Senior Design Final Report

Ames Laboratory Monitoring System

Spring15-Fall15

Customer: Ames Laboratory: Division of Materials Sciences and Engineering

Matthew Besser: Asst Scientist IV - Besser@ameslab.gov

Advisor: Leland Harker: Electronics Technician

Leharker@iastate.edu

Project Team: Scott Hood

Daniel Kieth

Eric Graika

Shiai Liu

Luke McDonald

1. *Introduction*

Project Definition

Our client was looking for the creation of a system that would act as a alert / notification system that monitors lab machinery during their experiment lifecycle. The machinery uses thermocouple and pressure resources to run its experimentation, these resources are important not only for maintaining the uptime and accuracy of the experiment, but the safety and health of the machinery. Our client therefor was looking for a way that would allow him to let his machinery run, without the need to continuously check his machines. Our group's task was to design from scratch a system that would read in sensor values, interpret the voltage output from the sensors, and notify personnel of potential risks.

Project Goals

Functional Requirements

- Our first goal was designing a project that would accomplish our client's needs. We needed to build something that would take in sensor readings, interpret those readings, and dictate whether or not machinery was at risk.
- Our second goal was making it affordable and reproducible. Our client realized that the equipment he was looking to make already existed, but these systems are expensive and don't meet the needs for a variety of experiments. He wanted to be able to build a singular system that could be used for a variety of experiments, while also accomplishing his basic needs.

Non-Functional Requirements

- Our first non-requirement goal was attempting to add extra functionality that would allow users to check their system from offsite. Creating a web client that was easy to use and manage by users with little programming knowledge.
- The second was real time viewing of data. Being able to see graphs and the data from their experiment run times. This we felt would add to the web clients functionality and allow scientists to view equipment status without constantly physically checking.

Definition of Common Terms and Abbreviations

Instrmon – Referring to the remote server that houses our SQL database, php scrips, and apache client.

Database – Referring to the SQL database housed on the instrmon remote server.

Web client – Referring to the HTML and PHP scripts that run the web user interface via the servers https apache client.

Emailer – The name of the script that acts as the systems notification system for the project.

System – Refers to the notification project as a whole, both the software and hardware aspects.

2. Project Design

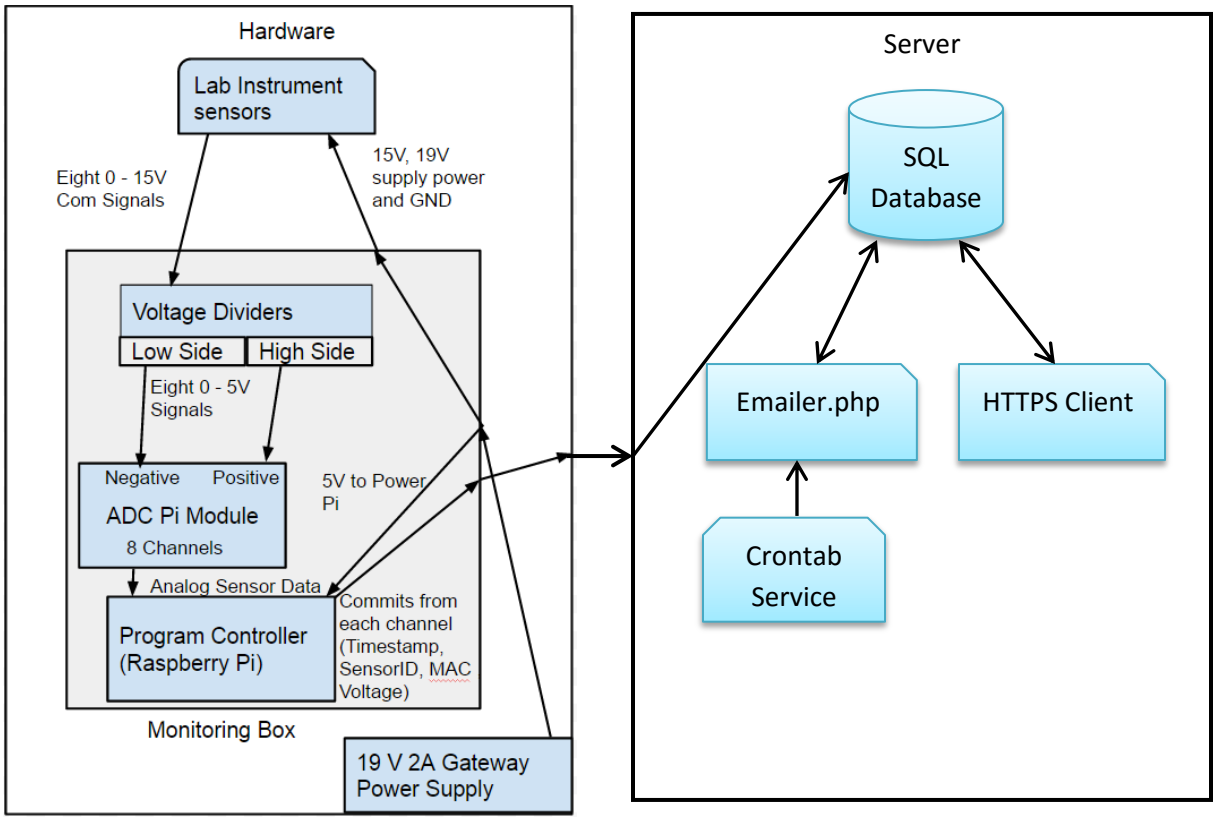
2.1. Outline

Our group based upon the projects requirements and goals decided to create from scratch a two part system, a hardware system and a server client side system that together would accomplish our goals and functionality for this project.

The hardware that is implemented is designed to read output signals from various thermocouple and gas transducers to be collected by a Raspberry Pi and an ADC Pi module. The ADC Pi module is intended to collect 0 to 5V signals from the equipment which is made possible by voltage dividers which downscale the voltage range from 0 to 15V to 0 to 5V. The sensor output signals come from thermocouple amplifiers which convert R, S, T, E, J and K type thermocouple into 0-15V signals or there from gas transducer signals which are 0-5V and 0-10V both of which are installed within the Ames Lab equipment. Once these signals are collected by the ADC Pi module the Raspberry Pi will commit the data to the database for use on the website.

The server client would then be designed to do all the back end work, interpreting, notifying, and providing extra user functionality. We needed something that could work within Ames Laboratory effectively, so we requested the creation of a virtual server to be housed on the labs network. This allowed access to all of Ames Lab security, access to important network paths, and capabilities to build and display user interface on the labs https network. This allowed us to create storage space for the sensor data using an SQL server, run PHP scripts to interpret the data and send notifications, and use an apache client to display user interface. The user interface would be the system that decides who to notify, when to notify, and give us the potential to build extra systems to add helpful functionality and real time viewing of the data.

2.1.1. Block Diagram



This model shows communication between the equipment and our monitoring box and then data being committed to the database to be processed and converted into useful information. The emailer and HTTPS Web Client communicate back and forth between the database sending in user input and data conversions.

