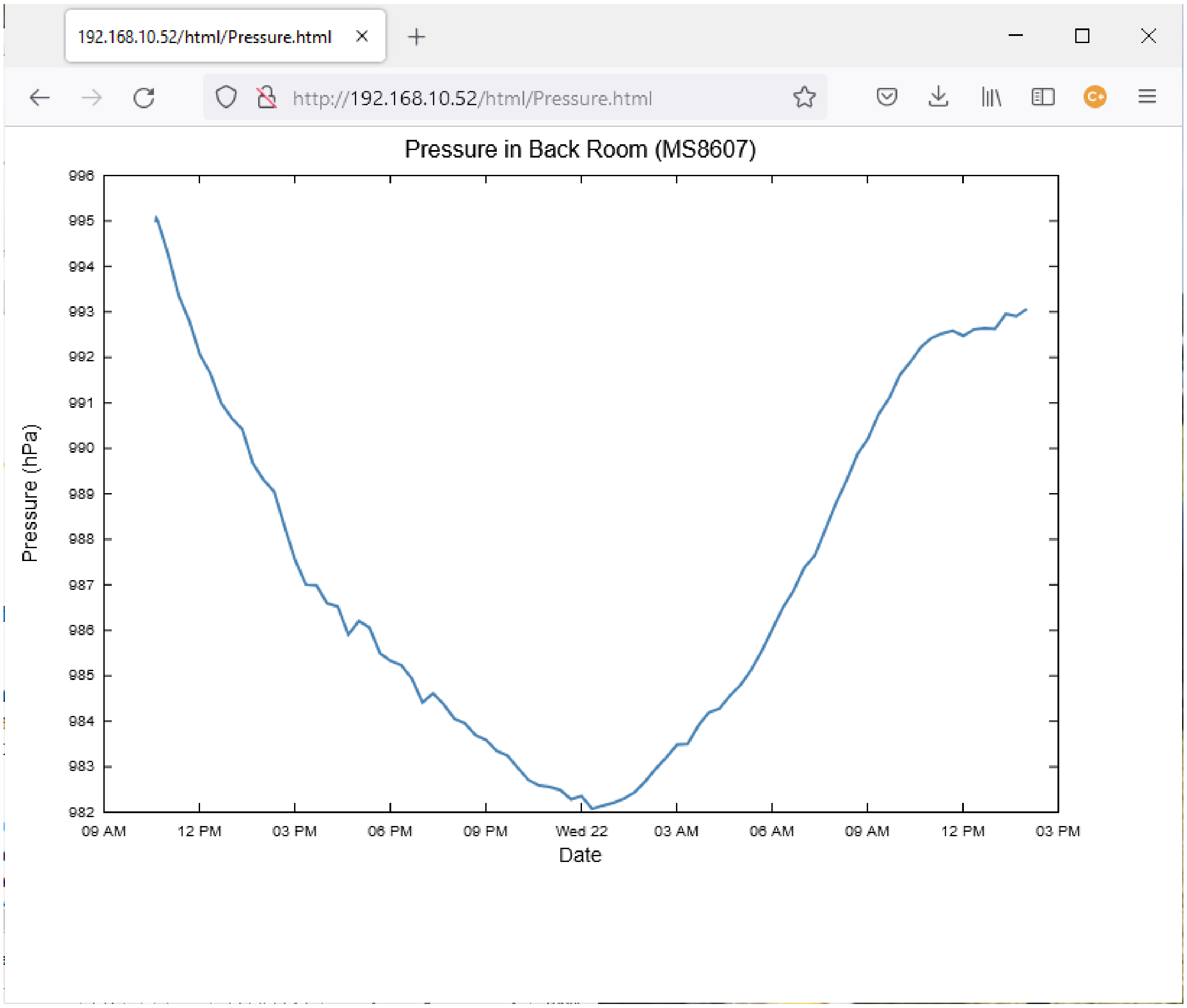
MS8607 Weather Station Graphs

Abstract

The purpose of this paper is to describe a set of codes for the Raspberry Pi that read and log an MS8607 PHT sensor, a Pressure, Humidity, Temperature sensor, then display the results in graphs on a web page. The objective is to use this code for a high altitude balloon mission. However, it may also be useful for a weather station. Source code is written in python3, html, and javascript. It can be viewed and downloaded from github.



