School of Science and Engineering

CPS 342 – Embedded Linux

Final Project

Scalable and Robust Infrastructure to Create a Building's Heat and Humidity Profile

Date Completed: 5/12/17		Semester: Spring 2017
Group Members	Department	Major Contribution
Stephen Ayre	CS	Data collaboration and user interface
Emerson Benn	CE	Network and server configuration
Ray Hall	EE	Data acquisition and local storage

Course Professor: Chirakkal Easwaran

Project Description:

Our project involved using a temperature/humidity sensor, along with a Raspberry Pi to collect data, store it locally and upload it to a remote server which would present the data in an easy to read format. Furthermore, our project would allow for multiple Pi to gather data which could be integrated and presented on the server.

Project Goals:

The goals of our project included implementing the sensor and web server as described. They also included creating a robust infrastructure that could be expanded as needed. Lastly, we desired a system that could be easily deployable and once installed, would be user friendly for a non-technical client.

Project Implementation Details:

Our Pi used an Adafruit DHT22 digital temperature and humidity sensor to detect the temperature in degrees Celsius and the percent humidity. Using a python script, this information was extracted and the temperature was converted to Fahrenheit. The measurements were then stored, along with the current date and time in a local database on the Pi. For the purposes of implementing the process on multiple Pi, but with limited hardware stock, two additional Pi were implemented using Adafruit DS18B20 digital temperature sensors. Those two additional Pi used a random-number generator to simulate humidity readings for the purpose of demonstration. Using a crontab editor, the Pi were programed to retrieve measurements once per hour.

To allow for data sharing, the three Raspberry Pis were connected to a Local Area Network and assigned static IP addresses. Because static IP addresses were not obtainable on the school's network, a group member's home network was used. To allow all members access to the Raspberry Pis while away from the LAN, a VPN server was run on the network's ASUS RT-N66R router. Group members were then given credentials for this VPN so that they could work on the project from their respective homes.

For the implementation, a single Pi would act as the host, collecting data from all other Pi and presenting it on a web server. On the Asus router, ports 80 and 443 were forwarded to the Pi's local IP address. This would allow users to access the web interface off network by using the router's public IP. The server Pi used the Apache2 server software. Within Apache, the basic_auth module was used to create user credentials. The server was configured to require authentication in order to access any web pages. The SSL module was also enabled to ensure user's credentials were encrypted when logging into the server. A self-signed certificate was created for the SSL encryption. In a real world implementation, an actual Certificate Authority would likely be used. Lastly, the Apache's config files were set up to redirect regular http traffic at port 80 to port 443 for https.

In order to incorporate the web coding aspect of the project, we needed to install an Apache2 server on one of the raspberry pi's system. Along with Apache, we also installed PHP5 with an SQLite3 library in order to communicate with the local database on the pi and the databases that were transferred to the pi via an SCP command.

PHP acts as a backend web language which retrieves the necessary data that will be displayed. In order to display the data in a user friendly way, we created charts using Chart.js, a JQuery library that is useful for displaying data in a colorful graph. There are also different dropdowns where the user can choose what temperature and humidity data to see on different dates or times.

Project Components:

To complete the project, we divided the work into three main components. Those included data acquisition and local storage, data collaboration, and data presentation.

Data acquisition involved the Adafruit DHT22 and DS18B20 sensors. They were connected to a breadboard, which was subsequently connected to the port pins of our Raspberry Pi.

