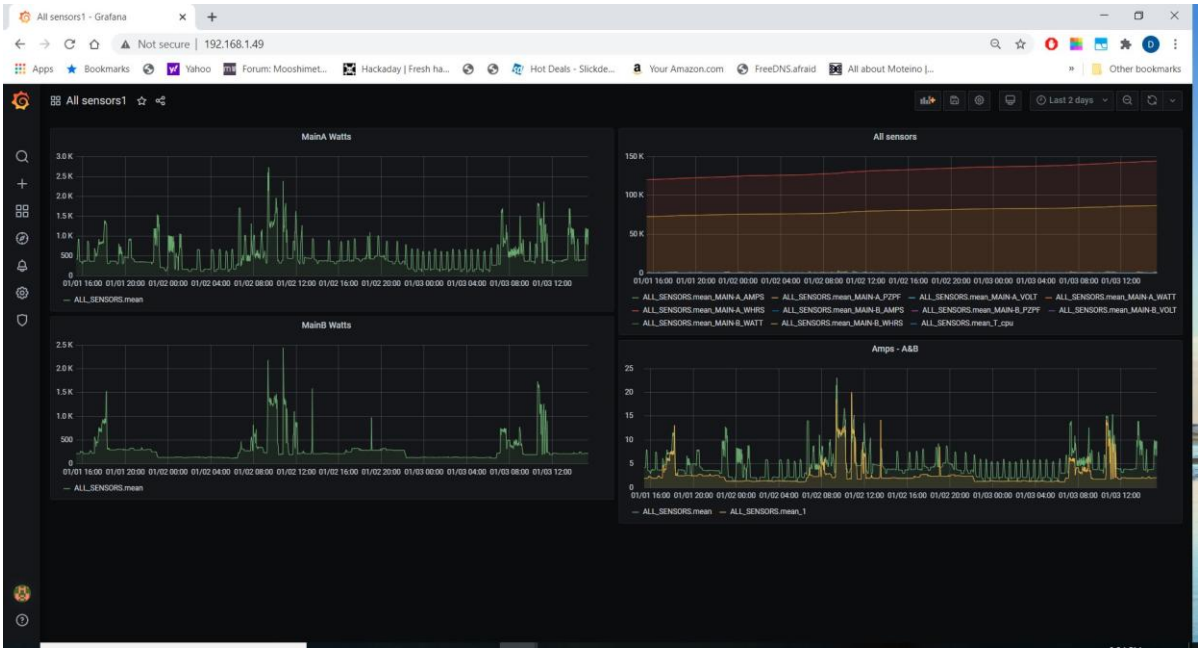
House Electrical Power Usage Monitor, Including Generator



This documents my approach to building a system to monitor my house electrical consumption. My electricity is supplied from either my electrical utility company or my small gasoline/propane portable generator. My environment is a USA standard 200A residential house. See Photo1.

Design Requirements:

- Monitor and store measurements for either my Utility Co. or my generator source.
- Measurements include V, A, W, PF, kWh, collected every 5 seconds
- Store measurements locally, no outside servers required
- View measurements with any browser, set email alerts
- Simple hardware to install, simple software to implement, simple to maintain
- Low cost sensors, low cost support hardware, and free, open source software

