ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

New Orleans JAZZED about Engineering Education

Paper ID #14531

Techniques in Data Visualization for Electrical Engineering: From Embedded Systems to the Internet

Mr. Jason McGuire, Sonoma State University

Jason McGuire is currently a graduate student at Sonoma State University and an intern at Pocket Radar, Inc. He received his B.S. in physics from Humboldt State University in Arcata, CA. His main areas of interest include: embedded systems, single board computer systems, sensor networks, digital communication and software-defined radio. He is currently working on a large internet connected micro-climate sensor network for a local wildlife preserve under a grant from Pacific Gas & Electric.

Dr. Farid Farahmand, Sonoma State University

Farid Farahmand is an Associate Professor in the Department of Engineering Science at Sonoma State University, CA, where he teaches Advanced Networking and Digital Systems. He is also the director of Advanced Internet Technology in the Interests of Society Laboratory. Farid's research interests are optical networks, applications of wireless sensor network technology to medical fields, delay tolerant networks. He is also interested in educational technologies and authored many papers focusing on eLearning and Active Learning models.

Techniques in Data Visualization for Electrical Engineering: From Embedded Systems to the Internet

1. Introduction

The emerging Internet-of-Things (IoT) concept is considered to be the next technological revolution [1]. It describes various technologies and research disciplines that enable the Internet to reach out into the real world of physical everyday objects. By 2020, it is expected that 25-50 billion "things" to be connected to the Internet. Gartner, the world's leading information technology research and advisory company, projects IoT will result in $1.9 trillion in global economic growth [2]-[3]. While today there are just 300,000 developers contributing to the IoT, a new report projects that an estimated 4.5 million developers are needed by 2020 [4]. This reflects a massive employment opportunity for future hardware and software engineers in IoT.

Not surprisingly, over the past decade, the IoT has become an attractive area for many students majoring in Electrical Engineering (EE) or Computer Engineering (CE). Building IoT-based projects involves developing a rich set of skills. Some of the skills can involve sensors, embedded systems, communication technologies (e.g., WiFi, Cellular, Ethernet, Bluetooth, ZigBee, 6LoWLAN), antenna design, smartphone app development, Internet web protocols and applications, cloud computing and virtualization, data analytics, data repository and visualization, and much more.

Our Electrical Engineering Department at Sonoma State University has observed an increase in the number of capstone projects selected by students involving the implementation of various wireless/wired monitoring systems. These systems measure various physical phenomenon such as a creek's water level and water flow, or counting the number of cars in a parking lot, or the power generated by a solar panel system. While our EE students are capable of completing and demonstrating the hardware design, they often struggle in areas such as data management and developing data visualization. Many student teams have used various platforms known as Cloud-based IoT services, that allow data logging on the cloud [5]. The key issue with utilizing such services is that many students don't fully understand the underlying technologies.

In fact, our student surveys from capstone courses indicate that while our EE students feel confident in areas such as embedded systems, microprocessors and interfacing such devices to various sensors and actuators, they find themselves less knowledgeable about concepts such as cloud applications and services, data visualization, and developing application programming interfaces (APIs). Unfortunately, throughout our EE curriculum no course formally addresses any of such topics.

Over the past year we have been developing a cloud-based data repository and visualization training exercise that teaches the students how to store and display their sensor data on a remote server. Our design methodology in this training is to teach students how to create a flexible unified framework that can operate with any TCP/IP-capable microcontroller or embedded-based system. This training exercise is offered to students as a workshop during the first two weeks of their two-semester capstone course.

We believe the proposed platform teaches students a number of valuable skills that with the advent of IoT can potentially make them more marketable upon their graduation. We emphasize that the educational value of this training exercise extends well beyond the typical implementation of LAMP (Linux, Apache Web Server, MySQL database, Perl, Python, or/and PHP) stack [6]. In this exercise the students also learn about cloud services, advantages and disadvantages of data pushing (or posting) and pulling for retrieving sensor data, developing APIs, creating and managing data feeds, and various approaches to plot and visualize the sensor data using available online plotting libraries, such as Google or Flot Charts.

This paper is organized as follow. In Section 2 we briefly elaborate on the background and course work of our EE students prior to taking the capstone course. In Section 3 we describe the structure of the workshop as it was presented to the students. Section 4 lays out the technical details of implementing data visualization and lists the steps that students are expected to complete. Various possible expansions to this exercise are discussed in Section 5, followed by a brief review of learning outcomes based on student feedback in Section 6. We conclude the paper with Section 7.

2. Student Background

Prior to taking the capstone course, all EE students are required to complete a 4-unit course discussing microcontrollers and microprocessors with heavy emphasis on Assembly language and C programming and a 4-unit course in Networking and Network Protocols with lab activities, covering packet tracing and Layer 1-2 protocols. At least half of the students will be taking a course in Introduction to Wireless Technologies during the first semester of their capstone course. The table below summarizes the learning outcomes of these course.

