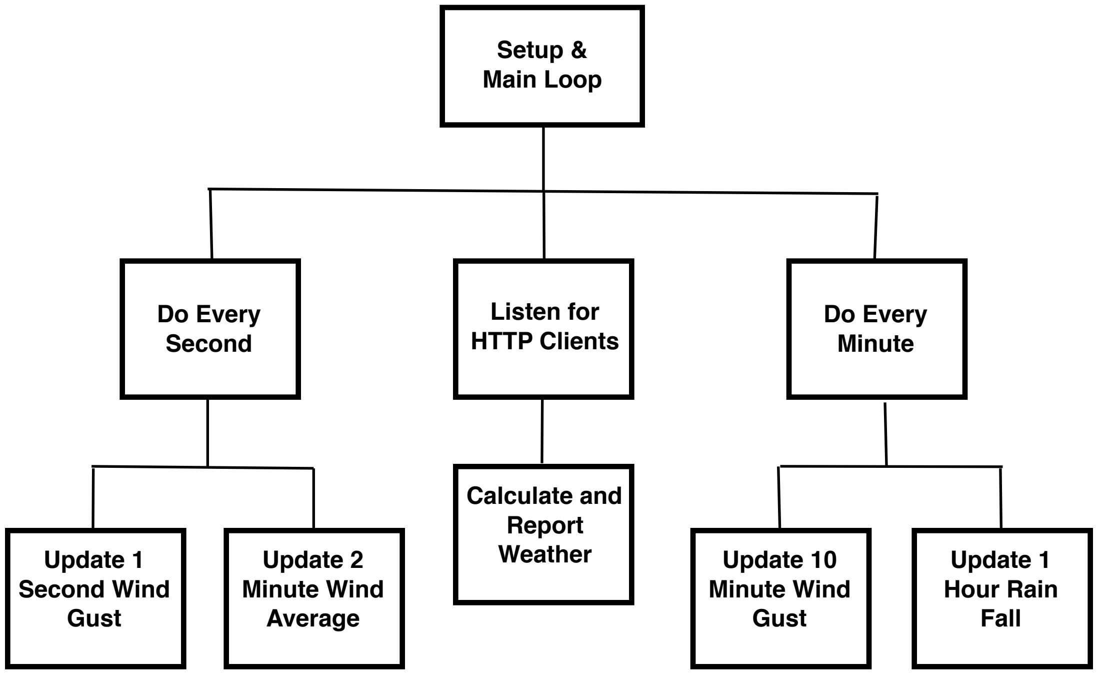
Setup & Main Loop
Do Every Second
Listen for HTTP Clients
Do Every Minute
Update 1 Second Wind Gust
Update 2 Minute Wind Average
Calculate and Report Weather
Update 10 Minute Wind Gust
Update 1 Hour Rain Fall

Figure 3. Micro-controller software block diagram showing high level components.

Server Software

Referring to figure 4, the server software consists essentially of two components: a web server component and an agent component. The web server component includes the HTML documents, which contain Javascript for displaying weather information. The agent, a Python script, manages data conversion and reformatting, updating a database, and routine weather station maintenance.

Events flow in the following manner. The agent periodically sends an HTTP request to the weather station. The weather station responds with a string containing the current weather information. Below is a typical example of the string containing the weather information

```
$,ws=10.0,wd=270,ws2=7.0,wd2=270,wgs=25.0,wgd=180,wgs10=12.0,wgd10=270,h=51.0
,t=76.8,p=101269.3,r=1.00,dr=5.00,b=4.3,l=2.4#
```

The agent converts specific data items to other physical units where required and formats the data for other services. Selected data items also get written to a round-robin database for permanent storage. Besides these functions, the agent manages generation of graphic charts for display in HTML documents, as well as sending periodic “reset” commands to the weather station.

The reset command, always sent exactly at midnight, resets the daily total rain fall and maximum wind gust back to zero. After formatting the weather data, the agent writes the data to the *Output Data File* for use by HTML documents. When a client browser requests the weather HTML document, Javascript embedded in the document periodically requests the output data file and displays the data in an HTML document. Although not shown in figure 4, the agent, as mentioned earlier, generates graphic charts for display in the HTML documents. The graphic charts are stored as image files which Javascript in the HTML documents can display in the weather web page.

