The process of gathering weather data by the use of technology has been around for decades. However with the ever expanding growth of its advances, the cost of the equipment for collecting the data has not diminished. This lack of decrease in equipment costs has no justification behind it when available resources, though not complete, could be fabricated together to create the same or even better system.
**Introduction**

A weather station is used to collect different meteorological data to help forecast the weather. Since technology advances so fast, it has enabled personal weather stations to be bought and used in homes, schools and other personal places. Some of these weather stations can be advance packed with copious amount of sensors that the general user would not need. However, there are personal weather stations that come with just the right amount of sensors to enable the user to have a general idea what the weather is going to be. The only problem with a lot of these personal weather stations is the price for the amount of functionality it brings. This is the niche we are trying to fill with our personal weather nodes.

The weather nodes are little translucent white cubes that can be dispersed around to collect accurate real-time weather data. The weather nodes all interconnect wirelessly to each other creating a network of nodes to collaborate as one unit. This allows the end user to collect different kinds of weather data per node. Each node will have the ability to hook up one of the custom sensors that are specifically made for the weather nodes. However, the user also has the ability to use other sensors that work with Linux, by interfacing with the weather node software.

Each one of these nodes can be configured to the user’s needs via an in-browser configuration page from the designated master node. Users could use only a single node and still have a full, functioning system. Although you can connect as many slave nodes as you want to a single master node. Each node has the ability to be toggled to become a master or a slave node, making the possibilities for multiple master nodes to create isolated data collection.

The nodes feature an interesting way to relay information to the user by using colors. Since the whole node is a white translucent cube, it uses an RBG led to defuse color onto the cube. An example of how a user would get information from the cube via color would be if the user was setting up a salve node from a master node and did not know the range it could communicate in. The cube would be green for excellent signal and accurately transitioning into red for being completely out of communication rage.
Existing Products

Davis Instruments 6152 Wireless Weather Station

The Davis Instruments 6152 has a large amount of sensors for tons of functionality packed into its system. It is broken up into two devices, one being the wireless console and the other is the wireless sensors that can be placed all together from up to 300 meters in line of sight. The system comes with sensors for humidity, rain, temperature, wind, direction and barometric pressure for a starting price of $535.95. [1] Additionally you can get optional sensors that will allow for measuring evapotranspiration, leaf wetness, soil moisture, solar radiation, UV radiation and UV meds.

The wireless console has a 6 x 3 ½ inch LCD grayscale display to view the weather data on. The screen has on screen graphing for averages, current conditions or highs and lows for the last 24 hours, days, months or years. It also contains all the information from the sensors displayed with a 2 ½ second delay. The console also has the ability to be connected to a pc to view and log the weather data via serial interface (WeatherLink). Software from Davis for communication to the console can be purchased for additional $165 [2]. However, there is some cheaper software from a 3rd party developer that has all the features of WeatherLink and additional ones like sending custom weather updates to your twitter status. [3]

Netatmo

Netatmo claims to be “The first Made for iPhone Weather Station” [4] making it have an interesting market place for personal weather stations. For $179.00 you get the IOS app and two wireless modules. The indoor module measures temperature with +0.3°C accuracy, humidity with +%3 accuracy and CO2 with +5% accuracy. The outdoor module measures temperature with +0.3°C accuracy, humidity with +%3 accuracy, barometric pressure with +/-1mbar and sound dB with the range of 35db to 120db. The whole system gets updated every 5 minutes to give you updated data.
Ambient Weather WS-1090

The WS-1090 is a compact, feature rich and cheap personal weather station going only for $99.99. The system has sensors to measure wind speed, wind direction, temperature, humidity, rainfall and barometric pressure. The data from these sensors is updated every 48 seconds wirelessly to the 5.75 x 4.25 inch grayscale touchscreen weather console. The weather console also has a usb port to connect to your pc to view the data with their free software. This will allow you to save as many sample intervals as you want if you record them on your pc. If you wanted to record them on the system itself, it is able to handle up to 4080 data points.