2.2 Hardware

2.2.1 Requirements

The requirement for the hardware side of the project is to be able to collect output signals from the various Ames Lab equipment thermocouple and gas transducers that are currently installed. In order to collect the sensor output signals from the equipment Ames Lab employees will have to wire the sensors they want to monitor to a 12 pin connector which is mounted on the equipment that the monitoring box will plug-in to in order to collect the output signals from the sensors. Sensors can have various outputs, the sensors that we planned to supported will be ones with output signals of 0-5V, 0-10V, 0-15V, and 4-24mA. Thermocouples will fit in this category because of thermocouple amplifier which increases a normal thermocouple signal of 0 to 53mV to 0-15V. By default gas transducer output signal fit in this category as will.

System supports:

- **Gas transducers**: so long as the M and B values for the linear conversion from voltage to PSI is known to the user and have an output signal less than 15V.
- **Thermocouples:** Supports thermocouple of type R, S, K, T, J and E.

Specification for Sensors

The following specification do not represent what in currently being used by Ames Lab rather what we believe our monitoring box is capable of supporting. What we can support has yet to be fully tested.

Thermocouples:

These sensors have a voltage output of 0 to 53mV before amplification. After amplification through the thermocouple amplifier the output signal is 0-15V this improves the resolution of the output signal by a 100 times so conversion to Celsius will be much more accurate. The amplifier allows us to use the conversion of 10mV per degree Celsius instead of the much less forgiving conversions using a polynomial equation to translate voltage into a usable digital signal that is in degrees Celsius which a thermocouple without amplification uses.

- Thermocouple Specifications
    - All thermocouples use voltages in the range of 0 - 15V after amplification in this system and conversion to Celsius is 1 degree Celsius to 10mV.
    - R-type Thermocouple
        - Reads temperatures from -50 to 1768C
    - S-type Thermocouple
        - Reads temperatures from -50 to 1768C
    - K-type Thermocouple
        - Reads temperatures from –270 to 1372C
    - T-type Thermocouple
        - Reads temperatures from -270 to 400C
    - J-type Thermocouple
        - Reads temperatures from -210 to 1200C
    - E-type Thermocouple
        - Reads temperatures from -270 to 1000C

Gas Transducers:

Gas transducers can be used by the equipment to measure supply tank pressure, environment pressure, and so on. To function properly a gas transducer requires a 12-36V input to be powered. It will then return a 0 to 5V, 0-10V, or 4-24 mA signal to the ADC module (The gas transducers we have be looking at). That will be converted into a usable digital signal by using the linear equation y = mX + b which can convert voltage or current into a PSI value which can then be converted into the desired pressure unit.

- Gas Transducer
    - Read in the range of 0 – 1500 PSI
    - Use voltages in the range of 0 – 5V, 0-10V, or 4- 24mA
    - Requires a voltage source between 12 – 36V for power
    - Conversion equation: y = mX + B
        - X is voltage from Gas Transducer
        - M is Range that Gas Transducer can measure divided by the max com signal voltage.
        - EX: M = (1500 – 50)PSI range / 10V Max com signal
        - B is the offset from 0
            - If range is -25 to 300 PSI then B is -25
            - If range is 50 to 1500 PSI then B is 50

Thermocouple Amplifiers:

Thermocouple Amplifiers are used to amplify thermocouples from 0 to 50mV to 0-15V which gets a better resolution with the higher voltage output to get a more accurate degree value. To use the signals from the amplifiers the monitoring box contains a voltage divider for each channel going to the ADC Pi. Both amplifiers that are being used require a 15V power source to accurately measure our desired range of temperatures.

- K-type Thermocouple Amplifiers (Old Part still useful if negative Celsius needs to be measured)
    - Using a 15V supply voltage
    - Conversion equation is: T = (Vo – 1.25)/5mV
    - Can make voltage representation of temperature for: -250 to 3000 C

- Variable Thermocouple Amplifiers (From Herdware.com)
    - Using a 15V supply voltage
    - Amplifies R,S,T,E,J and K type thermocouples
    - Conversion is: 10mV per 1 degree Celsius
    - Can make voltage representation of temperature for: 0 to 1500 C

Conversion Equations for the Website:

For the conversion from voltage to readable temperatures for thermocouples we used the following equation:

- 10mV to 1 degree Celius

For the conversion from voltage to readable pressure for Gas Transducers. We used the following equation:

- Conversion equation is y = mX + B
    - X is voltage
    - M is Operational Range divided by the max com signal voltage.
        - Example: M = (1500 – 50)PSI range / 10V Max com signal
    - B is the offset from 0
        - If range is -25 to 300 PSI then B is -25
        - If range is 50 to 1500 PSI then B is 50

For the conversion from current to readable pressure for Gas Transducers. The following equation can be used:

- Conversion equation is y = mX + B
    - X is current
        - Which is gotten by Voltage/ 3300 ohms (I = V/R)
            - 3300 ohms is the total resistance of the voltage dividers
    - M is Operational Range divided by the max com signal current.
        - Example: M = (1500 – 50)PSI range / 23mA com signal
    - B is the offset from 0
        - If range is -25 to 300 PSI then B is -25
        - If range is 50 to 1500 PSI then B is 50

2.2.2 Raspberry Pi / Monitoring Box

The monitoring box is designed to collect outputs signals via a 12-pin connector on the Ames Lab equipment. These output signals come from a system of gas transducers and k-type and R-type thermocouples that are attached to the Ames Lab equipment. The output signals from the sensors can be in the form of a voltage signal that is within 0 to 15V range or a current output. The signals will all be converted into 0-5v for the ADC Pi Module to read by voltage dividers. These 0-5v signals will then be collected by the ADC Pi and then scaled from the 0-5V signal to the original output signal voltage with in the Raspberry Pi. The Raspberry Pi will then commit the output signals to the "Volts" table with in the database. The website will handle the conversion from voltage to Celsius or PSI. Whether the voltage is converted into Celsius or PSI values depends on the setting the user input on initial setup of the sensor and monitoring box.

Components of Monitoring Box:

- Raspberry Pi ADC module
    - Reads 0 - 5V analog signals on up to 8 channels
    - Records voltages as a 17-bit value
    - Can measure micro volts

- Raspberry Pi
    - Receives data from ADC module and scales the voltage by 3 to its original voltage from the sensor
    - Then the Voltage value and additional information is committed to the "Volts" table in the database.

