Wireless Temperature and Humidity Sensors

Evan Carlin
1500 North Warner Street
Tacoma, WA 98406
evan@carlin.com

Gabe Lennon
glennon@pugetsound.edu

Mark Gilbert
University of Puget Sound
Tacoma, WA 98416
mgilber@pugetsound.edu

Matthew Bogert
University of Puget Sound, 1500
N. Warner St. Tacoma, WA,
98416
mabogert64@gmail.com

ABSTRACT

This paper describes and evaluates a system of temperature and humidity (T&H) sensors, and an associated web application built for the Lelooska Foundation. This project incorporates hardware and software to design, implement, and deploy a reliable climate monitoring system for the Foundation at a low cost.

1 INTRODUCTION

The Lelooska Foundation and Cultural Center is dedicated to preserving and sharing Native American history through Living History events and by curating a small museum. The museum is housed in an old wood framed building without proper insulation, which is damaging to their collection of fragile artifacts. To acquire funds for a new building so that the artifacts would be better protected, they first had to demonstrate a need. To prove this, they needed to collect data on the conditions of the museum, which included information about T&H. Prior to this project they had individual climate sensors that they had to check manually and record on pen and paper. This was not an ideal way to collect sufficient data that could be both analyzed and utilized in a grant application. Professional alternatives were not an option because the costs were beyond their budget. Our purpose was to provide them the affordable solution to measure T&H. They asked for an automated system that could record T&H in different areas of the museum, as well as a way to easily access the data.

This project consists of designing, constructing, and deploying T&H sensors to monitor the conditions in the museum. Along with the hardware system is a website for both the management of sensors and viewing of data. The network of sensors record T&H, transmit the readings to a local gateway, and the gateway then pushes the data to a web server for permanent storage and data analysis.

2 Background

There currently exist numerous imperfect options to monitor T&H in a museum. On one end of the spectrum lie cheap, low-tech sensors that display readings on an onboard screen but offer no way to store readings over a long period. On the other end lie sophisticated systems that meet the Foundation’s needs, but exceed their budget.

2.1 Lelooska's Previous Solution

Before we developed our system, the Lelooska Foundation used inexpensive sensors from a local hardware store. The sensors displayed the current T&H on a small digital screen and an employee at the museum had to periodically record the readings with pen and paper. This was extremely inefficient in both data retrieval and completeness.

It was time consuming for an employee to walk around to the numerous display cases and record the readings. Likewise, the transcription from pen-and-paper to digital format took up additional work time, and presented the opportunity for clerical error. In addition, data was only recorded when employees were onsite and had extra time to record information. This meant that climate conditions before and after work hours, and on holidays, were missed. The lack of a full, 24-hour climate record provided a limited picture as to what conditions the artifacts were subject to. This solution worked for a short period, but would not provide the functionality needed to fully actualize their goals.

2.2 Expensive Commercial Options

Due to the problems stated in the previous section the museum had investigated other options. DataQ Instruments has commercial systems for climate monitoring. The Model EL-WiFi-TH sensors record T&H, as well as transmit the data to online software for data visualization [7]. However, only one of the cheaper models cost $165 for a single sensor. Scaling costs, the Foundation would have invested thousands of dollars by the time they had a full system implemented. Other commercial systems provided similar functionality, but all were either expensive, or time-consuming to receive data [3] [4] [11].

2.3 Hobbyist IoT Options

Numerous hobbyists have implemented T&H sensors. These different implementations provide a wealth of knowledge on how to implement such a system, but none were perfectly suited for the museum. Some systems were not designed in such a way as to allow for monitoring of multiple display cases in one system [14]. Others did not functionally meet our needs and only displayed information on LCD screens, instead of transmitting readings [9] [12]. All of these systems were created by technically proficient hobbyists and meant to be used by other technical people. Our implementation required a solution that could be deployed to a non-technical end user. Though helpful, the final product could not be a carbon-copy from the online community.