Oregon Scientific WMR200A

The WMR200A is a midrange personal weather station that has some interesting features. It comes with a standard wireless touchscreen grayscale console that displays its data with. The WMR200A also comes standard with sensors to measure humidity, temperature, wind chill, wind speed, wind direction, barometric pressure and rainfall. However for the price of $349.99 the sensors are surprising not accurate and the base only gets a update every 60 seconds. For example the temperature accuracy is +/- 2°C and the humidity sensors are up to 7% off. Nonetheless the whole system can also connect up to 10 wireless sensors up to 300 feet away. To power those wireless sensors it also comes with a solar panel to power them along with their battery backup. A nice feature that this system has the ability to read WWVB-60 signals to get the time from the atomic clock at Fort Collins, Colorado.

Even though its sensors may not be that accurate, the system makes up for it in its software. It has the ability to be hooked up to the PC via a usb cable. This will make the system communicate with the free software that it comes with called Virtual Weather Station (Base Edition). This software displays any data from the console on your computer. It allows you to record and save data as well as stream it to weather underground where you can view you weather from the web.
WeatherHawk 916 Wireless Weather Station

The WeatherHawk 916 is considered a professional personal weather station. It runs for a starting price of $2,745 [8], but it has features that other personal weather stations do not. It comes with sensors to measure air temperature, relative humidity, barometric pressure, rain, solar radiation, wind speed and wind direction. The sensors accuracy is quite impressive with temperature +- 0.5°C, humidity +- 3% and rain resolution at 1mm. The system does not come with a base station with a console display; instead it comes with a wireless receiver base that plugs into your computer. The weather station has a range of ½ mile while within line of sight and over 7 miles with optional external antenna configurations. The WeatherHawk 916 also has the ability to store around 32,000 data points inside of its 128 kbytes of nonvolatile Flash RAM. This is the only weather station in this paper that claims to have real-time weather data. This is contrary to other weather stations that can have over a minute of wait time before weather data gets updated.

MK-III-LR

The MK-III-LR is a long range wireless weather station costing around $995 [9]. It provides data from up to 1 mile away within line of sight. It also has a close to real-time update interval of 2 seconds. The system comes with an array of accurate sensors including wind speed, wind direction, temperature, relative humidity, barometric pressure and rain fall. Compared to other systems its sensors have high accuracy, for example the temperature sensor is +- 0.25°C and the humidity sensors accuracy is +- 2%. The station also comes with a solar panel and battery that they guarantee 60 days of operation without out sun. Just like the WeatherHawk 916 this system comes with a wireless receiver that you can hook up to your pc to view the data.
Architecture

Functional Specification

*Hardware*

Figure 1: Cube Layout for cutting. The 7 holes on the squares above are where the magnets will be. These magnets will create the connections for the sensor cubes to the main cube.

- ATMega328-PU
- Raspberry PI
- ShiftBrite (SPI)
- DS2745U+
  - I2C address = 0x48
- Barometric Pressure Sensor - BMP085 (I2c)
  - I2C address = 0x77
- Ambient Light Sensor - TEMT6000 (Analog)
- Humidity sensor - HH10D (I2C)
  - I2C address = 0x51
- Compass Module - HMC6352 (I2C)
  - I2C address = 0x42
- Temperature sensor – DS18S20 (1-Wire)
- White Translucent Cube to house the weathers bread board and raspberry pi
  - 2600Mhz Lipo with solar cell on top
Software
- Custom Debian Linux distribution
  - Minified for maximum ram
  - Pre install packages to make everything work together
  - Node.js as the server for the whole cube
  - Hostap for second WIFI card
- GUI
  - Will be able to be accessed in browser by connecting to its WIFI.
  - Auto detects the nodes and shows each one and its sensors data real time.
  - When a new weather node connects it will create a node in the GUI in real time.
  - No refreshing is required since the webpage is dynamic.
- Server
  - Auto detects what sensor is connected and then has it configure the data correctly to transmit the data to the master server.
  - Knows if it’s the master node or slave node via toggle switch.
  - The master node also serves the webpage to the client via web browser.

Comparison of Raspberry Pi to Arduino

The table below shows the comparison of the two systems that we researched to possibly use as our solution while making the weather nodes. This data shows the Raspberry Pi has extreme power advantages to the Arduino. However, the Arduino has more Digital I/O pins that support PWM and has analog inputs.

