Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 5th, 2017

Remote Sensing in a Body of Water Using an Adafruit Feather

Jordan Adelman, Norissa Lamaute, Dan Reicher, Dallas van Norden, and Matt Ganis
Seidenberg School of Computer Science & Information Systems
Pace University
Pleasantville, NY 10570
ja79745p@pace.edu, nl27157n@pace.edu, dr52244p@pace.edu, dv69896n@pace.edu,
mganis@pace.edu

***Abstract*—The Internet of Things in a few words is the interconnected nature between billions of objects that industry experts refer to as "smart devices". These objects, which can be anything from a smartphone to an SUV, have enabled humans to begin the integration of our tangible world with networked computer systems. Utilizing the capabilities of an Arduino microcontroller, this project created a system to measure, manage, and broadcast temperature and oxygen levels in any given body of water. This study tested the system at Pace University's Choate Pond, and in due time, it will be expanded to the Hudson River. This concept involved a consistent dissemination of data over an indefinite period. A direct result of this was an incredibly large amount of information that needed to be stored in a logical manner. With the combined functions of Big Data management and Internet of Things devices, this project sought to prove an effective method for sensing general environmental data.**

***Index Terms*— Environment Sensing, Internet of Things, Big Data**

I. Introduction

The Choate Pond at Pace University is a 10-foot deep isothermal dimictic body of water. The main goal of this study is to develop a hardware and software package that will (a) measure several variables starting with water and air temperature and (b) create a platform for storing and reporting these findings. Another outcome of this research is the development of a curriculum for other universities who have similar interests.

This study aims to develop the hardware and software needed to develop an environment sensing system using Internet of Things (IoT) and Big Data. Members of the research team soldered, programmed, and installed a waterproof device which serves as a prototype for this project. This device is not only able to detect water temperature at three levels, but is also able to be used as a platform for measuring levels of various elements to deliver a comprehensive picture of the surrounding environment. This includes nitrogen levels or oxygen percentage. Further, the device is able to transmit the measurements to a database for further manipulation.

After this small-scale study in Choate Pond, the project will be expanded to capture the same type of data in larger bodies of water, such as the Hudson River.

We used the idea of potentially testing on the Hudson River to guide us in our development. Our team created a scalable model because while the Choate Pond is merely 10-feet deep, the Hudson River is reportedly up to 200-feet deep in some parts according to the National Water Quality Assessment Program [3].

The second section of this paper summarizes the research that was conducted. Section III discusses the development of a physical and virtual system. Section IV is a brief look at the future of the study, specifically where developments can and should be made. The final section is a summary and conclusion of the aforementioned materials.

II. Research

A. Solar Paneling

A major requirement when dealing with a self-sufficient system is power supply. A crude solution would involve the use of an electric generator. While this is technically viable, generators cannot be truly self-sufficient, and they depend on a significant amount of exposed wire, which would jeopardize the rest of the equipment. These shortcomings among several others are simply unacceptable for anything other than a short-term project. The sensor system that we developed thrives on big data gathered over a period of time. With that in mind, our goal was to identify the best possible power supply for a project with a potentially indefinite span. After thorough research and careful thought, the team decided to utilize solar panel technologies.

A Solar Electric Photovoltaic (PV) Module is the proper name of the equipment that we commonly refer to as solar panels. The Photovoltaic effect, in essence, is a conversion process where light changes to electricity at an atomic level. There were several variables based on current latitude that needed to be determined before deciding where to situate our solar panels [17]. The goal is to position these modules in such a way that they are absorbing the most amount of sunlight

possible without the need for any adjustments throughout the year.

This would be considered a fixed angle, and it should be noted that there is a downside when taking this approach. Since the sun is highest in the sky during the summer, and vice versa, adjusting the solar panels four times a year or even twice a year can result in increased energy absorption [15]. The increase, however, is not dramatic enough to warrant any changes in our project thus prompting us to move forward with a fixed angle module.

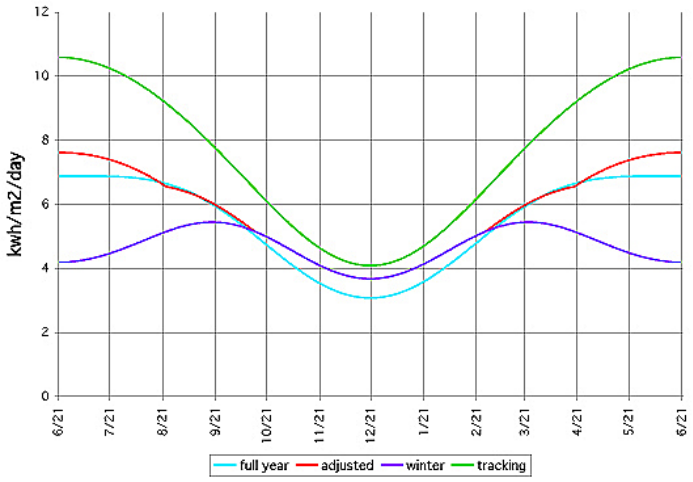

Fig. 1. Optimal PV module energy absorption rates for different adjustment patterns