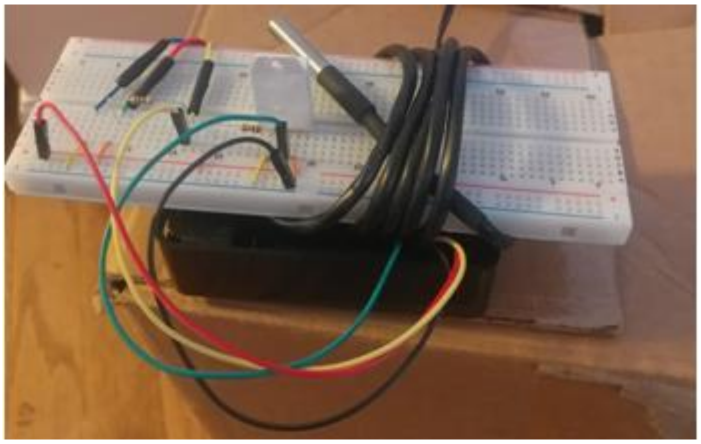
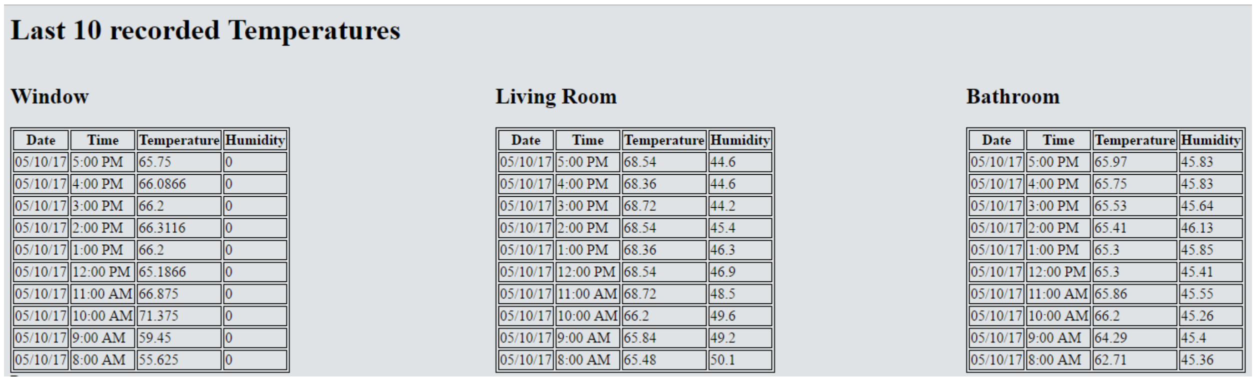


Figure 1: Breadboard showing humidity and temperature sensor connections

Data storage was accomplished using an SQLite3 database. The same python script which executed the data measurement included code which would also add the data to the database. The respective code for data acquisition and local storage is included and available in the code documentation on github (logData.py).

The data collaboration utilized one Pi as the host which would store all of the collective data. Using a secure copy, each Pi would transfer its database to the host. The secure copy occurred every minute.

The data presentation offered two visual representations to the user. The first was a chart which lists for each the Pi the 10 most recent temperature/humidity readings. Using PHP code, the host Pi would retrieve that data from each respective database and post it to the web server interface. The other data representation was in the form of a customizable chart. The user could specify the specific date and Pi(s) of interest and display the corresponding data in a graph. The code for data collaboration and for the web server interface are included and available in the code documentation on github.



Last 10 recorded Temperatures

Window

Date	Time	Temperature	Humidity
05/10/17	5:00 PM	65.75	0
05/10/17	4:00 PM	66.0866	0
05/10/17	3:00 PM	66.2	0
05/10/17	2:00 PM	66.3116	0
05/10/17	1:00 PM	66.2	0
05/10/17	12:00 PM	65.1866	0
05/10/17	11:00 AM	66.875	0
05/10/17	10:00 AM	71.375	0
05/10/17	9:00 AM	59.45	0
05/10/17	8:00 AM	55.625	0