<table>
<thead>
<tr>
<th>Features</th>
<th>Raspberry Pi</th>
<th>Arduino Uno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>ARM1176JZF-S (armv6k)</td>
<td>ATMega328P</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>700 MHz</td>
<td>16 MHz (8MHz without crystal)</td>
</tr>
<tr>
<td>RAM</td>
<td>256Mb (SDRAM)</td>
<td>2Kb (SRAM)</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>17 (of which 1 provides PWM output)</td>
<td>14 (of which 6 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>700mA</td>
<td>12mA</td>
</tr>
<tr>
<td>Price</td>
<td>$35</td>
<td>$30</td>
</tr>
</tbody>
</table>

Arduino Uno

For this comparison we used the most current project board by Arduino called “Arduino Uno”. The Arduino Uno provides a platform for programming embedded applications to the ATmega328. Running with the clock speed of 16 MHz the ATmega328 also provides eight GPIO\(^1\)’s and six PWM\(^2\)

---

\(^1\) GPIO – General Purpose Input Output

\(^2\) PWM – Pulse Wave Modulation
outputs, which also dual function as six more GPIOs. The board also includes six analog inputs, an ICSP header and a micro USB connection to communicate via serial to device.

Figure 2 – Block Diagram of the Arduino Uno [10]

The way users would communicate with the ATmega328 chip would be through the ATmega15U that is taking care of the USB to serial communication between the two chips. Arduino has developed an IDE for easy programing the ATmega328 via this communication. [11]

Analog Inputs

The pins (23-28) on the Atmega328 are dedicated to Analog input. For example one can use a potentiometer that could be hooked up to 5v power, ground and one of the analog inputs to get a reading, noting that the max voltage for the analog inputs is 5v. The way the reading works is through a built in analog-to-digital converter, the voltage first comes in between 0-5v. Then it is converted to digital, in the form of a number between 0-1023. This number range is the same range when dealing with the Atmega328 PWNs.

Figure 3 - ATmega328’s pin out the diagram to visualize the inputs and outputs. [12]

3 ICSP – In-Circuit Serial Programming
ATmega328 CPU

The Atmega328 is based on the Harvard architecture 8-bit RISC. This processor has an internal clock speed of 8Mhz. However, when connected to an external (16Mhz – 22Mhz) crystal you are able to get those speeds by flashing an edited version of the Arduino boot loader. The default for the Atmega328 is to use the external clock. The chip uses the Atmel AVR instruction set with each instruction being 16 bits long.

Raspberry Pi

The Raspberry Pi is a full computer that is the size of a credit-card. This tiny computer two USB ports, Ethernet, HDMI, RCA Video, 3.5 mm Audio jack, SD Card slot and an array of GPIOs. The computer has 256MB of ram that is shared between the CPU and GPU. This gives the computer the ability to display 1920x1080 through the HDMI without any performance issues. The most compelling feature of this computer is that its price is set for only $35.

Raspberry PI CPU/GPU is an armv6k clocked at 700Mhz, this gives an extreme increase in power compared to the speed of a ATMega328. However, it does lack the interface pins that the ATMega328 has. The Raspberry PI only has one functional PWN (without reusing some of the PWMs that are used for audio), this makes interfacing with I2C, SPI and other hardware interfaces hard to do unless you wanted to bit bang the protocols.

This small computer can use any operating system that supports ARM architecture. The armv6k also support operating systems that are complied with ARM hard float. This will give the whole system better performance since this CPU utilizes hardware. Without floating point, the operating system tries
to emulate in software floating point hardware.

**ATMega328P**

While the ATMega328P is not the centralized component to the node, it is the one that actually communicated with all the sensors to be able to relay the information to the raspberry pi. The basics for how the ATMega328P server the information to the raspberry pi is inside its code, and is constantly be listening on two of the analog inputs to sense if there is any voltage change. When the voltage is zero then there is no device hooked up to the node, this lets the ATMega328P know that it now needs to listen for a device to connect. When a device connects it puts power to what one of the analog pins with a unique voltage to the sensor. This is how we can tell what device is connected to run the proper code to update the Raspberry Pi with new information. After the device connects, it then reads the information from the sensor as fast as the sensor will allow it to be read. If there are any changes to the data received from the sensor, it then pushes out the new data through serial to the raspberry pi. This is done with all of the listed sensors expect for the analog sensors in this paper using I2C or 1-Wire protocol.

