This document is a record of my domestic PV project using a Raspberry Pi as a control system – status at November 2020 – ON HOLD waiting for improved finances……

Darrell Moores

Warning - I am not an electrician!!

Aim

Create a domestic power system that:

- Generates as much solar power as possible.
- Uses as much of that power as possible, stores power if possible and and exports excess to the grid.
- Uses the grid to supplement use when load exceeds generation.
- Provides a UPS capability to cover rare grid outages
- Provides EV charge capability to replace existing diesel car for local range travel only
- Provides V2H (Vehicle to Home) and V2G (Vehicle to grid) capabilities
- Makes use of cheap rate electricity and whatever other price models that are developed to integrate small scale generators into a comprehensive grid.
- Reduces current oil used for domestic heating and hot water heating.

Planning

Worked out that the SW facing garage roof can take 36 x 315w panels to provide 11.3kw max.

The house and garage sits within a restricted building area.

Had a chat to the electricity grid supplier about what could be connected to the grid. Their constraints are that only 4KW could be exported to the single phase domestic grid. Chatted about converting to a 3 phase supply but the cost is exorbitant.

September 2019

- Applied for planning permission for 36 panels using a Renusol fixing system to the metal corrugated roof.

December 2019

- Permission approved.

System design

A lot of research later, the system must conform to the following constraints:

The system must have a battery system to store excess KW during the day and provide it to the house at night.

- Bought a simple CT instrument and measured the main grid supply to the house, which shows an approximate overnight average use of 11KWH.

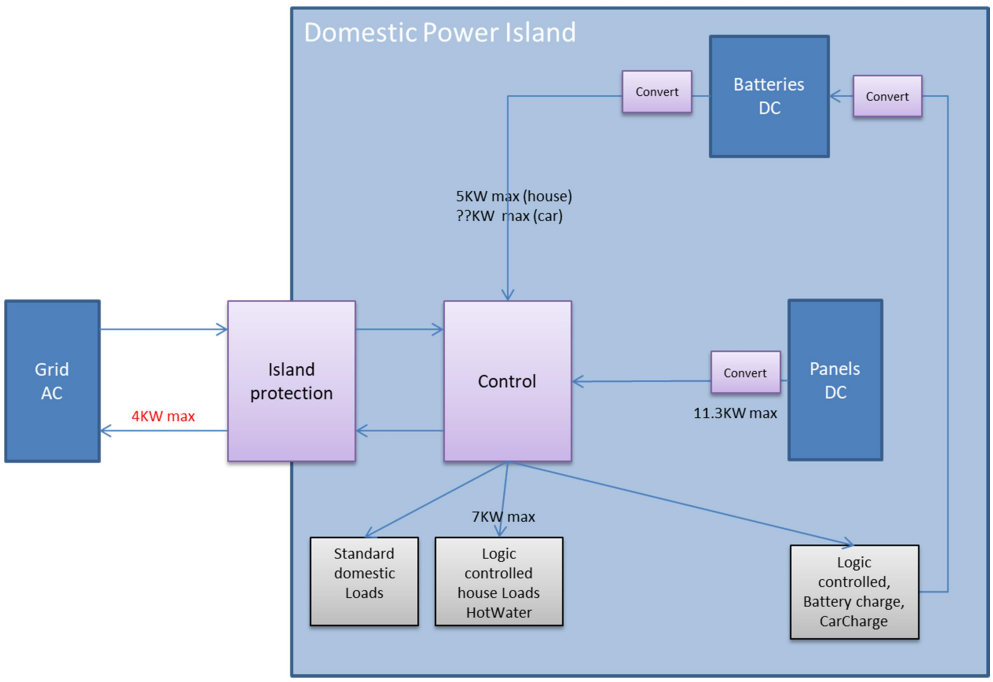
Any UPS capability must provide an ‘islanding’ mechanism to isolate the house circuit from the grid if the grid is turned off to prevent PV or battery power killing a grid worker.

I want to be able to control when the batteries (house and car) are charging and discharging.

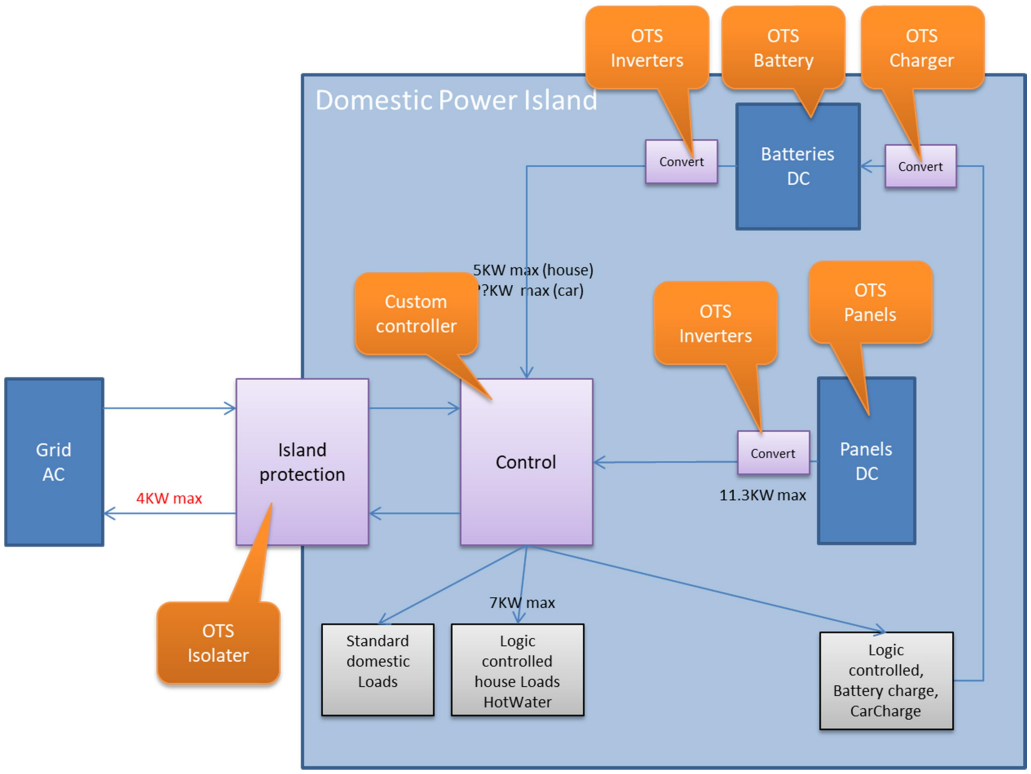
I want to be able to control when certain high power appliances (Hot water heater for instance) are used.