- Gateway Laptop Power Supply 0 – 19V , 2A
    - Used to power the Raspberry Pi with 5.1V and up to 500mA
    - Used to power K-type amplifiers with 15V
    - Used to power variable thermocouple amplifier with 15V
    - Used to power Gas transducer with 19V

2.2.3 Design Choices

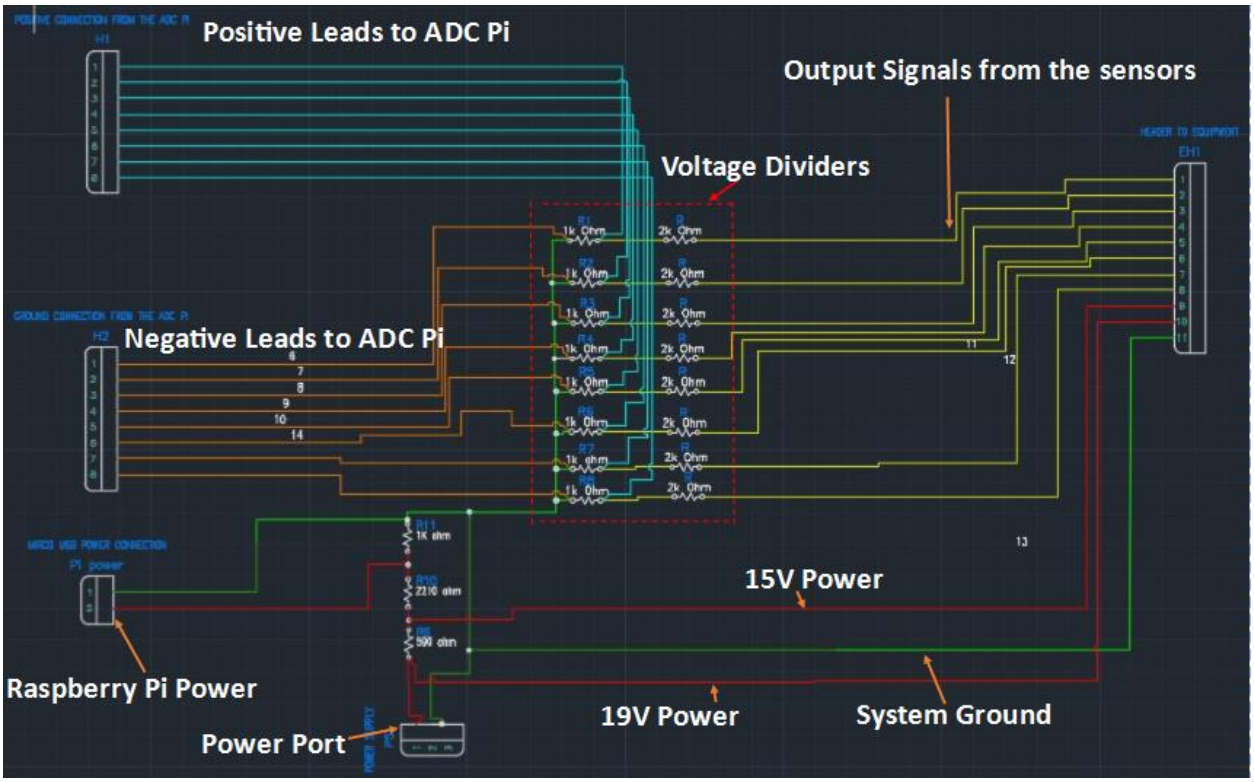
Raspberry Pi vs Arduino

Both the Raspberry Pi and Arduino are capable of capturing and storing voltage signals but the Arduino lacks a proper operating system which is required for this project. To allow for the voltage sender program run at boot time and send voltages to the database. In order for the Arduino to be able to communicate with the database it would need network access via an Ethernet module which would have to be purchased separately. While with the Raspberry Pi the Ethernet module was standard. In additional while doing research for how to capture voltages with a Raspberry Pi, we found a ADC Pi module which can collects 8 channels of 0 to 5V signals and converts them into a 17-bit float value which allows for voltage measurement accuracy to millivolts. This was helpful because it was necessary to measure voltages from an unamplified thermocouple. So we decided that the Raspberry Pi would work best.

Standardized channels vs Custom channels

Using standardized channels, which was assigning a select task to each channel, on the ADC Pi module was our original plan till we implemented thermocouple amplifiers. This took the 0 to 53mV thermocouple signal and amplified it to 0-15V which caused the channel to work with voltages on the same scale. That made using standardized port points less since all of the channels could handle any voltage with in the 0 to 15V range. The standardized channels were set up as follows: channels 1 through 4 were k-type thermocouples, channel 5 was R-type thermocouple, and channels 5 through 8 were gas transducers. The gas transducers already required voltage dividers since they could be 0 to 10V to keep signals in the 0-5V range for the ADC Pi to collect. Then with the addition of thermocouple amplifier which took the 0 to 53mV signal and amplified it to a 0 to 15v signal which made all thermocouple channels require a voltage divider. All channels required a voltage divider. So in order to satisfy the need for all 8 channels and either a thermocouple or a gas pressure sensor we used a standard voltage diver that converts signals too scale to 0 – 5V. This allows the ADC Pi module to collect any combination thermocouple or gas transducers signals.

2.2.4 Assembly and Wiring



The above wiring design is the internal wiring of the monitoring box which is designed to take in eight communication signals from the equipment sensors and to use voltage dividers to downscale output signals into 0 to 5V signals. The wiring also divides up the voltage from the gateway power into 19V, 15V, and 5V supplies for the gas transducers, thermocouple amplifiers, and the Raspberry Pi.

2.2.5 Network

Raspberry Pi Network settings

In order for the Raspberry Pi to work on the Ames Lab network it first has be register by its MAC address. Network access is required to be able to communicate with database. Once registered, the Raspberry Pi will have an assigned static IP address set by the DHCP server so there is no need to set the IP address locally on the Raspberry PI in its /etc/network/interface file. The DHCP server will lease only one IP address to the PI which is determined by its MAC address.

2.2.6 Internal Software

Raspberry Pi remote access

Remote access to the Raspberry is a useful feature since it normally takes a monitor with a HMDI port, mouse, and keyboard to interface with the Raspberry Pi to use it. With remote access the need for all that equipment to access the Raspberry Pi is eliminated. For remote access the Raspberry Pi uses the default windows remote desktop connection program (Microsoft Remote Desktop) which can be found preinstalled on new windows machine. The remote desktop program on the Raspberry Pi is setup to start up a remote session on boot-up so it can be used for remote troubleshooting so long as a network connection is available. Remote access also eliminates the need to physically interface to each Raspberry Pi to work on them. Also the box in which the Raspberry Pi is housing is quite small so hooking-up a mouse, keyboard, and monitor could be difficult.

Data collection programs:

The Raspberry Pi runs a program call sendervoltage.py which starts whenever the Pi is powered on to prevent need for user manual start the program each time the Pi loses power. Sendervoltage.py is started by a script called launcher.sh which is a shell script. Crontab (daemon which allows commands to be executed at specific times) holds a command line which runs launcher.sh which runs sendervoltage.py.

- Raspberry Pi (sendervoltage.py)
    - Written in Python
    - No user needs to modify code since no unique values are hardcoded
    - Scales voltages to proper value
    - Sends Voltage, time, Mac address, and channel id to the “Volts” table for each channel in the monitoring box
    - Sampling rate of one reading per 2 minutes

Scripts

Crontab: launcher.sh

```
cd /
cd home/pi/ABElectronics_Python_Libraries/ADCPi/
sudo python sendvoltage.py
cd /
```

Shell script which Crontab executes on boot up which starts sendvoltage.py

sendervoltage.py

```
#setup connection to the database
db = MySQLdb.connect(passwd=[redacted], db=[redacted], host=[redacted],port=[redacted],user=[redacted]);
curs=db.cursor()
```

```
    print ("Channel 1: %02f" % adc.read_voltage(1))
Volt = (adc.read_voltage(1)*3)
query = """INSERT INTO volts(sensorID, voltage, time, IPaddress) VALUES ( %s,%s,%s,%s)"""
curs.execute(query, ( 1,Volt,Time,readableMAC))
```

Code for committing information to the database for channel one

2.2.7 Testing

Phase 1: Checking resistance and circuit connection.