3 Implementation

Our project consists of three main components: The sensor nodes, a gateway computer, and a website backed by a database.

Throughout the project we used pre-built libraries and frameworks wherever possible. Each individual component of our project alone is not exceptionally novel, as they are built on extensive libraries. What is novel is how we integrated all of these individual parts into a greater system.

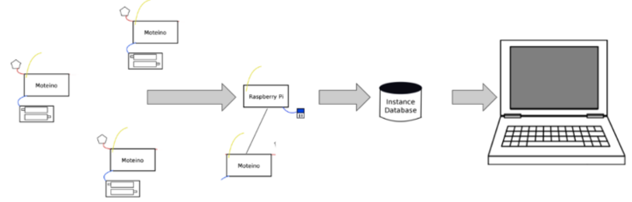


Figure 1: Diagram of the data flow from sensor nodes, to gateway, to website.

3.1 Sensor Nodes

The first iteration of our project was a single micro-processor/transceiver wired to multiple T&H sensors (see Appendix 6). This idea was too rigid and did not provide the flexibility needed to monitor the entire museum efficiently. We would have had to drill holes through the cases so that the sensors could be wired together. Drilling holes in cases and having wires laying around the museum was quite concerning to the Foundation. Likewise, wiring multiple sensors to one Moteino lead to vastly decreased battery life. Hard-wiring power was not an option due to inaccessible outlets. There was also a concern of data accuracy in wiring too many sensors to a single sensor node.

So we transitioned to a wireless modular approach (see Figure 2). Our two main constraints for selecting sensor node hardware was that they had to be wireless and run for as long as possible on battery power. To meet that end we opted to go with Moteino development boards, using an RFM69 transceiver, and DHT22 T&H sensors. The device is powered by 3 AA lithium-ion batteries.

The nodes spend the vast majority of their lives in "deep sleep." They boot up at a random interval (to reduce packet collision) every 12-15 minutes, take a reading, transmit the reading to the gateway, and go back to sleep. Under these conditions the batteries should last for 4 years.

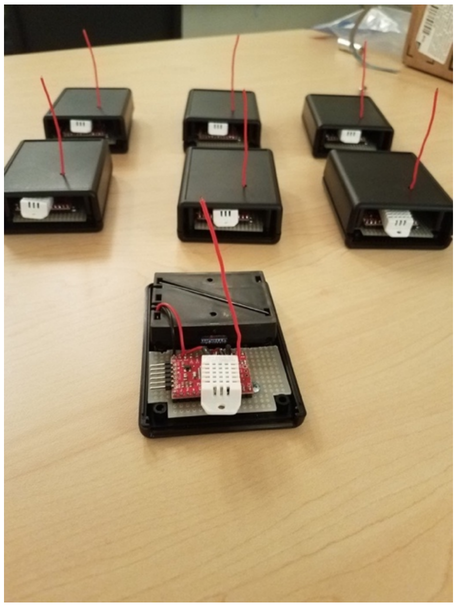


Figure 2: Our final sensor node design.

Figure 3: A sensor inside the museum.

3.1.1 Moteino

The Moteino product is a development board with an Atmel ATMegae328p micro-processor and RFM69W radio transceiver. We selected the board due to its cheap cost, low-power use and extensive library support.

The ATMega328p MPU is the same processor found on many Arduino development boards. This simplified the development process considerably. We could use the Arduino IDE without needing to install any extra libraries. Likewise, it meant that libraries and solutions to problems developed specifically for Arduinos worked on our chip with no modification.
The RFM69W transceivers provided a reliable low power way to transmit readings. The transceivers operate on an unlicensed 915 Mhz band. This meant that the 2.4 / 5 Ghz WiFi in the museum would not interfere with our sensors. Likewise, there is an excellent library for the RFM69 radios developed by the creators of the Moteinos. This meant that sending a packet was a

simple as doing radio.send(“a packet”). Likewise, network reliability systems such as packet ACKing came right out of the box. We changed radio.send(...) to radio.sendWithRetry(...) to achieve this.