The graph above demonstrates the different energy absorption levels for modules with varying angles. The blue, red, and purple lines represent a fixed, an adjusted (four times a year), and a winter angle, respectively. Each of these is compared to the green line which represents a (unattainable) 100% optimal situation [11]. The relatively small discrepancy between each of the different angles was the essential deciding factor when choosing to use a fixed angle solution.

The next step was to determine which direction to face our module. In general, a solar panel should always face true south in the Northern Hemisphere and true north in the Southern Hemisphere [15]. For our purposes in Choate Pond, we placed our module on the island facing true south. Using the following formula (1), we were able to determine the angle (to the nearest tenth of a degree) needed for our solar panels:

$$\angle = (\text{Latitude} * 0.76) + 3.1° \quad (1)$$

Using the approximate latitude of Choate Pond (41.13°), an optimal angle of 34.4° was determined. The calculation above assumes an altitude near sea level and is meant to generate an optimum angle for consistent exposure throughout the day [11].

B. Big Data

A byproduct of the Internet of Things is a massive influx of data. Every electronic device, whether it be a household appliance, a coffee webcam, or an aircraft engine over the Indian Ocean, is undergoing a transition toward the need to be "smarter", or having a greater interconnectivity leading to increased interactivity and/or autonomy. They collect and exchange data, and that data in turn becomes what is known as big data. The IT research firm Gartner defines big data as "high-volume, high-velocity, and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation" [19].

Gartner's definition is divided into three parts. The first part of the definition is attributed to Laney who coined the dimensions driving big data quantification as the three V's: volume (scale of data), velocity (real-time analytics and rate of data change), and variety (different forms of data). Later added is a fourth dimension called veracity, or the uncertainty of data. For the second part, big data is about finding cost-effective information processing solutions that fit with current technological capabilities. Finally, the third and most important part refers to the ultimate goal of creating value [1].

C. Cloud Computing

For an agile deployment, we took advantage of cloud computing services. Gartner defines cloud computing as "a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies" and lists five attributes of cloud computing as service-based, scalable and elastic, shared, metered by use, and uses Internet technologies [6]. There are three types of service models which are software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) (see Table I on next page) [2].

For our project, we had convenient access to IBM Bluemix. Bluemix is IBM's cloud platform that combines PaaS with IaaS and offers a catalog of cloud services. Among the services is the IBM Internet of Things service, which provides application program interfaces (API) to connect devices to Bluemix's IoTcloud [8].

D. Cloud Security

When dealing with Cloud platforms where sensitive data is stored and used, such as Node-RED, it is extremely important to have a strong level of cloud computing security. This is used as a line of user technologies and policies that follow regulatory compliance rules and keep important information, data applications, and infrastructure that supports cloud computing safe.

The cloud is a platform that is accessed and used by multiple people; therefore, the areas that need to be strongly monitored and protected are privacy, access control, and identity management. Organizations that work with a cloud computing provider need to be aware of the important need for strong cloud security thanks to the fact that many companies have now turned cloud computing providers for their data operations. With more companies now than ever before turning to cloud solutions, the need for strong cloud security solutions is at an all-time high.

If a cloud computing security system is to be successful, it should work on the security controls that the cloud computing provider uses to uphold the company's information integrity and compliance with important and mandatory regulations. Business continuity and data redundancy must also be

addressed before any changes are made in the event of an information breach in the cloud's security program [18].

A big fear in terms of security that many companies interested in cloud computing, as well as the providers of cloud computing often are forced to face is the potential of losing control over the IT infrastructure and applications once a cloud provider has managed to take over the system and started their work on managing it. However, this fear is often assuaged by the many security technologies available, as help to put them back in control and allow them to see the important information that they need to access.