I want to take advantage of cheap electricity time windows (like off-peak) if available.

At a high level the design looks like



- Investigated existing PV control systems, and while different vendors can supply aspects of the system, no single vendor supplies a comprehensive control system that satisfies the requirements. I could wait but decide to design my own - Gulp! I am not an electrician, but I do have antiquated programming experience.
- Decided to implement the design using OTS (Off the shelf) components controlled by a custom control system. How hard can it be?



Research and experiment

- Researched control systems and programming languages. Decided the simplest, most easy to understand, and initially cheap way to get started was to use a **Raspberry Pi** programmed using **python**.

Christmas 2019

- Added a Pi to my Christmas list. Santa provided.

January 2020.

- Researched domestic mains power wiring (did I mention I am not an electrician), and found out that the hot water tank has a 3KW heater on its own circuit with a 20a isolator, The main house fuse is 100a.
- Set up the pi, downloaded Python 3.7 to my laptop and searched for code and pre-existing components (Hats and boards) that used CT sensors and controlled switches.
- Researched how to switch mains circuits on and off, and worked my way through contactors, relays and isolators. Landed at SSRs (Solid State Relays) that are controlled by turning a low voltage LED on and off. SSRs can be used to turn on up to 100amp 240 volts.
- Researched PV batteries and found that the best storage to price is currently a Tesla battery that can store 13.5Kwh and can provide a max of 5kw power to the house/grid. But crickey are they expensive.
- Researched EV batteries to discover that I should limit max charge to 7KW, but as V2H/V2G is being developed, there is not much information on max charge they can provide.
- Researched grid suppliers to find that export tariffs and V2G tariffs are ‘in development’ so my system must be flexible and easily changed.

February 2020

Experiment 1 - SSR

- Bought an SSR (20amp with a 4v minimum control voltage), and a heat sink. Bought the Pi a breadboard, some wires, a fan and a case (base only as I have no idea how big the final component will be).
- Set up the Pi. Tried to write some python on my laptop, compile it to an executable and transfer it to the Pi. Doh! Cannot create a linux executable on a windows development system. GPIO python library does not work on a windows system even to prototype code. Set up Python on the Pi by updating the latest packages. Lots of searching on basic linux commands – what a load of unintelligible gobbledigook! Now have a simple script to turn on pin 18. Unfortunately, the GPIO pins that you can control via a program, only deliver 3.3V. The Pi does have a 5V pin and I naively assumed you could use this in a program controlled pin, but no. My SSR has a minimum control voltage of 4V.
- Connected the low side of the SSR to a 9V battery. The LED turns on fine. Connected It to 2 x 1.5V batteries, and the led light still came on, so will try this from the pi even though it is outside the required control range.
- Created a test board for the SSR protected by a 13amp fuse, and connected up a fan heater as a load.
- Connected up the Pi and hey presto I can turn the fan on and off. Success! I can now use program logic to turn mains circuits on and off.
- That is great, but I suspect that when the SSR gets hot that 3.3v may not be enough to keep it running. Researched how to generate a 5V program controlled pin.
- The SSR is also dangerous in its current test state. It has open 240V connections. Find and order a cover for the SSR.

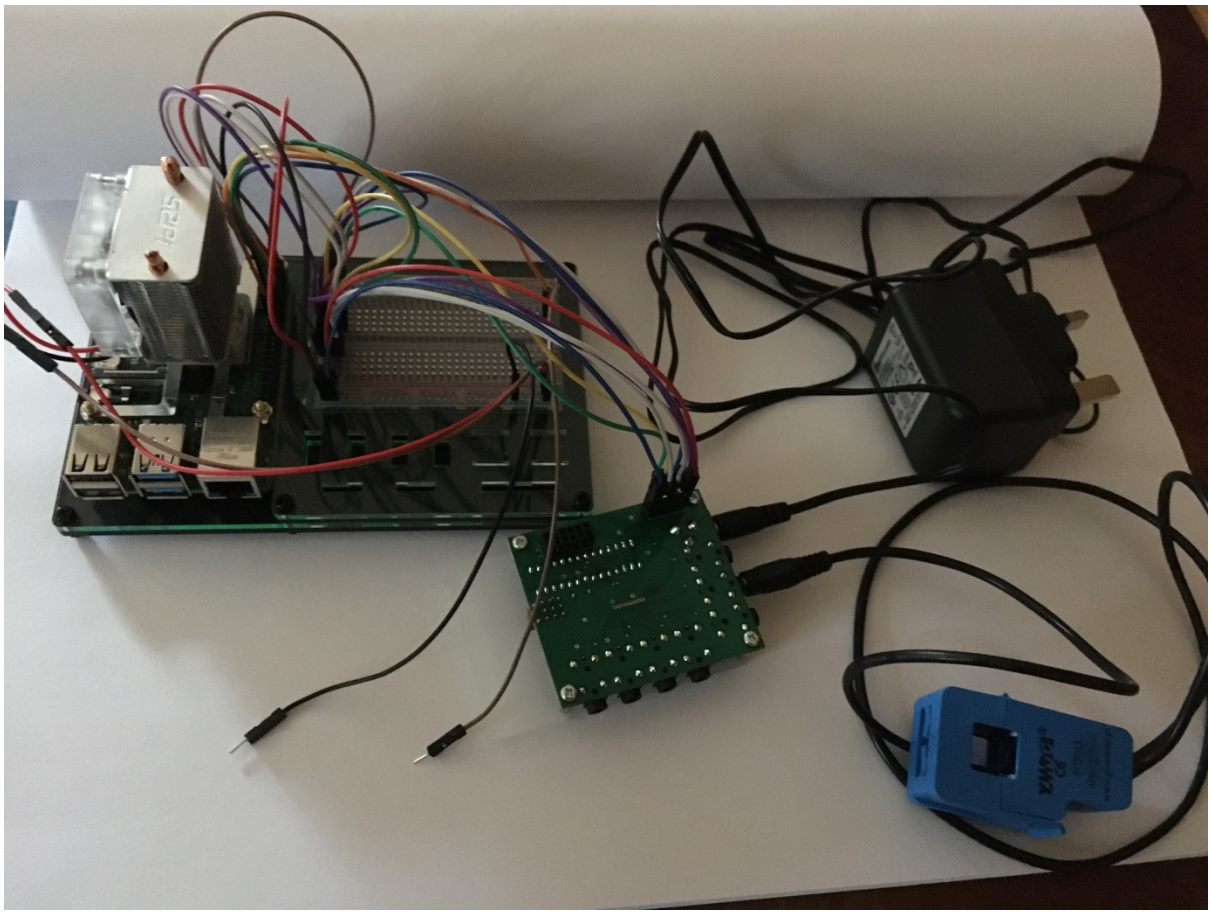
Experiment 2 – Python continuous execution and GUI

