

Energy Monitoring Lamp

James Fotherby, February 2021

End Result



Introduction

The concept is simple, the lamp glows a colour that represents the instantaneous electricity usage of the house. Blue obviously means good things whereas red means you're an evil environmental sadist (or perhaps baking some muffins).

The inspiration behind this project was me recently moving into a flat entirely powered by electricity. It was winter at first with sub-zero temperatures outside and it quickly became apparent how expensive electric radiator heating can be. To be precise, 5 times more expensive than gas. But I also quite like the simplicity of just the one energy supply.

I actually built this exact lamp and concept when I was 18 but it's been sitting in a cupboard for years. I had lost the original transmitter unit and it ran using a Propeller microcontroller made by parallax. I have long since stopped using that particular MCU so have retrofitted the original lamp with an Arduino and a few fresh ideas.

Transmitter Unit

Most digital electricity meters have a little red light that flashes 1000 times per kWh used. Hence the rate of flashing represents the rate of energy usage (ie. the power consumption).

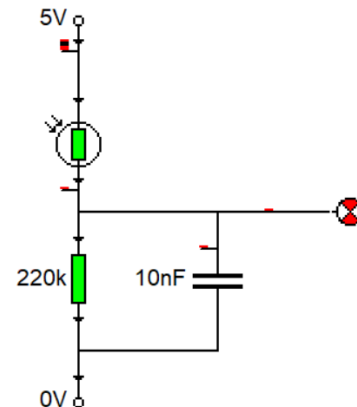


Figure 1 - Your classical electricity meter, the transmitter circuit for detecting flashes

It's simple enough to bluetack a photoresistor over this and write a simple program to run on an Arduino to detect the flashes. By using a RF transceiver unit we can broadcast these flashes to the neighbourhood. [Here's the transmitter code on Github.](#)