The ATMega328P controls the analog sensors similar way it senses what device is connected if it is connected regarding to the digital sensors. The chip listens to see if there is a change in voltage to know when the sensor is plugged in. Then it starts read the sensor every update tick to provide real time information back to the Raspberry Pi when there is a change in data. The only difference between how these sensors are detected for a connection from the digital sensors is that the analog sensors do not need to supply voltage through a resistor to provide a way of identifying what it is. This is because there is a separate plug for the analog sensor.

**Raspberry PI**

The Raspberry PI is the central part of this project. It is used to host the web client, communicate between the ATMega328P and to handle all of the configuration and data storage. The design flow of the communication between the Raspberry PI and the ATMega328P is through serial communication. This utilizes pins 14 (TXD) and 15 (RXD) on the Raspberry PI. Since the Raspberry PI does not contain any over-voltage protection and is not 5v tolerant we had to put 1k Ohm resistor between pin 15 (RXD) and the ATMega328P’s TX line. We also had to put another 1k Ohm resistor between pin 15 and ground making the voltage not go over 3.3 volts.

On the networking side, the Raspberry PI uses two usb wireless cards that support G,B and N. The chip that runs on each of the wireless cards is a Realtek 8192CU. This particular chip was not supported by the distribution of Linux that is being used for this project (Raspbian Wheezy compiled for ARM with Hard Float binaries). This created the challenge of getting that chip to work on that distribution without switching to a more expensive wireless card. However, we got the wireless cards to work by finding the kernel files for it and configuring it into the system manually. The way the project
utilizes these wireless cards is having one connecting to access point while the other creating its own access point and then bridging them together. After the user configures the master node, it will be able to scan and connect to other local access points around the node to connect with to enable internet access through the web client interface.

The last role the Raspberry Pi has is to host and run the client’s web interface. This is done with a series of programs written in Javascript, html5 and C. The server’s backend is the central part of the web client design. One of the responsibilities for the server is to serve the client to people accessing the node within a web browser. Another responsibility is to handle all of the communication between the ATmega328P to listen to the data and update the client when needed. The server also handles all of the saving and loading of the clients data, while also controlling the ShiftBrite through the ATmega328P.

**ShiftBrite**

ShiftBrite is a module containing a 140 degree viewing angle RGB LED with 8000mcd per color. It also has on the module the A6281 which is the IC that controls the RGB with 10-bit PWM for each color [14]. The module uses SPI for communication to the A6281 to control the RGB, making it a perfect solution to use with the Raspberry Pi since it lacks PWM pins to control a RGB LED.

**BMP085**

The Bosch BMP085 is a low-power and high-precision barometric pressure sensor. Its accuracy can be ± 0.03 hPa while measuring a range of 300 to 1100 [15]. This sensor also has a built in temperature sensor included inside the IC. The protocol to talk to the BMP085 is I2C making it only need two wires to communicate with the IC.

**TEMT6000**

This is a popular light sensor that you can find in a lot of consumer electronics today including your cellphone or notebook. It has a spectral bandwidth range of 360nm to 970nm with a wavelength of peak sensitivity of 570nm [16]. This sensor uses analog output to relay its data to a microcontroller.

**HH10D**

The HH10D is a Humidity sensor that has high accuracy of ±3% relative humidity. This sensor uses I2C protocol to communicate its data through. This particular module has a capacitive type humidity sensor, a CMOS capacitor to frequency converter and a EEPROM used to hold calibration factors [17]. The humidity sensor also has an impressive resolution of 0.3% relative humidity.

**HMC6352**

The Honeywell HMC6352 is a compass module that uses 2-axis magneto-resistive sensors, algorithms for heading computation and its onboard firmware to give back the heading through I2C protocol [18]. This compass module has a heading resolution of 0.5 degrees. This module also features stray magnetic field protection and temperature compensation.
DS18S20

This high-precision digital thermometer by Maxim Integrated uses a unique protocol called 1-Wire. This interface only requires 1 wire to for sending and receiving data through. On top of that, that same data line can be used to power this IC through a process they call parasitic power making it a true 1 line device. The thermometer has high accuracy of +/- 0.5°C with 9 bits of resolution [19]. Just like most 1-wire devices from maxim integrated, this sensor has a unique 64 bit unique serial code to address it by. This makes it possible to run multiple 1-wire devices with still only one wire.