Course	Learning Objectives	Modality
Microcontrollers & Microprocessors	- Fundamental architectures - ADC / Interrupts / GIOP / Timers / USAT / I2C / SPI / - PIC Microcontrollers and MPLAB IDE - C & Assembly Programing	Lecture + Lab
Networking & Network Protocols	- Linux OS - Layer 1-7 Protocols & Packet tracing - Hubs, Switches, and Routers - Client-server model & Socket API - Web applications such as HTTP / FTP / SSH / email - Python programing	Lecture + Lab
Introduction to Wireless Technologies and Networks	- WLAN & Wireless Protocols - Link budget & antennas - Wireless technologies - Testing and characterizing a ZigBee (hands-on)	Lecture

Table 1: Expected Student Background.

Our EE capstone is offered as a two-semester course. In the first semester, the students are expected to define a problem statement and a product idea, create a team, identify their customer base, create a development plan, prepare a funding proposal, and find an industry advisor. In the second semester each team is expected to communicate with their industry advisory and client, and complete a functional prototype [22].

During the first semester of their capstone, the students are encouraged to attend one or more 2-8 hour, free, workshops covering various topics, ranging from introduction to LabVIEW and AppInventor [7] to understanding spectrum analyzer basics. The workshops are typically designed by advanced undergraduate or graduate students, and supervised by a faculty member. The topics are selected based on student feedback and their importance in assisting students to find employment or internship positions. Generally, none of these topics are formally covered in the curriculum. Students who attend a workshop and complete the assigned activities will receive an official certificate of accomplishment from the EE department.

3. Data Visualization Activities and Learning Outcomes

We offer the Data Visualization training exercise as a workshop outside class time. The table below summarizes general activities developed for the 8-hour (three-day) workshop. We note that these activities can also be integrated in an existing course, should time permits.

Duration (hrs)	Content	Assignment
1 (day 1)	Overview of microcontrollers and various ways to interface them to sensors (PIC-based microcontrollers, SPI, I2C, USART) - Brief overview of embedded systems (Raspberry Pi)	Interface a PIC to a temp. sensor via I2C interface (optional)
2 (day 1)	Introduction to MySQL/PHP and their applications	Install Linux and MySQL
2 (day 2)	Basic understanding of Socket API	Programming with the Python Socket API
3 (day 3)	Displaying the data using Flot charts	Create a web-based chart to display existing data

Table 2: Summary of Data Visualization Course.

Each workshop, including the Data Visualization workshop, is required to have learning outcomes that are clearly aligned with ABET outcomes taught and assessed in the capstone course. In particular, in the Data Visualization workshop we have focused on the following subset of “a-k” outcomes defined by ABET [8]:

- An ability to design a system, component, or process to meet desired needs (“c”).
- A recognition of the need for, and an ability to engage in life-long learning (“i”).
- A recognition of contemporary issues (“j”).
- An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (“k”).

Therefore, in preparation of this workshop, we focused on the following learning objectives:

- To demonstrate an end-to-end knowledge of IoT;
- To become familiar with open-source software tools and utilities to display sensor data;
- To understand available online tools to store and visualize the sensor data.

In the following section we describe the implementation details of the Data Visualization exercise. We purposely list a few technical details (see Section 4.4) to highlight the student learning scope of this project.

4. Implementation Details of Data Visualization

Figure 1 depicts an overview of our proposed architecture which enables the students to extend the scope of their projects from just a local embedded system to a full IoT solution. We have selected this architecture due to its ease of setup and most of our students have already taken a course in microcontrollers and a course in networks and protocols