- Researched and experimented with Python code to run a continuous loop that would request data from sensors, decide what to do, and record actions taken. Now have a loop and a simple CSV log to store sensor information and switch events.
- Decided that I wanted to keep access to the control system 'in-house'. So no external internet connections. Also decided to keep communication simple, to cut down reliance on multiple technologies. All sensors should be wired, and all control signals should also be wired.
- GUI should be direct from the Pi, but as a future development, a Bluetooth data transfer to a remote (within Bluetooth range) app would be useful.
- Running 2 continuous loops, one for the control system and one for the GUI seems to be a problem. May have to investigate 2 processes using a common data store. For a Bluetooth remote GUI will probably have to do this anyway. Also would be nice to compile the GUI into an app for ios, android, windows and linux. Hmm, maybe my initial choice of the TKinter GUI framework wont cut it.

Experiment 3 – CT sensor

Ordered a CT board (Chacal hat -7 power and 1 voltage inputs), a CT clamp, a transformer for the voltage input. Connected it to the Pi, but had to turn the Chacal hat over and connect up using the breadboard because of my ridiculously large fan. The Chacal set up instructions turn off bluetooth on the Pi, I think this is because the hat uses it as the serial port for the CT sensors. I want to use Bluetooth eventually for my app, so instead of using ttyAMA0 I have used ttyS0.

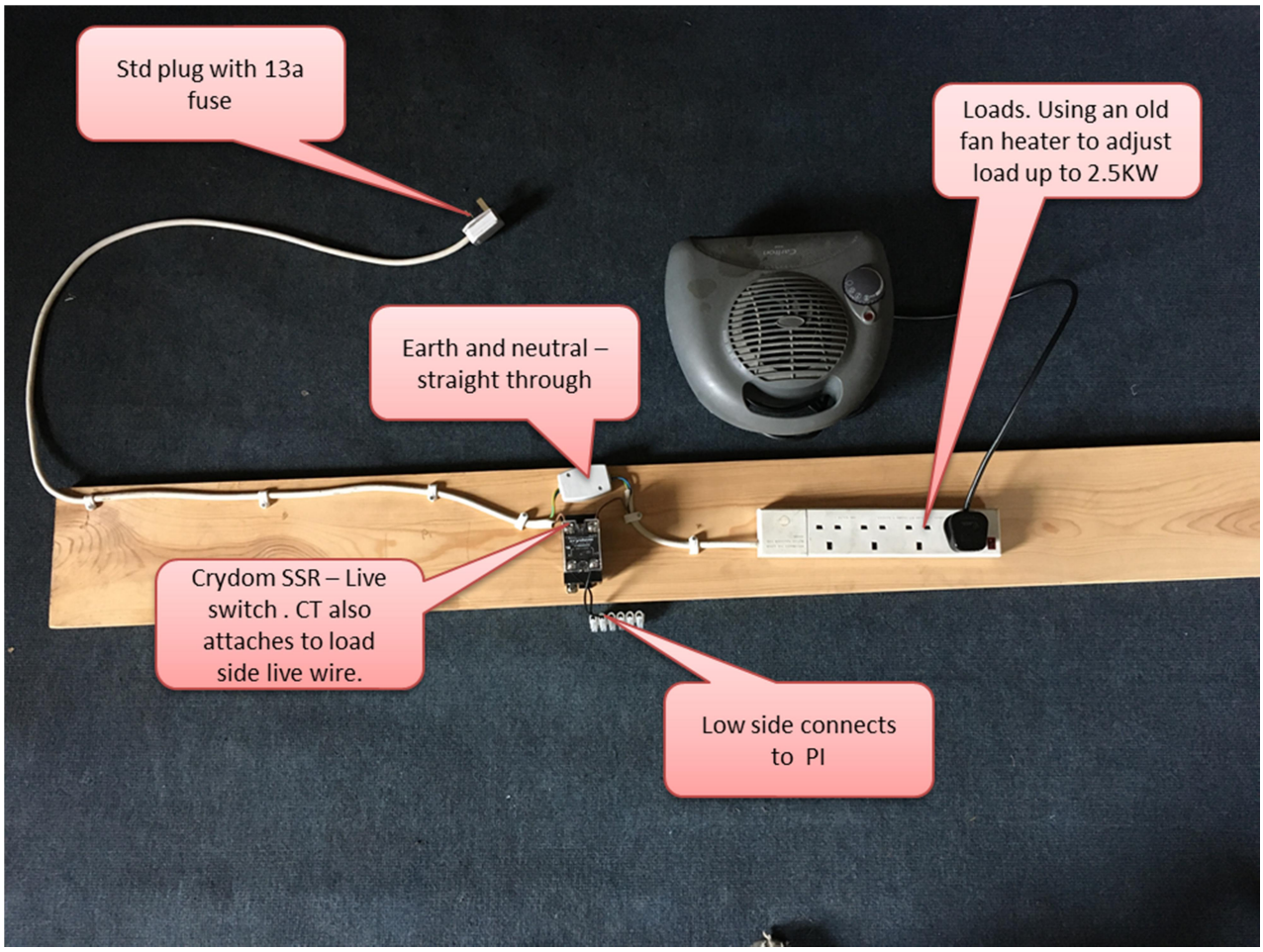
Pi looks like this:



The Pi set up is getting messy with fiddly wires and breadboard connections. Will need to research how to 'productionise' the set up so it becomes less fragile, but not until all experiments are over.

