

**OpenWeather++**

**OpenSource Forcasting**

**Tyler Hoeficker**

# 1.0 Executive Summary

The OpenWeather++ project is a collection of technologies based around individual and crowd sourced atmospheric data collection. These technologies include Dallas Semiconductor’s One Wire protocol and it’s integrated circuits, Arduino compatible AT Mega based microcontrollers, and Argent Data Systems weather monitoring circuits. While the use of propriety technologies is currently necessary the project aims to be as open source as possible. All source code used in this project is published under the GNU Public License and is freely available to the public via GitHub.com.

The project aims for three fundamental goals. The first is to provide anyone with basic understanding of Arduino based microcontrollers a guide to construct a standardized weather station. The second fundamental goal of the OpenWeather++ project is to centralize the libraries used in this project. Before this project the required libraries where fragmented across the web with no clear explanation to their use or implementation of version control. The project will host an open repository on GitHub for distribution and version control, as well as a build guide with examples. The third and final fundamental goal is to provide a web based API and framework for accessing the organizing the data into information

Two future goals for the project include expansion into other platforms, such as the ARM based Raspberry Pi, and community based source maintenance.

A future fourth goal for the project is to develop an open source community to maintain the source code and provide further development.

# 2.0 Problem Definition

The OpenWeather++ project arose out of library issues when constructing a weather station kit purchased from SparkFun.com. Most of the libraries where incomplete or outdates with poorly written usage examples. The increased number of weather monitoring modules only furthered this problem, which eventually led to a need for a central repository for source maintenance.

After assembling and updating the necessary libraries came the issue of correctly using methods defined in the libraries. Most code assembled was uncommented and without documentation.

# 3.0 Scope

## 3.1 Target Audience

OpenWeather++ is targeted towards individuals with a willingness to learn, or currently holding the following skill sets: Soldering, C++ based programming, and microcontroller driven development. OpenWeather++ provides these individuals with a base source code, examples and documentation to construct a fully operational weather station.

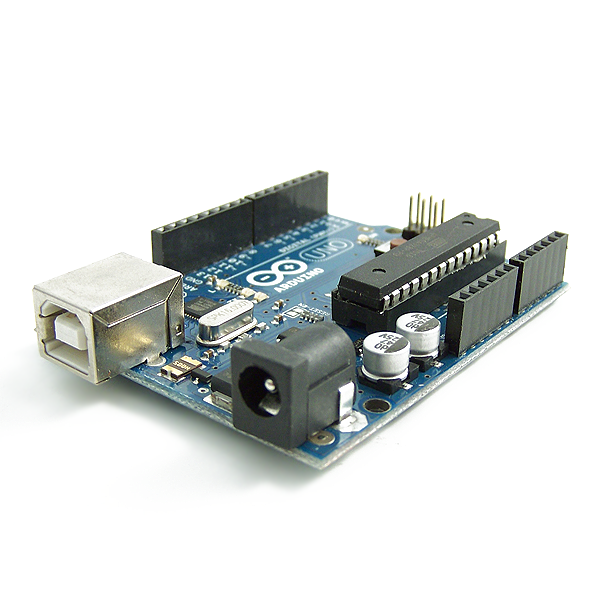
## 3.2 Development Scope

At the time of the first version of this document Aug 14, 2012, OpenWeather++ is in a prototype alpha stage.

