Design of PowerCost Application

Table of Contents

1. Introduction

Due to my house’s significant usage of electricity, I needed to get a handle on how much electricity I was using on a daily and hourly basis. Cost estimates are little more involved as kWh is charged at different rates up to 600 kWh and above.

1.1. Background

I purchased an EMU-2 home energy display from Puget Sound Energy (PSE). The EMU-2 unit is manufactured by Rainforest Automation. PSE intended customers to use the EMU-2 to ascertain how much electricity each appliance uses in an effort to reduce electricity usage. A customer would put batteries in the unit, walk around and turn on and off various appliances to see the usage.

My intent from the start was to track the total household kWh electricity usage over time. I can see what hours of the day have the most usage and I can track the daily usage as I progress through the monthly billing cycle.

I searched for a way to script the collection of data. I found the emu_power Python library that automated the serial interface between my PC running Python and the EMU-2 device. The same Python script can run on Linux and Windows computers with the only difference being the device name. Windows 11 emulates the serial interface over USB.

emu_power is based on the XML spec for the Rainforest RAVEN API. The spec is similar to the API that the EMU-2 device uses. Please see **emu-power 1.51** at https://pypi.org/project/emu-power/

1.2. How to Run

My first version of the application used Windows batch files to run Python. The batch file contained all of the script parameters. I converted the batch file to PowerShell to better control the output of the start up script and this output could contain any Python syntax errors.

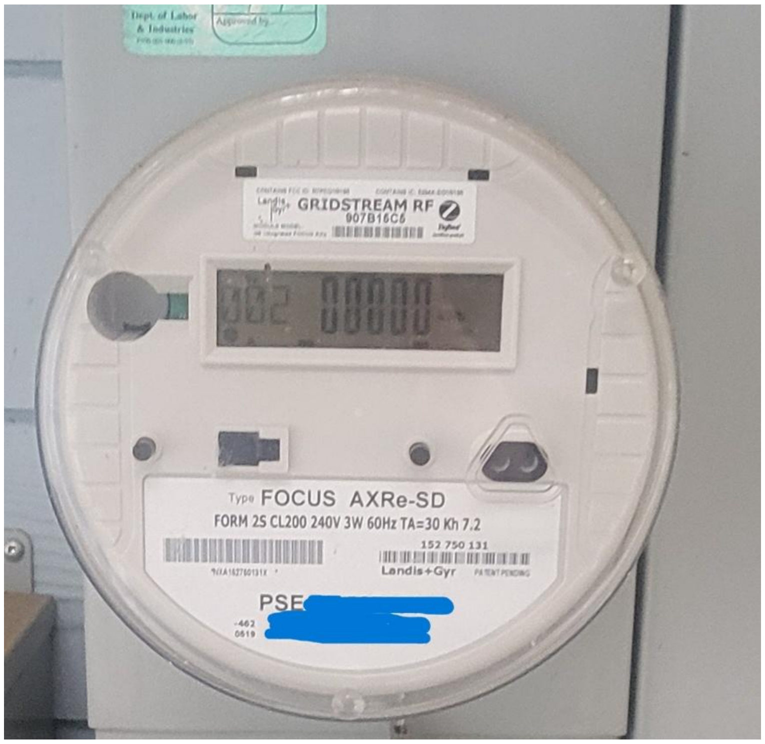
In this first version, the starting point was a Windows Scheduled Task or something akin to a UNIX daemon. The trigger was to run the BAT/PowerShell script at system startup, This may have added unnecessary complications since the Python script is intended to run for an indefinite period of time. Windows Task Manager would be used to stop/kill the script. The PowerShell script can also be started from the command line. This scheme may need to be re-examined, that is, how often is the script started via how long does each run take? I had to restart the serial connection every hour since device would eventually lose track.