Connected up to my test board so I could turn on the SSR and attached loads and see the difference in load via the CT sensor.

Board looks like this:



Woohoo! I can now instrument all the loads in the system, and use the data to control mains power switches via a program.

Python test script now looks like:

```
import serial
import RPi.GPIO as gpio
import time
# set up chacal hat to use serial input
ser = serial.Serial('/dev/ttyS0', 38400)

#set up pin 17 to control SSR
gpio.setmode(gpio.BCM)
gpio.setup(17, gpio.OUT)

#turn the SSR off
gpio.output(17, gpio.LOW)
print ("Off")
# read the power
response = ser.readline()
print(response)
# wait for 10 secs then turn the power on
time.sleep(10)
gpio.output(17, gpio.HIGH)
```

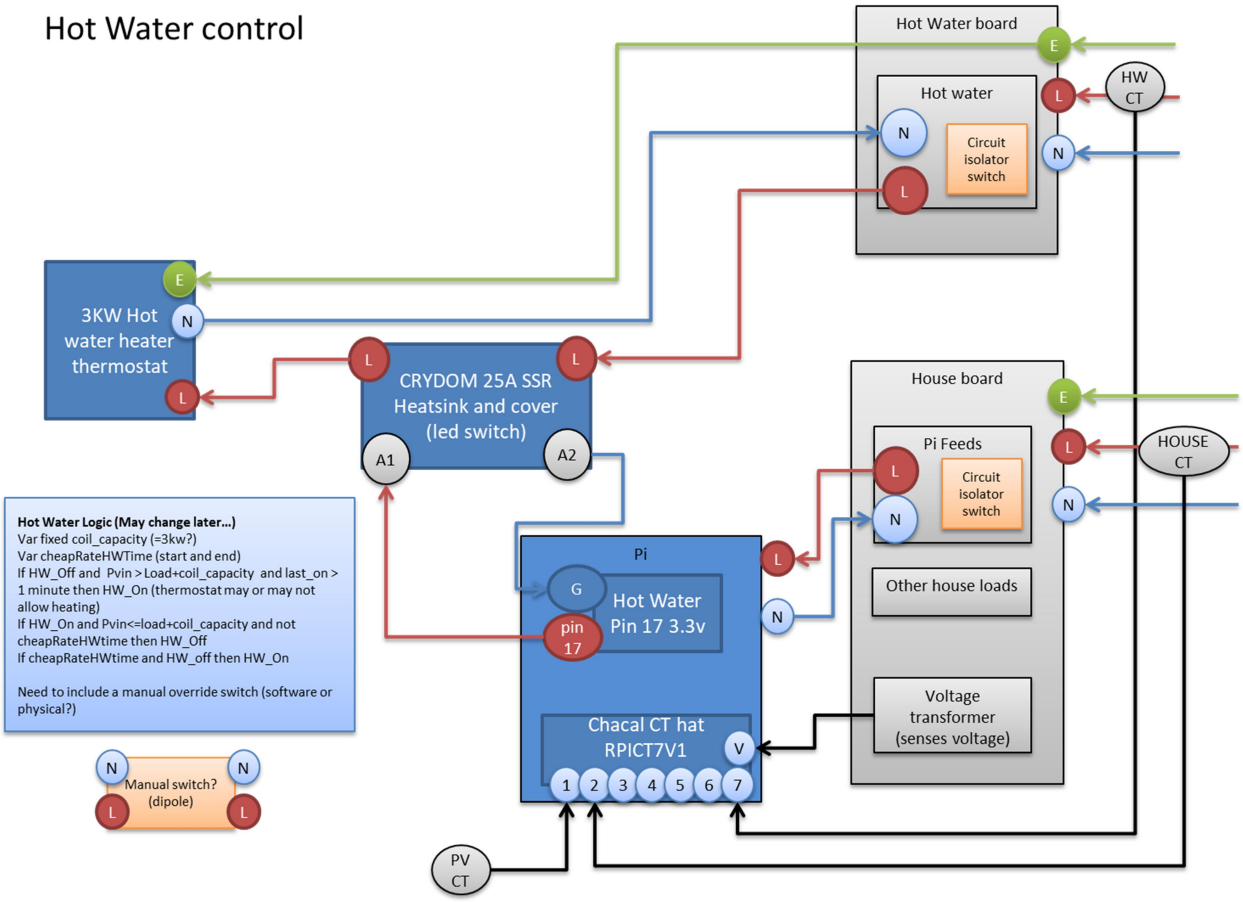
```
print ("On")

# For 10 times, get the power readings (during this time
# I am using the fan controls to change the power load)
# The chacal hat has a 5 second delay between reading the sensors
count=1
print("reading")
while count<10:
   response = ser.readline()
   print(response)
   count=count+1
#Turn the SSR off
gpio.output(17, gpio.LOW)
print ("Off")
#Clean up pins and serial port
gpio.cleanup()
print("clean")
print("close")
ser.close()
```

CT Results look like:

<<Add results>>

I now have all the components necessary to control my hot water system, if I moved the components from the test board to the house board like this:



Experiment 4 – connect SSR and CT to GUI controls – multithreading – can it work for a long time?

Connected the GUI control experiment to the SSR experiment, and coded up a separate thread to read the CT sensors and create a log file entry every 2 seconds. The TKinter GUI runs in its own thread and I can use it to control the state of the SSR and see the loads change both on the GUI and in the sensor log. Have programmed it to record a new sensor log file for every day, I currently have a separate file to record user actions. I might try to rationalise these into a single log file when I look at graphing the logs and overlaying the user and automated state change actions.

Connected up the loads to a humidifier (with humidistat) and an electric radiator (with thermostat), ran it for 48 hours and monitored VM size, RAM and temperature. Hardly touch the CPU (3%) so heat is minimal, and RAM and VM size stayed constant throughout the 48 hr period. Sensor log files are well within limits.

Great. It should be possible to run this system continuously. It will contain more logic(so more CPU usage) when complete, but it looks to be well within bounds.

Experiment – Graph views of the logs

This is easy for python so will skip this till later.

Experiment – battery charge/discharge control

I want programmatic control of the battery charge and discharge times and rates, but I want to delegate to a battery charge controller to control the cell distribution and monitor temperature and charge status. Not sure how this will work. Commercial Battery controllers seem to be morphing into system controllers, for instance the tesla PW2 with gateway, but they do not yet do V2H or V2G. The EV chargers are also morphing towards system controllers,. What is required is a single system controller that just treats an EV as another battery. – Research required.