# 4.0 Requirements Analysis

## 4.1 Hardware

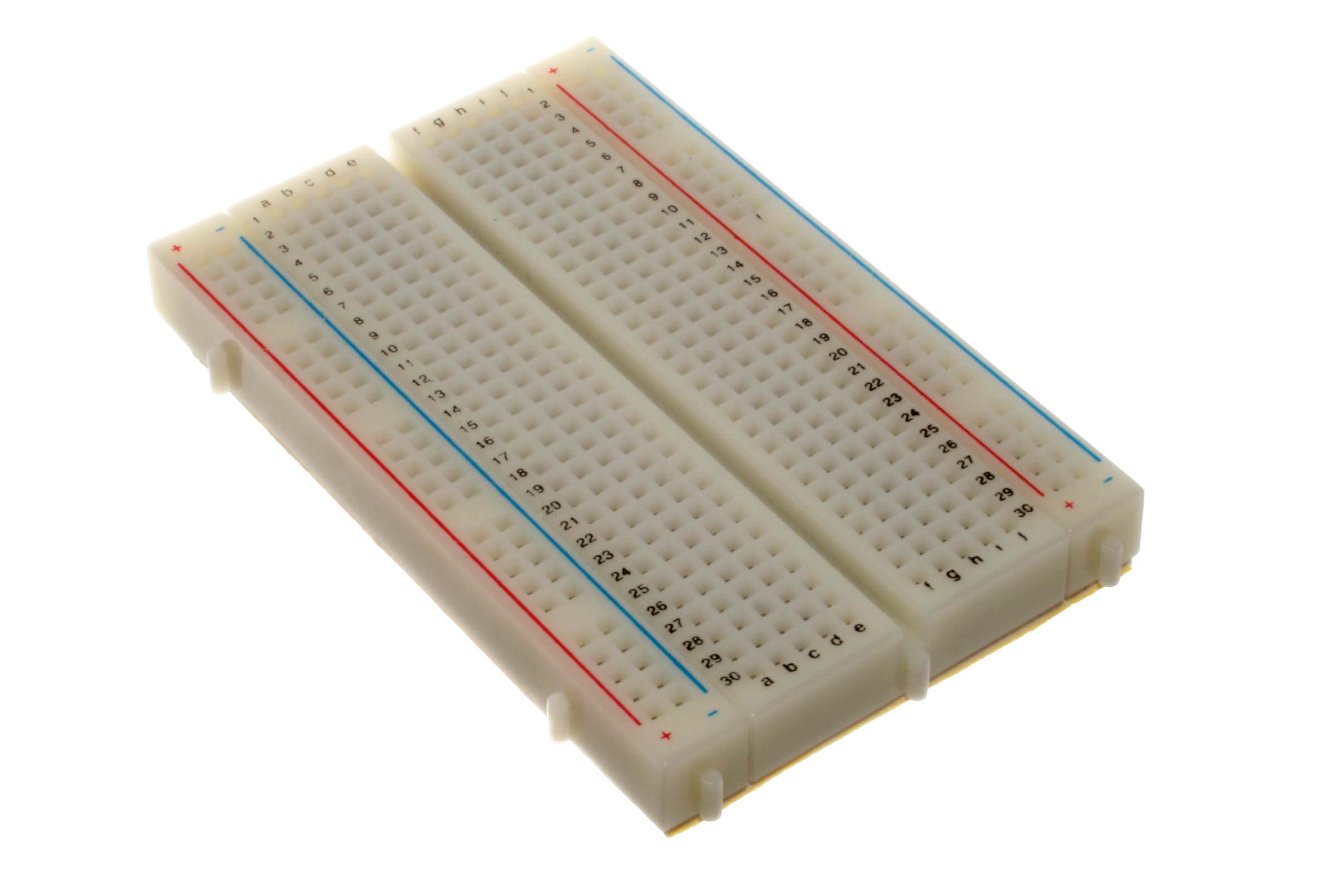
### 4.1.1 Arduino Compatible Prototyping Board (Arduino UNO)



Description: Atmel AVR Atmega 328 based development board. The Arduino UNO, while not required for a final prototype is necessary for testing and development. The Arduino UNO contains 6 analog input/output pins, 13 digital input/output pins, pulse width modulation powered by a Atmel Atmega328 processor. The Atmega328 processor is capable of storing 32 kilobytes of programming instructions that easily fits the 10 kilobytes of programming instructions needed for OpenWeather++.

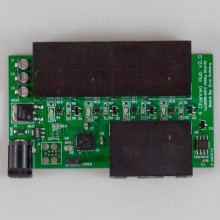
<http://www.arduino.cc/>

### 4.1.2 Solder less Breadboard and Pin Jumpers



Description: Solder less breadboards are used for prototyping electronic designs before creating more permanent PCB layouts. The pin jumpers provide a means of connecting electrical components to each on breadboard. Breadboards can be purchased from local electronics shops and a variety of online retailers including amazon.com, mouser.com, adafruit.com and Sparkfun.com