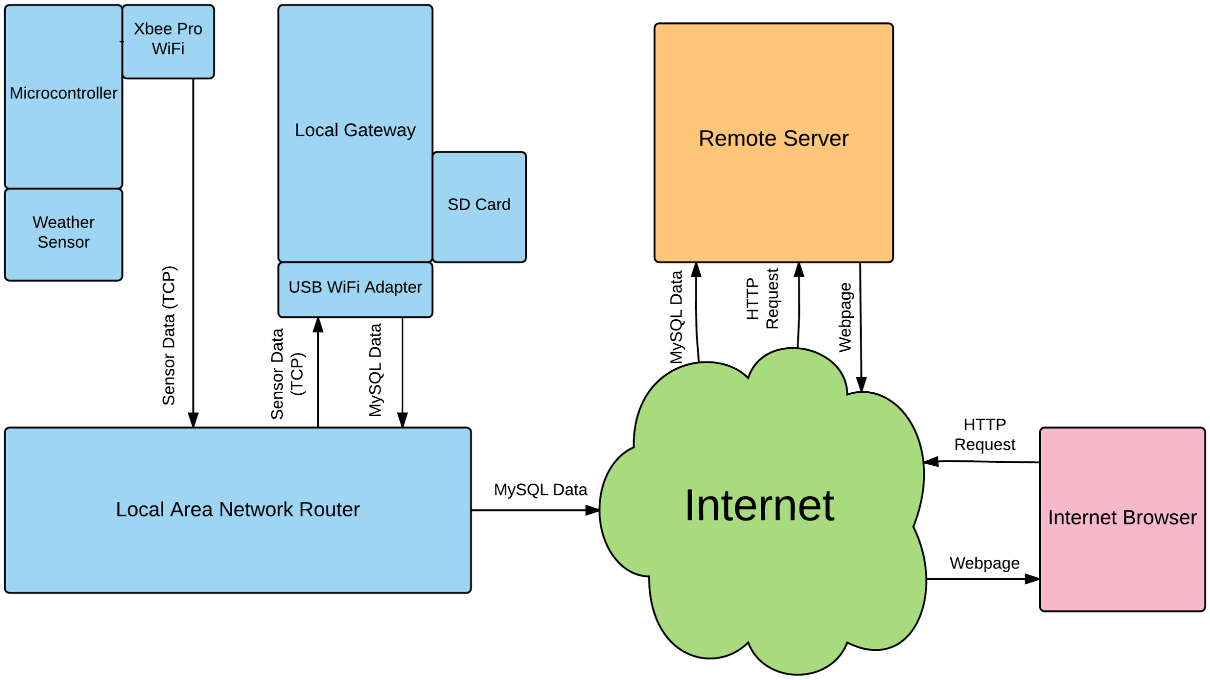


Figure 1: Hardware Architecture.

As illustrated in Figure 1, the architecture is divided into three key blocks: the microcontroller system, the local gateway and the remote server. In the following sections we explain each block and discuss how it can be configured. In each case we discuss ways students can modify the proposed configuration and enhance their own setup. We also elaborate on the potential challenges students may face as they attempt to complete this architecture of IoT for the first time.

4.1 Sensor Data and Data Gathering

The overall focus of this workshop is to add visualization and database capability to an existing MCU-based IoT project. We focus on using the PIC18F25K20 microcontroller [9] and interfacing it with a BME280 weather sensor [10], which provides temperature, relative humidity and barometric pressure readings. The PIC microcontroller was chosen due to student familiarity of its architecture and operation. The weather sensor was selected due to the interest of many of our students in environmental monitoring. Clearly, there are many alternative microcontrollers and sensors on the market that interested students can consider.

In order to store and visualize the sensor data, the collected data must first be transmitted to the local gateway (LGW). The physical communication between the LGW and the microcontroller can be established using a direct serial connection or wireless technologies such WiFi, Bluetooth, 6LoWPAN, or RF radio pair (e.g., popular Xbee). In this experiment, we assume the microprocessor communicates with the LGW, through the existing wireless LAN (WLAN), via an Xbee Pro WiFi adapter that is capable of sending TCP messages on a local WLAN, shown in Figure 1. The Xbee Pro was selected due to its ease of use for students, simple graphical user interface, and serial connection interface (USART) that is well-understood by our EE students. In addition, using Xbee provides a great learning experience in understanding how to send messages over IP, a concept extensively covered in the Networking & Network Protocols course.

It must be noted that depending on the capabilities of the LGW module, it is possible to interface it with different sensors and eliminate the need for having the microcontroller block. One advantage in doing so is that it removes the complication of communication between the microcontroller and the Internet gateway. However, in our proposed approach illustrated in Figure 1, the students have greater flexibility in terms of locating their sensor node (e.g., combination of the sensor and the microcontroller) and interfacing multiple sensor nodes to the LGW.

In many IoT designs, the microcontroller sends sensor data in TCP packets directly to the IP address of the server, on a specific port. This requires running an application on the server to continuously listen to the assigned port and receive the packets. Such an application must also handle security threats from malicious users or bots, adding another layer of complexity. In our experiment, having the LGW eliminates such complexities, as it directly provides the application layer in the Internet Protocol suite. We elaborate on the function the LGW in the following section.

4.2. Local Gateway Configuration

The purpose of having a local gateway is to establish a secure path from the microcontroller to the remote server for data storage and visualization. The LGW can be a single-board computer such as a Raspberry Pi or BeagleBone Black, capable of running Linux [11] or a dedicated Linux-based PC. We used a Raspberry Pi 2 Model B to serve as our local gateway. The Raspberry Pi Model B was selected as it is well documented on the Internet, and readily available for students to purchase at a low price.

In this experiment we used the Linux operating system due to its simplicity and portability. Furthermore, all students are expected to be familiar with Linux, as they explored it in their networking course.

In order to communicate with the remote server, students are required to install MySQL on the LGW. MySQL [12] is a free relational database software that can operate in a lightweight Linux environment already existing on a single-board computer or a remote server.

An alternative architecture is to provide HTTP access via the LGW. However, this requires students to successfully navigate the firewall rules of their own network and dedicate sufficient bandwidth for remote users. In our proposed architecture we leave these details to be handled by a rented or shared server space. Thus, the students simply configure the LGW to act as a secure link between the embedded system and the remote server.

4.3. Gateway Hardware Setup

Setting up the Raspberry Pi involves imaging a Linux distribution onto a SD card. The Raspbian distribution was selected for this project [13]. Once the operating system has been installed, the Raspberry Pi can be directly connected to an Ethernet cable on the same local area network as the microcontroller, or a USB WiFi dongle can be used to connect it to an existing WLAN. It is also necessary to establish the Raspberry Pi with a local static IP address (e.g., 192.168. x.x), so that the destination address in the microcontroller code does not need to be dynamically updated.

Once the Raspberry Pi has been successfully connected to the local network, students must test the data path from the microcontroller to the local gateway in order to ensure that the sensor data is being correctly received by the LGW. This can be achieved by utilizing a Linux utility, such as netcat,[14] to monitor all messages received on a specific port on the LGW. Netcat can be installed directly on the Raspberry Pi by typing *apt-get install netcat*.