TABLE I
COMPARISON TABLE AMONG THE THREE MODELS OF SERVICES SAAS, PAAS, AND IAAS [2]

SaaS model	PaaS model	IaaS model
Characteristics:		
- Users are provided with accessible applications. - Access via web to commercial software. - No need to manage software. - APIs integrate different software. - UI powered by "thin client" applications. - Stateless and loosely coupled. - Modular. - Semantic interoperability. - Centralized hosting / delivery. - Uniform platform for delivery. - Open collaboration / sharing.	- Users are provided with a platform for developing applications. - Separate development environments. - Web-based user interface creation tools. - Web service and database integration via common standards. - Support for development team collaboration. - Tools for billing and subscription management. - Customizable / programmable UI. - Unlimited database customizations. - Solid workflow engine/capabilities. - Flexible "service-enabled" integration model.	- Users are provided with virtualized hardware and storage. - Dynamic/self-scaling. - Alterable cost, utility pricing model; Pay-as-you-go model. - Ability to provide single hardware to many users. The costs are less due to this sharing of infrastructure. - Supported OS and platform independent. - Applications / frameworks.
Typical level of control granted to cloud consumer:		
Usage and usage-related configuration	Limited administrative	Full administrative
Consumer activities:		
Configure cloud services.	Test, develop, manage, and deploy cloud-based solutions and services.	Configure and setup bare infrastructure; install, manage, and monitor any required software.
Provider activities:		
Manage, maintain, and implement cloud services; and monitor usage of consumers.	Pre-configure platform; provision underlying infrastructure, middleware, and other required IT resources; and monitor usage of consumers.	Manage and provision storage and processing; and monitor usage of consumers.
Services:		
Email, CRM, website testing, virtual desktop, wiki, blog, automation	Service and application testing, development, integration, and deployment	Virtual machine, OS, message queue, network, storage, CPU, memory, backup service

Cloud access security brokers and other important tools can target hybrid cloud computing environments and multi-cloud environments before they have a chance to add to the list of potential vulnerabilities that companies are already trying to prevent. The most important thing enterprises can do to take on cloud security problems is to be productive and proactive against cloud security risks by forming a strong and comprehensive strategy to protect themselves.

Some companies may want to transition into a public cloud infrastructure because it is efficient and inexpensive; however, they should understand the necessity of secure private cloud computing services for security reasons. Public cloud services can easily introduce new IT risks due to insufficient protection levels. The biggest cloud security risks could be shadow IT, compliance issues, and improper encryption techniques. For all businesses that go the way of the cloud eventually, they should take a hard look at their current IT strategy and then be willing to completely revamp it to accommodate cloud computing security strategies. They should never assume that their cloud computing provider can handle all the security needs of the company, as there's no way to think of every possible risk. While using the cloud, it is important to make sure whatever technologies that are influenced by the cloud computing services are constantly updated to reflect the updates from the services used. The research team followed this same strategy, as ignoring Cloud Security could have derailed the whole project if the data becomes lost or intercepted during transport.

E. Database Management System

When the topic of storing, manipulating, and querying data comes up, the conversation eventually leads to choosing the right database management system (DBMS). A DBMS is general-purpose software for collecting and accessing interrelated data. The goals of a DBMS are an attractive user interface, efficiency, fault tolerance, security, integrity, and concurrency control [20]. For this project, we needed to choose a database system that is relevant to and fits within the project demands. The scope of the project involves using three remote sensors to collect temperature data (along with additional variables in the future) every minute for an indeterminate amount of time at three separate depths. Collected temperature data is time-stamped with expected temperature values ranging from -25.0 to 40.0 degrees Celsius -- the record high and low for Pleasantville, NY is -23.33 and 37.78 degrees Celsius, respectively [22]. If

temperature data is collected uninterrupted for an entire year, there would be over 1.5 million data points. This amount of data puts an emphasis on the efficiency goal of a DBMS.

To narrow down the list of database management systems, we considered one that is scalable for big data, within our time and resource constraints, and readily available in IBM Bluemix for IoT. Fig. 2 depicts four classes of databases as they came to fruition. Among these classes, the most common one is a relational, or SQL, database, but it lacks the "horizontal" scaling required by big data applications. This flaw has led to the era of NoSQL databases, which are used to analyze the majority of data on the Internet. Likewise, performance challenges have led to NewSQL databases which are capable of analytics within a database. Polystore is a recent innovation that will not be explored further here due to its infancy [10].

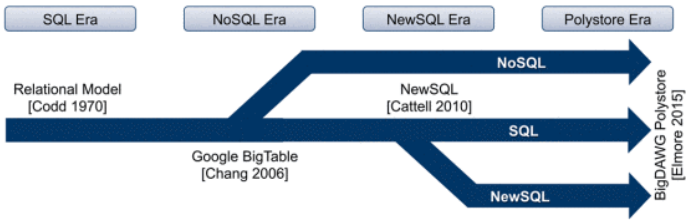

Fig. 2. Evolution of SQL, NoSQL, NewSQL, and polystore databases. Each class of database delivered new mathematics, functionality, and performance focused on new application areas [10].

Each class of database is designed for specific applications using a different approach in terms of data models and mathematical theories (see Fig. 3). Databases specially designed for specific applications can be 100x faster than general-purpose databases [10]. Due to time and resource constraints, building a special-purpose DBMS from the ground up is not feasible. Instead, one must be chosen from among those that are available on IBM Bluemix for IoT and that suit our application.