Checked each resistors value before assembly the internal wiring to verify resistance value was with in its tolerance rating. After which internal wiring is assembled and checked for completed connections by check for resistance from the barrel connector power terminal to the sensor output wires from the equipment. After that the connection from the sensor output wires to the negative and positive ADC Pi wire leads was tested. Once verified, any sensors hooked up to the 12-pin connector will have their output signals recorded by the ADC Pi module.

Phase 2: Committing programmed voltage values to database.

Testing was done by sending dummy values from the Raspberry Pi to the database to verify the connection. Set up valid timestamp, MAC, and sensor channel id to be sent to the database to verify that the database will accept the each of the values in their current format.

Phase 3: Committing voltage values to database.

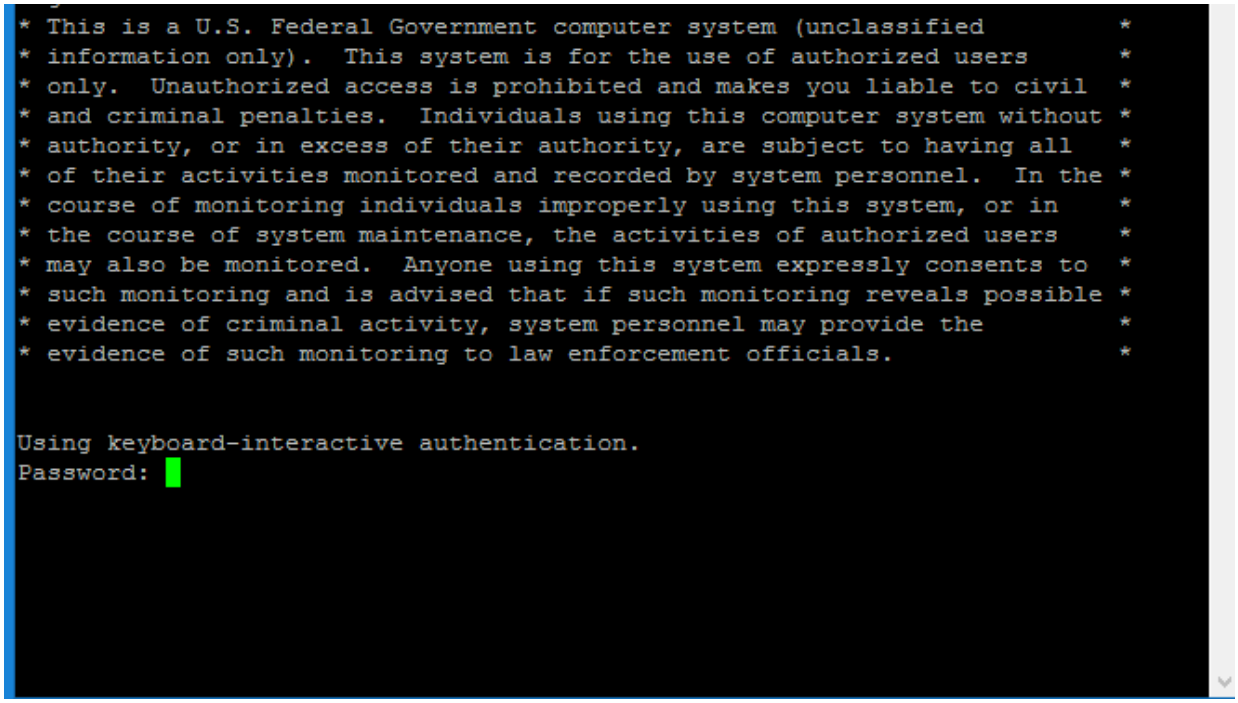
Hook up a K-type thermocouple to channel 1 of the ADC Pi module with a thermocouple amplifier to verify that the thermocouple amplifier will send a correct voltage value relative to the room temperature of the room the thermocouple is set up in. It ran for a couple days to populate the database with enough value to check for conversion errors or transmission errors.

2.3. Software

The software is run on a Centos 6 virtual machine with the following specs.

- Dual Core Processor
- 4 GB's RAM
- 80 Gigabyte Disk Space.

The server was provided to us by Ames Laboratory Informational Systems department who will be maintaining the server stability after our group finishes its work on the project. This server is housed on the Server network group and has been modified so its apache client can be view by all Ames Laboratory Wired Networks. This provides a extra measure of security, so only people wired on the network, or who VPN on to the internal network have access to view the https client. Direct server access is only capable for those who have Ames Laboratory accounts and are given permission to access the server through VPN and also given access to the correct server groups inside the server itself. This creates a thick layer of security for our application, with no risk of individuals injecting attacks onto the PHP scripts and SQL server.



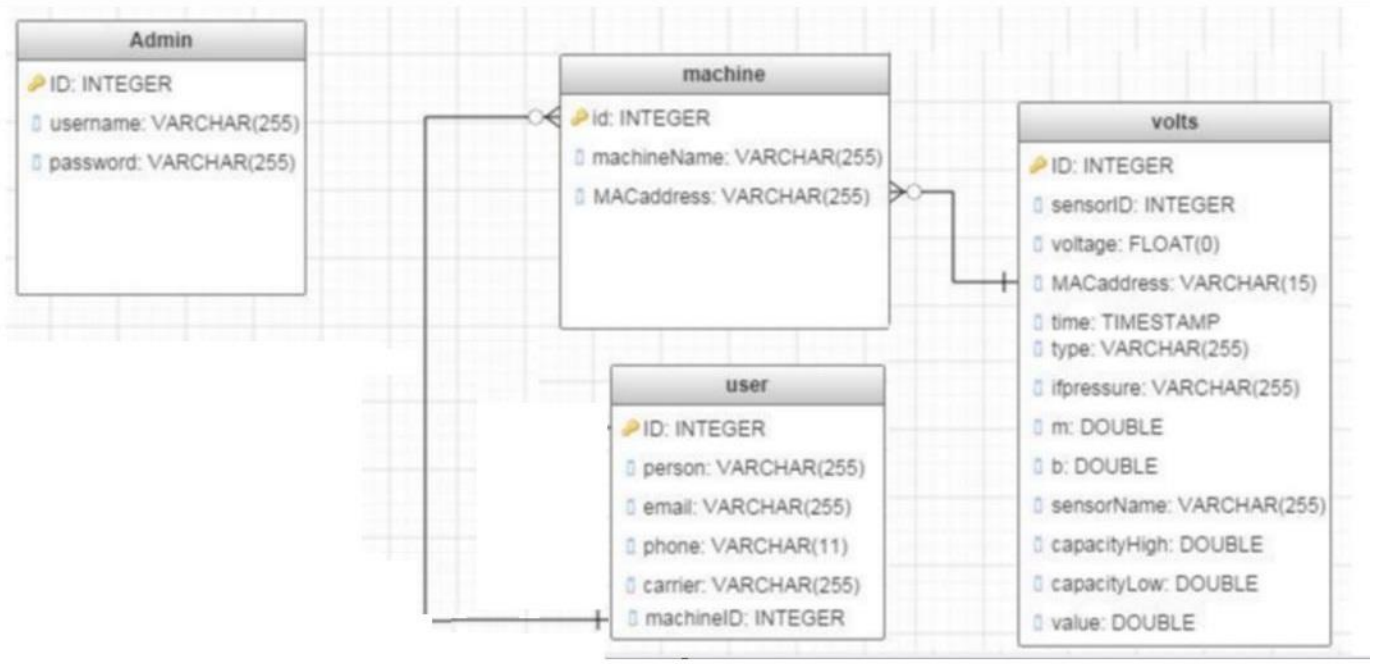
The server houses an SQL Database, an Apache Web Client, and all the background PHP scripts that run the back end interface for the web client. Another functionality used in our project is called Crontab which is a job scheduler to run applications on the server and given times. We use this to run our value converters every 2 minutes, and our emailer notification system every hour.