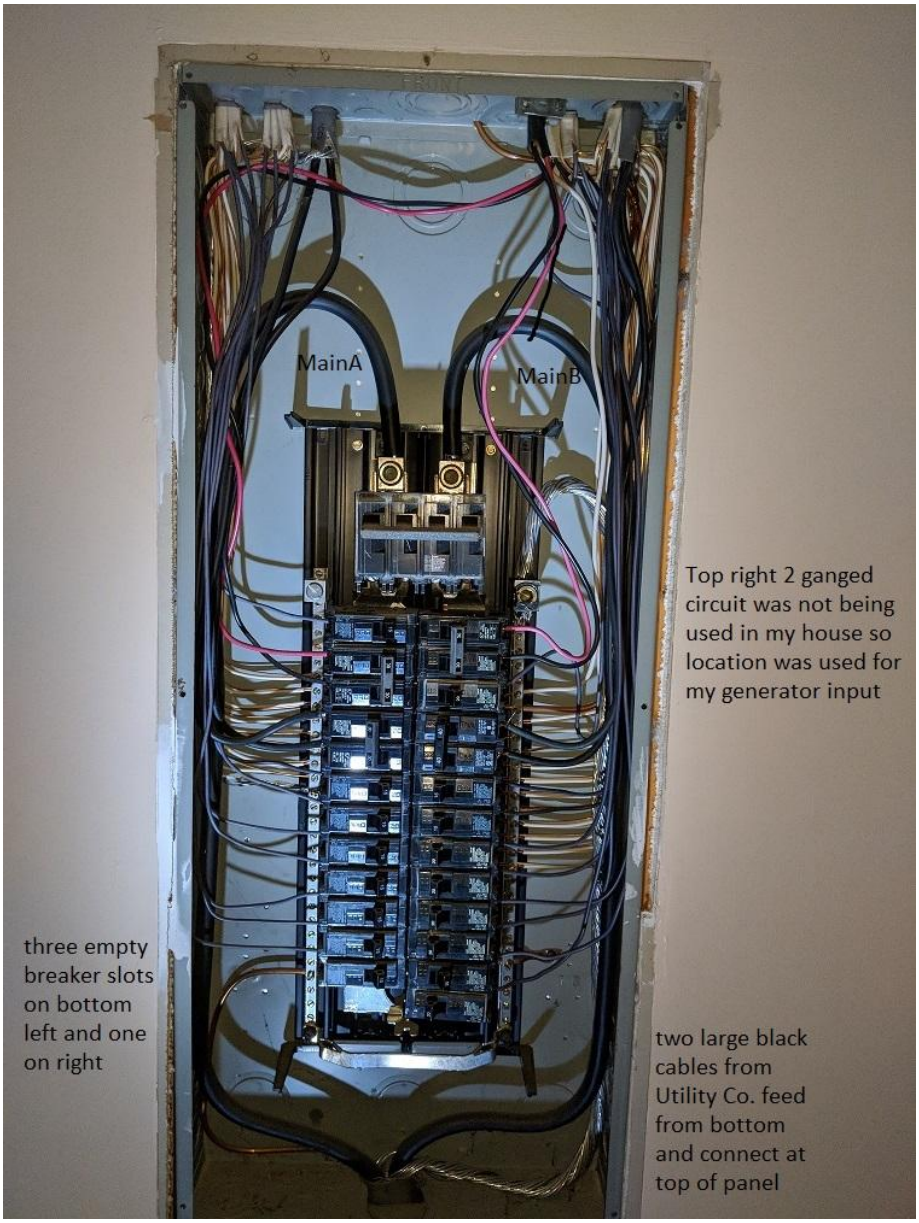


Photo1

First, let me say this is a rather large project with lots of pieces and parts, both hardware and software. A great site, https://openenergymonitor.org, certainly has a lot of good information, but the amount of information and options was a little overwhelming. Using internet searches, I could not find any single step by step tutorial of both the hardware and software pieces which included all of my requirements and which was simple enough for me to understand and tailor. However, I found enough bits and pieces from the previous work of many “giants” on the internet to cobble my system together. I am very grateful for each person before me who took the time and effort to post their work for others to use. It is with that same appreciative spirit that I share this project. I claim little original content for myself other than distilling and organizing much material into detailed step-by-step instructions for this project. Best wishes to all who attempt this.

Top Level Concept:

Use very inexpensive, high accuracy non-contact current sensor/modules from China that output processed data via serial USB. Put this hardware on the inside bottom of my house main breaker panel. Run the USB cables below the breaker panel to a drywall cutout where I put a Raspberry Pi, which hosts my python sensor software, Influx database for storage, and Grafana to view graphs from any browser via my network wifi.

Safety Concerns:

Clip-on split current sensors are used which wrap around the insulated wires, so no direct connection to bare power wires is needed. However, always remember the main 240VAC wires from the power utility company are always hot, even when the main breakers are off. Although not the subject of these instructions, my generator input has a simple, but safe, interlock installed in the breaker panel box which prevents back-feeding the Utility Power Grid when my generator is being used. See Photo2. The PZEM modules are very low power so in-line fuses (100mA – 200mA) are used for added safety. The serial USB outputs are optically-isolated.

Software development tools:

On my Win10 computer I use https://www.putty.org/ to SSH into the RPi. Putty hostname will be your Pi assigned address, something like 192.168.1.123. Port will be 22 for SSH. I also use https://filezilla-project.org/download.php?type=client at port 22 to copy files from Win10 computer. I use https://notepad-plus-plus.org/ for code changes on my Win10 computer. Note for more complex python development I would probably use PyCharm, but Notepad++ is simple and it works fine for small scripts.

Hardware Parts required:

Raspberry Pi 3B+ with case and 5VDC adapter, Amazon or ebay ($35 + $10 + $10 = $55)

(3) PZEM-016 AC Single Phase Current Voltage Power Meter Electric Energy Ammeter RS485 Modbus Frequency PF Monitor 100A With Split CT&USB from AliExpress (3x $15 = $45). Suggest you buy a spare since future supply may not be guaranteed.

(3) USB extension cables, Amazon or ebay ($15)

(2) circuit breakers, 15A, for your breaker panel box, local big box stores (2 x $5 = $10)

(2) In-line fuse 120VAC holder and fuse for the 2 breakers (just to be super safe since current draw is only 0.1 to 0.2A). Amazon or ebay is a good source. I got an entire kit for future projects: BOJACK 5x20 mm Fuse Holder Inline Screw Type with 16 AWG Red Wire + 15 Values 150 Pcs 5x20mm Fast Blow Glass Fuses 250 V 0.1 0.2 0.25 0.5 1 1.5 2 3 4 5 8 10 12 15 20 A Assortment Kit ($15).