	SQL	NoSQL	NewSQL	Polystore
Example	PostgreSQL	Accumulo	SciDB	BigDAWG
Application	Transactions	Search	Analysis	All
Data Model	Relational Tables	Key-Value Pairs	Sparse Matrices	Associative Arrays
Math	Set Theory	Graph Theory	Linear Algebra	Associative Algebra
Consistency	■			■
Volume		■	■	■
Velocity		■	■	■
Variety		■		■
Analytics			■	■
Usability	■			■

Fig. 3. Focus areas of SQL, NoSQL, NewSQL, and Polystore databases [10].

For our endeavor, almost all operations involve inserting new data, and deleting or modifying data is rare. Further, adding new data is sequential with discrete intervals. Using a simple approach to structure our data, we would get:

```
CREATE TABLE station (station_id INT,
  description TEXT);
CREATE TABLE temp (id INT, station_id
  INT, time TIMESTAMP, temp_value
  FLOAT);
```

Notwithstanding its simplicity, this approach would create several inefficiencies. Copying the data would require a statement for every record, which in our case means over five hundred thousand statements per sensor. A better way to structure our data is to use an array DBMS that is separated into page size-friendly chunks. For example, we could use two tables -- one for the station_id and counters and one for the temperature data -- and separate the data into chunks by day.

```
CREATE TABLE station (station_id INT,
  last_day DATE, last_pos INT,
  description TEXT);
  CREATE TABLE temp (station INT, day
  DATE, temp_value FLOAT []);
```

Additional improvements in the database model can be made which has led to the emergence of array and time-series database management systems. In fact, the SQL: 1999 revision added support for arrays, and the main feature adopted by the SQL: 2011 revision is improved support for temporal databases [12].

F. MQTT

In order for the IoT aspect of this project to work out, it needs a messaging protocol to send data collected about temperatures, oxygen content, etc. to an online database. This is where MQTT (MQ Telemetry Transport), designed in 1999 by Dr. Andy Stanford-Clark of IBM and Arlen Nipper of Eurotech, comes in. MQTT is used for pushing information (with a maximum message size of 256MB) in a publish/subscribe system, and it works with devices in less than ideal network conditions. The MQTT standard is designed to be a workaround for bad network conditions, as it is not very resource-intensive. It is also very reliable. Thanks to MQTT, M2M (machine to machine) and IoT are able to function without a mass amount of bandwidth or battery power. MQTT is already used in many projects, such as a British Sign Language avatar rendering, location-based messaging accessibility, a house that can use Twitter, and FloodNet – an environmental information project that warns someone if river levels become too high [13].

MQTT was originally developed to monitor an oil pipeline in the desert while using little bandwidth and battery because it used, at the time, an expensive satellite link. MQTT's publish/subscribe system is much different from HTTP's request/response system. This allows MQTT to be event-driven and send data to users in the form of 256MB messages. Since MQTT uses a broker, every user can subscribe to see the data that they want, and the broker will send them messages when necessary. This allows clients to not have to communicate with each other making MQTT very scalable.

Currently, MQTT v3.1.1 has a royalty-free license, and became an OASIS Standard on November 7th, 2014. It is used by IBM in the WebSphere MQ Telemetry project to be implemented as a scalable project. There is also a list of

software on GitHub where MQTT is implemented, working for brokers/servers, devices that are known as “Things” by MQTT, plugins for different software, tools, applications, and client libraries. In order to work, MQTT uses standard TCP/IP ports. The Internet Assigned Numbers Authority has reserved port 1883 to be used with MQTT. Port 8883 was also registered, and uses MQTT over SSL.

Because MQTT is so bare-bones and is not resource-intensive, the security support is not efficient. Version 3.1 of MQTT does allow users to send a username and password in a sent packet, but SSL is widely responsible for the rest of the network encryption. Using SSL can be done outside of the MQTT protocol, and it does hog a lot of network resources. This could possibly defeat the whole purpose of using MQTT, so those who want network encryption might wish to look at protocols other than MQTT. Some applications are made to encrypt data that they send and receive, but MQTT was designed not to do this so that it could remain bare-bones. By not encrypting the data that is sent back and forth, MQTT could possibly open the opportunity for unwanted traffic, such as MAC spoofing, IP spoofing, and other negative actions. If a company using MQTT controls the system they are using and know their clients’ IP addresses, they should restrict access to defined IP ranges.