So, the concepts of battery charging and discharging, grid connection, car charging, V2H, V2G and export limits are all tied up together.

To tease them out, lets try to deal with them one at a time.

Export limitation: Modern smart inverters, and battery chargers are moving their capabilities towards system controllers. For instance a smart battery charge controller can tell PV inverters to change the frequency of their output so they produce less power. I would rather do something simpler that I am in control of. Using a simple switch on a PV string I can vary the AC current produced by the panels. Having 9 strings of 4 solar panels, each string can generate a max of 1.25kw, and putting a 1.5Kw inverter on every string each with an SSR on the AC side, and a single CT clamp to measure the total PV being released, and another on the grid supply to measure export power, should allow me to reduce/increase PV tand keep within the 4kw export limit.

House battery charge: Adding an SSR before a battery charge controller will allow me to turn it on when there is enough spare PV to charge the battery. This would just appear as another load on the system. When battery charging is on I would turn off the grid (unless I want to charge the battery from the grid)

House battery use - There are several scenarios:

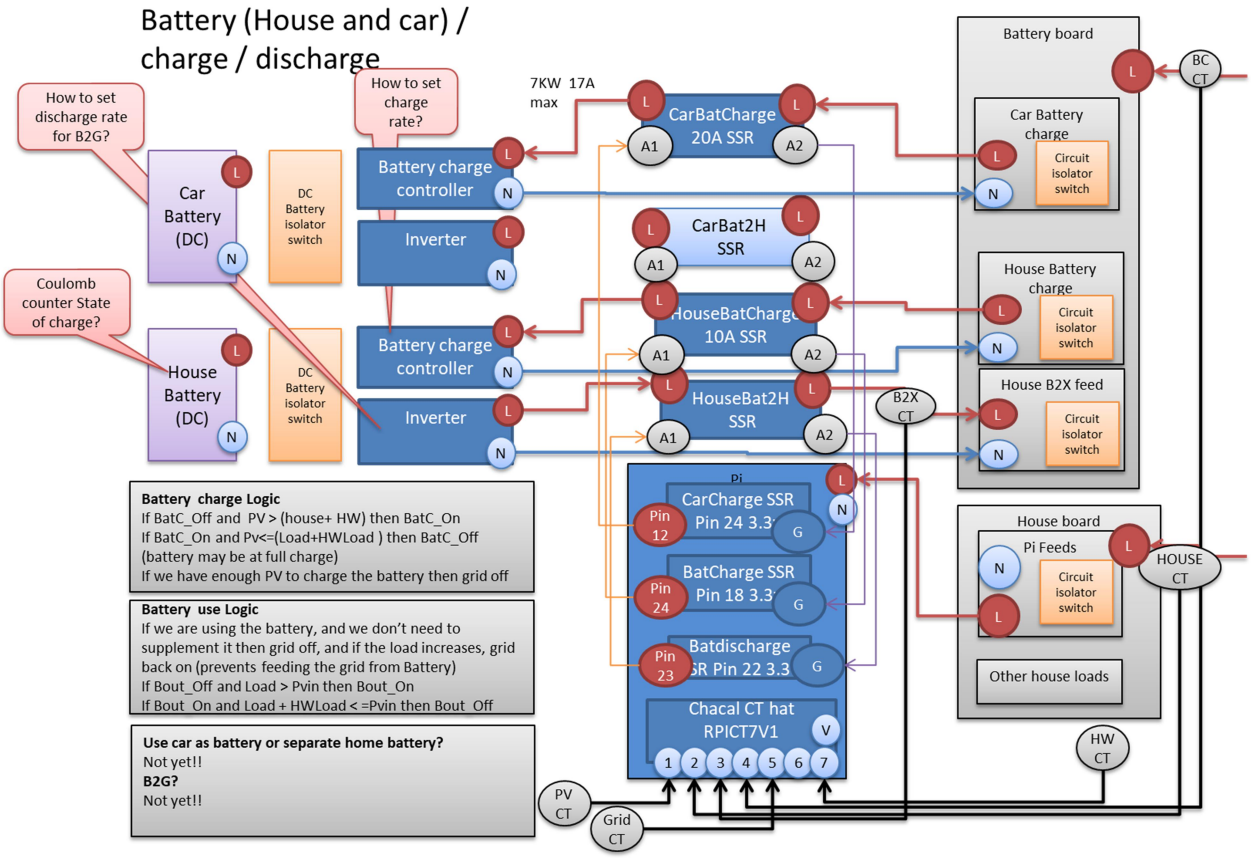
1. The house load is greater than the available PV, and the battery can supply the excess (this will cover early evening periods). If I know the house load and the PV output and the battery discharge rate and its state of charge I would switch off the battery charge switch, switch on the battery inverter to power the loads
2. The house load is greater than the available PV plus the battery supply (likely at early morning periods), and must be supplemented from the grid. In this case I would turn on the battery inverter (to get what I can from it) and turn on the grid. Because the battery inverter will generate AC at a higher voltage then the grid, then the battery voltage should be used first. The battery should be turned off when it hits a minimum SOC.
3. The grid has failed. Islanding requirements state that the system must be detached from the grid if it fails so that PV is not exported. So I must detect that the grid has failed, detach from the grid and move into scenarios 4-6?.
4. Grid has failed, house load > PV + battery. We don't have enough to satisfy the load, what happens in this scenario? I assume some loads stop working but might need an experiment.
5. Grid has failed, PV > house load. Turn off battery. Turn on charge and hot water if enough PV. Do we need to turn off excess inverters, or just leave everything running? nothing should be exported.
6. Grid has failed, PV<load. Turn on battery.

V2H: I want another plug on an EV for exporting to an inverter, but they don't have them. If I build my own EV I can add one (is this other project??) and hacking an EV voids the warranty. Bi directional car chargers are starting to appear but these are expensive mainly because they are also trying to be system controllers. Essentially V2H should be simple, but one for the future.

V2G: Comes with all sorts of complications.