Various pieces of wire, wire nuts, tape, and other small pieces, local big box stores ($xx)

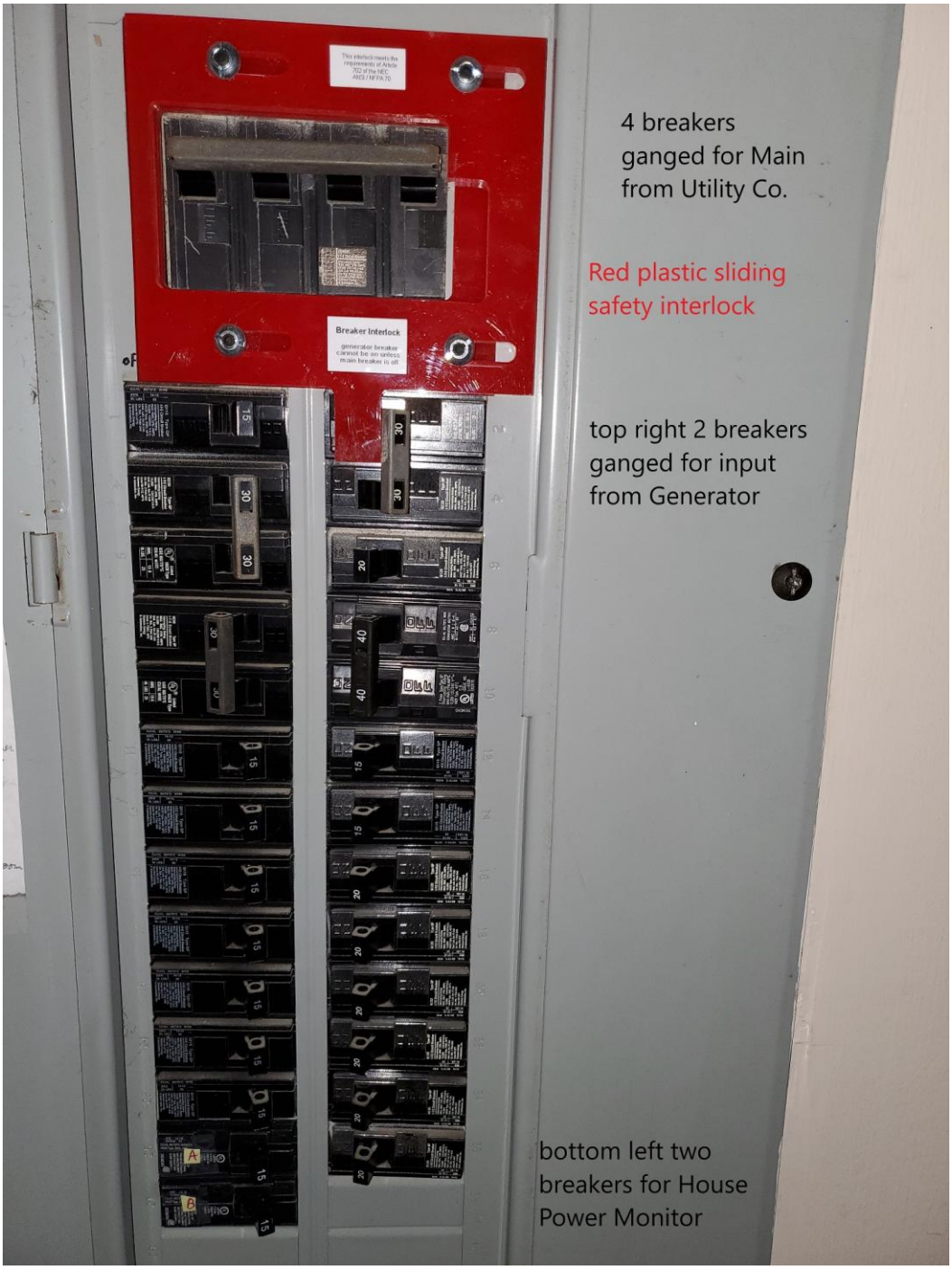


Photo2

Detail Steps:

Steps 0 thru 4 were inspired by the following great work https://simonhearne.com/2020/pi-influx-grafana/ I copied much of the material here, without pictures, for completeness, and modified for my system.