3.1.2 LowPower Library

In order to achieve the smallest power draw the Moteinos are put to sleep whenever they are not reading or transmitting. To put the Moteinos to deep sleep we are using a library called LowPower from Rocket Scream [10]. This library allows the processor and all peripherals except for a small hardware timer to be powered down. The hardware time is left running and can wake the processor back up when a reading needs to be taken. This cuts down on our current draw from 10 mA at full wake and transmit to 0.007 mA in deep sleep.

3.1.3 DHT22

The DHT22 are low cost ($10) T&H sensors [6]. They provide ±0.5°C temperature accuracy and 2-5% humidity accuracy. Adafruit has developed an excellent library for working with the sensors that make reading them a breeze. To read the humidity is as simple as dht.readHumidity().

The DHT22 was wired to an analog pin for power. This was done to reduce current draw during sleep (from 0.057 mA to 0.007 mA). So, each time the sensor is read the power pin must first be set to HIGH and then one must wait 800ms to let the power settle. Then a reading can be taken and the power pin set back to LOW.

3.2 Gateway

Inside of the museum we are using a Raspberry Pi 3 as the gateway computer between the sensors and our web-app. The Raspberry Pi is connected to power and Wifi [2]. Connected to the Pi via USB is a Moteino that receives the packets from the sensor nodes and writes them to the USB.

The Pi runs a Python script, which reads messages from the USB port the gateway Moteino is connected to [8]. It then appends a timestamp to the readings, formats them as a JSON and POSTS the data to an API on our website.

3.3 Webapp

The web application component of this project was built using Ruby on Rails. Our Rails app is hosted on an EC2 instance running an Apache web-server. Rails was selected due to its fast speed of deployment and large community package support.

3.3.1 System security

The museum required that the sensor information only be accessible by the museum staff. To accomplish this, we implemented username and password authentication. This was done using the Clearance Authentication gem. It provides strong security and nice features such as email links for forgotten passwords and the ability to update passwords.

We also implemented two different privilege levels on the website. As museum staff changes administrator level users can add and delete users on the website.

3.3.2 Renaming Sensors

To maximize the efficiency of the relatively small number of sensors, the museum can rename the sensors in the database to accurately reflect their position within the museum. This allows the museum to determine which cases they need data from at a given time, and allocate the sensors appropriately.
To accomplish this, we made the ‘node_name’ attribute in the sensor table to not cascade on update. This means updates will not apply to previous readings. Which allows historic data and new data to remain accurate.

3.3.4 Graphing

The museum wanted full access to their data in an easy to digest format for grant writing. To accomplish this, we gave them the ability to download all readings over a specific date range as an Excel file [1]. This allows them to perform the exact data analysis they need for each individual grant proposal.

The museum also wanted to be able to easily view day-to-day T&H changes. We chose to provide graphs of the data from each sensor over the course of the most recent day using the ChartKick gem [5]. This allows them to understand how different actions such as conducting tours (large groups of people in a small space) or turning on fans change the conditions in the museum.

4 Evaluation

4.1 The Price Tag

The grand total for constructing the entire system was approximately $600. From initial comparison to the DataQ option we saved the Foundation over $1,000. However, the savings add up further when breaking down the costs of implementation. The gateway setup (Raspberry Pi, sensor, and casings) was approximately $75 of the total; each sensor placed in the museum cost roughly $35. In the event seven more sensors are constructed for the whole museum setup, the Foundation only has to pay an additional $245. The DataQ option (at $165 a sensor) would have resulted in an additional $1155 to the grand total. We provided a solution that is financially beneficial in both the short term and long term.

4.2 System Stability

4.2.1 Hardware

The sensor nodes have not sent abnormal data, and they have all connected to the system. The Pi functions and transmits data to the website well. Error logs have remained empty ever since implementing them, which is a cautiously optimistic sign. T&H readings from the DHT22s have all been within reasonable ranges.