Introduction

As can be seen from the other posts here on StratOhio, one of my interests is to develop and fly high altitude balloon payloads. I would like have a web server onboard the payload to display and download data sets. Some of the principle measurements are pressure, humidity, and temperature (PHT), just like with a ground weather station. Code for a balloon mission should run more frequently than for a ground station. A ground station might update data every hour, or 30, or 20 minutes. Here in Cleveland, where it said [like many other places] that if you don't like the weather, wait 5 minutes. A 5 minute interval might be suitable.

For a balloon payload, where the rise rate may be about 1000 ft/min, a 30 second or 20 second interval is more appropriate. A balloon mission may experience temperatures down to -60C and pressure under 10 millibar (mb) at altitudes above 100kft. While not specified for the entire range, the MS8607 comes close to the requirements for an inexpensive digital sensor. It reads down to 10 mb, and -40C. I do not know yet whether the sensor bottoms out under extreme conditions or simply become less accurate. That should be cleared up on the next mission. Out of range measurements may be better handled by analog sensors but it will be interesting to see how the digital sensor performs.

A ground station does not require the same range extremes. In Cleveland, the outdoor temperature rarely reaches -20C and the pressure of a very strong storm might be 950 mb, so the extreme ranges of the MS8607 are not needed. I have seen the BME280 recommended as a weather station PHT sensor. It has a bottom pressure limit of 300mb, corresponding to about 30 kft. However, it appears that the Adafruit code for the MS8607 and BME280 are similar.

I will use a Raspberry Pi Zero W[H] micro computer. It can easily run data acquisition code, as well as support an Apache2 web server. A couple projects already exist for running a weather station on a Raspberry Pi. As a balloon payload, I want to be able to run a web server offline, that is independent of the world wide web. So rather than use Google charts, I will use D3. The D3 javascript code can be downloaded to the hosting web server.

I have a graph format that I like, so some of this work will go into developing a template or a baseline chart that conforms to that format. I have put the code for this effort onto github as a StratoOhio project so it will be easy for others to access.

There are a couple weather station projects already posted on the web. I found the raspberry weather project to be very helpful. I shamelessly used the python code as a template for my read_sense.py code. However, I did convert it to python3, and converted the sensor from DHT22 to the MS8607. Also, instead of logging data to an mysql server, I write to a csv file directly to `/var/www/html/data/sense.csv`. This makes it easy to download, and to copy to backup files.

I won't claim my code is particularly elegant. I am not a java script or html programmer. I have kludged enough together to do what I want. It appears to work on my system. I make no guarantees. Your mileage may vary.

Note: StratoPi is already used by Sfera Labs. Any use of the term Strato-Pi, for this project to operate Raspberry Pi's in the Stratosphere, is not intended to violate other trademarks, or detract from StratoPi.

Approach

I used the current version of the pi operating system, Bullseye, the regular Desktop version. The desktop display can be turned off, but when running `ssh` with X pass-through, it has support for X-windows and GTK which makes it useful for a headless system, i.e. without a monitor or keyboard attached. I think the baseline operating system includes python3, but may need pip3 installed. After running `apt update` and `apt upgrade`, start installing software. Consult the Adafruit pages.

I will use python to write the data acquisition code.

```
sudo apt install pip3
sudo pip3 install Adafruit-Blinka
sudo pip3 install adafruit-circuitpython-ms8607
```

I have put my code on GitHub, https://github.com/stratohio/sensor_page The python code is src/read_sense.py

Data will be logged as a comma-separated-value file in `/var/www/html/data/sense.csv` so it can accessed from the web pages.