Step 0: Initial Setup

Follow these steps first if you have a brand new RPi:

1. Download the latest *lite* Raspbian (Raspberry OS) image from https://www.raspberrypi.org/software/operating-systems/
2. Get a reasonable micro SD card – I used 32GB
3. Burn the image to your SD card, I used https://www.balena.io/etcher/
4. Once Balena is finished, re-mount your card by physically removing and re-inserting it. Do not reformat. Then copy the empty file called ssh and wifi config file wpa_supplicant.conf into the root of the SD card - this enables SSH access which we'll need later.
5. Insert the SD card into your Pi, connect to your router via ethernet and power on.
6. Determine the auto-assigned IP address of the Pi by logging in to your router interface (see a guide on finding your router IP address here) and navigating to LAN / DHCP settings - the pi should be recognized as raspberrypi. Now is a good time to assign a static IP for your Pi to make life easier in the future. You'll need to power cycle your Pi if you change from the auto assigned IP address.
7. SSH into the Pi at the assigned address, something like 192.168.1.123. Port will always be 22 for SSH. You should probably update the password for the pi user now by running passwd at the $ prompt once connected to the Pi.
8. Expand sd card by running sudo raspi-config at the $ prompt, then selecting Advanced Options then Expand Filesystem.
9. While in raspi-config, you should change the serial config by: select "5 Interfacing Options", Enter and arrow to select "P6 Serial", Enter. Then select "No" to login shell to be accessible over serial, then "Yes" to want to make use of the Serial Port Hardware. Once the Raspberry Pi has made the changes, you should see the following text appear on your screen.

   "The serial login shell is disabled

   The serial interface is enabled".

   Before these changes fully take effect, we must first restart the Raspberry Pi, `sudo reboot`

10. Install python3

    $ sudo apt-get install python3-pip

    Note after this if you see instructions which use pip, you will need to substitute pip3 instead. Or you can assign an alias.

Step 1: Getting up to date

First off we'll make sure everything is up to date. This could take a while, especially on a new Pi:

```
$ sudo apt update
$ sudo apt upgrade -y
```

Step 2: Install Influxdb

First we add Influx repositories to apt:

```
$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
$ source /etc/os-release
$ echo "deb https://repos.influxdata.com/debian $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
```

Update apt with the new repos, & install.

```
$ sudo apt update && sudo apt install -y influxdb
```

$ influxd version shows version 1.8.3 for my system (v2.x requires 64bit OS)

Then start the influxdb service and set it to run at boot:

```
$ sudo systemctl unmask influxdb.service
$ sudo systemctl start influxdb
$ sudo systemctl enable influxdb.service
```

You can confirm this works by issuing a sudo reboot

We should now be able to run the influx client with influx at the $ prompt and create a user for later (for my system I use a single admin user grafana1 for simplicity):

```
$ influx
> create database housePower
> use housePower

> create user grafana1 with password 'your_password' with all privileges #Note, I made my Pi,
```

InfluxDB, and Grafana passwords all the same to make it easier for me.

> grant all privileges on housePower to grafana1

> show users

> user admin

If done successfully, here will be the output:

---- -----

Grafana1 true

That's it! You can now exit the Influx client by typing exit.

Step 3: Install Grafana

Again we need to add the Grafana packages to apt:

$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee

/etc/apt/sources.list.d/grafana.list

We can now update and install the binaries:

$ sudo apt update && sudo apt install -y grafana

Then simply enable the service and set to run at boot:

$ sudo systemctl unmask grafana-server.service

$ sudo systemctl start grafana-server

$ sudo systemctl enable grafana-server.service

You can confirm this works by issuing a sudo reboot.

we can check that grafana is up by loading it in a browser: http://<Pi_ipaddress>:3000. Notice the default port for grafana is 3000. If so, you can log in with the username = admin and default password = admin and set a new admin password **your_password**

Step 4: Add Influx as a Grafana data source

Now we have both Influx and Grafana running, we can stitch them together. Log in to your Grafana instance with your browser (http://<Pi_ipaddress>**:3000)** and head to “Data Sources”. Select “Add new Data Source” and find InfluxDB under “Timeseries Databases”.

As we are running both services on the same Pi, set the URL to localhost (localhost is the same as 127.0.0.1) and use the default influx port of 8086

Name: influx Default on

URL: http://127.0.0.1:8086

We then need to add the database, user and password that we set earlier:

For my system

Database is housePower

User is grafana1 password your_password

HTTP Method: GET

That’s all we need! Now go ahead and hit “Save & Test” to connect everything together:

PZEM016 modules for house power

Now it is time to set up the system to measure house power. I use three pzem modules since for my application I want to measure my power company supplied power, I call mainA and mainB, as well as my single phase backup generator, which I call gen. So on the RPi, install the following:

$ sudo pip3 install minimalmodbus # used to read pzem modules

$ sudo apt install python3-gpiozero # used to get RPi cpu temperature

Now a little hardware work to get the three pzems wired for testing. I set my testing up near my main computer to make it more convenient for debugging. See Photo3. Each pzem ships with a wiring picture and a diagram is on the bottom of the module. Pretty straight forward with all connections as screw terminals. Need to do lots of wire stripping. I used some leftover clear speaker wire for the RS485 to USB converter. I used some different 2-conductor wire spliced onto the [too short] current sensor wires. Black and wires stripped out of Romex 14/2 for the 120VAC connections to the pzems and a official Rasberry Pi 2.5A usb 5VDC power source. This is the same wire inside my house breaker panel for 15A circuits.