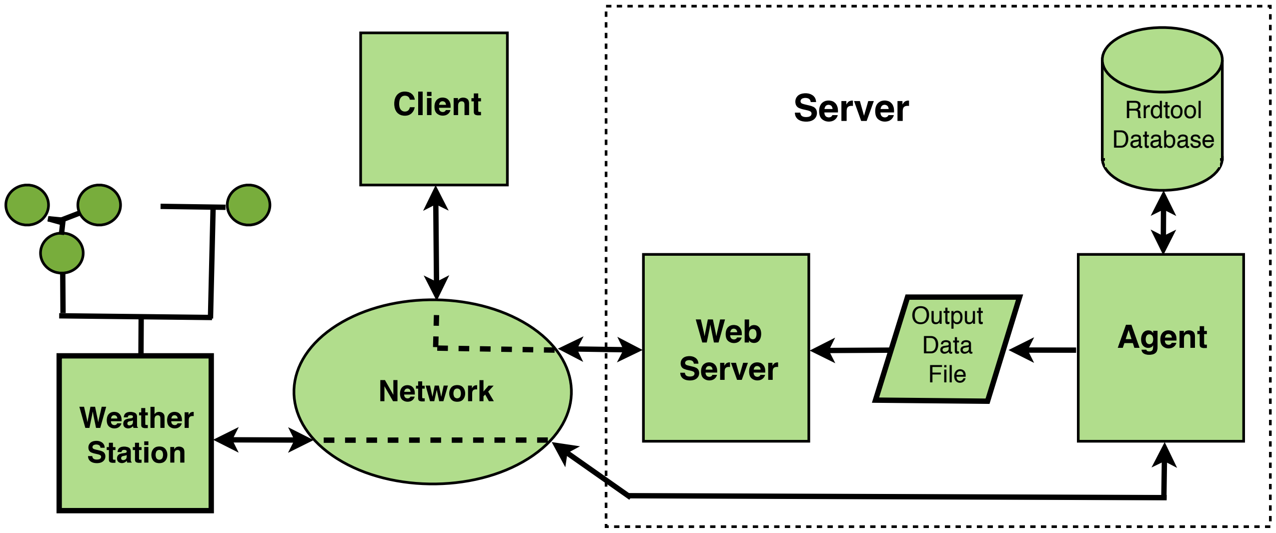

Figure 4. Block diagram of server software components showing data i/o, agent process, web server, and database elements.

Building the DIY Weather Station

Parts List

To build the weather station, see the list below for recommended parts and materials. Consult the supplier's web site for detailed instructions on how to hookup and use individual parts.

Description	Quantity	Supplier	Part #
Red Board (Arduino Uno equivalent)	1	Sparkfun	DEV-12757
Weather Shield	1	Sparkfun	PRT-12081
ESP8266 Wifi Shield	1	Sparkfun	PRT-13287
Weather Meters	1	Sparkfun	SEN-08942
Sunny Buddy MPPT Solar Charger	1	Sparkfun	PRT-12885
Polymer Lithium Ion Battery 6Ah	1	Sparkfun	PRT-08484
Solar Cell 20 Watt	1	Banggood	SKU286611
Arduino Stackable Header Kit	3	Sparkfun	PRT-11417
RJ-11 6 Pin Connector	2	Sparkfun	PRT-00132

Description	Quantity	Supplier	Part #
JST Jumper 2 Wire Assembly	1	Sparkfun	PRT-09914
Solar Radiation Shield (housing)	1	Ambient	SRS100LX
Raspberry Pi 2, Model B (optional, if you need a server)	1	Maker Shed	MKRPI8

Supplier Websites
Ordering information, detailed specifications, and hookup guides are available online at

- Sparkfun: sparkfun.com
- Ambient LLC: ambientweather.com
- Maker Shed: www.makershed.com
- Banggood: banggood.com

Some additional hardware will be required, such as: a plastic card on which to mount the Arduino and shield assembly, solar charger, and LiPo battery. Also the weather meters, solar shield housing, and battery will need to be mounted on some sort of mast. (The author used an inexpensive camera tripod, available from amazon.com.)

For construction ideas and more details see the excellent weather station tutorial at https://learn.sparkfun.com/tutorials/weather-station-wirelessly-connected-to-wunderground.