4.4. Gateway Software Setup

The LGW must be configured so that it can receive sensor data from the microcontroller and then connect to the remote server and place the data into the database, as shown in Figure 1. The transmission of data between the microcontroller and the LGW (e.g., Raspberry Pi) follows the client-server model with the Raspberry Pi acting as the server. In this case, all connections are handled by the socket API. A socket API is an application programming interface that allows a program like Python to handle the network socket connections. A network socket represents the two end points in a communication link. The server listens for incoming TCP connections from clients and provides a socket (communication link), which is specific to a single TCP connection [15].

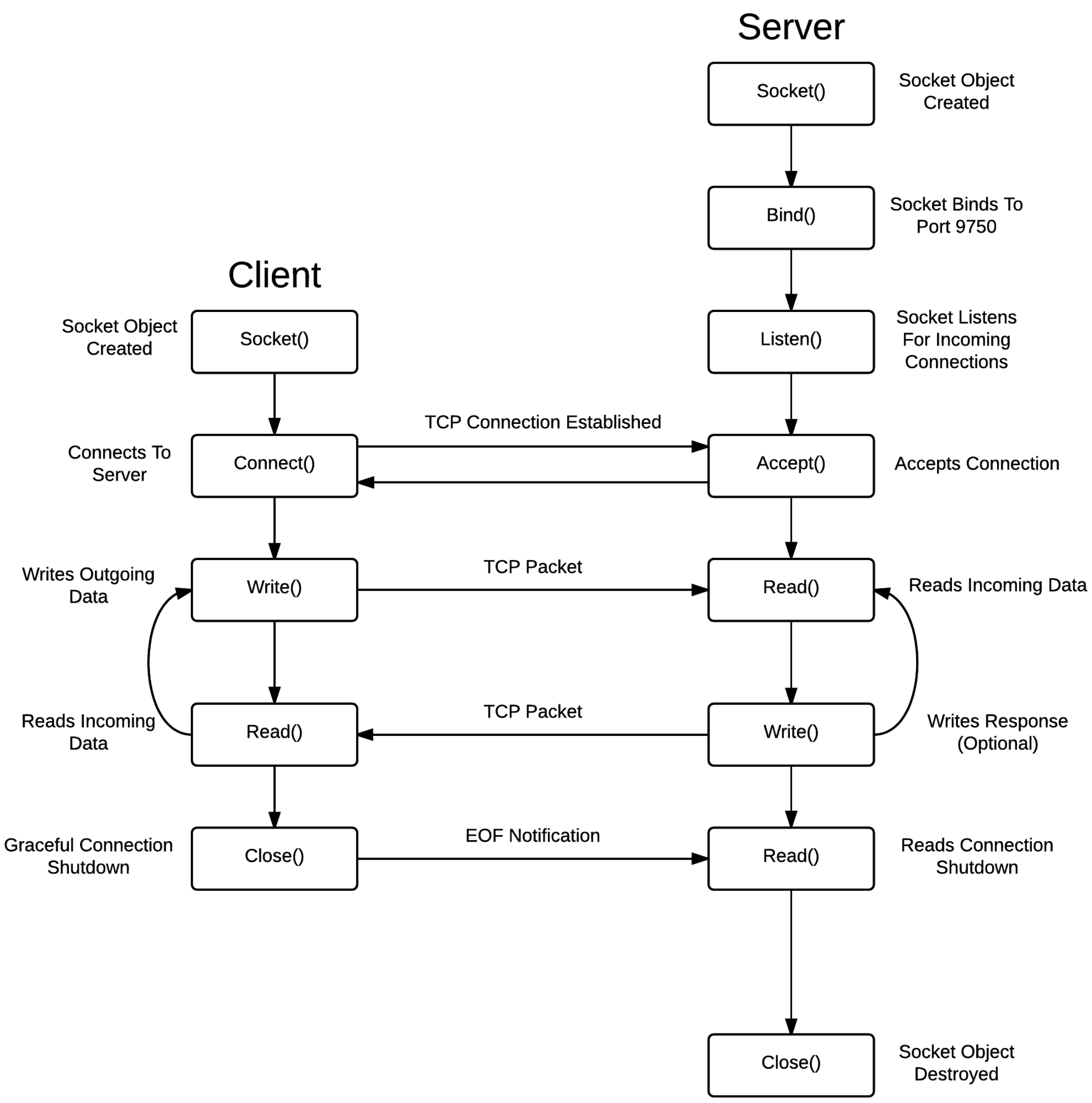


Figure 2: Socket API.

Figure 2 provides a visual description of the detailed steps that students must accomplish in order to establish the communication pathway between the LGW, the remote sensor, and the remote server. In the following paragraphs, in order to demonstrate the learning scope of this activity for students, we briefly describe each step in Figure 2 as explained to students on day 2 of the Data Visualization workshop (see Section 3). We note that depending on the background of students, some of these details can be skipped.

4.5. Creating Socket API

The client socket functions are handled by the Xbee WiFi adapter. The server side socket functions must be implemented on the network gateway. These functions can be implemented using one of the various socket APIs available to the students. In this experiment we use the Python socket API, which follows the general socket API as shown in Figure 2, but uses Python specific syntax [16].