Another way to help alleviate the strain of MQTT lacking essential security features is to use a load balancer, which proxies traffic to different MQTT brokers. This helps to keep downstream systems from overloading. This is useful for when there is high traffic and the MQTT brokers are becoming overwhelmed. A demilitarized zone is also extremely useful thanks to its additional firewall. Since there are now firewalls from different vendors, if one aspect of MQTT is compromised, the rest of it will remain safe. Lastly, keeping MQTT and its software up to date is always the simplest way to keep everything running safely and smoothly [14].

There have been several companies that use MQTT extensively due to its low resource load, such as Eclipse and Facebook. Facebook has used MQTT for the Messenger app for notifications and updates. The MQTT protocol is great for the messenger app, as it is extremely efficient with resources, and does not use much of a device’s CPU, battery, or network thanks to the low resource overhead of the protocol.

There are many services other than MQTT that also performs the function of moving data between a series of devices, including CoAP, XMPP, DDS, AMQP, and SOAP. MQTT is bare-bones and easy to program with on IoT devices. Nevertheless, it requires a server to work, which makes it a bad choice for a local network, as it requires an additional broker. MQTT can also be considered a device to server communicator. CoAP is used for document transportation between devices and servers. XMPP is useful for connecting devices and users, and it is considered a device to server protocol because the users are connected to the servers. SOAP is not used very often as a potential protocol for IoT as it has not had very favorable reviews, although it will suffice if needed. DDS is a device to device protocol that allows machines to interact with each other. Lastly, AMQP is used for server to server connection. It can be used for device to server connection if needed, but that is a complicated undertaking. Due to all the inefficiencies in other devices related to this study, MQTT was chosen as the working messaging protocol [9].

G. Node-Red

An inherent problem when dealing with the Internet of Things is the development of an affinity. One of the largest advantages that these smart devices offer is their ability to collect and subsequently exchange data. This ability, however, is arguably pointless if the data cannot be controlled in a logical manner. The innate flaw here is an absence of organization, which is characteristically hindering to just about any situation. In an effort to prevent any unnecessary confusion, our team decided to use the Node-RED programming tool as a means of organizing the transmission of data between our IoT devices.

Node-RED is a software tool developed by IBM that features a browser-based flow editor to "wire" together devices, APIs and web services. Flow-based programming "…is a way of describing an application’s behavior as a network of black-boxes, or “nodes” as they are called in Node-RED. Each node has a well-defined purpose; it is given some data, it does something with that data and then it passes that data on." [16]. The true potential of Node-RED lies in its ability to transform robust IoT systems into a visual representation of synchronous events. This tool will ultimately conduct the flow of data between our physical/virtual devices and structure our project in a scalable fashion.

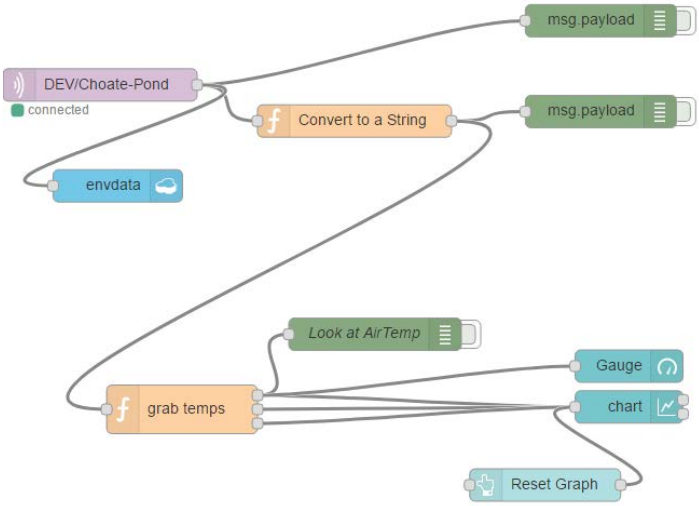

Fig. 4. Node-RED Flow

Fig. 4 represents an image from the backend view of a sample Node-RED application. This flow consists of several nodes that will listen for information published on a specified MQTT channel and push that info to a payload and then to a line chart/gauge for display. The result is shown in Fig. 5 below.

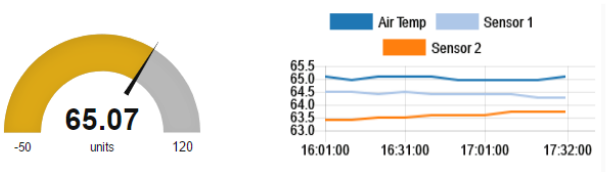

Fig. 5. Node-RED Output

The implementation of Node-RED has had three significant effects on the project. The first advantage described earlier in this section is the new found sense of organization. Node-RED serves as a functional and visual representation of the big data flow. Another added benefit of this tool is its ability to address some of the scalability issues with this project. The flow-based programming simplifies the concept of expanding the breadth of sensors. For example, adding another MQTT broker would be as simple as dragging out another MQTT node and entering the proper information. The third and final benefit seen from using this tool lies in the platform it creates. The ultimate goal is to create an application that will analyze data or at least view its history. Node-RED creates and stores its flow using JSON which can be imported/exported to almost any environment.