Assembly Notes

1. The wifi shield should be plugged into the Arduino board, and the weather shield should be plugged into the wifi shield. This arraignment allows for the most free air flow around the pressure and humidity sensors and, moreover, distances the temperature sensor from possible heat sources on the Arduino board and wifi shield.
2. The Arduino-wifi-weather-shield stack should be mounted, along with the solar charge controller and battery, on a plastic card (figure 6) such that the entire assembly can be easily slid into the solar shield housing (figure 5).
3. The power output leads coming from the charge controller should be connected directly to the 5 Volt and GND pins on the weather shield header. (See figure 5.) While the Sparkfun Redboard, weather, and wifi shields will all work powered from a 3.8 volt LiPo battery, the power must be connected directly to the “5 Volt” rail. Powering through the VIN pin requires an input voltage of 8 to 16 volts, which the LiPo battery cannot provide.
4. Average daily solar energy falling on a solar cell varies greatly with latitude and climatic conditions. The selection of a 20 Watt solar cell is based on the climatic conditions of western Oregon, where periods of heavy overcast skies can last for days. During such times typical solar cells produce as little as 5% of their rated output. The solar cell must be sized to provide adequate output to both power the electronics and keep the battery charged. Desert areas subject to daily clear skies may require only a 5 Watt (or less) solar cell.

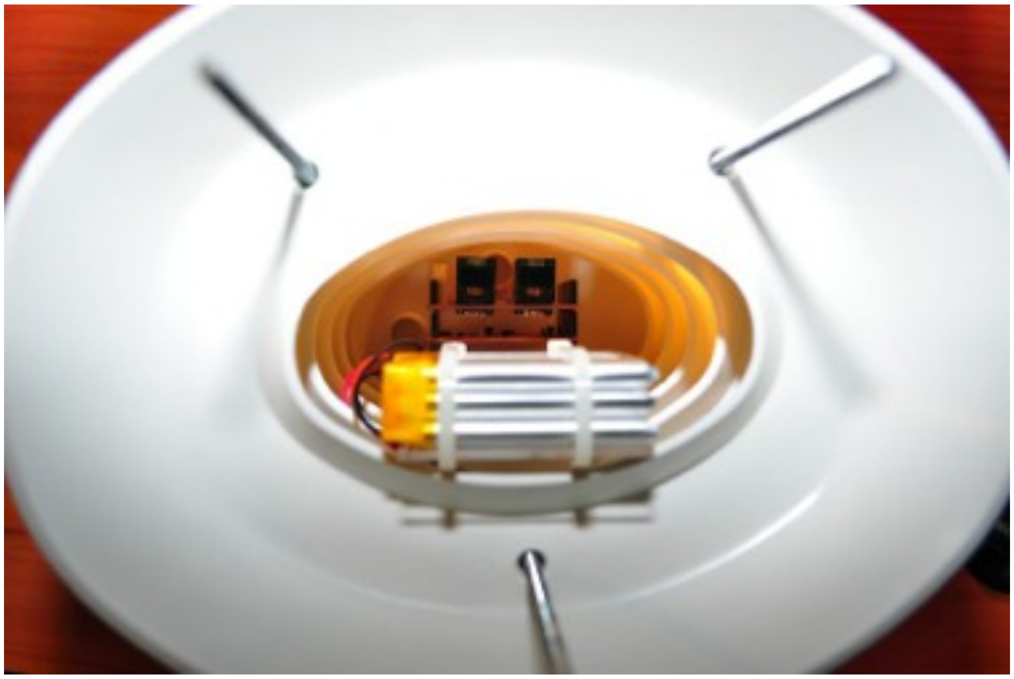

Figure 5. Photograph showing how weather station electronics fit inside the Ambient solar radiation shield assembly.