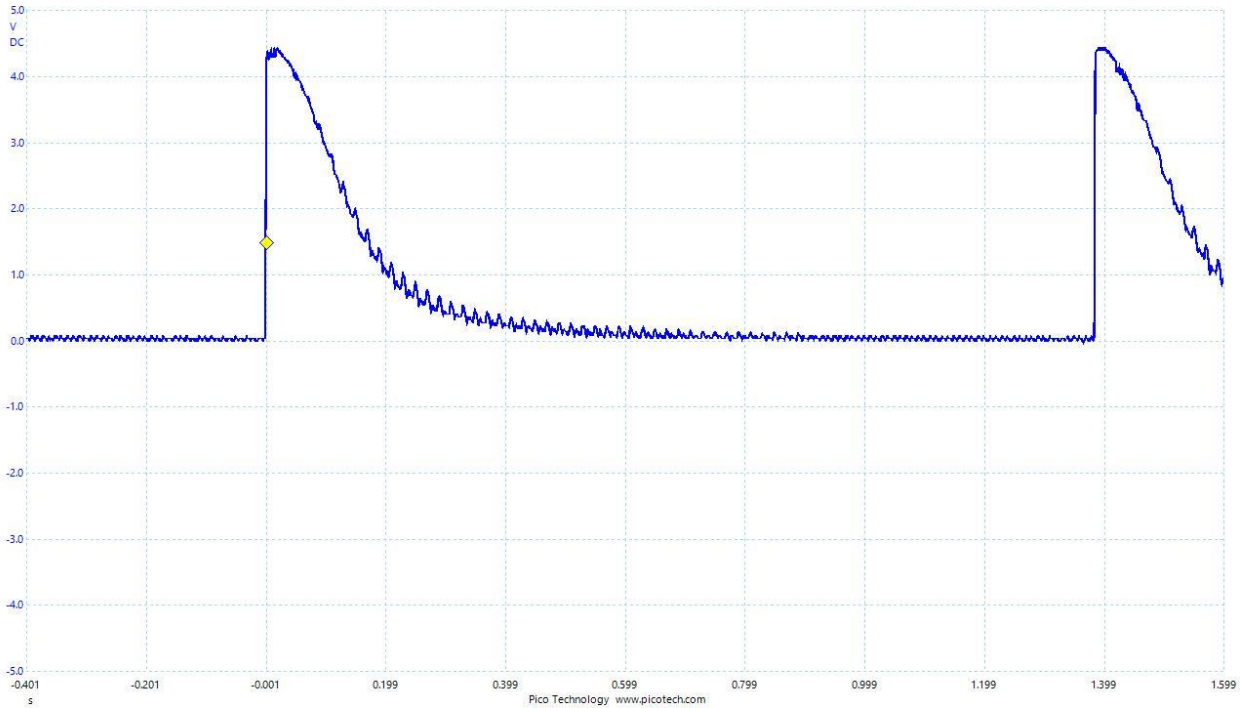


Figure 2 - Pulses as measured by photoresistor. Width ~200ms. At this moment I'm using 4.32kW of power as there is 833ms between the pulses (dishwasher + dryer)

I used a pair of [SiK telemetry transceiver units](#) which were super easy to interface with. They basically provide a wireless UART port and have a micro USB which I could plug a 5V power adapter into.

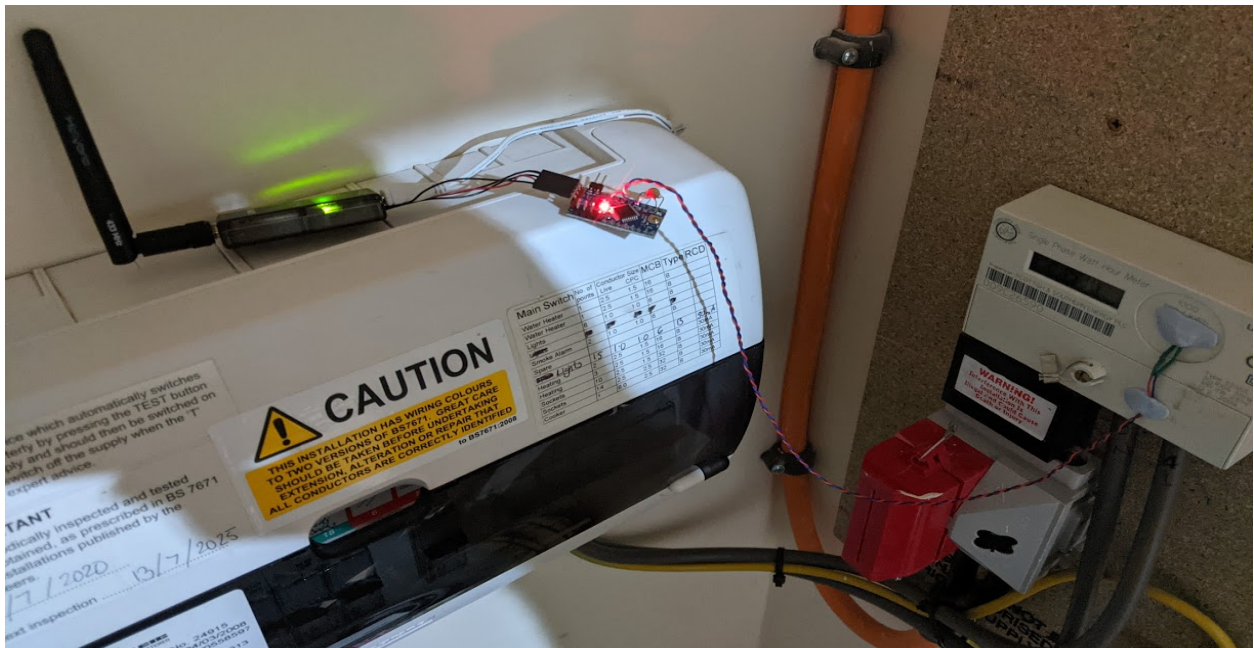


Figure 3 - Transmitter setup, photoresistor bluetacked to meter

Receiver Unit

The receiver unit has a couple jobs:

1. Time the interval between received flashes to calculate the power usage
2. Based on the current power usage, select a colour to glow (the hue)
3. PWM the RGB LEDs to produce the colour we want