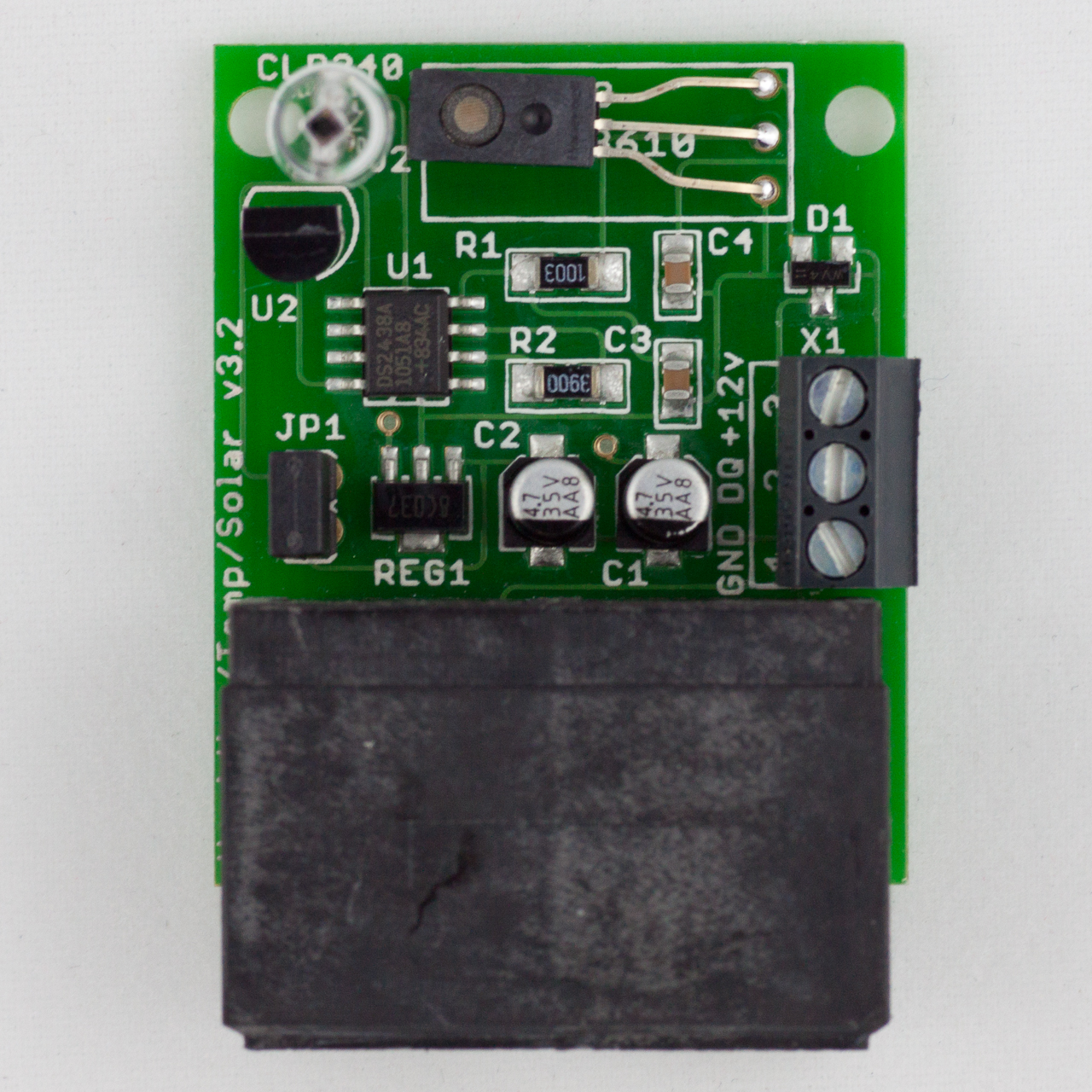
### 4.1.3 Hobby Boards 6 or 4 Channel Power Hub



Description: A hub to provide the required power of five volts across the One-Wire bus for operating multiple modules. Connections provided via RJ-45, with a barrel jack to accommodate DC in from the accompanying AC to DC power adapter.