I looked into using Google Charts, However, since I want to fly this on a balloon payload not connected to the world wide web, I want charting code that I can download to my computer. I decided on d3.js.   I found this example from d3noob to be very helpful.

Setup

Make sure python3 and pip3 are installed. I use `geany` as an ide. Make sure its execute preferences point to python3 instead of python2.

Install apache2, and get a web page running. A search of “raspberry pi web server apache2” comes up with a number of helpful guides. I used the Random Nerd Tutorial for setting up a LAMP server (Linux Apache Mysql PHP) as a guide to set up my server.   Make sure the user who runs read_sense.py as a crontab job is a member of the `www-data` group and has write access to `/var/www/html` and its subdirectories. And make sure that `www-data` has read [&excecute?] access. I add the subdirectories `data` (csv log files), `src`(javascript), `css`(css formating),`images`, `html`(display html) and `doc`(useful information). I do not include an `index.html` in these subirectories so I can get a directory listing to access the contained files.

As an html editor, I use bluefish.

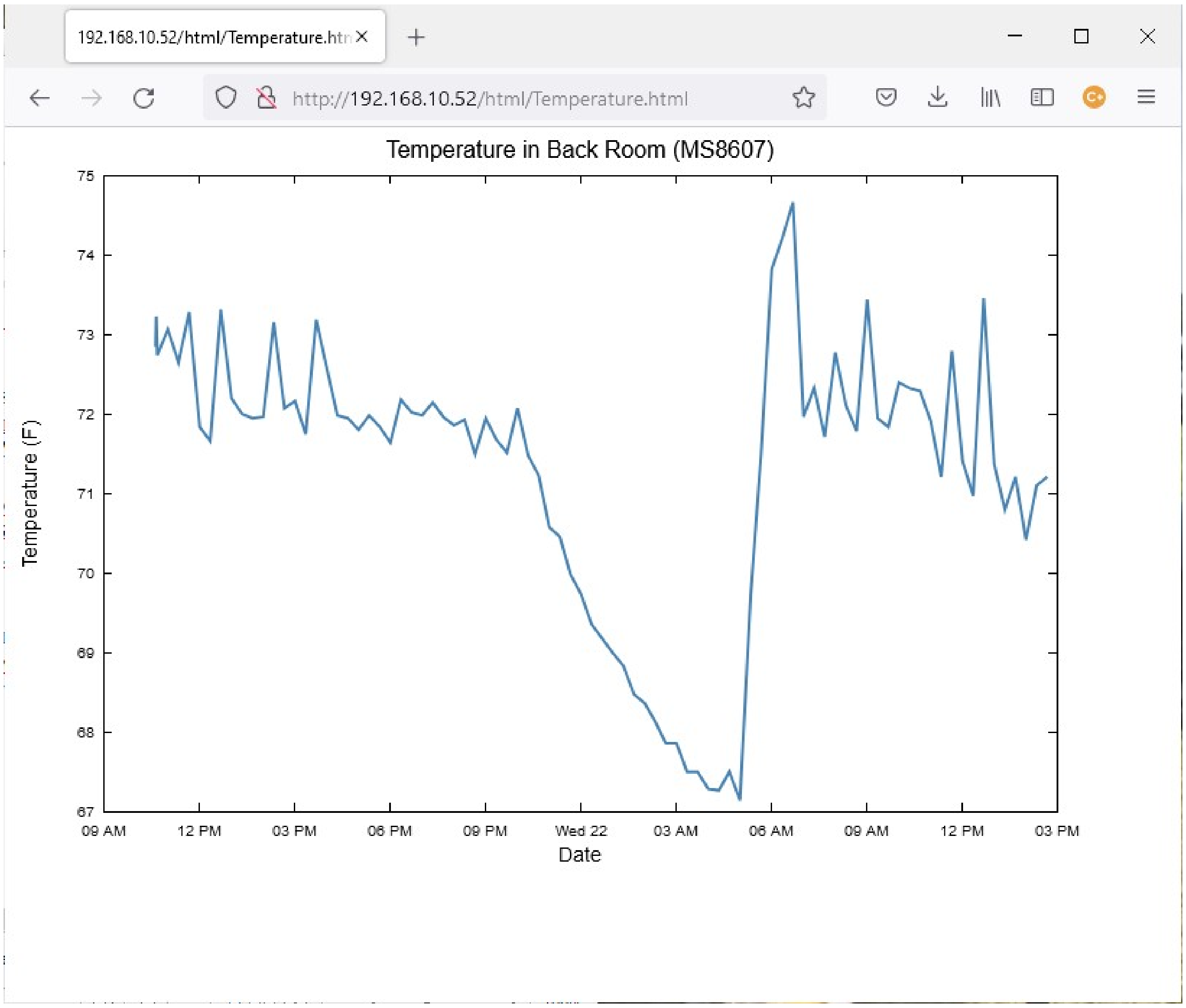
```
sudo apt install bluefish
```

My initial efforts to download d3.js did not work, but I did find (kludge?) a way. I opened d3.v7.js in a browser, select all and copy. Then I pasted into a file `/var/www/html/src/d3.v7.js`. There is a `d3.v7.min.js` that is a smaller file at the expense of readability. It appears to remove spaces and linefeeds.

Discussion

I have a graph design that I like and have used over the years. It uses X and Y labels, as well as a Title. I set these as constants early in the code to make them easy to change or fix. I set the axes and tic marks go all the way around the plot. AxisRight is translated over by a `width`. AxisTop is placed at (0,0) to provide the extra axes. While not as good as a full grid, plot values can be estimated. Another personal preference is that the tic go inward. This accomplished by sitting the ticksize negative. Currently, I have no numbers on the top and right axis. To accomplish this, tickFormat is set to “”.

It appears that reading the csv data set is handled by the web server. Simply running the code locally, in a browser does not work. It needs a server such as Apache2.