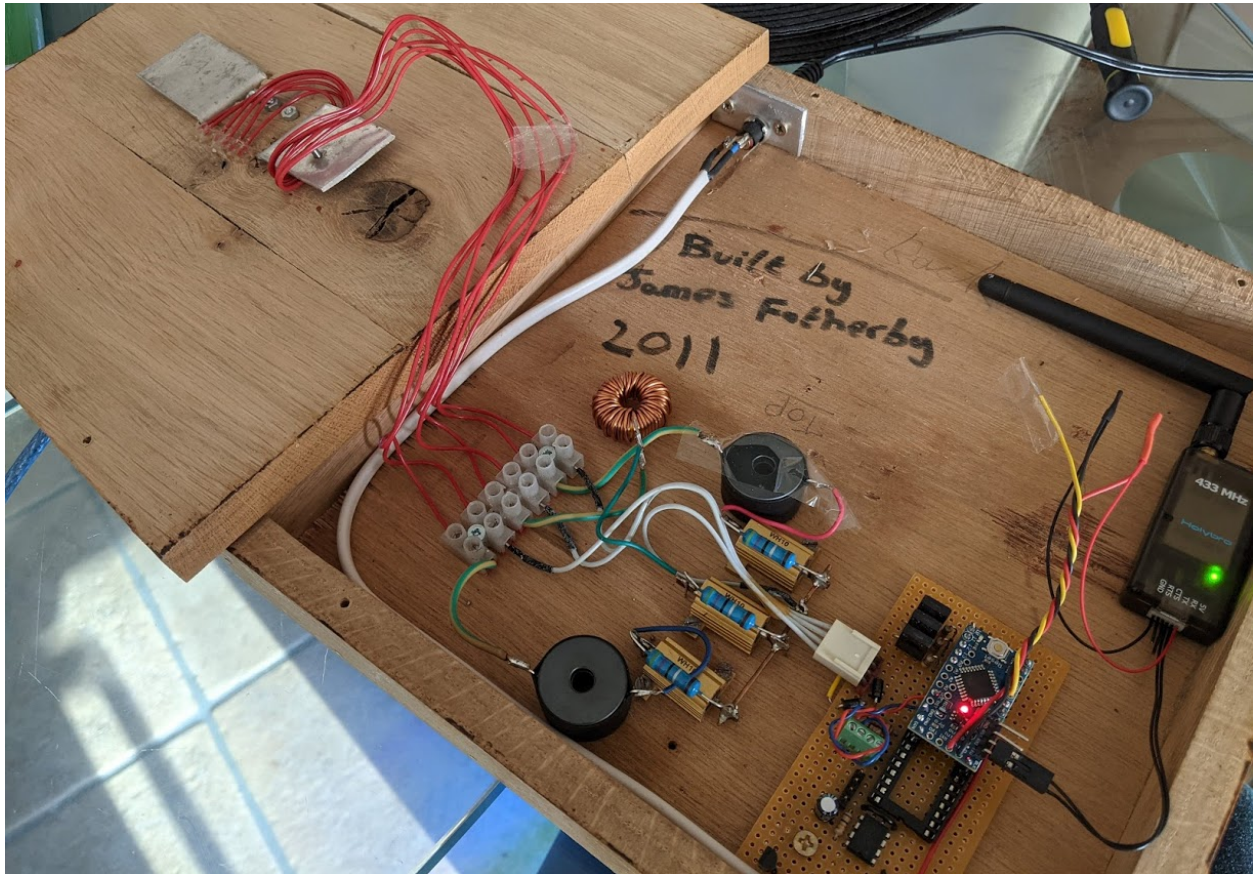


Figure 4 - Lamp innards - LED drive circuitry, transceiver and arduino pro mini (That chip in the bottom right corner is not connected to anything)

We use PWM at 62.5kHz to modulate the current through the LEDs. A buck converter circuit is employed for each LED. This is better than simply PWMing the LEDs directly from the 12v supply through a resistor because it's much more efficient.