<http://www.hobby-boards.com/store/products/4-Channel-Hub.html>

### 4.1.4 Hobby Boards Humidity/Temperature/Solar Sensor

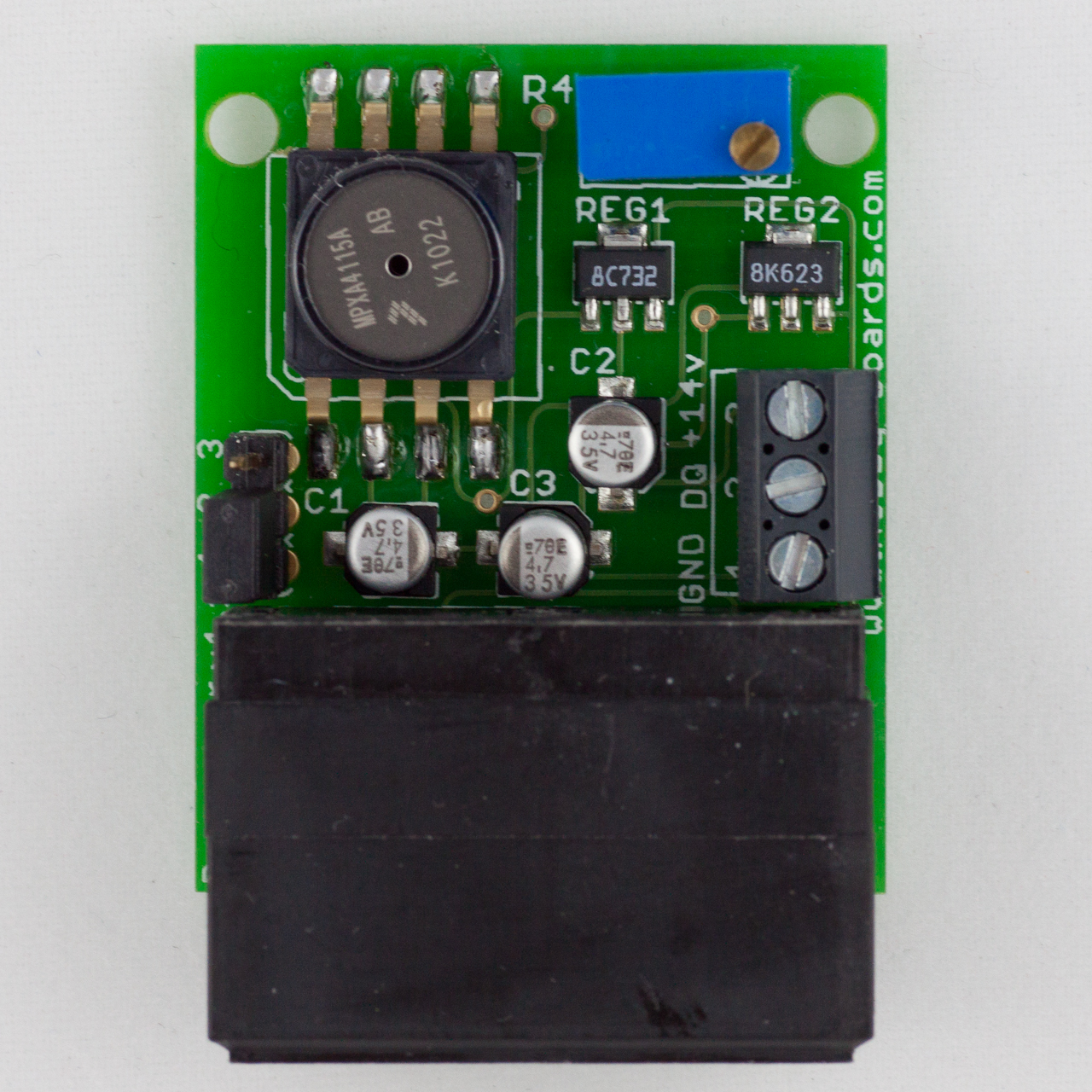


Description: HobbyBoards.com humidity, temperature, and solar sensing module provides a single node for gathering humidity, temperature and visible light levels. The temperature is calculated using a DS1820 digital thermometer communication over the OneWire protocol. Humidity sensing is provided by a HIH-3610, and visible light levels measured by a Clarion CLD240 photodiode. Being that the last to sensors are analog sensors, a DS2438 battery monitor is used to convert their analog data into a digital form. The module required 5 volts to operate.