1. The utility companies want control of this process.
2. I want a simple process that can charge my car from PV or cheap rate grid power and export it back at a beneficial price. In order to export it, I cannot break the export limit. So an inverter must be able to set the AC power generated, and I must be able to blend it with PV and the house battery so the total does not breach the limit.

A battery layout diagram would look something like (diagram does not show V2X connections)



March 2020

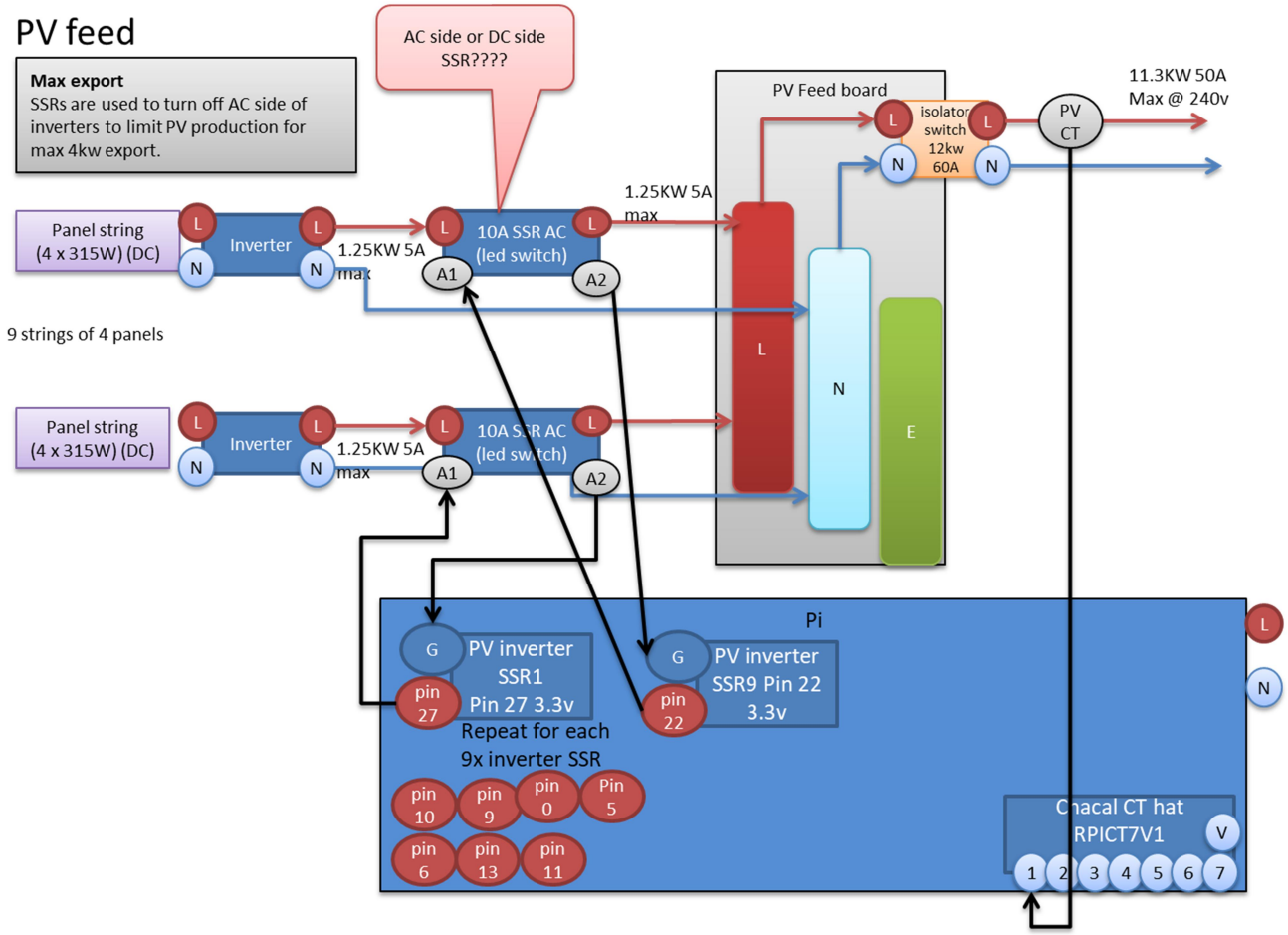
Experiment – PV inverters

I now have a GUI which can control 9 SSRs, one for each string of 4 panels.

But where to put the switch? Between Panel string and inverter (DC) or between inverter and system (AC)? If the panels are disconnected from the inverter then because there is no load the

panels will stop generating. If the inverter is disconnected from the system then does the inverter provide a load that keeps the panels generating, so it must absorb the power?

The eventual physical layout would look like this (showing the SSRs at the AC side which may need to change)



COVID19 – waiting for the stock market to restore my coronavirus degraded savings! Have to put this project off for a couple of years till I can afford the panels, inverters, SSRs and battery.

Will still make progress with the GUI to:

- Display a ‘Now’ view of the various feeds and loads.
- Display a historical view to allow the user to see previously recorded data.
- Provide manual controls of all SSRs