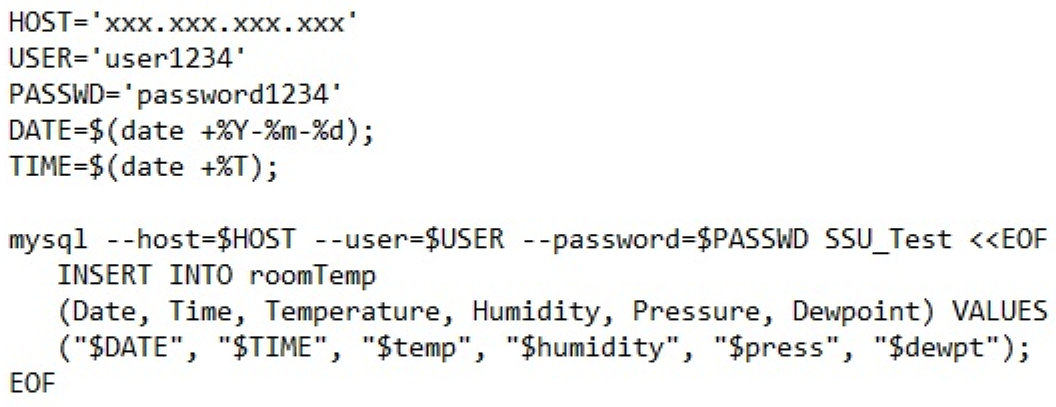
The students must call each of the functions in Figure 2 within a Python script to achieve the TCP connection and receive sensor data from the microcontroller. The first function, *socket()*, creates a socket object and is generally called with two arguments: the address family and the type of socket connection. The address family argument is almost always AF_INET, which sets the socket up to receive IPv4 addresses. The type of socket connection for TCP is SOCK_STREAM, which is a connection-oriented socket, opposite of SOCK_DGRAM which is connection-less and associated with UDP packets.

In order to bind to a specific port number the next function that must be called is b*ind()* which binds the socket to a specific IP address/port number pair. Since the students want to be able to listen to all devices that sends the LGW data, typically the IP address portion of the argument is a set of empty single quotes, which binds to all IP addresses on the network. The specific port number must be explicitly stated. An example call would be *socket.bind(('',9750))*

Students must next call the *listen()* function, which takes as its argument the number of connections to be allowed in queue. The *listen()* function is called to instruct the server to begin listening for incoming TCP connections. The default value is 0, and can be up to 5 connections in queue [16]. Once the socket is bound to a specific address and listening, it can accept a connection from the client using the *accept()* function. The accept function returns a new connection and address. The connection can use the *recv()* function, which allows students to get the data that has been sent. Finally, when there is no more data the connection can be closed with the *close()* function.

It is helpful to package the data from the microcontroller in a way that makes it easy to separate at the gateway. For example weather data pieces could be set in one line as “,70.54, 54.05, 100.51, …,” where it is lead, separated and terminated by a control character like a comma. This allows students to then use the string processing functions in Python to strip the TCP messages received into individual variables. Once the data pieces are individual variables a Shell script can be called from within the Python script to create a MySQL connection, append a time stamp and insert the data in the remote server.

To insert the data into the remote server, MySQL must be installed on the Raspberry Pi using the *apt-get install mysql-server* command. Once MySQL is installed on the Raspberry Pi, the students can insert command line commands into a shell script as shown in Figure 3 below. Data is placed into a specific table using the MySQL command INSERT INTO <table name> followed by open parentheses containing the variable field names as set up in the MySQL table followed by VALUES with another set of parentheses containing the data variables to be placed in the table.



```
HOST='xxx.xxx.xxx.xxx'
USER='user1234'
PASSWD='password1234'
DATE=$(date +%Y-%m-%d);
TIME=$(date +%T);

mysql --host=$HOST --user=$USER --password=$PASSWD SSU_Test <<EOF
   INSERT INTO roomTemp
   (Date, Time, Temperature, Humidity, Pressure, Dewpoint) VALUES
   ("$DATE", "$TIME", "$temp", "$humidity", "$press", "$dewpt");
EOF
```

Figure 3: Sample MySQL Code

4.6. Setting Up the Shared Server

There are several options available to students or the instructor in setting up the remote server. Some universities offer virtual servers to individual departments, allowing them to install and maintain applications such as PHP, MySQL and Apache. Server space can also be rented from various companies [17]. Once an option is selected, a full LAMP (Linux, Apache, MySQL and PHP) stack must be installed on the server [18]. After installing MySQL on the remote server, students can create a database based on a hierarchical structure shown in Figure 4. At the top of the hierarchical structure is the database, which houses as many tables as the students create. Each table contains data fields such as date, time, temperature, etc., where entries can be made in those specific fields.