<http://www.hobby-boards.com/store/products/Humidity%7B47%7DTemp%7B47%7DSolar.html>

*Note: HobbyBoards.com may have changed or will change certain sensors on the module in the future. It is likely that the provided code in OpenWeather++ will work, however it would be still be wise to check your version to see if any code modifications are needed.*

### 4.1.5 Hobby Boards Barometer



Description: The Barometer measures atmospheric pressure, and can be used on its own or as a part of your total weather monitoring system. It reads pressure from 28-32 inHg (948-1083 mb). Its resolution is approximately 0.01 inHg (0.34 mb). Since indoor and outdoor pressure are typically close to the same, we recommend installing this board indoors to avoid possible sensitivity to moisture in an outdoor setting.

This device comes with dual RJ45 connectors, as well as screw terminals, for easy connection to your 1-wire network, and requires power of 15-24 volts DC.

<http://www.hobby-boards.com/store/products/Barometer.html>

### 4.1.6 Hobby Boards Lightning Detector



Description: The Lightning Detector counts lightning strikes, giving you a fascinating new perspective on the electrical storms in your area. The lightning detector needs to be well grounded. We house ours in a length of PVC pipe, which is inexpensive and easy to assemble.

With the 24" antenna this lightning detector will be able to pick up lightning more than 50 miles away. This device comes with dual RJ45 connectors for easy connection to your 1-wire network.

The Lightning Detector requires a 9v battery (not included), this battery should last for at least a couple of years.

<http://www.hobby-boards.com/store/products/Lightning-Detector.html>

### 4.1.7 Development Computer

Description: An x86 based computer to run the Arduino API for development. The computer must be able to run the Arduino API as well as communicate to the Arduino UNO via USB.

### 4.1.8 Web Server

Description: This requirement is optional if the OpenWeather++ build is not using the Wunderground API. The webserver will not only serve as a data collection node, but will also be responsible for serving the weather data to end-users.

## 4.2 Software

### 4.2.1 Arduino Integrated Development Environment (IDE)

Description: Graphical Integrated Development Environment used for development and prototyping with Arduino boards.

<http://arduino.cc/en/Main/Software>

### 4.2.2 Dallas Semiconductor One-Wire Arduino Library

Description: Library for AVR chips that allow the processors of the Arduino to speak and listen to the One Wire protocol.

Included in OpenWeather++ Repository

### 4.2.3 Dallas Semiconductor DS2438 Smart Battery Monitor Library (Patched)

Description: Updated library for communicating and switching modes on the DS2438 Smart Battery Monitor used to meter humidity and solar levels.

Included in OpenWeather++ Repository

### 4.2.4 Dallas Semiconductor DS2409 Library One Wire Powered Hub Library

Description:

Included in OpenWeather++ Repository

### 4.2.5 Dallas Semiconductor DS2423 Analog to Digital IC Library

Description:

Included in OpenWeather++ Repository

### 4.2.6 An Operating System running Apache Web Server, MySQL, and PHP

Description: This requirement is necessary if the OpenWeather++ implementation is to require personally hosted data. Examples: Windows Vista or greater, Linux, BSD variants, Mac OS X

### 4.2.7 MySQL version 4 or greater

Description: An open source database management system used for storing the weather data sent from an OpenWeather++ station. Advanced users may substitute their own database management system, but documentation will not be provided.

<http://www.mysql.com/>

### 4.2.8 PHP version 5 or greater

Description: A server side language used for the backend distribution and services for web apps and web pages.

http://www.php.net

### 4.2.9 Apache Web Server

Descript: An open source web server for serving the weather data to a web app.

http://httpd.apache.org/

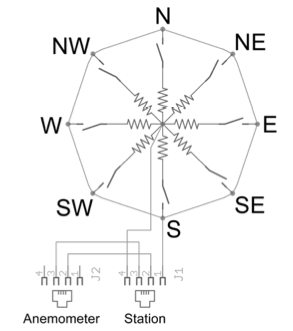
# 5.0 Design Specifications

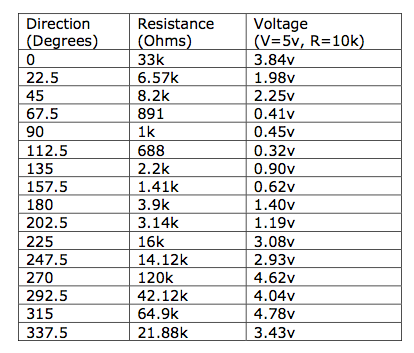
## 5.1 Anemometer Wiring and Programming