April 2020

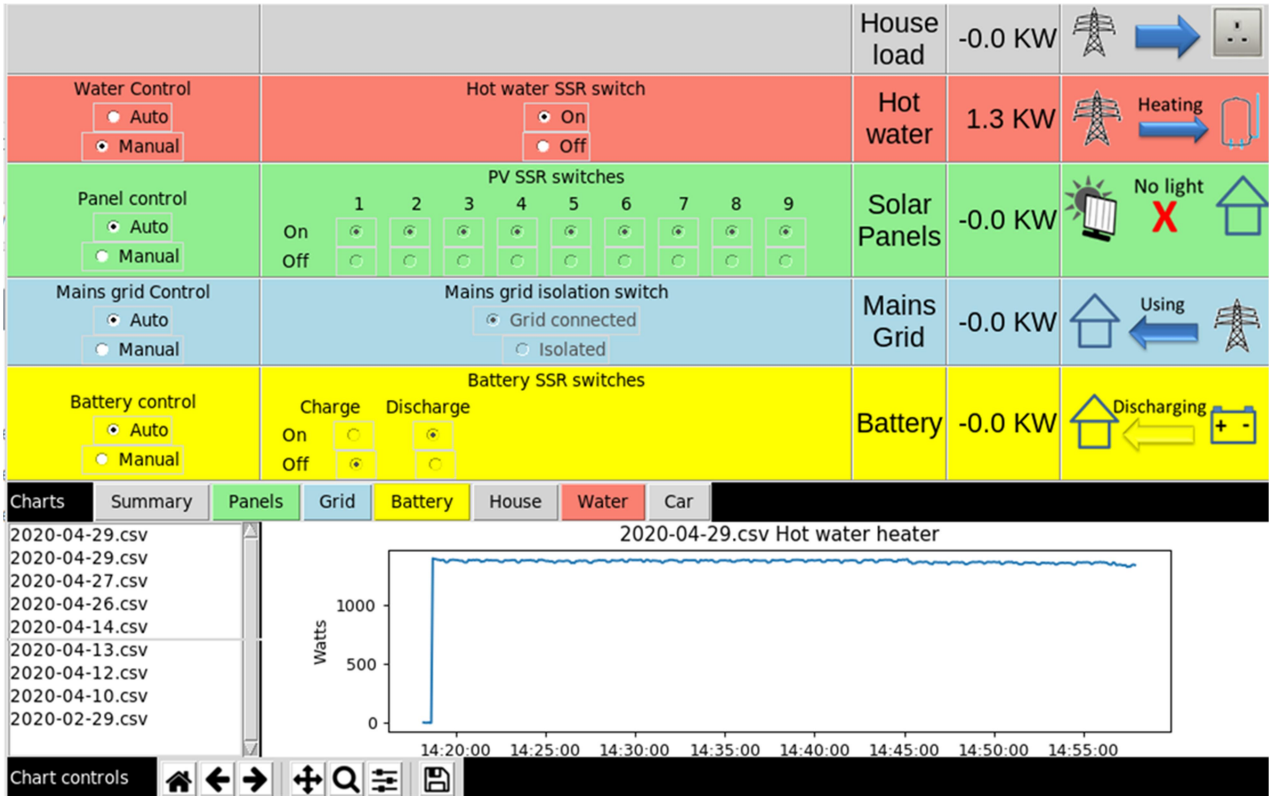
Made good progress on the GUI:

- Graphs of CT records can now be displayed
- I have a ‘day summary’ graph

Made progress connecting the GUI back into the control thread, as if the Pi had access to all the SSRs.

Have developed automated processing logic to turn on/off SSRs depending on CT data, and have given the user override control which will turn off aspects of automation. Still need to think about some logic where manual control breaks the rules.

GUI now looks like (with only 1 CT clamp attached):

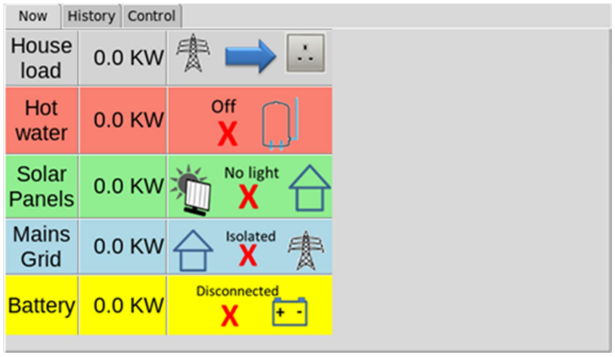


I really only need the top right corner to see what is going on now. I could hide the user actions away behind a tab, same with the charts, but it will do for now. The icons on the right change depending on the values of the feeds and loads. There are still some GUI bugs to sort out.

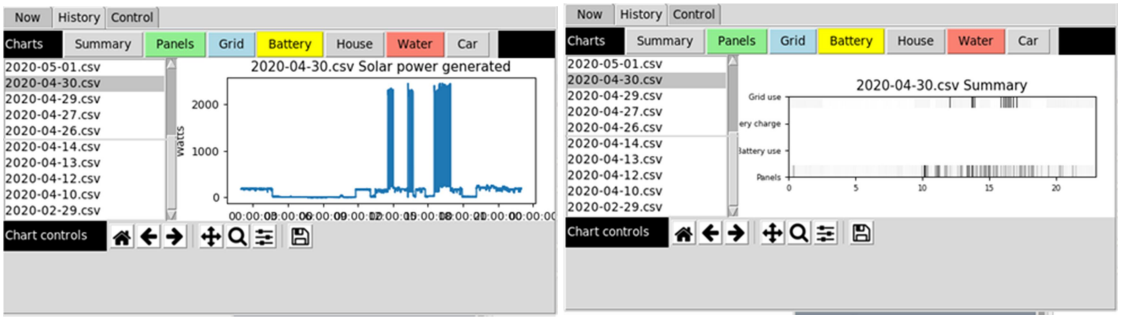
Running another live experiment with my 2 CT clamps and a single SSR switch over a few days to test for memory leaks etc. It is also a good way to start looking at the automation rules as the switch changes are at least reflected in the GUI.

I am going to have to set up a test CT file and use it as input in a testmode. This is the only way I can test out the full set of automation rules.

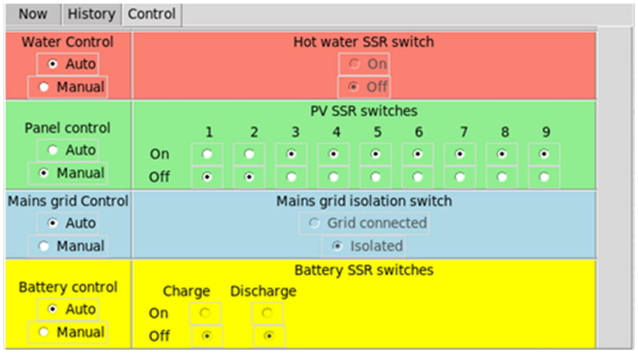
Experiment to use the GUI on my phone and ipad: Downloaded RD client to my iphone (OK) and ipad (not ok, the ipad is too old to support the latest IOS versions required for RD client). Connected the client to the Pi and can now operate the GUI from my iphone. Reorganised the GUI to a tabbed pane so the screen estate fits better on the small screen. This provides a much better GUI with monitoring information on tab 1,



historical information on tab 2, which contains a list of daily data captures, and a set of tabs to look at each flow and an overall summary which I have configured as a heatmap (matlab colormap).



and user controls on tab 3.