III. ARCHITECTURE

To accomplish this project, we used hardware and software devoted to IoT – Arduino, temperature sensors, soldered pipes, an Otter Box, Solar Panels for energy, Node-Red programming, Wi-Fi signals, and MongoDB on IBM Bluemix.

A. Adafruit Feather

The research team used the Adafruit Feather because of the temperature sensors attached to it that meet the environmental sensing needs of the study. The sensors are in a waterproof coating and travel to a pipe's connection matrix. This allows easy attachment of power and I/O lines from the Arduino's microcontroller. The Adafruit Feather needs code written for it to connect to Wi-Fi, sense temperatures, send data to the Cloud for storage, and be able to sleep for five minutes between temperature readings in order to save on stored solar energy.

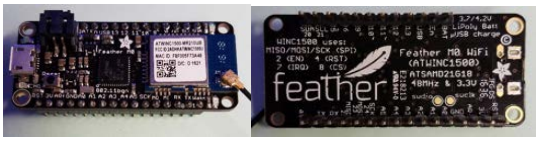

Fig. 6. Adafruit Feather Top and Bottom

The sensors are exposed in the pond, but the microcontroller needs to stay dry and powered on. First, we placed the electronics on the island, and tethered them to the pipe in the water. The battery for the electronics on the island charges via solar panels.

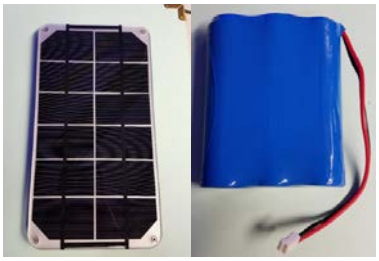

Fig. 7. Solar Panel and 3.7v Power Cell

B. Pipe Design

The pipe needs to hold three waterproof temperature sensors that can be queried. The wires that connect the sensors to the microcontroller need to be waterproofed, as parts of the device are submerged. The pipe has a connection matrix to allow for power and I/O lines from the microcontroller to connect. The sensors are separated by 1-2 feet inside the pipe. There is also a water sensor inside the pipe to see if the pipe is taking in water. The first prototype for this pipe design is named "Tiny Tim".

C. Code

The main goal of this iteration of the project is to extract and display the temperatures from three points in Choate Pond. In order to do this, the structure of both the apparatus and the coding has to be effective.

In the physical construction of the structure, there is a temperature sensor at three different levels. The thermometers are secured to a waterproof pipe which protects the wires and connect the sensors to the Arduino which is located at the top of this device. The Arduino contains the code that reads the temperature of the water every minute and the authentication code for the wireless router, both of which are written in C/C++.

Using Node-Red, the team created a trigger that will activate the temperature reading script on the Feather. The Node-Red process automation also creates an MQTT listener on the framework which, when triggered by a message, can automatically transform the data from a string into a JSON object so that it might be sent into the database with relative ease.

The Node-Red framework also allows the device to connect through the internet using its order of operations. When the Node-Red sends the correct information to the router, it recognizes the Arduino as a registered device and then it allows the Arduino to connect so that the MQTT messages can be transmitted seamlessly. If the team was managing the server, they would have complete access to the code and all of the data; however, the time and money spent on managing the security, data compression, and the networking issues could potentially outweigh the benefits of having complete and open access to all hardware and software involved in the project. Due to these reasons, the team has decided to manage the data on a cloud platform; i.e. Node-Red.

The team used a MongoDB database to organize the collected data. There is one database for each sensor, which in our first prototype includes one sensor/database for each level of the pond. The tables are organized by the date, and inside of each table, the collection is organized first by the hour from 00-23, then by the minute from 00- 59, and finally by the

temperature (measured in Fahrenheit) from 000-100. So that a sample temperature might appear to be in database 2, table 03052017, and collection 0548063. This would tell us that for sensor 2 on March 5, 2017, the temperature was 63 degrees Fahrenheit at 5:48 AM.

The Node-Red code stamps the data received by the receiver with the date and the time in which the message was received then proceeds to input it into the correct location within the database by the use of a sorting function automatically inside of the cloud server. The database connection code is triggered by the reception of the string sent by means of MQTT, where the temperature is sent over as an argument. This database integration script also includes a check that the network is connected. If the Adafruit Feather is not connected to the WiFi, Node-Red invokes the script which connects the Arduino to the server over Wi-Fi. It continues to attempt to connect the Arduino to the Wi-Fi until the connection is confirmed. In case connection to Wi-Fi proves impossible, measurements are stored in a local database located in a flash drive connected to the Arduino to prevent data from being lost. However, as this code is still not entirely completed, the backend developer is still able to manipulate various features of this framework code.