The Argent Data Systems produced anemometer connects to the Arduino development board via an RJ-11 cable which is compatible with both RJ-11 and RJ-45 ports. The center two pins on the RJ-11 are used for communicating with the Arduino Uno development board. On the positive side of the connection a 10k ohm pull up resistor is necessary for reading interrupts. Each time the anemometer completes a full revolution it closes a magnet-based contact that can be measured as a digital interrupt by the Arduino Uno.

## 5.2 Wind Direction Wiring and Code

Wind direction is read in from the Argent Data Systems Wind Vane. The wind vane uses eight magnetic switches to measure the position of the wind vane. Each magnetic switch holds a different resistance value and two magnetic switches can be tripped at the same time allowing for a sixteen calculable positions. The wind vane uses the two outer wires on the same RJ-11 socket that the anemometer uses. As with the anemometer, reading the connection requires a 10k ohm pull-up resistor on the positive line. Exact wiring is as follows: Arduino analog pin 5 is connected to RJ-45 port 8 along with a 10k pull up resistor connected to a 5v line. RJ-45 pin 4 is connected to the Arduino ground pin.





*­­Note: These resistances are used read via analog in on the Arduino and their values mapped accordingly. When the weather station is emplaced it must be have its read in north aligned with true north.*

The following code is used to calculate wind direction

#define PIN\_VANE 5 // Analog 5

#define uint unsigned int

#define ulong unsigned long

#define MSECS\_CALC\_WIND\_DIR 1000

ulong nextCalcDir; // When we next calc the direction

ulong time;

#define NUMDIRS 8

ulong adc[NUMDIRS] = {26, 45, 77, 118, 161, 196, 220, 256};

// These directions match 1-for-1 with the values in adc, but

// will have to be adjusted as noted above. Modify 'dirOffset'

// to which direction is 'away' (it's West here).

char \*strVals[NUMDIRS] = {"W","NW","N","SW","NE","S","SE","E"};

byte dirOffset=0;

void setup(void)

{

nextCalcDir = millis() + MSECS\_CALC\_WIND\_DIR;

unsigned long startTime = millis();

delay(100);

}

void loop(void)

{

if (time >= nextCalcDir)

{

calcWindDir();

nextCalcDir = time + MSECS\_CALC\_WIND\_DIR;

}

}

void calcWindDir() {

int val;

byte x, reading;

val = analogRead(PIN\_VANE);

val >>=2; // Shift to 255 range

reading = val;

// Look the reading up in directions table. Find the first value

// that's >= to what we got.

for (x=0; x<NUMDIRS; x++) {

if (adc[x] >= reading)

break;

}

x = (x + dirOffset) % 8; // Adjust for orientation

Serial.print("Dir: ");

Serial.print(strVals[x]);

Serial.println();

}

## 5.3 Rain Gauge Wiring and Code

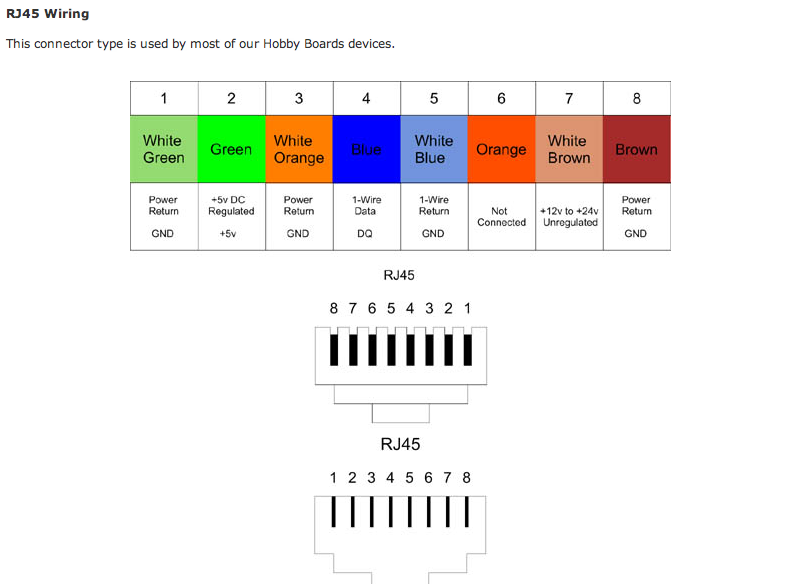
The rain gauge module uses a tip bucket design, meaning that every time the bucket fills with liquid it tips over to either side contacting a connection and causing a digital interrupt. The gauge itself connects to the Arduino by way of a CAT2 cable and RJ-45 port. Data is sent from RJ-45 pin 8 to Arduino digital pin 4. Note that this connection requires a 10k pull up resistor connected to a 5-volt power supply. The connation is grounded at RJ-45 pin 4.