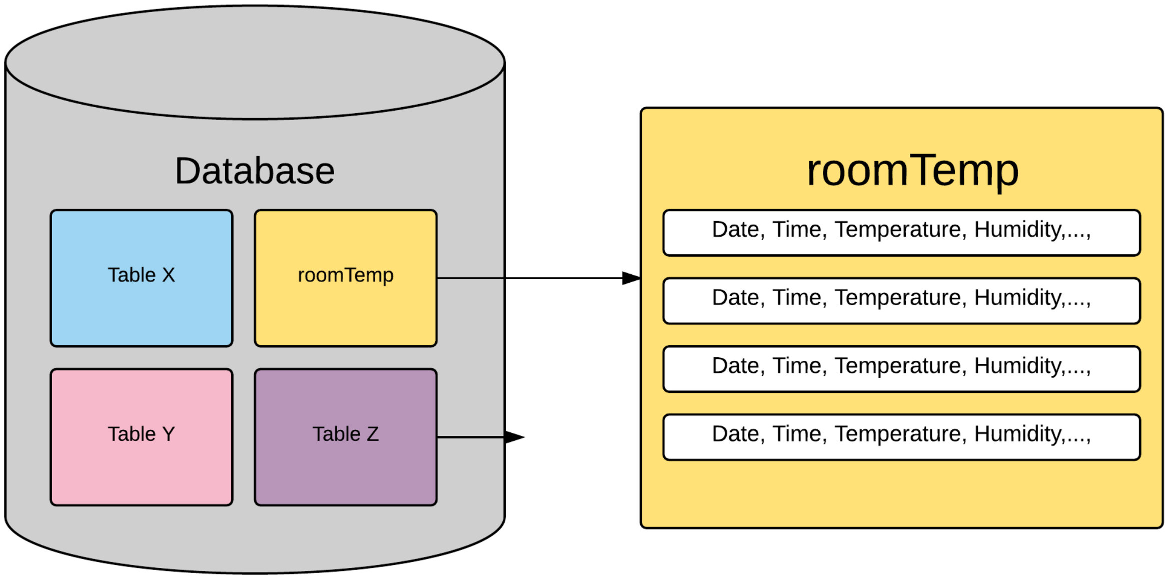
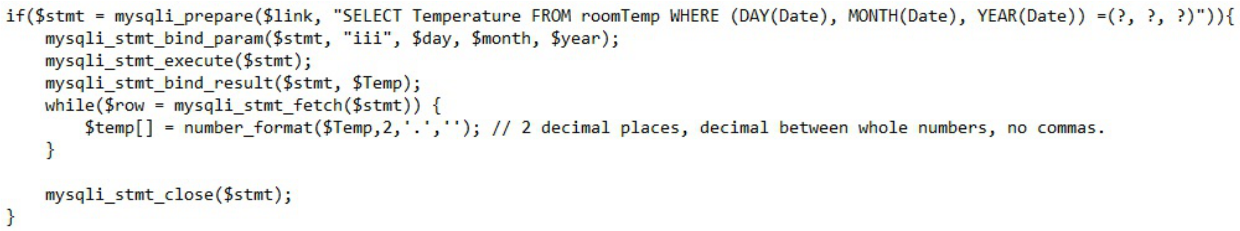

Database
Table X
roomTemp
Table Y
Table Z
roomTemp
Date, Time, Temperature, Humidity,...,
Date, Time, Temperature, Humidity,...,
Date, Time, Temperature, Humidity,...,
Date, Time, Temperature, Humidity,...,

Figure 4: MySQL Hierarchical Structure

The data must be accessed automatically each time the website has a visitor and then visualized. The software package used for querying the database in our method is PHP, a server-side scripting language that offers a handful of services such as the ability to query SQL databases, read/write files on the server, and much more [19]. Sample code for querying a MySQL database in provided in Figure 5. As shown in Figure 6, PHP runs on the remote server and acts as a link between MySQL database and the web browser. Using PHP scripts it is possible to grab all the data out of the MySQL database for a specific time range or date and create a data array that can be passed to a plotting library as shown in Figure 5 below.



```
if($stmt = mysqli_prepare($link, "SELECT Temperature FROM roomTemp WHERE (DAY(Date), MONTH(Date), YEAR(Date)) =(?, ?, ?)")){
    mysqli_stmt_bind_param($stmt, "iii", $day, $month, $year);
    mysqli_stmt_execute($stmt);
    mysqli_stmt_bind_result($stmt, $Temp);
    while($row = mysqli_stmt_fetch($stmt)) {
        $temp[] = number_format($Temp,2,'.',''); // 2 decimal places, decimal between whole numbers, no commas.
    }

    mysqli_stmt_close($stmt);
}
```

Figure 5: Sample PHP Code to Query a MySQL Database

Browsers are only capable of displaying basic HTML code, so when a user requests a PHP based site, the PHP must be converted into HTML. This detail is largely handled by the Apache HTTP server software. A general mapping of the process can be viewed in Figure 6 below. Apache runs on the remote server and provides HTTP access to users who visit the website. When a visitor enters the URL of the website, Apache looks for a file named *index.html* or *index.php* to return to the user for viewing. If it is a *.html* file it returns it directly, if it is a *.php* file it executes the PHP code and returns the HTML generated by PHP. There is no configuration of Apache required, other than it be installed on the remote server.