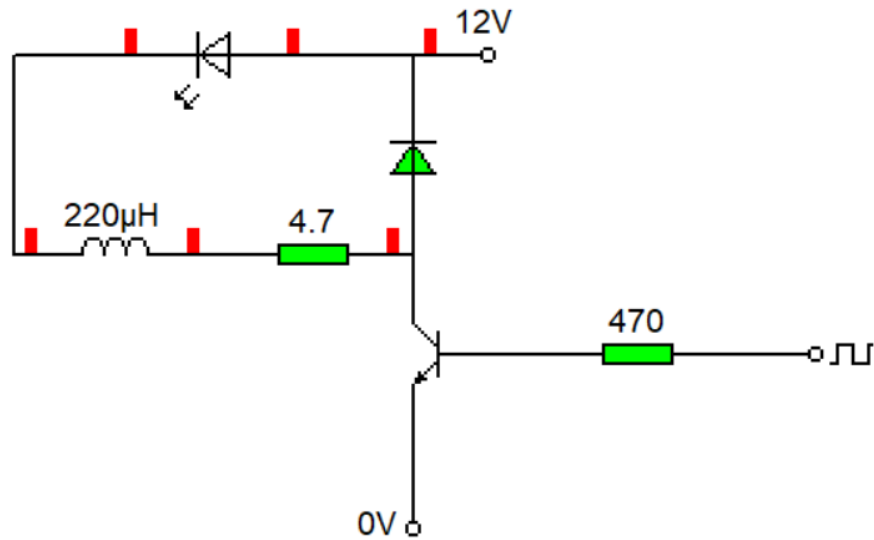


Figure 5 - Buck converter used to drive each LED

When the transistor switches on, the current through the LED ramps up due to the 220uH inductance. When switched off the current freewheels through the diode and the LED current decays. This gives a classical triangular waveform as shown in the figure.

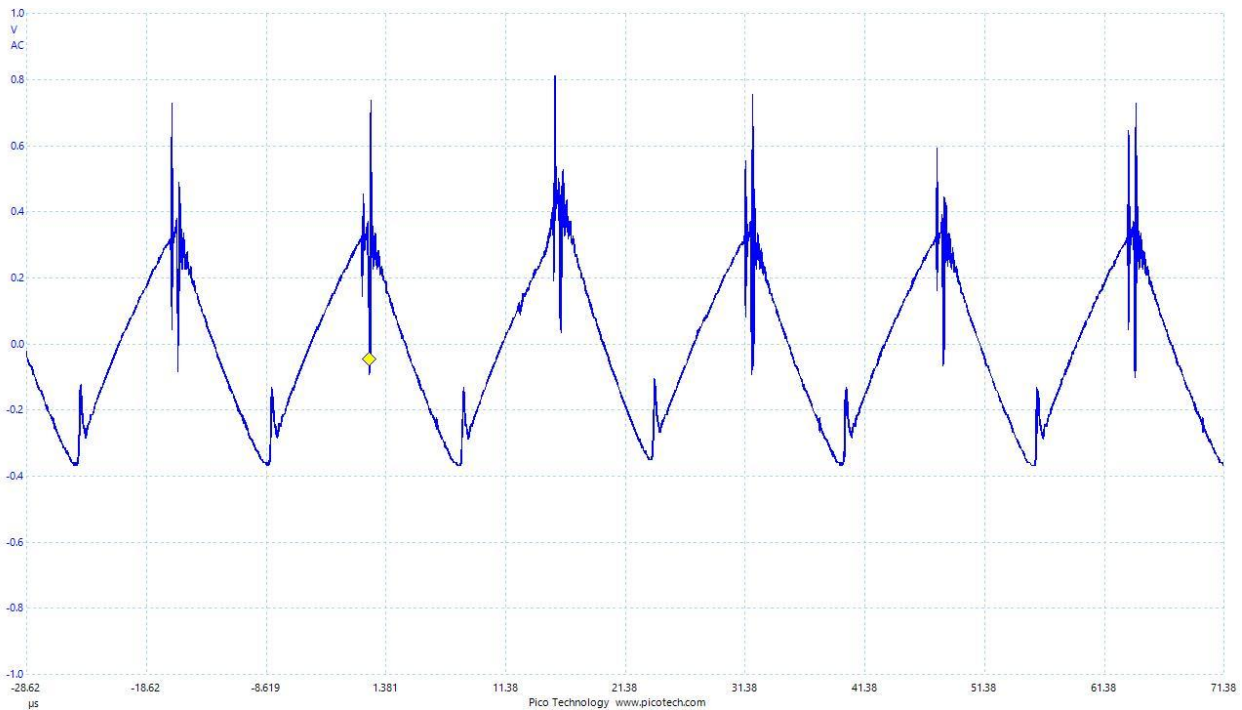


Figure 6 - Triangular LED drive current waveform (+ interference)

The 4.7R resistor is important. The reason is not immediately obvious but without the resistance it becomes difficult to predict the current for a given duty cycle. If we had a current sensor in the

loop we could adjust our PWM duty based on the measured current. However that adds a lot of complexity for little benefit over this setup.