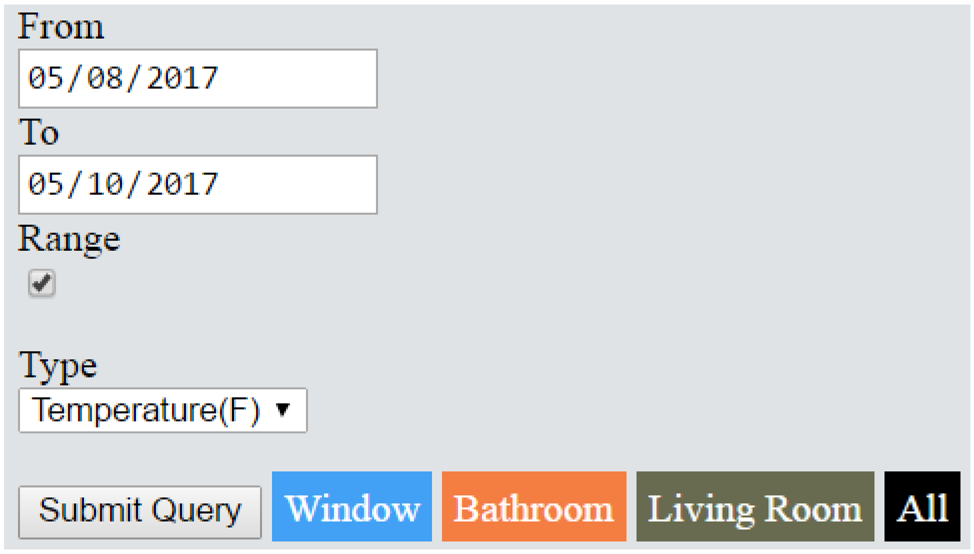
Living Room

Date	Time	Temperature	Humidity
05/10/17	5:00 PM	68.54	44.6
05/10/17	4:00 PM	68.36	44.6
05/10/17	3:00 PM	68.72	44.2
05/10/17	2:00 PM	68.54	45.4
05/10/17	1:00 PM	68.36	46.3
05/10/17	12:00 PM	68.54	46.9
05/10/17	11:00 AM	68.72	48.5
05/10/17	10:00 AM	66.2	49.6
05/10/17	9:00 AM	65.84	49.2
05/10/17	8:00 AM	65.48	50.1

Bathroom

Date	Time	Temperature	Humidity
05/10/17	5:00 PM	65.97	45.83
05/10/17	4:00 PM	65.75	45.83
05/10/17	3:00 PM	65.53	45.64
05/10/17	2:00 PM	65.41	46.13
05/10/17	1:00 PM	65.3	45.85
05/10/17	12:00 PM	65.3	45.41
05/10/17	11:00 AM	65.86	45.55
05/10/17	10:00 AM	66.2	45.26
05/10/17	9:00 AM	64.29	45.4
05/10/17	8:00 AM	62.71	45.36

Figure 2: Chart displaying 10 most recent measurements



From: 05/08/2017

To: 05/10/2017

Range: ✓

Type: Temperature(F)