4.2.2 Software

From running on a personal instance, to migrating it to the Foundation's website, there has not been a crash. We intentionally developed the system to run stably on its current software. If no one attempts to update the software, we can promise the website will stay online. The script running on the Pi has been proven to work as intended, rebooting itself and getting data flowing to the website again after having lost power.

4.3 Battery Efficiency

After running the sensors for several weeks, batteries have not yet needed to be changed. Using batteries was a primary concern of the museum; they presumed batteries would need to be replaced every three weeks, adding considerable investments in both time and money. Given that batteries do not have to be replaced every few weeks, we further accomplished the goal of cost minimization.

4.4 Packet Collisions

Throughout the design process we have considered handling packet collisions. We decided a single retransmission would be sufficient for the system. In the event of a collision the sensor node trys one retransmission. We selected this method because in our testing this created the best balance between battery use and packet loss. However, we have observed packet collisions on a near-daily basis: approximately one collision per day.

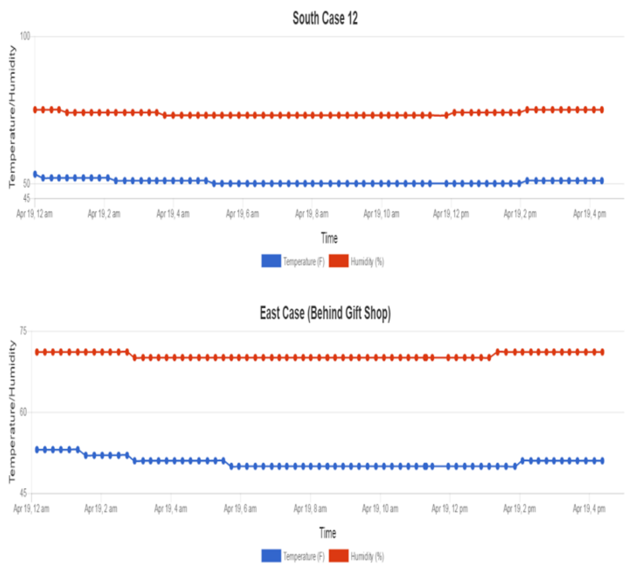


Figure 4: A lost reading due to packet collision.

As seen in Figure 4 right before 12:00 PM, there is an observably larger gap between data points for both sensors, implying collision.

Functionally, this is not an impediment; one collision a day means the Lelooska Foundation loses 0.04 to 0.2 % of their readings. However, this is a peculiar phenomenon to observe. The system currently has 10 sensors, each transmitting once every 12 to 15 minutes. These transmissions only takes a fraction of a second, leaving a small window for collision to occur.

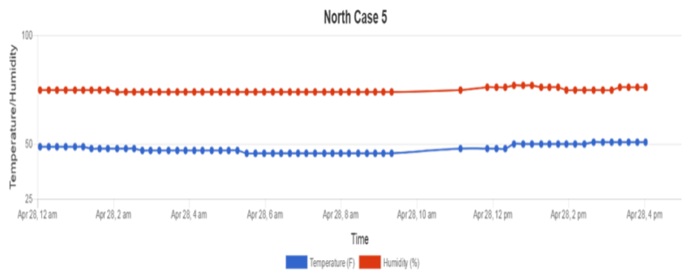


Figure 5: Packet loss due to interference

Another form of packet loss that we have observed is due to transmission interference that occurs inside the museum. We have only observed this once but it was for a 3-hour period (pictured in figure 5). During this time period numerous other sensors also had lost packets. Although, interestingly "North Case 5" experienced by far the greatest packet loss.

Unfortunately, the source of interference is unknown, which means we haven't fully ruled out the possibility of a hardware malfunction. However, given that over several weeks, this is the only significant functional gap in data, we are optimistic of the system's efficiency.