Once the data is stored on MongoDB, the front end developers are able to extract the data to create any display that would best express the information within the Node-Red framework.

The code for MQTT was written as follows:

```
{Loc: [location] / Body: [body of water
  sensor is in] / TS: [timestamp] / N:
  [name of device being used] / S:
  [sensor number] / {id=1 (sensor1), /
  Temp=x} / {id=2 (sensor2) / Temp=y}]}
```

Data gathered from the sensors will then be returned.

D. Data Visualization

The final development of this project is to transform the information (now stored in a database) from a group of related numbers to something visually interpretable. This technique is referred to as data visualization and can be described as "…the science of visual representation of "data", defined as information which has been abstracted in some schematic form, including attributes or variables for the units of information." [5]. For the purpose of this particular project, a dynamic webpage is sufficient. The end goal is to create a scalable system that will design graphs and charts based on real-time data. The first step in the creation of this system is to take a closer look at the fundamental principles of an informative visual display.

Reading information from a visual source (as opposed to a numerical source) can represent comparisons and causalities in a much better light, but only if the visual is created properly. Vitaly Friedman, co-founder and editor-in-chief of Smashing Magazine, explains that "to convey ideas effectively, both aesthetic form and functionality need to go hand in hand…by communicating its key-aspects in a more intuitive way." [4]. Just like any type of dynamic webpage, the functionality and the aesthetics must agree with each otherwise user experience, and the accuracy of information, will suffer greatly. The following are four key points (in no particular order) to observe when creating a visual display:

1) Induce the viewer to think about the substance rather than about methodology, graphic design, the technology of graphic production, or something else.
2) Avoid distorting what the data has to say.
3) Make large data sets coherent.
4) And serve a reasonably clear purpose: description, exploration, tabulation or decoration [21].

Ignoring any of these points could result in a misleading graph which would defeat the entire purpose of this project. These four points are general guidelines to observe when determining an appropriate tool for a particular project. Because this study is an abbreviated portion of a much larger project, it did not require the breadth of a full visualization tool. With that in mind, this iteration of the study aimed to conclude with some suggestions based on our findings from the semester.

IV. Future Research

The next major development in this project is visualization of data in a controlled, consistent environment. Every component described in this study is chosen with scalability as the foremost concern. The Hudson River will introduce several different variables that will inevitably require a more extensive means of visualizing data. The following is a suggestion for a data visualization tool that can be implemented to a Node-Red flow.

A. Google Charts (Visualization Tool)

Google Charts is the successor service to Google Charts API and was released in February 2010. It enables users to create line, bar, pie and radar charts among many other designs. The main difference between the now deprecated Google Charts API and current Google Visualization API is the latter's ability to connect charts and data sources all over the web.

The Visualization API is a web service dedicated to creating graphical charts, which suggests a higher quality of representation than the capabilities of Node-Red alone. The output produced by our Node-Red flow (see Figures 4 and 5) is sufficient for this phase of the project, however, a more versatile solution is ideal. The JavaScript platform makes implementing Google Charts to a Node-Red flow a rather trivial process.

V. Conclusion

The main purpose of this iteration of the project was to create a scalable prototype of a sensor system using concepts of Big Data storage, and Internet of Things devices. By using Node-Red to execute the computational aspects of this project, the team was able to create a paradigm for the individuals who will continue to develop this system. The team also decided on storing the data in a cloud server, BlueMix, which has the ability to be expand or conform to the needs of the user. These tools combined with the physical model were all chosen with

mutability in mind. Our software and hardware architecture have sustained initial trials and should foreseeably maintain their efficacy throughout the course of any associated long-term project.

ACKNOWLEDGMENT

The project team consists of the Research Paper writers, the project customer – Matthew Ganis, the Seidenberg School of Computer Science and Information Systems, the Lubin School of Business, and the Dyson Environmental Center. The stake in the project for the Seidenberg School is to turn the end results of the research into a curriculum that will attract future students to Seidenberg School of CSIS. Lubin hopes to turn the curriculum into one that they can monetize and then sell to other universities to use. Dyson Environmental Center is interested in the data collected and what further research can be done with it.

REFERENCES

[1] D. Bachlechner and T. Leimbach. "Big data challenges: Impact, potential responses and research needs," *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, Balaclava, 2016, pp. 257-264. Web. Mar 2017. <http://ieeexplore.ieee.org.rlib.pace.edu/stamp/stamp.jsp?tp=&arnumber=7737349&isnumber=7737292>.