I used wire nuts to connect all three pzems to two temporary power cords with a standard 120VAC plugs. I put electrical tape over the 120VAC connections of the pzems. The other connections are low voltage, and RS485 is optically isolated. I used usb extenders to get extra length; either usb2 or usb3 variety will work. I used another cheap 120VAC 2-wire extension cord which I separated the two conductors (keep each conductors insulated) so I could clip on the three current sensors around only the hot wire. The hot wire has the smaller terminal/socket; the wider terminal/socket is neutral.

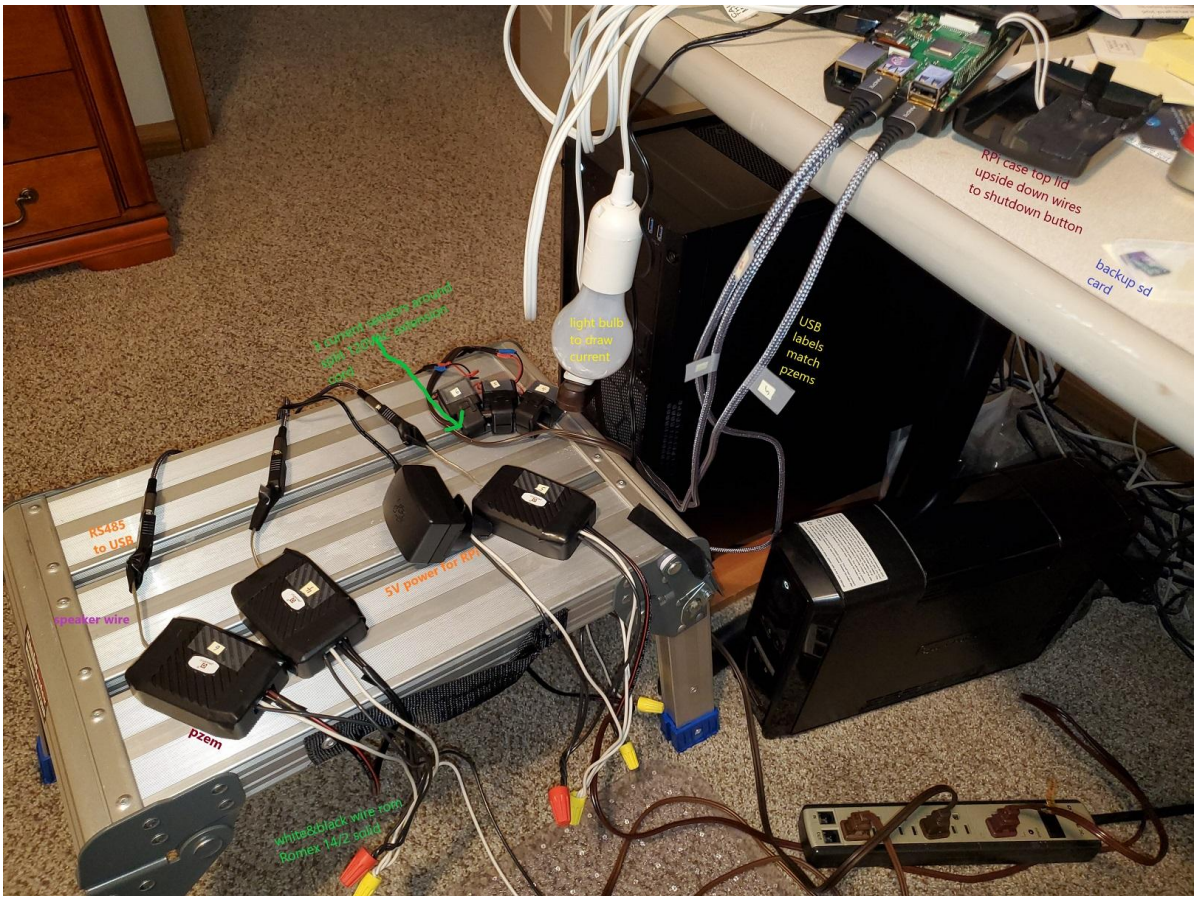


Photo3

Suggest you start with plugging in just one pzem into the RPi, any usb port. Use the pzemTest.py to verify the RPi can talk to the pzem. Then comment out the code to change the slaveAddress to something other than the factory default 1. Do the same for the other two pzems. I choose SA 4, 5, 6 for my houseMonitor usage. I put those numbers on the pzems, current sensors, and usb end that plugsinto the RPi.

To satisfy my curiosity, I took the following measurements (with usb meter and multimeter) of my setup with everything on and pzem.py script running.