5 Discussion

We feel that our implementation of this project met and, in many ways, exceeded the expectations set by our customer. The initial concept for the sensor network stated that "Real-time, detailed measurements probably aren't needed. Getting daily summaries with the high, low, and average humidity and temperature would be more than enough" [13. Our implementation builds on top of these specifications considerably. Currently our system has been deployed for 4 weeks and has taken ~32,000 readings. The proposal additionally suggested that the site be remotely configurable so off-site technical work could be done if changes are required without necessitating the intervention of museum staff. We implemented this by turning on port-forwarding to the Raspberry Pi and assigning it a static IP address. This allows technical support to remotely SSH into the Pi.

A great deal of attention was put into ensuring that this project could be as stable as possible. Given our experience and time allotted we are relatively optimistic in providing substantial reliability. The project itself delivered what the Lelooska Foundation had asked for, providing a stable flow of easily digestible information that requires very little time investment from the foundation itself. It is true that other climate monitoring systems could provide higher resolution readings, however this was not a necessity given the Foundation's requirements for this system. In this sense, hardware is one of the few truly hard limits on our implementation. If for some reason higher resolution data was required, it is very likely that a different T&H sensor would need to be used, which might in turn necessitate the selection of other different hardware as well. Apart from this (so far) entirely hypothetical limitation, the majority of this system was designed with scalability firmly in mind. The code on the sensor nodes, gateway, and Pi is all easily modifiable by an informed programmer, and the web app is built on a well-documented framework.

Overall, we believe that our system compares extremely favorably to commercially available systems designed to work on this scale. The needs of the Foundation are met, the system is easily operable and maintainable for non-technical users, and systems are in place to allow for any technically-minded people with relevant knowledge to extend our system in meaningful ways.

6. Future work

6.1 Exterior Sensors

We discussed the prospect of providing sensors for the outside surroundings of the museum as well. Providing data on both the exterior and interior would further bolster data insights. However, we ran out of time in finding a viable solution for retrieving that data. The Moteino sensors would have suffered in exposure to the variable and humid conditions of Washington; each rainy day would pose the risk of short-circuiting the sensor. Another option considered was taking routine readings from the Weather Channel (via weather.com) and adding those to the database. However, that would have been a functionally incomplete approach. Because weather can vary over small geographic spans, measurements for Vancouver, Washington, as a whole would not have provided as detailed of a picture as measurements at the exact geographic coordinates of the museum. Constructing a weather-resistant exterior sensor would be a beneficial extension of this project. The software is in place, so it would be largely hardware-focused effort.

6.2 More Extensive Data Visualization

The Foundation cared primarily about the raw data, because they did not know how grant organizations would want to see their climate trends. For that reason, our graphing remained rather rudimentary. However, since we had access to all of the data, there lay countless possibilities for data visualization. We could have even used a Tableau extension and enabled the museum to store all of their data visualization via a Tableau server. Given that there are only two variables being recorded, Tableau would be quite bulky for the museum's needs, but could also provide many opportunities for data insights.

6.3 Dynamic Mapping

From a UX perspective, we considered implementing a dynamic map of the museum for the front page of the dashboard. It would have shown which cases had sensors, their most recent readings, as well as indicators for optimal/suboptimal conditions (e.g. red cases are at a dangerous temperature/humidity). However, as a stretch goal that would have required additional research into UI/UX, this did not manifest.

7 Conclusion

While the individual pieces of the hardware are not novel, we developed an innovative and reproducible product that is both competitive to commercial options and cost-effective. Along with this the client is very pleased with the result, having received over thirty-thousand readings as of writing this paper. Coming into this project we wanted to deliver a product that was, at a minimum, stable and able to take readings and we thoroughly exceeded this ambition.

This project was also beneficial to our own learning as we were exposed to new coding languages and hardware. Most of us did not know Ruby or had much hardware experience or knowledge. We also gained valuable insight into working with a real client.