DS2745U+

The DS2745U+ is an 8 pin IC that can measure voltage, current-flow and temperature. This IC is designed for monitoring batteries in one low cost single chip. This chip comes in a SOIC form, making it not able to be dropped into a bread board for testing like a DIP chip would allow. We had to make a break out board for this small chip that is about the size of a grain of rice placed diagonally along the chip. After creating the board we used I2C protocol to communicate with the chip with the ATmega328P.

![Figure 5 – A photo of the board we created to break out the pins to enable communication.](image)

To start communication with the chip you first need to transmit its hardware address, which by default is 0x48 in hex. This however has a programmable address that can be changed after the first communication is established. After the request for transmitting has been connected, then we wrote the memory address that we wanted to read from. On this chip, there is two address for each measurement. For example if we wanted to get the temperature, we would have to read from addresses 0x0A (MSB) and 0x0B (LSB).

Measuring UV

Every weather station seems to give UV index reading. We have the TEMT6000 for light sensing, but that sensor, even though it goes as low as 360nm in the spectral range, its lens protects it from UV and IR waves. [16]

To actually calculate the UV index requires extremely expensive sensors ranging from $50 to $1000+, and the use of two satellites operated by the National Oceanic and Atmospheric Administration measuring the current total ozone amounts over the whole globe. [20] The full spectrum range to measure is 290nm to 400nm, which is the full spectrum of UV-A and UV-B. However, this leaves out other radiation the sun gives off which should be a part of the UV index calculation, such as UV-C. This could be because of the very low amount that reaches the earth and the fact that UV-C’s wavelength range is extremely low (280nm-100nm).
Noticing the fact the sensors just for UV index sensing cost $50 - $1000, it defeats the purpose of this project which is to make a cheap solution to meteorology data collection. Nonetheless, we discovered the ability to use LEDs as photodiodes. This makes it possible to buy cheap UV LEDs and use them as photodiodes. The way this works is by hooking up the LED’s negative lead to positive voltage and the positive lead to low voltage. This will charge up the capacitance of the LED making the LED not come on. After that, we will cut off the voltage of the negative lead and count how long it takes for the voltage to drop to logic zero giving us how intense the light is.

Capabilities

Web Interface

Looking at the picture above you can see that there are grey nodes. Each one of those nodes is represented by meteorological information provided by each sensor. For example, we can have mutable “Temperature” nodes since there can be more than temperature sensor or other sensor that also provides the temperature plugged in at the same time. The web interface can also have a duplicate node (Type and Sensor) but it can be from a slave node. This will be noted on the bottom of the grey nodes what cube this data is coming from.

Another feature of the web interface is to have the ability to view this data on your screen in real time. Users can sit back and watch the data while it gets seamlessly updated without a single refresh using HTML5 web sockets. When a node has new data to be updated with the node will let the user now by fading in and out in a flash like manner for that particular node. This gives the user a easy
way to tell which ones are being updated without having to focus on the screen.

Along with the real time data updates, the web interface also possesses the feature of removing
a node from the users screen if the sensor is unplugged. This will make the screen feel less cluttered
with the removal of stale data. Along with that ability it also naturally comes along with fading back into
place when the sensor is plugged back in.

**Server**

**Weather Cube Networking**

```javascript
function StartHostAP()
{
    if(hostap !== null)
    {
        d.log(2,"Killing HostAP");
        exec('killall hostap',function(error, stdout, stderr) {
            if (error) throw error;
            hostap.kill();
        });
    }

    setTimeout(function(){
        exec('brctl addif br0 wlan5', puts);
        hostap = exec('/root/Hostapd/hostap /root/Hostapd/hostapd.conf', puts);
        console.log('HostAP started at: %d', hostap.pid);
        hostap.on('exit', function(code, signal) {
            console.log('exit with code %s and signal %s', code, signal);
        });
    },1000)
}
```

This function first enables the program Hostapd. Hostapd is software that helps with the
configuration of the Weather Cube’s seconded wifi by putting the device into AP mode. This allows users
to connect to the Weather Cube’s wifi to gain access to the web interface. It also allows the user to
share the Weather Cube’s main wifi connection if it is connected to the internet via another wireless
modem.