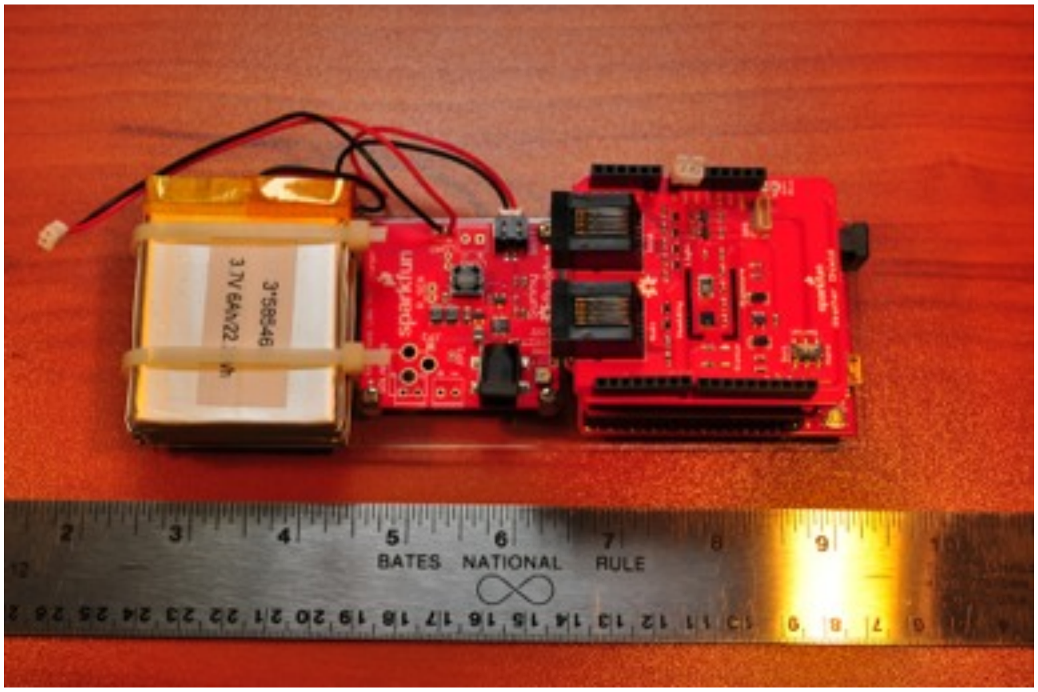

Figure 6. Photograph showing weather station electronic components. Note how the weather shield and wifi shields are plugged into the Arduino board at the bottom.

Software Installation

1. Get the latest version of the Arduino IDE from https://www.arduino.cc/en/Main/Software. The IDE is available for Mac, Windows, and Linux operating systems.
2. Before the weather station sketch can be compiled, three device driver libraries must be installed. When you run the Arduino IDE for the first time, the IDE creates a folder **Arduino/sketch/libraries**. Look for this folder in your home documents folder. Download the two sensor libraries from Sparkfun's web site at https://cdn.sparkfun.com/assets/b/5/9/7/f/52cd8187ce395fa7158b456c.zip. Unzip the downloaded folder and copy the HTU21D humidity sensor and MPL3115A2 pressure sensor library folders into your Arduino/sketch/libraries folder.
3. Download the ESP8266 wifi shield device driver library from https://github.com/sparkfun/SparkFun_ESP8266_AT_Arduino_Library/archive/master.zip. Unzip the downloaded folder, and place the unzipped folder in your Arduino/sketch/libraries folder.
4. If you are using the Sparkfun Arduino Redboard as the micro-controller, consult Sparkfun's web site for details on using the Redboard with the Arduino IDE. You may need to install some custom USB drivers from Sparkfun.
5. Using a USB cable connect the computer running the Arduino IDE to the Arduino's USB port. Be sure to disconnect the power leads from the charge controller *before* connecting the USB cable.
6. Follow the Arduino instructions for connecting the Arduino to your computer and configuring the IDE for the Uno. Normally this means navigating to the Tools->Board menu item and selecting the Uno, and going to the Tools->Port menu item and selecting USB port to which the Uno is connected.
7. Before the WeatherStation sketch can be compiled and downloaded to the Arduino, the Arduino EEPROM must first be initialized with your wifi configuration and weather station password. From the Arduino folder on the project website, download the **SetupEeprom** Arduino sketch and open it in the Arduino IDE.
8. Look for the preprocessor defines at the top of the file. These defines will need to be edited. Look for the following three lines

   ```
   #define STATION_PASSWORD "{weather station password (required)}"
   #define SS_ID "{your wifi access point's SSID}"
   #define WPA_PASSWD "{your wifi access point's WPA password}"
   ```

   SS_ID and WPA_PASSWD are self-explanatory. STATION_PASSWORD is a password that must be sent to the weather station in order for changes of SSID or WPA password to take effect. Changes to station password, SSID, and WPA password are added to the URL for the weather station. Although this is not a secure method, it is expected that changes to the wifi configuration will not occur frequently enough for abuse to be a concern.
9. Compile and download the sketch. If the sketch is running properly, in the IDE serial monitor window you should see status messages indicating progress of writing data to EEPROM.
10. From the Arduino folder on the project website, download the **WeatherStation** Arduino sketch, and open it in the Arduino IDE.