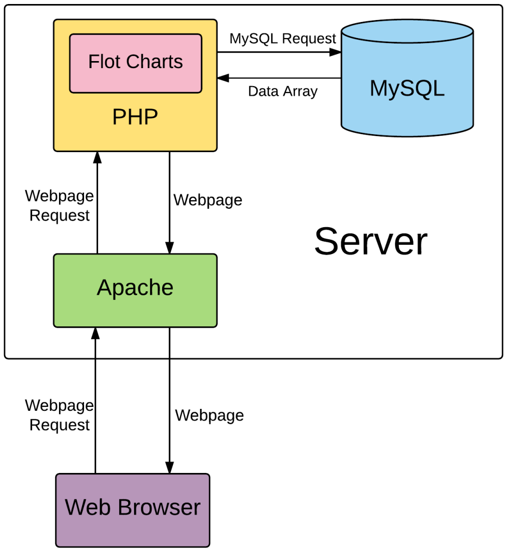
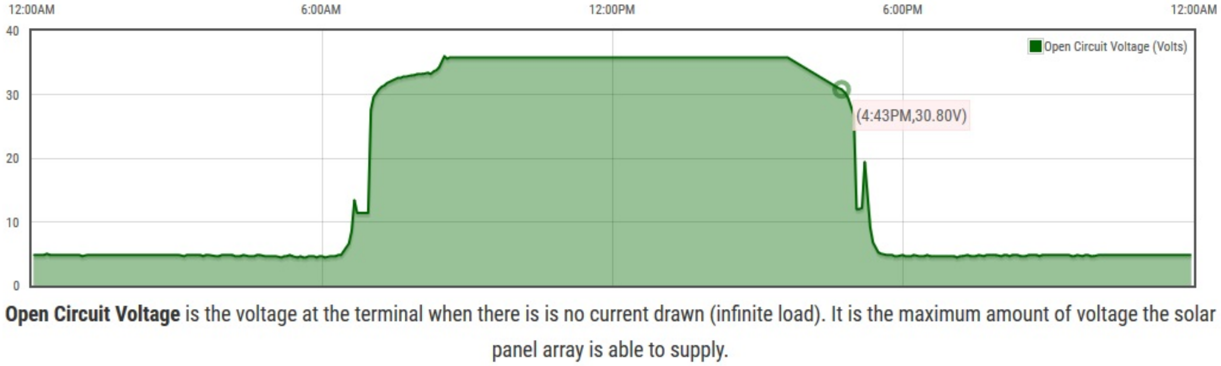


Figure 6: Server Relationships

4.6. Implementing the Data Visualization

The final goal of this experiment is to provide a way for students to display the sensor data on the Internet in an attractive and informative way. Once a database has been established and data from the sensor node populates it, students can choose from several different plotting options such as Google charts, Flot charts, and many more [20]. In this experiment, Flot charts are used exclusively [21].

Flot charts were selected for their relatively ease of configuration to create highly customized, attractive graphs, and have a large amount of control over how the graphs appear and how end users can interact with them. The graphs can be dynamically updated to allow website visitors to select ranges of data to view and also create and download comma separated value (CSV) files.



Open Circuit Voltage is the voltage at the terminal when there is is no current drawn (infinite load). It is the maximum amount of voltage the solar panel array is able to supply.

Figure 7: Sample Flot Graph - Solar Panel Monitoring

The data array created with a PHP query of the MySQL database can be passed directly to Flot charts and plotted. Setting up the Flot charts requires downloading several JavaScript plug-ins and writing code in JavaScript. There are a handful of examples on Flot's website [21].

Using Flot charts, students can perform a number of interesting exercises with the generated plot. The displayed graph can be a line graph, a bar graph, or points. The graph can also be filled in and have different opacity options applied. Students have the option of making the graph interactive, where a user can view the specific data points by hovering over it with his mouse. Shown in Figure 6 is an example line graph over a 24 hour period using the hoverable functionality. There are coloring options to stylize the grid behind the data, and various options for labeling the axes. Students can also select the interval of time in which the data should be displayed. This is established by the query made by PHP to MySQL. If students want to display the data for a specific time frame, this can be done by passing a range to MySQL for the specific data to be returned. The range of time can then be passed to the x-axis of the Flot chart to correctly display the time series. After these options are selected, and the JavaScript code is placed into a file and uploaded to the remote server in the public HTML directory. This allows visitors to see the graphs created by students.

5. Expansions and Future Works

There are several features to this experiment that could be implemented by students who excel at the initial setup. In this section we briefly describe possible approaches to improve and enhance the features of the proposed cloud experiment. We first elaborate on enhancements to data visualization and web design and then we point out potential approaches to enhance hardware capabilities of the system.

The first topic of expansion would be to ask students to change the webpage to allow the web page user to select and display data from different time intervals. In this case, students can include a drop down menu for: one day, one week, one month, one year, or whatever the data supports.

Another interesting extension to this experiment is to ask students to modify the web site to display the data using different scientific units, such as Fahrenheit or Celsius. Interested students can also modify the web site to allow visitors to download the data in various formats, such as comma separated value format.

Students can also develop software on the Raspberry Pi that enables it to communicate with multiple sensor nodes, each sending a different set of data. In this case, the Raspberry Pi would have to identify where the data came from and the database table it belongs in. This could be accomplished by including the table name with the data from the microcontroller and reading the address that is returned from the *accept()* function in the Python API.

On the hardware side of the project, power consumption is always a good area to improve. Interested students can use sleep settings and a watchdog timer reset to maximize battery life while still gathering data at an acceptable rate. This can also lead to students learning more about interrupt subroutines to wake the microcontroller from sleep mode if the specific project requires such a feature (e.g. a pulsing wind speed meter).