[2] M. U. Bokhari, Q. M. Shallal and Y. K. Tamandani. "Cloud computing service models: A comparative study," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 890-895. Web. Mar 2017. <http://ieeexplore.ieee.org.rlib.pace.edu/stamp/stamp.jsp?arnumber=7724392>.

[3] W.O. Freeman. “NATIONAL WATER QUALITY ASSESSMENT PROGRAM- The Hudson River Basin”. USGS New York Water Science Center. 1991. Web. Mar 2017. <https://ny.water.usgs.gov/projects/hdsn/fctsht/su.html>

[4] V. Friedman. "Data Visualization and Infographics – Smashing Magazine." Smashing Magazine. N.p., 03 Nov. 2016. Web. 3 Mar 2017. <https://www.smashingmagazine.com/2008/01/monday-inspiration-data-visualization-and-infographics/>.

[5] M. Friendly. "The Golden Age of Statistical Graphics." Statistical Science 23.4 (2008): 502-35. Math.YorkU.CA. York University, 24 Aug. 2009. Web. 3 Mar 2017. <http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf>.

[6] Gartner. “Gartner Highlights Five Attributes of Cloud Computing,” June 23, 2009. Web. Mar 2017. <http://www.gartner.com/newsroom/id/1035013>.

[7] Google. "Google Charts FAQ." Google Developers. Google, 23 Feb. 2017. Web. 03 Mar. 2017. <https://developers.google.com/chart/interactive/faq>.

[8] IBM. “What is Bluemix?” 21 Mar.1 2017. Web. Apr 2017. <https://console.ng.bluemix.net/docs/overview/whatisbluemix.html#bluemixoverview>.

[9] S. Kedar. "Which Is the Best Protocol to Use for IoT Implementation?" Quora. N.p., 09 Jan. 2016. Web. 7 Mar 2017. <https://www.quora.com/Which-is-the-best-protocol-to-use-for-IOT-implementation-MQTT-CoAP-XMPP-SOAP-UPnP>.

[10] J. Kepner et al. "Associative array model of SQL, NoSQL, and NewSQL databases," *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, 2016, pp. 1-9. Web. Mar 2017. <http://ieeexplore.ieee.org.rlib.pace.edu/stamp/stamp.jsp?arnumber=7761647>.

[11] C. Landau. "Optimum Tilt of Solar Panels." Optimum Tilt of Solar Panels. N.p., 11 Nov. 2015. Web. 7 Feb 2017. <http://www.solarpaneltilt.com/>.

[12] R. McGann. “SQL Version Analysis,” Web. Mar 2017. <https://www.cs.colostate.edu/~cs430dl/yr2016su/more_resources/SQL%20History.pdf>.

[13] "MQTT." MQTT RSS. N.p., n.d. Web. 7 Mar 2017. <http://mqtt.org/>.

[14] "MQTT Security Fundamentals – Securing MQTT Systems." HiveMQ. N.p., 14 Sept. 2015. Web. 7 Mar 2017. <http://www.hivemq.com/blog/mqtt-security-fundamentals-securing-mqtt-systems>.

[15] R. Perez, and S. Coleman. “PV Module Angles. Ashland: Home Power,” Aug. & Sept. 1993. PDF. <http://www.spyordie007.com/ahanwpdfs/PVANGLES.PDF>.

[16] RED, Node -. "About." Node-RED. The JS Foundation, 2016. Web. 29 Mar 2017. <https://nodered.org/about/>.

[17] Rensselaer Polytechnic Institute. “Photovoltaic Lighting.” Lighting Research Center. 2006. Web. 8 Feb 2017. <http://www.lrc.rpi.edu/programs/NLPIP/lightingAnswers/photovoltaic/abstract.asp>.

[18] M. Rouse. "What Is Cloud Computing Security? - Definition from WhatIs.com." *SearchCompliance*. N.p., n.d. Web. 2 Apr 2017. <http://searchcompliance.techtarget.com/definition/cloud-computing-security>.

[19] S. Sicular. “Definition Consists of Three Parts, Not to Be Confused with Three "V"s,” Gartner Inc., 2013, Web. Mar 2017. <https://research.gartner.com/definition-whatis-big-data?resId=3002918&srcId=1-8163325102>.

[20] A. Silberschatz. H.F. Korth, and S. Sudarshan. “*Database Systems Concepts*. 6th ed.” New York, New York: McGraw-Hill, 2011, ch. 1, 14 , pp 1-5, 627-629.

[21] E. Tufte. “The Visual Display of Quantitative Information.” Graphic Press: Cheshire, Connecticut, 2015. Print.

[22] The Weather Company. Web. Mar. 2017. Web. Mar 2017. <https://weather.com/weather/monthly/l/10570:4:US>.