RPi3B+ draws about 0.5A from 5VDC and around 70mA from 120VAC

Each pzen draws less than 5 mA from the 120VAC

Since the current draw is so low and to be super safe in the final installation inside my breaker panel, I used a in-line fuse holder with 200mA fuse from the breaker that supplies the RPi adapter and one pzen. I used a 100mA fuse from the breaker which supplies the two pzens connected together.

Integration – get it all to work together

This is a big leap. Once all three pzems have different slaveAddresses and can communicate one at a time plugged into the RPi, it is time to plug all three in. The run the pzem.py script.

If everything is exactly the same as mine, it may just work. You can set up a panel in Grafana to see the data. You can skip to the next section.

However, your situation may be slightly different, so here are some tips and comments. Determine your RPi usb names.

Use these commands at the $ prompt on the RPi:

```
ls /dev/ttyUSB*        # To list out the USB-serial ports that are active.
```

Or for more information about the USB buses and connected devices, use

```
dmesg|grep ttyUSB*
```

The serial ports "/dev/ttyUSB0", "/dev/ttyUSB1", and "/dev/ttyUSB2" appear here, depending on how many devices you have plugged into RPi. The "/dev/ttyUSB0" is the adapter that was physically identified first by the operating system. If you situation is different, you can modify the pzem.py scipt.

Just in case you are tempted to make too many changes to the script, I give you some background info. If you have multiple devices connected to the USB ports of your Raspberry PI, the operating system automatically assigns USB names which could unexpectedly change after a reboot. Your pzem that you thought was assigned to ttyUSB1 could be suddenly assigned the ttyUSB0 port name for example.

I tried several different methods I found on the internet, all unsuccessfully. Finally, I just created a function which loops through the three serial ports and matches them to

each of the three pzem slaveAddresses. I only have call this once at the start of the program.

The bottom line key to all of this is I know that the pzem with the slaveAddress of 4 has its current sensor physically wrapped around my utility power MainA cable. That usb cable can be plugged into any RPi port and my pzem.py script will map it correctly.

Auto-Starting the pzem.py script at boot

The script is pzem.service

sudo nano /lib/systemd/system/pzem.service

Once saved, you need to change the permissions of the file with:

sudo chmod 644 /lib/systemd/system/pzem.service

Which will make it readable by all, and only writable by the owner.

You want the service is started each time the Raspberry Pi boots. Do this with:

sudo systemctl daemon-reload   # updates the list of all services

sudo systemctl enable pzem.service   #enable for next boot

After which reboot your Raspberry Pi.

Here are some additional commands which can be used if you need them:

sudo systemctl status pzem.service

sudo systemctl stop pzem.service

sudo systemctl start pzem.service   # start for this boot

python print statements and errors output to   /var/log/syslog

as well as to   /var/log/daemon.log   <------- much easier to read in notepad++

shutdown button - optional

Optional, but If you are interested: Make a shutdown button without the need for a running script by adding the one line below to the file /boot/config.txt. This assumes a physical button is connected between GND and GPIO 17 (physical pin 11) or use any other unused pin.

Use command sudo nano /boot/config.txt

Add the following line, then exit.

dtoverlay=gpio-shutdown,gpio_pin=17,active_low=1,gpio_pull=up

When momentarily pressed, the system is stopped correctly as if the shutdown now command had been given. Remove power and reapply power to restart the RPi.

Pi built-in watchdog timer

This is an optional step. But you may be interested, so I pulled from several internet searches to provide the following. The Raspberry Pi has a hardware watchdog built in that will power cycle it if the chip is not refreshed by the RPi OS within a certain interval.

To start using the watchdog, you can still use the old watchdog daemon, but since 2012 systemd has had built in support for watchdogs that doesn't require installing anything else and offers better compatibility with the shutdown/reboot process.

sudo nano /etc/systemd/system.conf # to edit, uncomment and set the following lines:

RuntimeWatchdogSec=10
RebootWatchdogSec=10min ---- OR ---- ShutdownWatchdogSec=10min
DefaultTimeoutStopSec=20

What the lines above say is:

- refresh the hardware watchdog every 10 seconds. if for some reason the refresh fails (I believe after 3 intervals; i.e. 30s) power cycle the system
- on shutdown, if the system (not your scripts) takes more than 10 minutes to reboot, power cycle the system

- on shutdown if my scripts take more than 20 sec, then power cycle the system

I read where ShutdownWatchdogSec is no longer used, it was renamed to RebootWatchdogSec in July 2019. I verified my older version of the OS file still used ShutdownWatchdogSec, so I did not change it.

If you are using a Pi 2 or older, set dtparam=watchdog=on in /boot/config.txt and reboot. This is enabled by default from the Raspberry Pi 3 onwards, so you can skip this step with recent models.