11. Compile the sketch and download it to the Arduino. Now complete the next section "Installing the Server Software", and then return step 12 below.
12. Once the server software is installed, you can verify if the sketch is running properly by opening up the Arduino IDE's serial monitor window. You should see the startup message followed periodically by the data string the Arduino sends in response to HTTP requests.

This completes the weather station software installation.

Installing the Server Software

Dependencies

- The server software should be installed on a recent Linux distribution such as Debian. (The author has successfully developed and installed the software on a Raspberry Pi running the Raspbian operating system.)
- Apache2 should be installed and configured to allow serving HTML documents from the user's public_html folder.
- Rrdtool should be installed - type **sudo apt-get install rrdtool**
- Python 2.7 usually comes pre-installed in virtually all Linux distributions. Type "python" at a command line prompt to verify Python has been installed.

Software Inventory

The software items that need to be installed on the server are

Repository html folder:
- weather.html
- index.html
- static/chalk.jpg

Repository bin folder:
- createWeatherRrd.py
- weatherAgent.py
- startwea
- stopwea

The above software items may be found in the respective folders in the project repository.

Installation

Note that the following installation procedure assumes that weather HTML documents will reside in the user's public_html folder. Typically the full path name to this folder will be something like "/home/{user}/public_html".

1. In the public_html folder, use **mkdir** to create a folder named "weather" to contain the weather HTML files from the project repository html folder.
2. Download from the html folder in the project repository all the contained files and folders. Move the contained files and folders to the folder created in step 1.
3. The output data file (see figure 4) gets overwritten frequently and should not be stored on disk. Rather it is dynamic content that should be stored in the temporary

file system resident in ram. Similarly the graphic image files get overwritten frequently and should also be stored in the temporary file system. Assure that the server /tmp folder gets mounted to the temporary file system. This can be done by adding the following line to the /etc/fstab file

```
tmpfs /tmp tmpfs nodev,nosuid,size=50M 0 0
```

4. The web service cannot freely access files and folders outside of the public_html folder. Therefore, in the public_html/weather folder create a symbolic link to the temporary file system by running

   ln -s /tmp/weather dynamic

   The web service will look for dynamic content by following the “dynamic” link to the /tmp/weather folder.
5. If it does not already exist, use **mkdir** to create in the user home folder a folder named “bin”. For example, the full path name should look like “/home/{user}/bin’.
6. From the bin folder in the project repository, download the contained files into the bin folder created in step 6. In most Linux installations the user’s bash profile will automatically add the user’s bin folder to the command search path. If such is the case, then the agent can be started up by simply typing **weatherAgent.py** followed by ENTER.
7. In the user home folder create a folder named “database”. In the bin folder run the python script **createWeatherRrd.py**. Running this script creates an empty round robin database file where the agent will store weather data as it arrives from the weather station. This script should be run once and then kept in a secure place. Running it accidentally at some future date will result in *total loss of all previously stored data*.
8. In the user’s home folder create a folder named “log”. For convenience two scripts have been provided to make it easy to turn the agent on and off. The **startwea** script starts up the agent and causes all diagnostic output and error messages to be written to a log file in the log folder. The **stopwea** stops the agent from running.

This completes installation of the server software.

References and Resources

The Sparkfun hookup guides provide a wealth of detailed application notes for Sparkfun products. The following hookup guides should be consulted before building the weather station:

- Weather Shield https://learn.sparkfun.com/tutorials/weather-shield-hookup-guide
- ESP8266 Wifi Shield https://learn.sparkfun.com/tutorials/esp8266-wifi-shield-hookup-guide
- RedBoard https://learn.sparkfun.com/tutorials/redboard-hookup-guide
- Sunny Buddy Solar Charger https://learn.sparkfun.com/tutorials/sunny-buddy-solar-charger-v13-hookup-guide-
- Weather station tutorial https://learn.sparkfun.com/tutorials/weather-station-wirelessly-connected-to-wunderground

The following tutorials are useful for more in depth understanding of programming Arduino:

- C programming https://www.gnu.org/software/gnu-c-manual/
- Arduino programming https://www.arduino.cc/en/Tutorial/Foundations

The following tutorials are useful for more in depth understanding of the server software:

- Javascript http://www.w3schools.com/js/default.asp
- PHP http://www.w3schools.com/php/default.asp
- HTML http://www.w3schools.com/html/default.asp
- Rrdtool http://oss.oetiker.ch/rrdtool/
- Python http://greenteapress.com/thinkpython/thinkpython.html