2.3.1. Requirements

The requirements for the software were very basic, the client wanted an easy to use interface with built in functionality to make it resistant to user error. Due to the nature of the project the key users were scientists and the user interface needed to be plain and easy for non-technical users to understand. On top of that non-web savvy personnel would be the ones in charge of the project after we were completed, so it needed to be something that was easily taught to them. Other than these simply most software design choices were up to us and Ames Lab Informational Systems department to decide.

Informational Systems required us to create service accounts and go through their authentication processes in order for the server to communicate to the Ames Lab network. This was handled by opening up HTTPS ports and bypassing firewalls in order to safely encrypt information.

2.3.2. SQL Database

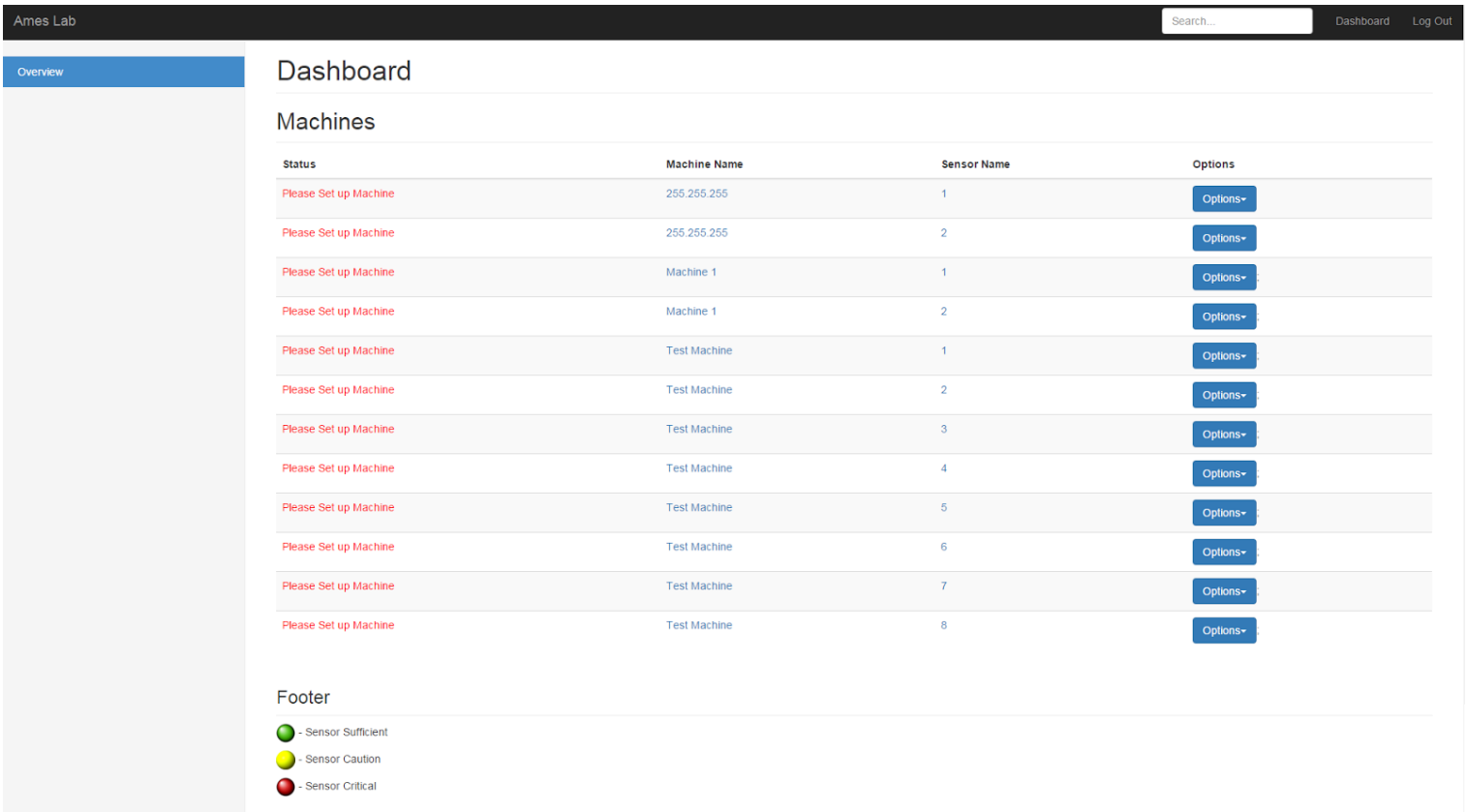


Our sql database has four tables. Admin is used only for logging into the website. The admin table only has one row so there is only one username and password that may log in. The other tables are all connected via the MACaddress which is one of the things values that gets pushed from the Raspberry Pi. The first row pushed to the volts table by each sensor is the row that will hold all the information for that sensor. The machine table is what links all the sensors to what machine it is and uses the machine ID to know what users to notify.2.3.3. PHP Scripts