6. Outcomes

Over the last year, sixteen engineering students attending the capstone class have participated in the Data Visualization workshop (see Section 3). The workshop was given by a graduate student and supervised by a faculty member. At the end of the workshop, all participating students were asked to complete a mandatory short survey. The survey was primarily based on quantitative questions with the last question asking for student comments and suggestions. We note that this is an ongoing project and in order to fully understand its learning impact more data is required.

All of the students taking the capstone project attended the workshop. Almost all of the students felt IoT, LAMP, and Data Visualization were important concepts that they should be familiar with before they graduate and seek employment. On average, participating students spent 12 hours working on the assigned exercises, as described in Section 3, outside of the workshop. The majority (81 percent) of students expressed that the introduction to MySQL and PHP Section in the workshop should be expanded by a few more hours. While most students felt they were comfortable with the concept of the LAMP, for many (93 percent) using Flot and graphics was a completely new experience. Almost all students were very satisfied with the workshop and felt they had gained a high level of understanding about data visualization. Many participating students (65 percent) indicated that they are planning to incorporate the IoT experiment in their own senior design project.

1. Conclusion

In this paper we introduced our ongoing work in designing and implementing a cloud-based data repository and visualization service. The focus of this exercise was to teach students ways to implement their own sensor data visualization capability. Using data visualization students can present their sensor data and assign meaning to the collected data and demonstrate the importance of the collected data. The students received this training during the first two weeks of their two-semester capstone course. Due to the structure of our capstone course, we offered this training as an 8-hour workshop outside the class time and highly encouraged all the students enrolled in the capstone course to participate. Throughout the workshop students reviewed important concepts in MySQL, PHP, LAMP, and Flot charts, and understood what it takes to transfer, store, and visualize the data on a remote server. Overall, the majority of students were satisfied with the learning objectives of the workshop, however, they expressed that certain sections of the workshop could be extended by a few more hours.

References

[1] M. A. Feki, F. Kawsar, M. Boussard, and L. Trappeniers, "The Internet of Things: The Next Technological Revolution," Computer, vol. 46, no. 2, pp. 24–25, 2013.

[2] Press Release: Gartner Says By 2020, More Than Half of Major New Business Processes and Systems Will Incorporate Some Element of the Internet of Things, 14 January 2016, Online. January 2016 < http://www.gartner.com/newsroom/id/3185623>.

[3] D. Evans, "The internet of things, how the next evolution of the internet is changing everything," in Cisco Internet Business Solutions Group (IBSG) white paper, April 2011.

[4] VisionMobile, IoT: Breaking free from Internet and things. June 2014. Online. January 2016 < http://www.visionmobile.com/product/iot-breaking-free-internet-things/>.

[5] Postscapes: Tracking Internet of Things. Online. January 2016 < http://postscapes.com/internet-of-things-platforms >.

[6] LAMP Guidelines, Online. January 2016 < https://www.linode.com/docs/websites/lamp/>

[7] Introduction to AppInventor, Online. January 2016 <http://www.appinventor.org/content/CourseInABox/Intro/WelcomeToAI>

[8] ABET Engineering Accreditation Commission, "Criteria for Accrediting Engineering Programs: Effective for Reviews During the 2014-2015 Accreditation Cycle," Oct. 26, 2013. <http://www.abet.org/eac-criteria-2014-2015/>

[9] PIC18F25K20 – 8-bit PIC Microcontrollers. Online. January 2016. <http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC18F25K20>

[10] BME280 Weather Sensor Datasheet. Online. January 2016. <https://www.adafruit.com/datasheets/BST-BME280_DS001-10.pdf>

[11] Choose Between Raspberry Pi or BeagleBone Black. February 2015. Online. January 2016. ,http://makezine.com/2014/02/25/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/>

[12] Why MySQL. Online January 2016.< https://www.mysql.com/why-mysql/>

[13] Raspbery Pi Basics: Installing Raspian. Online. January 2016. < https://www.howtoforge.com/tutorial/howto-install-raspbian-on-raspberry-pi/>

[14] Netcat – SecTools Top Network Security Tools. Online. January 2016. <http://sectools.org/tool/netcat/>

[15] Socket Programming. Online. January 2016. <http://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html>

[16] Socket – Low-Level Network Interface. Online. January 2016. <https://docs.python.org/2/library/socket.html>

[17] The Best Web Hosting Services For 2016. January 2016. Online. January 2016. <http://www.pcmag.com/article2/0,2817,2424725,00.asp>

[18] How To Install Linux, Apache, MySQL And PHP On Ubuntu. Online. January 2016. <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu>

[19] PHP5 Introduction. Online. January 2016.< http://www.w3schools.com/php/php_intro.asp>

[20] The 15 Best JavaScript Plotting Libraries. Online. January 2016. <http://www.sitepoint.com/15-best-javascript-charting-libraries/>

[21] Flot: Attractive JavaScript plotting for jQuery. Online. January 2016. <http://www.flotcharts.org/>

[22] F. Farahmand, K. Ely "Generating Start-up Relevance in Capstone Projects ," White paper.