8 Acknowledgements

8.1 The Lelooska Foundation

We would like to thank the Lelooska Foundation for entrusting us with this project, and for all the assistance they provided over the semester. We especially want to thank them for laying an Ethernet cable to the museum which made this project possible.

8.2 Professor Richards

Professor Richards, as a member of the Lelooska Foundation board of directors, was extremely helpful as a proxy between us and the Foundation. He helped facilitate communication as well as help inform the Foundation about some of the technical decisions we made. He also migrated the museum's site onto an EC2 instance, which made placing our website alongside the museum's site an easier process. Along with funding the hardware costs, his assistance was paramount in helping us deliver fully functioning solution.

8.3 LowPower Lab

We have to extend our gratitude to this robust community of developers. Having never worked on a hardware implementation before, we would not have been nearly as successful without the great deal of information provided by the Low Power Lab community.

REFERENCES

[1] #362 Exporting CSV And Excel - Railscasts". *Railscasts.com*. N.p., 2017. Web.

[2] Abhiaram, Shaba. "Sabhiram/Raspberry-Wifi-Conf". *GitHub*. N.p., 2017. Web.

[3]"Amazon.Com: Meade Instruments TS33C-M T&H Sensor With LCD, White: Home & Kitchen". *Amazon.com*. N.p., 2017. Web.

[4] "Amazon.Com: Room Alert 3E: Home Improvement". *Amazon.com*. N.p., 2017. Web.

[5] "Chartkick". *GitHub*. N.p., 2017. Web.

[6] "DHT22 Temperature-Humidity Sensor". *Adafruit.com*. N.p., 2017. Web. 11 Jan. 2017.

[7] "EL-Wifi-TH Wireless T&H Data Logger". *Dataq.com*. N.p., 2017. Web.

[8] Kedros, loannis. "Running Python Script At Boot". *Embedded Day*. N.p., 2017. Web.

[9] Leitao, Richard. "Wireless Temperature Sensor". *Instructables.com*. N.p., 2017. Web. 11 Jan. 2017.

[10] "LowPowerLab/RFM69". *GitHub*. N.p., 2017. Web. 11 Jan. 2017.

[11] "Monnit Wireless Temperature Sensor - Commercial Coin Cell Powered | MNS-9-W1-TS-ST". *Monnit.com*. N.p., 2017. Web.

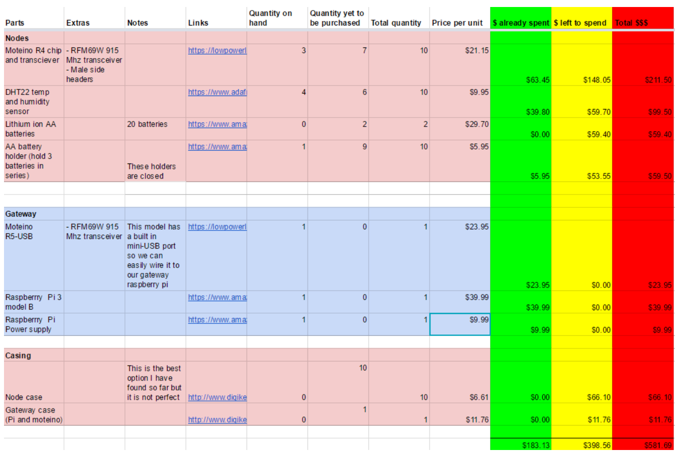
[12] "Portable Arduino Temp/Humidity Sensor With LCD". *Arduino Project Hub*. N.p., 2017. Web.

[13] Richards, Brad. "CS 440 Projects". *Cs.pugetsound.edu*./~brichards/Classes/440/project_pitches.html., 2017. Web.

[14] "Secret Arduino Voltmeter – Measure Battery Voltage". *Provideyourown.com*. N.p., 2017. Web.

Appendix

1.