Figure 3: User interface for selecting graph data

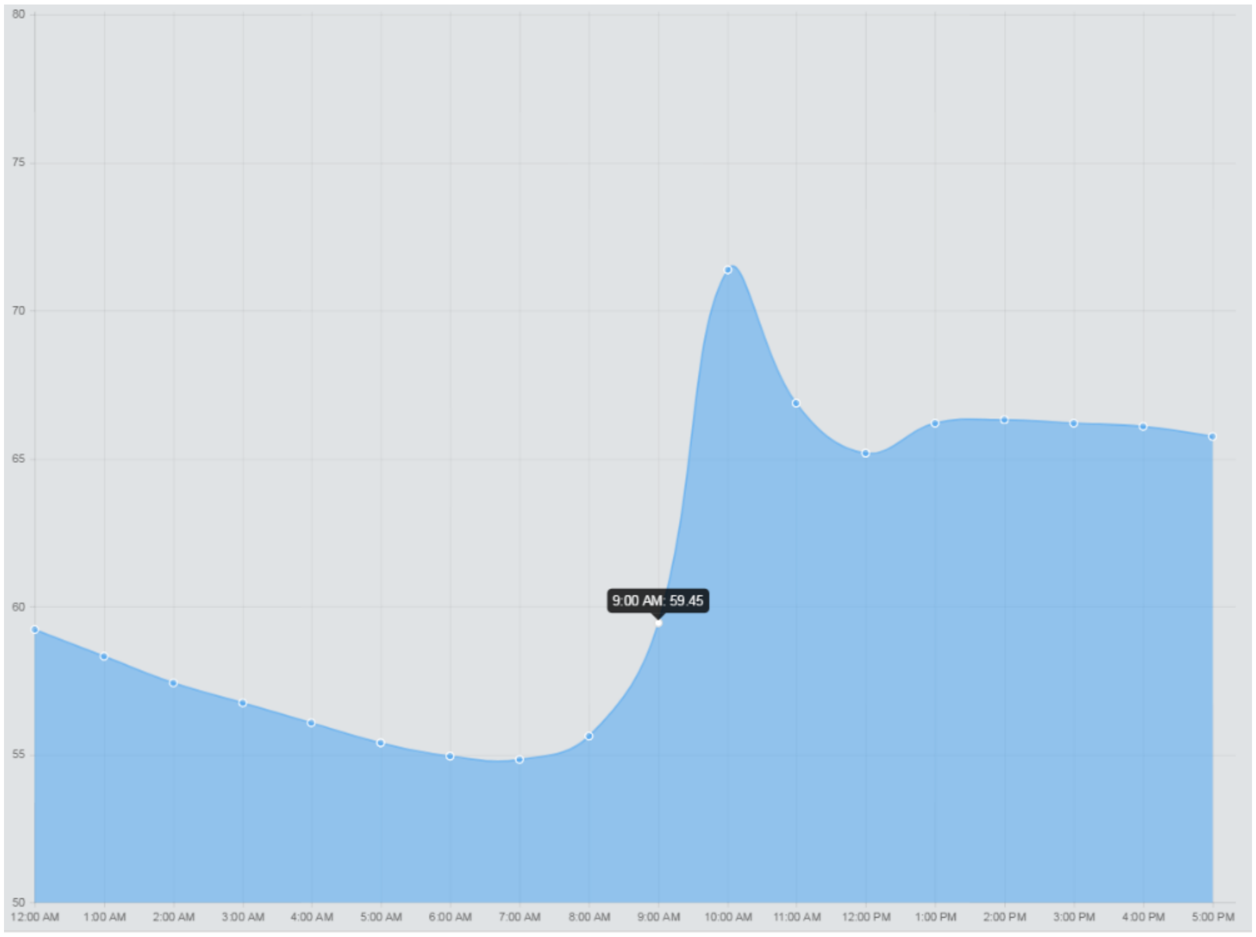


Figure 4: Customizable graphs illustrating temperature or humidity profile

Project Summary: (Good / Bad)

Ultimately, our project demonstrated a working implementation of a building's temperature and humidity profile. We were able to set up our devices to acquire the data automatically, collaborate it to a single server, and present it to a remote viewer. The easiest portion of this project was the actual data collection and local storage. Working with any individual Raspberry Pi is quite straight-forward. A simple python program was implemented to retrieve the data and store it into an SQLite3 database. This portion of the project was completed quickly.

Running an individual web server on a Pi which could upload its own data was also fairly easy. The most challenging part of our project was determining the best way to export the data from a Pi to a remote container. And then, subsequently determining how to retrieve and display that data from the remote container to a web user interface. There are many platforms available commercially to handle this type of data storage, however it was in our interests to develop a working method which was cost free. Initially, we considered using Google Drive as a 3rd party container for our data, which we would pull from to display on a website. We were able to successfully upload data to Google Drive, however we ran into complications when it came to merging the data from the Drive to our website. For this reason, we abandoned this method and decided to use a locally hosted server for easy collaboration between Pi. Once we committed to using the Apache server, the most challenging part of the project became using PHP code to create a user-friendly interface to display the data.

Group Member Contributions:

Our project group consisted of a CS major (Stephen Ayre), a CE major (Emerson Benn) and an EE major (Ray Hall). The diversity of our disciplines allowed us to easily divide our work for completion. The data acquisition and local storage was spearheaded by Ray Hall. The setup of our VPN for remotely accessing the Pi along with introducing SSL encryption with self-signed certificate for secure password entry was coordinated by Emerson Benn. And, most importantly, Stephen Ayre was able to tie everything together and wrote the code to display the data on the user interface.

Our research in this project can be expanded for further use. Ray Hall would be interested in investigating additional sensors and the possibility of using the Raspberry Pi for DSP or pairing it with a dedicated microprocessor and using the Pi for data transfer. Emerson Benn would be interested in networking multiple pis to the same server and giving different pis various capabilities for collecting different data. Stephen Ayre would be interested in incorporating more features on the dashboard in order to create more user interaction and different ways to view the data.

References/Resources:

https://www.tutorialspoint.com/python/index.htm

https://www.tutorialspoint.com/sqlite/index.htm

https://docs.python.org/2/library/sqlite3.html?highlight=sqlite#

https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-apache-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04