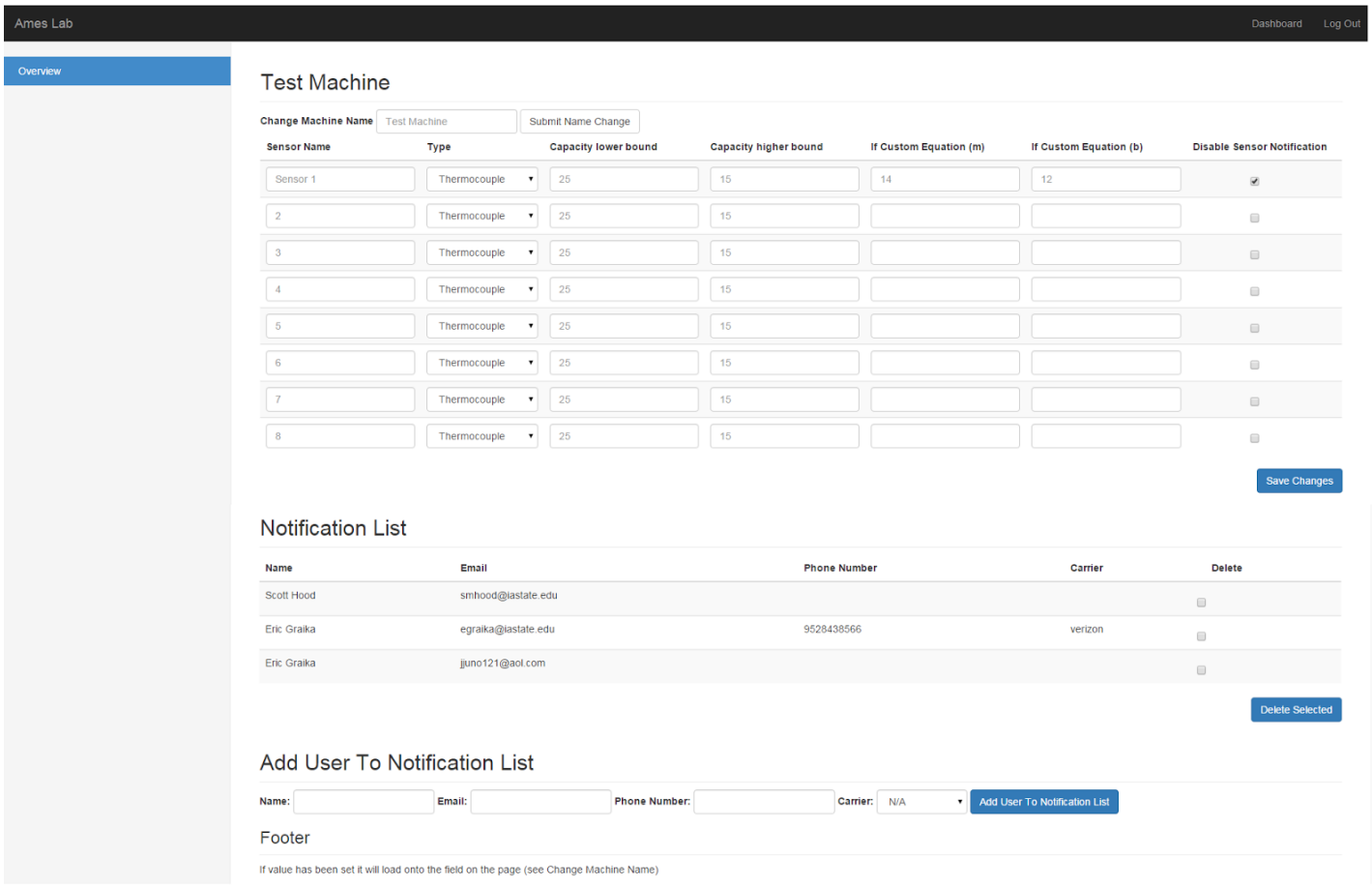
2.3.3 PHP Scripts

User Interface

The home page of our web server is called the Dashboard. This page displays all the sensors that are currently active. It knows to bring up only sensors that have been set to a type other than the default which is “not active”. If other values still need to be set a red text will display under the status says “Please Set up Machine”. This can be done only by an admin logging in at the top right of the page. After logging in it allows the admin to click the link to the specified machine that needs to be edited. That page is Machine Edit page and will be explained after the Dashboard. But also on the Dashboard we have a link to the sensor name which will redirect the user to a page that has a graph of the data recently pulled from the sensor. To the right of that is the options which has three options: View, Edit Machine, and Delete Sensor. View is the same as clicking the sensor name link taking you to the graph of the sensor. Edit Machine is the same as clicking the link to the machine edit page and will only be seen if an admin is logged in. Delete Sensor will also only be seen if an admin is logged in and will delete all the data from the sensor so that the admin can reset the sensor. Finally, at the bottom of the page is the footer which shows the different states the status of the sensors can be. Green being everything is all good, yellow being that a sensor is approaching a threshold, and red being a sensor needs to be attended to.



The Edit Machine page allows a logged in person to change the machine name, set up sensors, and edit notification lists for the machine. Only sensors that have been pushing data will be displayed and after first pushing data will have no values in any fields. To set up a sensor the admin will have to set a type and a higher and lower bound. If the admin selects PSI or MPa then they will also have to set the m and b for the custom equation y=mx+b and finally if they want the sensor to be included in notifications. Below that is the notification list which will display only users who have been added to this list. The user may check as many out of this list as they want and delete them. When adding someone to the list name and email are required. Phone and Carrier are not required but when inputting a phone number the format must be xxxxxxxxxx where x is 0-9.



Testing Machine Edit:

One of the biggest problems we had in figuring out why values wouldn't update was variables. We weren't sure if variables were being passed through php pages. One way we checked to see if these variables were being passed through was by doing a variable dump onto the page and then suspending the page. We could see all the variables passed through the form posts. After seeing that variables were passing through php pages we checked to see if queries were actually updating and populating the correct fields.

2.3.5 Email System:

General Information

The email system works within the server as a background script behind the web interface. The server is setup with authentication to run SMTP commands within the server. The server reroutes any SMTP to be sent through mail hub the primary mailing service for Ames Laboratory, and uses a pre-built service account provided by the Ames Laboratory Informational Systems department. This allows capabilities to send outbound emails to not only Ames lab emails, but other domains. It also allows for output emails to be sent as text messages, however having a limitation on the amount of characters that can be sent. The script itself is written in PHP due to its built in functionality with SMTP and in order to maintain consistency with our other written scripts. The script itself is then run using Crontab job scheduler which is built into the server as a service device at the first minute of every hour.

Script Details:

The emailer.php acts as both a notification system and a converting system. In order to determine whether or not an email should be sent, it must first convert the values and check for out of bounds variables.

The script searches through all machines that have been set up with a "Type" for the sensor, in order to accurately convert the values as well as to keep from searching through the sensors that are not being used by the machine. It goes through retrieving the values it needs, and then starts making the conversions.

Conversions:

- Thermocouple (Celsius): *$Celsius = $voltage / 0.1;*
- Pressure (PSI): *$PSI = $m * $voltage + $b;*
    - $m and $b are the values for any gas custom equation.
- Pressure (MPa): *$MPa = ($m * $voltage + $b) / 145.0377;*
- Possible future conversions being: Torr and mTorr later implemented down the road.
- While the conversions are happening, the script sets the converted values back into the database in the same row in the "value" column to be easily displayed by the web interface.

```
if($type == "thermocouple"){
    $value = $volt / 0.1;
    if($value > $upper || $value < $lower){
        $query = "UPDATE volts SET value =" .  $value " where ID = '" . $rowID;
        if (!mysql_query($sql)) {
            die('Error: ' . mysql_error());
        }
        emailer($ID,$Name,$sensor);
        break;
    }
    else{
        $query = "UPDATE volts SET value =" .  $value " where ID = '" . $rowID;
        if (!mysql_query($sql)) {
            die('Error: ' . mysql_error());
        }
    }
}
```

If the converter finds an error with pi values, it then calls the function “emailer()” to handle sending out a warning with the given machine name to all individuals associated with that machine.

It queries through all the users associated to the machine at risk, checks if an email or phone number was provided, and sends a notification. Email is a simply call to get the email provided by the user input, while the phone number must go through if statements in order to email using the correct carrier syntax.

