EmonLib Maths

Real Power

The formula presently used is:

$$average\ power = \frac{k}{n}\sum_{0}^{n-1}[(lastFilteredV + PHASECAL * (filteredV - lastFilteredV)) * filteredI]$$

where *k* is a calibration constant depending on the voltage and current calibration constants, the ADC reference voltage and the ADC resolution. *n* is the sample count.

Because PHASECAL is a constant, the equation can be rearranged as

$$average\ power = \frac{k}{n}.(PHASECAL.\sum_{0}^{n-1}[filteredV \times filteredI] - (PHASECAL - 1).\sum_{0}^{n-1}[lastFilteredV \times filteredI])$$

The implication of this is that by transmitting the two sums to emonHub and performing the final stage of the maths there, it is possible to perform phase calibration inside emonHub. Writing $k = k_1 \times k_2$, where k_1 is dependent on the ADC reference voltage and the ADC resolution, and k_2 is dependent on the calibration constants, it is possible and desirable to apply *n* and k_1 before the sums are transmitted, as these values are already known to the ATMega front-end processor and need not be known to emonHub, and k_2 afterwards as that is primarily dependent on the installed sensors and gives the user easy access to calibrate.

Viz:

$$A = \frac{k_1}{n}.\sum_{0}^{n-1}[filteredV \times filteredI]$$

$$B = \frac{k_1}{n}.\sum_{0}^{n-1}[lastFilteredV \times filteredI]$$

where $k_1 = (ADC\ full scale\ voltage / ADC\ counts)^2$

$$average\ power = k_2.(A \times PHASECAL - B \times (PHASECAL - 1))$$

where $k_2 = Ical \times Vcal / scalefactor$

As it stands, the linear interpolation/extrapolation generates an amplitude change. This can be avoided, for a pure sine wave, by modifying the coefficients:

$$average\ power = k_2.(A \times X + B \times Y)$$

where

$\Delta = sampling\ interval$
$\Phi = phase\ error\ difference\ between\ two\ transformers$
$Y = \sin\Phi / \sin\Delta$
$X = \cos\Phi - Y \times \cos\Delta$

Unfortunately, a wave that is not a perfect sinusoid will still be distorted, but to a lesser extent.

Transmitting the values *A* & *B*

As it stands, for the emonTx

$$k_2 \approx \frac{1.0}{average\ power}$$

at maximum power. This follows because the voltage and current inputs of the emonTx are a little less than and a little more than 1 V respectively at nominal voltage and maximum current.
These values could be transmitted as floating point variables inside the emonPi, but to do the same with the emonTx would require a significantly longer message when using the JeeLib library and the RF modules.

Therefore, it is proposed to multiply *A, B* and the corresponding rms voltage & current values by a constant, before transmitting them as signed integers as is customary. A suitable constant is 18432 (0x4800, lending itself to easy integer maths if necessary), which should be adequate for a 100% amplitude sinusoidal current wave at the input of a 5 V Arduino, (5 V p-p = 1.768 V rms, times 18432 = 32583)
and for the maximum UK voltage with the standard adapter (0.862 V rms at the input, 15888 counts)
and is also clearly satisfactory for real power at 1.768 V × 0.944 V = 1.669 V (30763 counts).
This will make the value of constant k_2

k_2 = Ical × Vcal ÷ 18432

Rms Voltage and Current.

Similarly, the voltage and current calibration constants respectively can be separated and applied in emonHub, while the sample count, ADC reference voltage and the ADC resolution are applied in the ATMega front-end processor.
In these cases, the appropriate scale factors respectively for emonHub will be

k_2 = Vcal ÷ 18432
k_2 = Ical ÷ 18432

The combined result of moving the ‘sensor-dependent’ calibration constants into emonHub is to remove all need for calibration from the emonTx, and from the ‘emon’ part of the emonPi, hence removing the need for a programmer, and removing the need to recompile and reload the sketch for the Atmel 328P.

Removal of ADC input bias voltage offset.

Initially, the offset arising from the input bias voltage was removed by a high pass filter. This, while effective, could not be satisfactorily initialised, with the consequence that a large spike would appear in the output as the filter settled, requiring the output to be inhibited for the first few sampling periods. This filter was replaced by straight arithmetic

subtraction of the offset, the actual offset value being obtained by a low pass filter, which could be initialised to the expected value. However, this still allowed a small degree of ripple to enter the readings.

However, it is not necessary to remove the offset immediately. If it can be assumed that the offset remains stable over the sampling period, it can be removed after sampling is complete. To do this in a practical manner, the nominal offset is removed by subtracting half the ADC count (simply to reduce the size of the numbers) and the average input value is obtained by totalling the samples and dividing by the sample count. The rms of the combined signal plus the residual offset is obtained in the usual way, by multiplying each sample by itself and accumulating the total. The true rms average is then obtained from

$$[rms\ of\ signal+offset]=\sqrt{(signal^2+offset^2)}.$$

This calculation is used for voltage and current. For the real power calculation, it is only necessary to subtract the product of voltage offset and current offset (the ‘offset power’) from the average power (the average of the instantaneous powers).

Additional Notes.

Default values for calcVI (number of crossings & timeout) and calcIrms (number of samples) allow these methods to be called without parameters if desired.
Should it be desired to calibrate the ADC supply measurement by the readVcc method (normally this is required and used only when a tightly regulated supply is unavailable), a ‘setter’ `setRef(unsigned long _vCal)` is provided. The default is the nominal value of 1126400.

The real UK voltage has been recorded with a crest factor (ratio peak/rms) = 1.38
The expected value is 1.414

Acknowledgements.

A major part of this work arises from suggestions made by @ursi (Andries) and @mafheldt (Mike Afheldt)
at https://community.openenergymonitor.org/t/emonlib-inaccurate-power-factor/3790 and
https://community.openenergymonitor.org/t/rms-calculations-in-emonlib-and-learn-documentation/3749/3