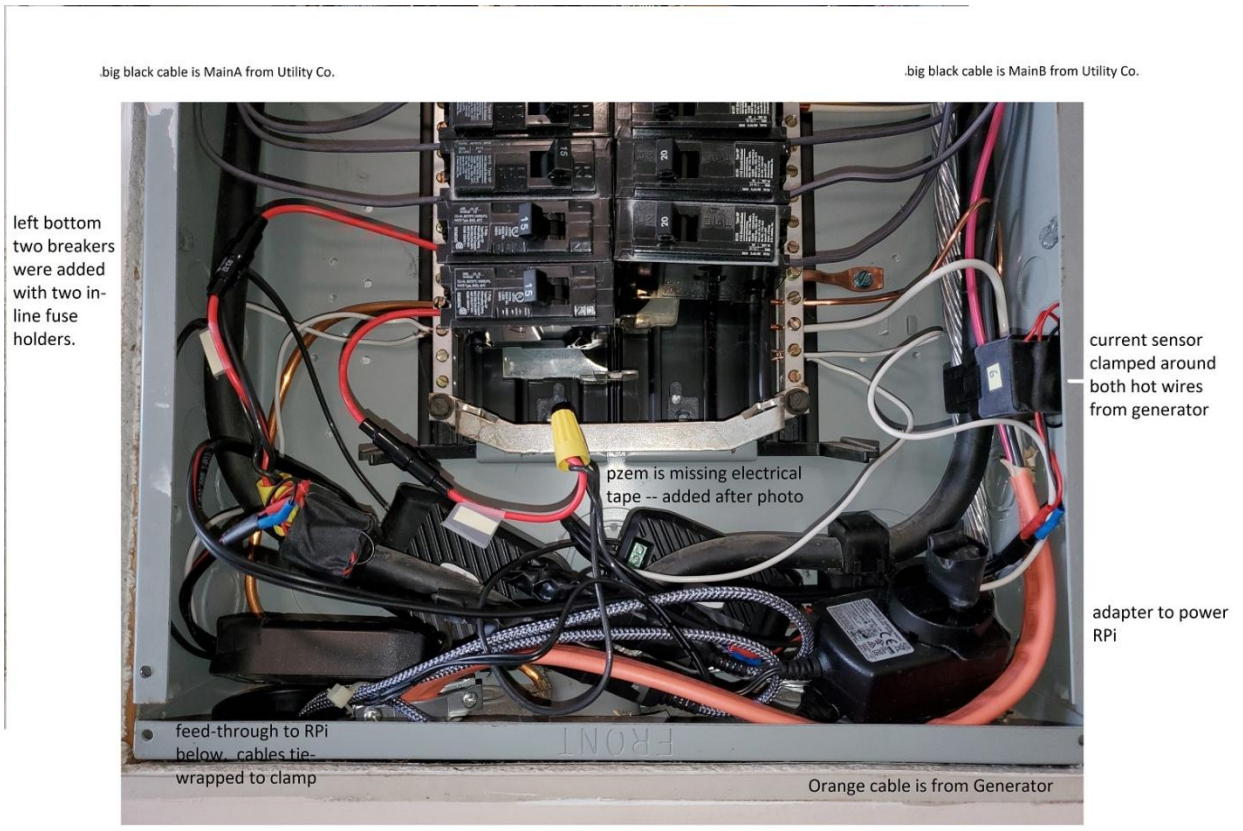
You don't need to run systemctl enable watchdog (that's for the old method mentioned above) but you will need to run systemctl daemon-reload (or reboot the RPi) after making these changes before they will take effect.
The best way to verify that the watchdog works is to overload the system with garbage process forks. If you run this the RPi will become unresponsive and the watchdog should kick in. Your system should be up and running again after about a minute:
:(){ :|:& };:

Paste that into a shell (i.e. after $ prompt) and your system will be taken down. I verified it bombed, PuTTY and Grafana became unresponsive, pi rebooted and InfluxDB and Grafana were operational once again.

Final installation

WARNING! This is some serious stuff. If you are not comfortable, get help on this part. The main 240VAC wires from the power utility company are always hot, even when the main breakers are off.

Turn MainBreakers off . Install the pzem related hardware inside house breaker panel. See Photo4:



Only one wire can be connected to a breaker screw, so use wire nuts to combine pzems for MainA and Gen to breaker on side A of the panel . MainB and USB power adapter are connected to the breaker on side B of the panel. This is important so each side A and B can properly calculate power factor, watts, etc. Keep in mind that my generator only puts out 120VAC single phase, so the pzem module can be powered from EITHER A side or B side. The pzem split core current sensor must wrap around BOTH of the two Generator hot wires (for A &B). Also, in most typical house panel boxes, installing the two new breakers one below the other will insure one is on A and the other on B.

I mounted the RPi outside of the breaker panel box. See Photo5. I used a standard 4-gang (i.e. made for 4 switches) plastic electrical box recessed in a garage drywall cutout. I drilled a large hole in the top of the box to pass the 3 usb extension cables along with the RPi usb power cable to the RPi. The clear Lexan polycarbonate cover from the big box stores was easy to cut with a utility knife and snap. It is basically flush with the drywall using standard outlet screws to the box or just a couple of 6-32 thumbscrews for easy access.