Phone’s Options Provided:

- Sprint: "@messaging.sprintpcs.com"
- Verizon: "@vtext.com"
- T-Mobile: "@tmomail.net"
- AT&T: "@txt.att.net"
- U.S. Cellular: "@email.uscc.net"

```
if($phone != NULL){
    if($carrier == "Sprint"){
        $to = $phone + "@messaging.sprintpcs.com";
        $subject = "Warning: Machine at Risk";
        $txt = "Please Check Machine: " . $machineName . " Sensor: " . $sensor;
        $headers = "From: instrmon@ameslab.gov";

        mail($to,$subject,$txt,$headers);
    }
    else if($carrier == "Verizon"){
        $to = $phone + "@vtext.com";
        $subject = "Warning: Machine at Risk";
        $txt = "Please Check Machine: " . $machineName . " Sensor: " . $sensor;
        $headers = "From: instrmon@ameslab.gov";

        mail($to,$subject,$txt,$headers);
    }
```

Email will be sent in the following format:

Testing:

Phase One - SMTP Capabilities & Crontab

The first phase of testing was simply testing the servers authentication of the SMTP system. After set up and rerouted to the service account, ran a simple 5 line script to test if email was sent.

Once successful, we attempted to set up crontab to run script every hour to check scheduler's capabilities. We let scheduler run for a couple of hours and checked to see if hours waited matched number of emails.

Phase Two - SQL Retrieval

The second phase testing was the retrieval of multiple user emails associated with a fixed machine. We assigned the same machine id to a group of users with emails and phone numbers in order to test its capabilities with different domains and sending emails to text. We were initially unsure if we would be able to send emails to non-Ames Lab domain email addresses, with testing we found we could send to any domain name. Also this testing taught us what restrictions we had sending email to text.

Phase Three - Conversions

The third stage of testing was making sure that the after catching an error during the conversions that the mailer would send correctly and not spam the users. To test the use of dummy values were used to create users to be emailed / texted.

During this time we also wanted to test how the user input of a carrier would work. Initially we were just given the address as a whole, however in order to decrease user error and simplicity we decided to simply take the users phone number, select a carrier, and allow the system to convert it to its correct @ format.

Phase Four - Final Testing

The final testing was taking in user input from the web client, and attempting a clean run of the system. We would be able to test this by first running the system, seeing if it ran through converted and displayed the values without sending any errors. The next run through we inserted values that would cause error, and see if the system would catch the error and alert the appropriate individuals.

Once this was checked we knew that the notification system was working properly, and further work on this system would be done through allowing the system to run and catching bugs and errors in the system through real time use.

Conclusion

We found that these tools were the best option for our system due to a couple of reasons:

1. To keep consistency with PHP language usage.
2. PHP provides easy functionality for SMTP. Other applications we found were either not supported by Linux, or didn't have the adaptability and functionality to be implemented into our work.
3. Conversions and SQL query retrieval proven successful in other scripts.
4. Easy to add additional conversions.
5. Works well with Crontab job scheduler which we used in the hardware aspect already.

Through these reasons, the testing process, and personal choice we found this process best suited both our client's needs and our own.

2.3.6 Graphing

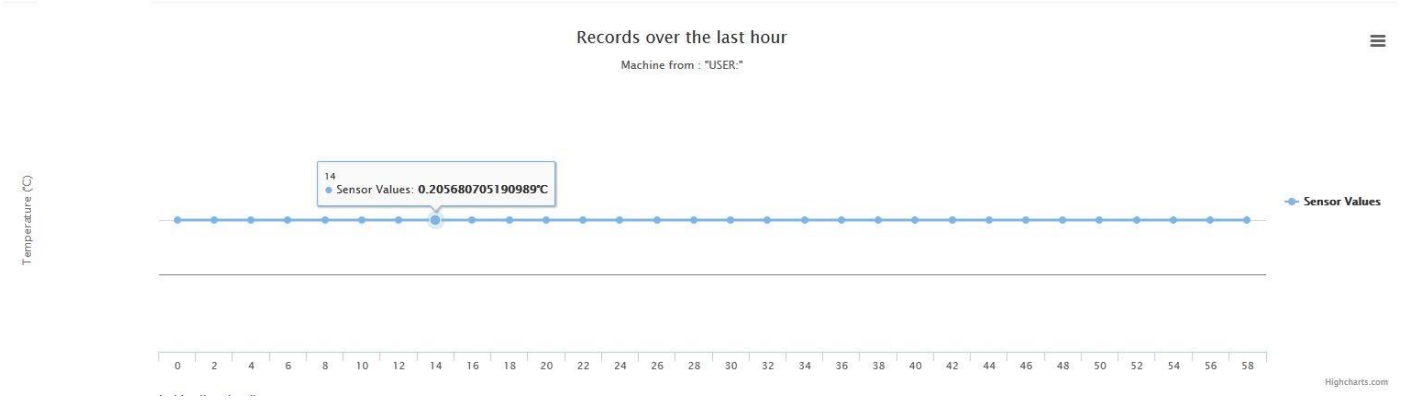
General Information

Our group in conjunction with our client decided that we wanted to have a system for remotely checking on the status on each of the sensors. This way the user can remotely check the status of the system without having to email or call security or an employee at Ames Lab. The idea was that the user should be able to see the values of the instrument that they are recording over the last hour so they could perhaps tell when something might have gone wrong or if the instrument is still functioning as normal. In order to do this we decided to write a script and implement a javascript library called Highcharts to graph our database values from the desired machine. We would use PHP to pull the converted values from the database much like the mailer script and then graph those values so the user can easily see the trendline over time.

Graph Script:

The graph script acts primarily as a webpage with a little bit of PHP thrown in so we can retrieve database values. Since the pi records instrument values every 2 minutes we decided it was best to graph the last hour of information recorded from the pi for the user. This way there would be 30 recorded points over the course of the graph. We decided to implement it this way so that the graph was not overcrowded with information, if there were more data points, it might be confusing or hard to look at. The nice thing about highcharts is that if you hover over any point on the graph it will display that data point value for the user. This is another reason we decided to only have 30 points, because if you have too many of them it might be tricky for the user to hover over the desired point.

From the main website page, a user can simply click on the website link to be brought to the graphing page. Then depending on which sensor is associated with the user the query is run for that particular sensorID. The SQL query returns the most recent 30 values for the particular machine. So this means that the first value on the graph is the most recent recorded value and then as you go along it goes back another 2 minutes per value. There is an example picture listed below.



An additional functionality that comes with highcharts is that the user can print or download several different versions of the graph by clicking on the right hand side of the graph drop down. They can download it as a PNG, JPEG, and PDF if they need to send a picture or results to someone else which is really nice additional functionality.

Testing:

Phase One - Bootstrap and Highcharts Capabilities

The first thing we needed to ensure that we could use both bootstrap, which is our web framework as well as highcharts which is our graphing javascript library on or Ames Lab sanctioned server. After some initial testing we discovered that bootstrap and highcharts would work fine with our server. After this we began uploading files as normal

Phase Two - SQL Retrieval

The second phase of testing was to ensure that the web page would be able to successfully retrieve data from the database and send it to our page so it could be graphed. We had a little bit of trouble with this but once we got the converter working we just used a lot of the same retrieval code to use for the graph page. We did this initially by just printing each of the returned values to the screen rather than placing them in the graph. Testing this also helped refine our queries since our original few queries were slightly wrong. They would either return way too many values which would make the graph display look wrong or would be out of order in some way.