1.3. Database

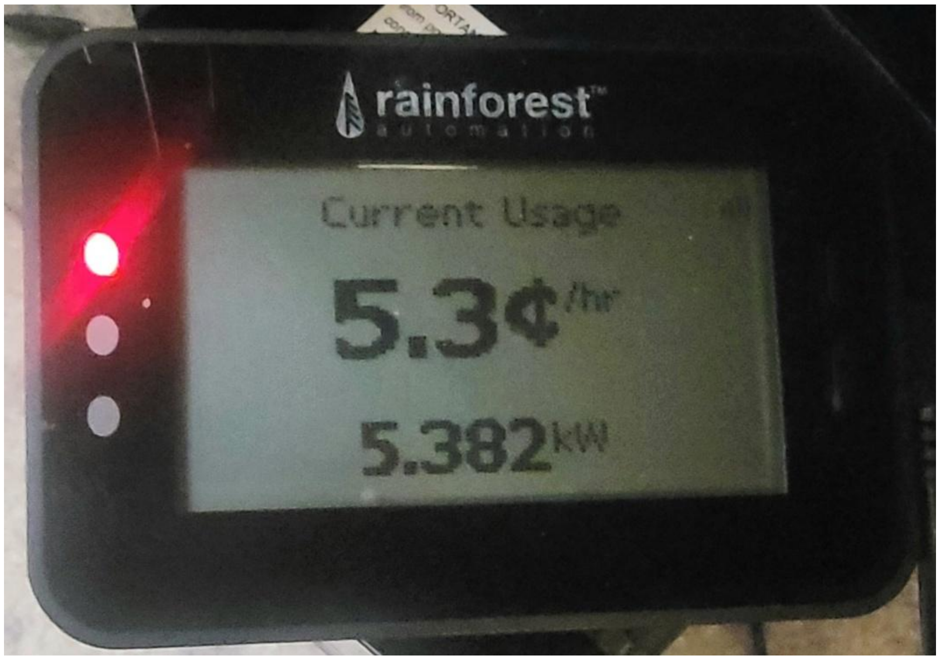
I decided to use a MySQL database due to its ease of use and my experience working with DB2 and Oracle. This design decision facilitate subsequent data analysis via simple SQL scripts of stored procedures. I could use the MySQL Workbench to export the table into CSV and then into Excel.

2. Power Meter and EMU

Puget Sound Energy power meter.

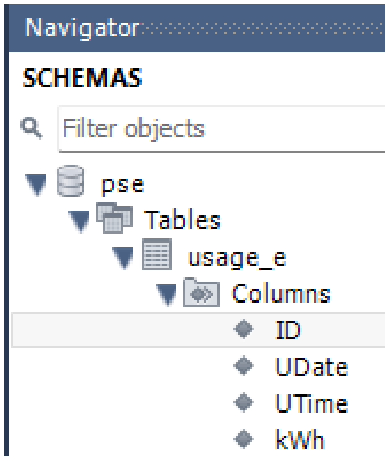


RainForest Automation EMU



3. Database

For the first application version, there was only one MySQL table that contained kWh amounts for each hour. The script takes a sample reading every minute and then adds up these readings and divides by 60 to get an average for the hour. This averaging is done to smooth out any temporary power spikes. Subsequent tables could contains reports of different aspects of power usage. For example, a stored procedure could sum up or calculate an average power usage for each hour of the day or a procedure could look at the daily power usage. These two analysis efforts could look for usage patterns during the day or week.



Creation

```
create database pse;

CREATE TABLE pse.usage_e` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `UDate` DATE NOT NULL,
  `UTime` TIME NOT NULL,
  `kWh` DECIMAL(7,3) NULL DEFAULT 0.0,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `I_USAGE_E_UNIQUE` (`ID` ASC) VISIBLE)
COMMENT = 'Puget Sound Energy Electricity Usage for The
Ponderosa';
```

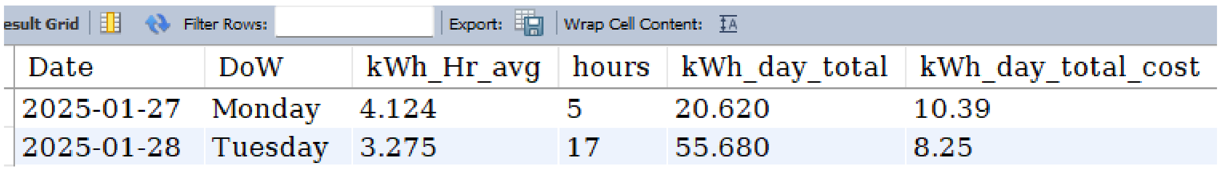
The usage_e table uses an auto-incrementing integer `ID` which also makes each row unique. `ID` is not directly needed in queries and is part of the unique index for faster queries. `UDate` and `UTime` are in separate columns to allow easier querying by hour or day. `kWh` is formatted as decimal value with 3 digits of accuracy due to EMU device's limitations. Allowing 7 digits for the value, or 4 digits to the left of the decimal is a bit of overkill as the kWh usage for an hour is less than 10.

Design of PowerCost Application

The following SQL attempts a minor amount of reporting of daily kWh usage and its approximate cost.