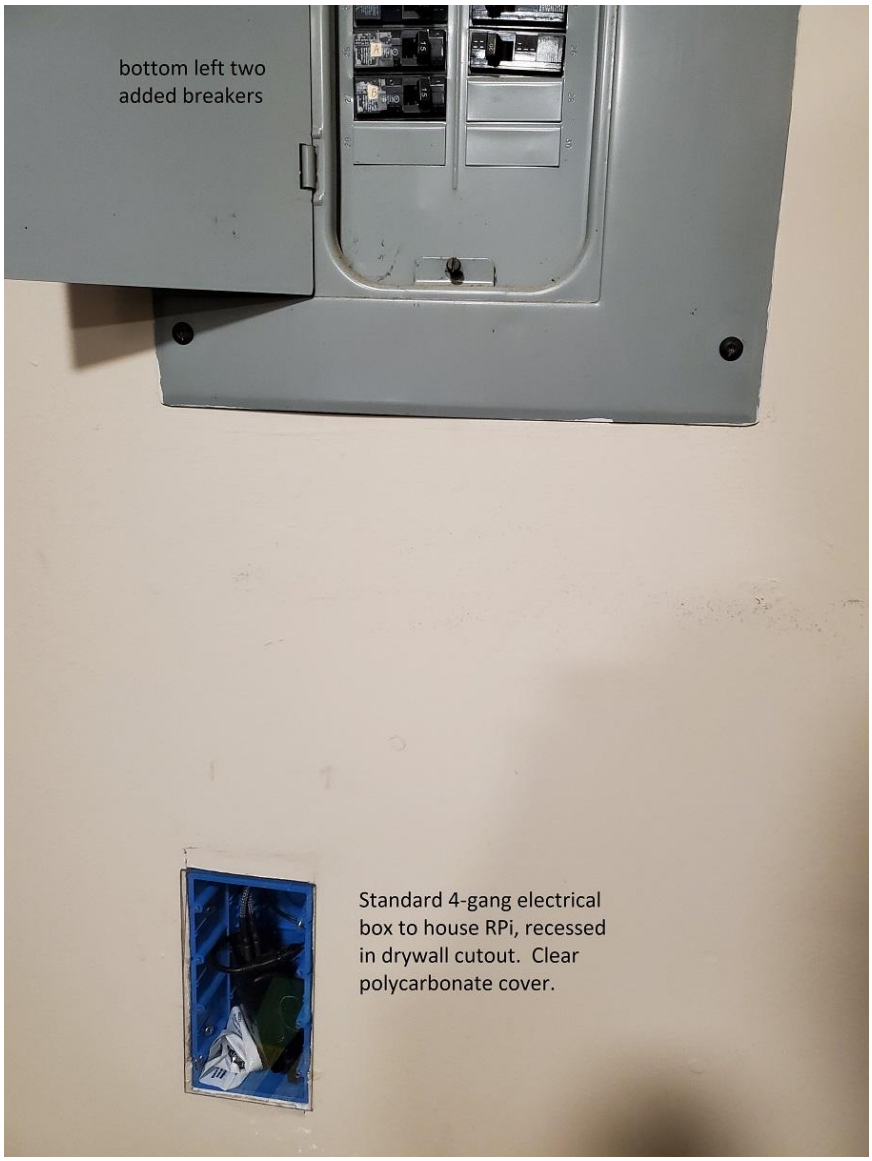


Photo5

You are done. Close everything up and perform final verification that all is working well. Over time, you will develop different Grafana panels to view your house power data, send email alerts, and many more capabilities thanks to Grafana.

Make a backup copy of your entire sd card.

Optional, but recommended. If you have rpi-clone installed on your RPi, below is the command to make a copy of the entire sd card, on the fly, without removing the card, and while the RPi is running. This is the typical two partition clone – currently running microSD card to USB microSD adapter stick plugged into RPi usb port:

```
$  sudo rpi-clone sda
```

This took less than 5 minutes.

InfluxDB backups

Optional, but recommend some form of backup of your data on a periodic basis. For example to perform an entire InfluxDB backup use the ib.py script:

$ python3 ib.py

Some other info:

Once you plug in a USB drive, the OS will detect the drive and partitions, and assign a name to it. For me the name is /dev/sda1

To list the mounted devices: $ lsblk

List the files: $ sudo ls /media/influx-backup # Note after the drive is unmounted, you cannot see the individual files using sudo ls or FileZilla

copy this to look at files $ sudo mount /dev/sda1 /media/influx-backup

copy this to unmount $ sudo umount /dev/sda1

To make the folder and everyting below the folder r/w/e by anyone:
$ sudo chmod -R 777 /media/influx-backup