To Recap:

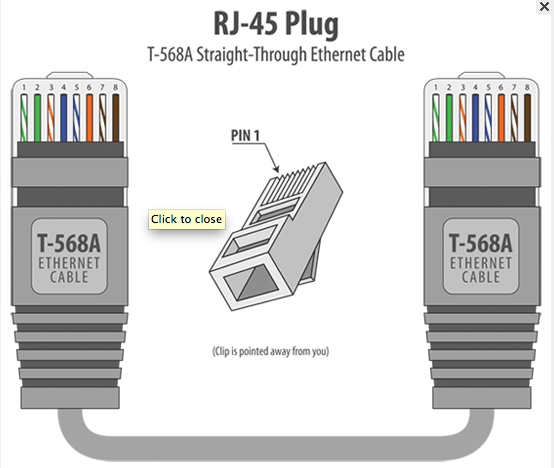
Arduino pin 4 -> 10k pull up resistor to 5v -> RJ-45 pin 8

Arduino ground (or power supply ground) -> RJ-45 pin 4

## 5.4 Wiring of One Wire Based Devices

One Wire modules rely on the use of RJ45 ports along with CAT 5 or CAT 6 cables for communication and power. A “Strait-Through” Ethernet wiring scheme is to be used with the Ethernet cables.

Power is supplied from the One Wire Hub while data is transferred over Ethernet pins four and five (these are colors blue and blue/white respectively). Pin four provides data and is connected to digital pin four on the Arduino while pin five is connected to ground.



## 5.5 Rain Gauge Wiring and Code

# 6.0 External Documents

# 7.0 Development Methodology

spiral or agile

# 8.0 Testing

## 8.1 Rain Gauge Testing

The rain gauge is designed to trip a contact when its tip bucket fills to a designated level. According to the manufacturer’s documentation each tip represents .011 inches of precipitation. To test weather the source code complies with this measurement complete the following steps.

Step 1: Set a variable in the source code to increment each time the tip bucket trips the rain gauge relay.

Step 2: Assign a *serial.print(increment variable name)* to the end of the rain gauge function.

## 8.2 Anemometer Testing

Initial testing of the anemometer is done by spinning the anemometer module while observing the results of writing the anemometer variable to serial with *serial.write();* The faster the gauge is spun, the higher its report rate should be.

After the anemometer has passed its initial test, a more thorough test must be performed to judge its tolerance. A motor vehicle with speedometer, computer for reading back serial communications, two individuals, and calm weather conditions are needed to perform this second test. The first individual will operate the vehicle at 4 designated speeds: 10mph, 20 mph, 30 mph, and 40pmh. The second individual will hold the anemometer out the passenger window of the vehicle to simulate wind at the predestinated speeds. A tolerance of 5% is acceptable at any speed.

## 8.3 Wind Vane Testing

# 9.0 Maintenance and Life Cycle

# 10.0 Definitions of terminology used in this document

AVR: Advanced Virtual RISC. Architecture of microcontrollers produced by the Atmel Corporation.

ARM: Advanced RISC Machines. A company that designs microprocessor architectures.

PHP: Pre-Hypertext Processor. A backend web server language.

MySQL: A relational database management system used for storing data.

Ethernet: A Layer 2 network standard.

CAT2/5: A Layer 1 network standard.