Looking at the first if statement we have to check to see if there is already a child process of
Hostapd spawned by Node and if there is we need to kill all processes of it. When first writing this
function we ran into the issue of killing the PID of the child processes which would end the process but
would not kill all processes related to Hostapd. This causes malfunction of Hostapd if it is started again
without killing all processes. This was solved by installing the package “psmisc” which included a
program “killall” which kills all the processes containing the string that is passed in as the first
parameter.

```javascript
function PingCheck(cb)
{
    exec('ping google.com -c 1',function(error, stdout, stderr) {
        if(stdout.search("0% packet loss") > -1){cb(true);}
    });
}
```
This function is called every time the device is connected back online (able to ping Google). The reason for this is because if the state of the network is changed on the Weather Cube's first wifi (the one that searches for wireless routers to connect to) it needs to be restarted or the bridge connection fails.

```javascript
function ChangeNetwork(type, ssid, key, cb) {
  if (type == "wpa") {
    fs.readFile(`/etc/network/interfaces`, 'utf8', function (err, data) {
      if (err) throw err;
      var lines = data.split(/\r?\n/);
      //line 8 = wpa-ssid
      //line 9 = wpa-psk
      d.log(2, lines[9]);
      if (ssid == "#" || key == "#") {
        lines[8] = "#";
        lines[9] = "#";
      } else {
        lines[8] = "wpa-ssid \"+ssid+"\"";
        lines[9] = "wpa-psk \"+key+"\"";
      }
      data = lines.join('\n');
      fs.writeFile(`/etc/network/interfaces`, data, function (err) {
        if (data[0] != "") {
          if (err) {d.log(3, err); throw err;}
          d.log(1, "Network Settings Saved!");
          d.log(2, "Restarting Network!");
          currentNetworkType = "wpa";
          if (ssid == "#" || key == "#") {
            exec(`/etc/init.d/networking_stop`, function (error, stdout, stderr) {
              d.log(1, "Network Stopping");
              cb();
              if (error) throw error;
            });
          } else {
            exec(`/etc/init.d/networking_start`, function (error, stdout, stderr) {
              d.log(1, "Network Starting");
              cb();
              if (error) throw error;
            });
          }
        } else {
          d.log(3, "Error saving network settings!")
        }
      });
  }
}
```

Looking at the function above shows how we the server has the ability to use any open, WEP or WAP based WIFI node to connect to with its primary WIFI card. Depending on what type the user needs to use create a whole different way of how the Linux box needs to be configured. This came with hours of trial and error to get it right and working well.
Weather Cube Arduino Library

This library creates an easy way for users to be able to take the Weather Cube and program it to do things. It has the functionality of setting call back functions for when there is a sensor update with new data and if a sensor has been disconnected or connected. This makes for the user not have to deal with all the individual communication with hardware that are using various communication protocols.

```c
void sensorUpdate(byte type, float data) {
    if(type == 81) {
        rh = data;
        Serial.println("Humidity:"+fts(rh,2)+"\%:"+"HH110D");
    } else if(type == 119) {
        hg = data/3386;
        Serial.println("Pressure:"+fts(hg,3)+"\:"+"BMP085");
        altitude = calcAltitude(data);
        Serial.println("Altitude:"+fts(altitude*3.28084,2)+"\:"+"BMP085");
    } else if(type == 120) {
        temperature = data;
        Serial.println("Temperature:"+fts((1.8*temperature)+32, 2)+" F:"+"BMP085");
    } else if(type == 33) {
        heading = data;
        Serial.println("Heading:"+fts(heading,1)+" getDir(heading)");
    } else {
        Serial.println("Err: "+String(type)+" Err "+fts(data,4));
    }
}

void statUpdate(byte type, byte stat) {
    Serial.println("Stat:"+String(stat)+"+String(type));
}
```

After we created this library we used it for the Weather Cube project. The way the flow of the code works is that if there is an update then it sends that update to the Raspberry PI via serial communications. This promotes an asynchronous environment which makes Node running on the Raspberry PI work very smooth since it is event based.
Continuing Forward

There are copious amounts of things that could be added on to this. For starters, we would like to finish the main scope of the project that was outlined in the proposal, creating a very good base line to make this project extremely adaptable to any developer or user. Some of the missing features include web client features like recording data and saving it. Another main feature that was left out was multiple cubes networking with each other. All the foundation to make that happen is there, however the time was not there to be able to finish.
Works Cited