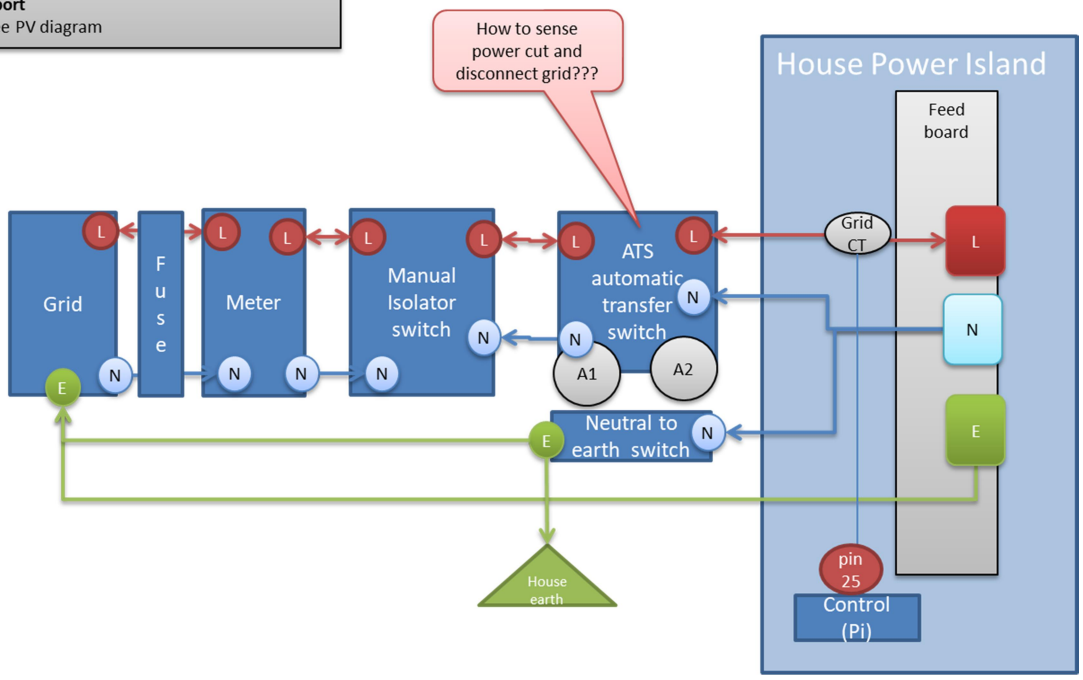


Starting and stopping the python program is a pain as I have to navigate the directory structure. Will have to think about how to deploy the executable in an easy to run format if this is to become a commercial offering (ie compile the code and run as an executable rather than in Thonny)
To come……

Experiment – Islanding

Grid connection and Islanding

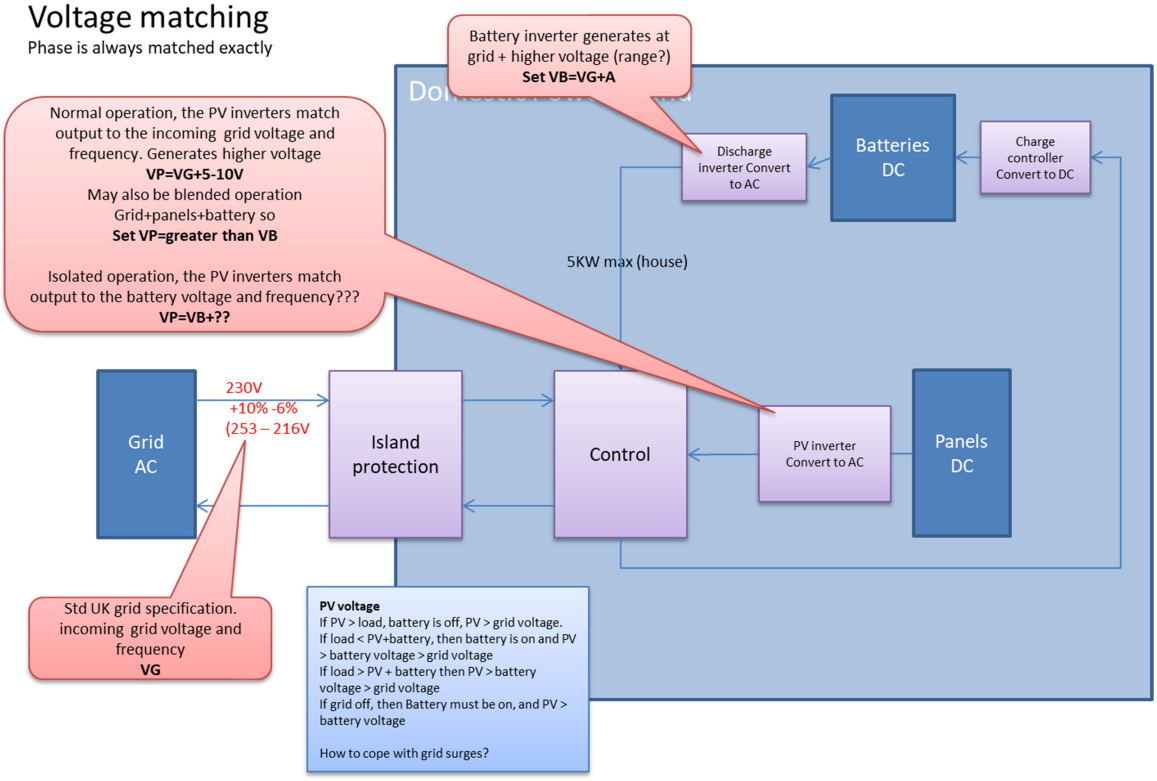
Grid isolation (islanding)
If the grid goes down, the House must disconnect from the grid within 4 seconds.
UPS
If the grid goes down the battery and PV should continue to power the island. (UPS capability)
Max export
4KW –see PV diagram



Experiment – How do PV, battery and grid work together. Need to test out the diagram below.

Voltage matching

Phase is always matched exactly



Python (this is changing daily!)

I now have over 1000 lines of py code, although Most of this is TKinter gui control code. It sits nicely on the Pi running at 2% cpu and the python 3 vm at a static170MB ram over a 4 day period.

It is however a monolithic program, but it does use a set of behaviour based classes. To do – break out into components which can be imported.

Current status
Covid impact has reduced money available to implement this, so it is on hold till things recover.