Phase Three - Using retrieved values in graph

After making sure the returned values were correct by comparing them to our database values, the last step was to make sure that the values could be joined together to make a nice graph. The nice thing is since the graph does not have a hard coded max value so if we needed to update our queries during testing or in the future we did not have to mess with the graph too much. One issue we did run into was that the graph would reverse all of the returned values when it was placed in the chart. Thankfully, testing the page resolved that issue.

Conclusion

We found that these tools were the best option for our system due to a couple of reasons:

1. PHP had been used previously in our project so it made sense to keep it consistent
2. Bootstrap had already been previously used for our website.
3. SQL query retrieval already proven by other scripts
4. Highcharts used because one of our members already has experience with using it previously

3. Appendix I: User Guide

General User

General Users only have the capabilities of viewing the machines listed and viewing data.

View Machines

- The home page automatically allows the view of all the machines that are pushing data to the SQL database.
- A user will be able to view the machines and each individual sensor by the given machine name and the given sensor name for quick and easy identification.

View Graphs

- The user can view the graphing page by selecting the name of the sensor they wish to see. It then takes the user to another page that allows them to view the recent history of the raspberry pi readings for that sensor via a graph.

Search for Machines

- To search for a particular machine, in the top right corner on the home page you may type in text and it will parse through the machine and sensor names in order to find the ID that you are looking for.

Admin

Admin Users have all the capabilities of the General User on top of the capabilities to edit machines sensors and delete data from database.

Edit Sensors

The admin to edit the sensor must click on the Machine Name for the machine sensors they wish to edit. This gives them the capabilities of changing the following:

- Machine Name
- Sensor Names
- Sensor Types
- Upper and Lower Graphing Error Bounds.
- (If a gas sensor) M and B for custom gas pressure equation use.

Add to Notification List

The admin also in the bottom of the page has the capabilities of adding people who wish to be notified by the machine. In this the admin can give the following information:

- Person Name
- Email Address
- Phone Address
- Phone Carrier

In order for phone to be listed a carrier and a phone number must be provided in order for the auto formatting to take place.

Delete Machine

On the home page for the options button, a new option will become available for admin user, which allows them to delete the machine entirely. Doing so will erase all the machines data and sensor information from the SQL server and allow it to start recording a new experiments information.

4. Appendix II: Cost Comparisons

Parts to build one monitoring box

Cost	Quantity	Item Name
$29	1	6” by 6” Electrical Box
$30	1	Raspberry Pi 2
$15	1	Micro SD Card if not included with Pi
$26	1	ADC Pi Module
$2.56	1	Barrel Power Connector
$6	1	6” Din Rail
$10	2	Din Rail to Raspberry Pi clips
$.95	1	Micro USB plug
$24	1	Ethernet bulkhead Connector
$2	1	1ft Cat5e Ethernet Cable
$11	1	Cable Stress Relief
$4	1	Terminal Strip
$4	1	Grounding Bar

Parts to Buy on Demand

Cost	Quantity	Item Name
$180	100ft	12 Wire 24 Gauge Cable
$6.39	Additional	Gateway Power Supplies
$20	1	Thermocouple Amplifier
Varies		Gas Transducers

Parts Supplied by Ames Lab Electrical Shop

1. QTY: 8 Item: 1 or 2 watt 2.2k resistors
2. QTY: 10 Item: 1 or 2 watt 1k resistors 1% tolerance
3. QTY: 8 Item: 1 or 2 watt 100 resistors
4. QTY: 1 Item: 1 or 2 watt 960 resistors 1% tolerance
5. Flux
6. Solder
7. Heat Shrink
8. 22 gauge wire (4ft Green, 5ft Blue, 5ft Orange) **Make sure it is copper wire**

Total Cost to Produce a Monitoring Box is around $153 excluding shipping charges.

5. Appendix III: Design Considerations and Changes

During the design phase of this system other considerations and additions to the system were suggested but never incorporated into the design or implemented. Time, team knowledge, and Ames Lab restrictions are all valid reasons as to why the following considerations never went beyond brainstorming.

Getting Data to SQL

Our previous plan was simply to make SSH calls to the Raspberry Pi in order to retrieve data from the sensors. We thought that we could commit the data and information directly to text files and pull them to the server to be interpreted using SQL scripts. This would also allowed us to detect errors that were created by the network such as a connection loss. However after experimenting and seeing the amount of work and complexity it would cause we started looking for different solutions. We would have needed to make twice as many scripts in order to interpret the data and place them in the database for storage, the scripts would also need to be changed when a new machine was created due to the pathway set up to retrieve said files. Overall we found this idea process to be ineffective and decided to change our data retrieval with the system we provided in this document.

Pushing Sensor Data

We originally thought we would be able to select the sensors to be sent via the hardware level of the project. However after talking with our client and realizing that the sensors had no smart technology to identify its activity we realized we needed to push all the sensors data every time a push to the database was made. This caused confusion on how to check whether a sensor was active, and needing to put checks in the emailer and user interface which allowed users to turn off or delete sensors that were unimportant to the particular experiment.

Graphing Systems

Previously when expressing ideas to the Informational Systems department we were attempting to decide what to use for graphing view. One of the suggestions by a member of the department was Cacti Graphing Solutions. This application would have done a lot of the work for us and was already used on the Ames Laboratory system. However due to it taking a lot of the work and experience out of the project we decided against using it. Group mates felt more comfortable using PHP scripts and designing something they could understand and develop towards the projects use then attempting to learn a system that was pre developed and had more functionality then the system was required to have.

Physical Long Term Storage on Pi Device

One of the ideas of the project was to convert the values of the sensors on the hardware side, and then save that data for long term use on a built in SSD card. However one complication was that the sensors had no way of telling the hardware system what type of sensor it was and what it was measuring. The sensors only output a voltage to the Raspberry Pi, so we were unable to convert the values on the hardware and save them to a text file. We later planned to attempt to build something on the Web Client side, however due to the change, and the project priorities, we were unable to make this feature a possibility by the time of the presentation.

SQL Table Structure

During the design process and changes on how we planned to store data, create functionality, and different outlooks on interpretation the design of the table changed multiple times. Table changes included removing a “Notify” table, which would have acted as a table that held the machine ID and list of users to be notified. This would have then allowed us to create another table called “Person” which would of held their personal information. However for this project we felt it easier to simply remove the notify table, finding it redundant, and combine aspects of the notify table into the user table so that we could have less queries within the PHP scripts going through the database. Other changes were adding different columns and changing columns variable type to better interpret data on the web client level.

Email Inputs

Previously we were simply going to create a form that would ask for a messaging address and name of person. However one of the requirements we found was to attempt to make the user interface resistant too error, so we found it to be a bad idea for people to have the capabilities to input wrong formats into the important messenger values. What we did was increase the amount of SQL tables and the form inputs so that it would be more simplistic for non-technical users. We added functionality that would ask for a user phone number and their provider, so that within the scripting we could set the appropriate format that would send messages correctly. So a user would simply need to put their phone number in, select their provider, and a new string would be created making *phone_number@carrier_messaging_domain*.