Parts	Extras	Notes	Links	Quantity on hand	Quantity yet to be purchased	Total quantity	Price per unit	$ already spent	$ left to spend	Total $$$
Nodes										
Moteino R4 chip and transciever	- RFM69W 915 Mhz transceiver - Male side headers		https://lowpowerl	3	7	10	$21.15	$63.45	$148.05	$211.50
DHT22 temp and humidity sensor			https://www.adafi	4	6	10	$9.95	$39.80	$59.70	$99.50
Lithium ion AA batteries		20 batteries	https://www.ama	0	2	2	$29.70	$0.00	$59.40	$59.40
AA battery holder (hold 3 batteries in series)		These holders are closed	https://www.ama	1	9	10	$5.95	$5.95	$53.55	$59.50
Gateway										
Moteino R5-USB	- RFM69W 915 Mhz transceiver	This model has a built in mini-USB port so we can easily wire it to our gateway raspberry pi	https://lowpowerl	1	0	1	$23.95	$23.95	$0.00	$23.95
Raspberrry Pi 3 model B			https://www.ama	1	0	1	$39.99	$39.99	$0.00	$39.99
Raspberrry Pi Power supply			https://www.ama	1	0	1	$9.99	$9.99	$0.00	$9.99
Casing										
Node case		This is the best option I have found so far but it is not perfect	http://www.digike	0	10	10	$6.61	$0.00	$66.10	$66.10
Gateway case (Pi and moteino)			http://www.digike	0	1	1	$11.76	$0.00	$11.76	$11.76
								$183.13	$398.56	$581.69

Our list of parts (and respective costs) for the system

2.

```
//Try sending the packet
// If no ACK is received try once more
if(!radio.sendWithRetry(GATEWAYID, payload, strlen(payload))){
  radio.send(GATEWAYID, payload, strlen(payload)); //No ACK so try again
}
```

Our network protocol for the Moteino's

3.

```
if (radio.ACKRequested())
{
  byte theNodeID = radio.SENDERID;
  radio.sendACK();
}
```

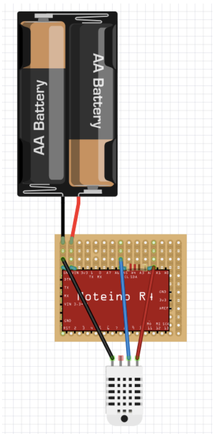
Our network protocol for the gateway node

4.

```
//Turn on power to the pin and let it stabilize before reading
pinMode(POWER_PIN, OUTPUT);
digitalWrite(POWER_PIN, HIGH);
delay(800);

//Take the readings
float h = dht.readHumidity();
float t = dht.readTemperature(true);  //true => temp. in farenheit
long v = readVcc();

//Turn off power to the DHT22
digitalWrite(POWER_PIN, LOW);
long readVcc() {
  // Read 1.1V reference against AVcc
  // set the reference to Vcc and the measurement to the internal 1.1V reference
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);

  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA, ADSC)); // measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  long result = (high << 8) | low;

  result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  return result; // Vcc in millivolts
}
```

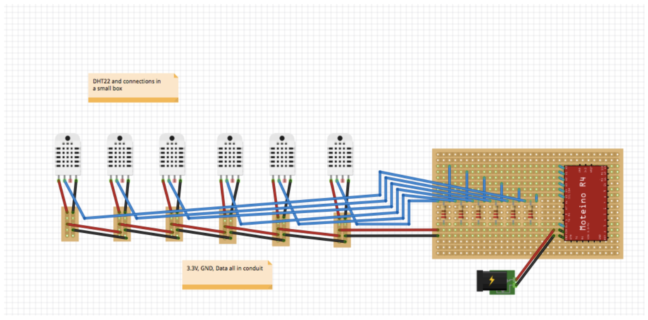
Taking readings, as well as software implementation for measuring voltage

5.



The modular-design sensor (1-node, 1-sensor)

6.


DHT22 and connections in a small box
3.3V, GND, Data all in conduit
Moteino R4

The hardwired sensor model (multiple sensors to a node)