I tried to set the graph up as a template, to make it easy to adappt in multiple applications. It is not completely portable. The data file is read in the `d3.csv()` line. The file name could be set as a constant near the beginning of the code. Headings on the first line identify names of the comma-seperated-values. These are read into arrays in the `data.forEach()` block of code. I read most of the data into arrays, `d.date`, `d.temp`, `d.pres`, etc. The temperature data is converted from C to F as it is read in. To plot a new line, the scale has to be adjusted (`y.domain()`) for the new y data, and the line description, `var valueline = d3.line()` block, needs to point to the appropriate y data.

I have set the graph html routines to just draw a plot, `Temperature.html`, `Pressure.html`, and `Humidity.html`. The `MS8607_sensors.html` page incorporates all three graphs using the <iframe> directive.

Conclusions

The python3 and html code described here can read sensors attached to a Raspberry Pi Zero. An installed web server can display a plot of the logged data.

Additional calculations that could be included, probably in the python code, are: dew point, a measure of absolute humidity, altitude, and sea level pressure for a given altitude.

Additions to the graphing capability could include plotting a second line, such at temperature and dew point, and adding the capability of a second y-scale so that, for example, temperature and pressure could be plotted together.

References

https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout bme280 description and use.

https://learn.adafruit.com/adafruit-te-ms8607-pht-sensor/python-circuitpython ms8607 description and use.

https://d3js.org/ D3.js web site

https://www.raspberryweather.com/ graphing temperature web page with a raspberry pi

https://projects.raspberrypi.org/en/projects/build-your-own-weather-station Weather station construction.

https://www.adafruit.com/product/2652 BME280 Stemma QT

https://www.adafruit.com/product/4716 MS8607

https://www.tutorialsteacher.com/d3js/what-is-d3js D3,js tutorial

https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/ apache2 installation.

https://www.adafruit.com/product/4688 SparkFun Qwiic / STEMMA QT HAT for Raspberry Pi

https://developers.google.com/chart/interactive/docs Google Charts

https://gist.github.com/d3noob/15e4f2a49d0d25468e76ab6717cd95e7 Line chart code.