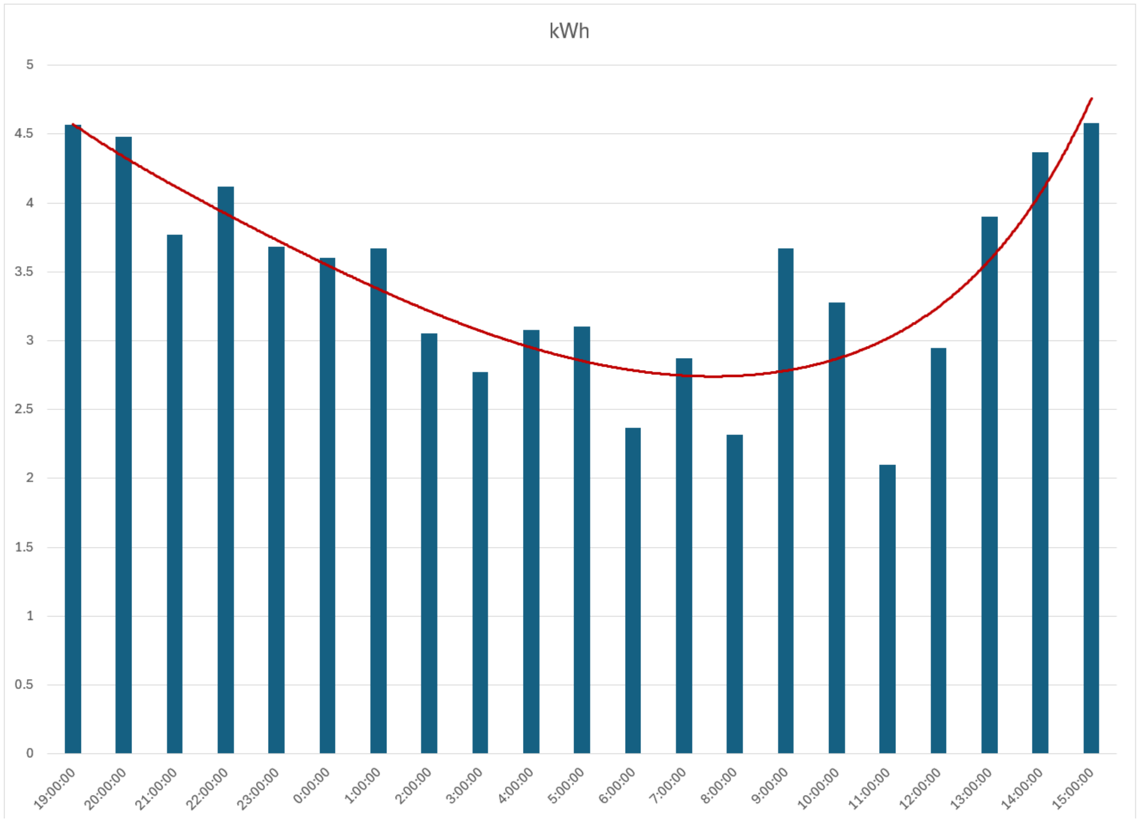
```
Simple Select for Daily Totals

select
   UDate as Date

,ELT(dayofweek(UDate),'Sunday','Monday','Tuesday','Wednesday','
Thursday','Friday','Saturday') as DoW
   ,round(sum(kWh)/count(kWh),3) as kWh_Hr_avg
   ,count(kWh) as hours
   ,sum(kWh) as kWh_day_total
   ,round(((sum(kWh)/count(kWh))*24*0.105),2) as
kWh_day_total_cost
from
   pse.usage_e
group by
   UDate
order by
   UDate
;
```

I included "hours" to check if the script did not run for the full 24 hours of each day. In the case below, I started the script well into 2025-01-27 and I did the query with several hours left in 2025-01-28. In other cases, the script may not get 24 readings in a given day.



Date	DoW	kWh_Hr_avg	hours	kWh_day_total	kWh_day_total_cost
2025-01-27	Monday	4.124	5	20.620	10.39
2025-01-28	Tuesday	3.275	17	55.680	8.25

<u>Direct Export to CSV and then Excel</u>



3.1. xxx

xxx

```
# Stuff
```

```
zzzz
```

4. Startup Script

The PowerShell script absorbs the complexity into itself for application startup where this is done from the command line, shortcut or Windows Started Task. It also captures the output of the startup to a file.

```
# Set timestamp to be used in the log file.
$ts   = Get-Date -Format "yyyyMMdd-HHmm"
$log = "D:\a\EMU-2\logs\launch_" + $ts + ".txt"

# Run Program
# -----------
$basedir = "D:\git\powercost\src"
$script = "$basedir\Ponderosa_Electricity_Usage.py"
$Stream = [System.IO.StreamWriter]::new($log)
$Stream.WriteLine("script   = " + $script)
$Stream.WriteLine("basedir  = " + $basedir)
$Stream.WriteLine("log      = " + $log)

# Redirect standard output to the StreamWriter
[System.Console]::SetOut($Stream)

$process = Start-Process -FilePath "D:\Python\pythonw.exe" -
ArgumentList "$script  --ini
$basedir\Ponderosa_Electricity_Usage.ini" -WorkingDirectory $basedir
-PassThru -NoNewWindow

$Stream.WriteLine("PID      = " + $process.Id)
$Stream.WriteLine("Name     = " + $process.Name)
$Stream.Close()

exit
```