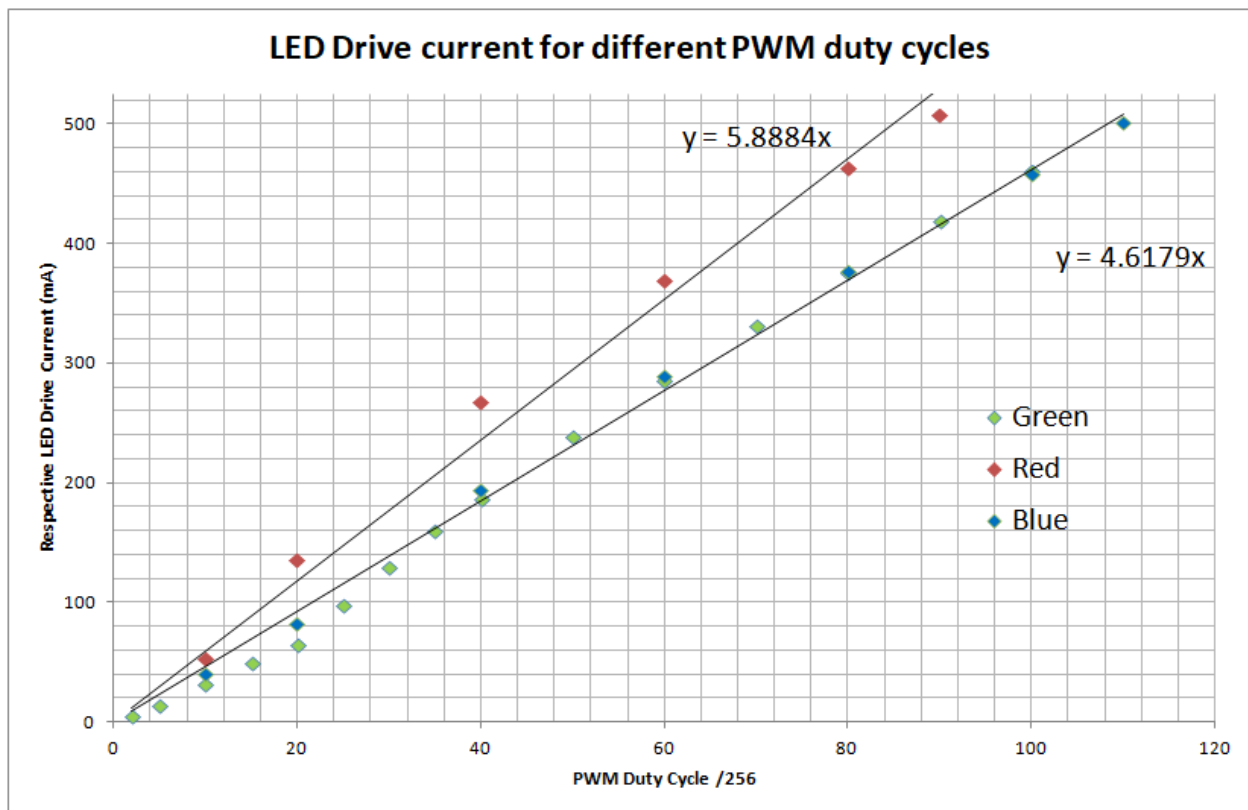


Figure 7 - LED drive currents for different duty cycles

The forward voltage drop of the LED is approximately 3V. With a 100% duty cycle the LED current would be $I = (12 - 3) / 4.7 = 1.9\text{Amps}$. I've used [these LEDs](#) which have a maximum current of 700mA. So it's important never to apply anywhere close to 100% duty!

I took current measurements for each LED against different duty cycles. These are plotted in the graph. We can see that at 60/256 (23% duty) we get a current of 365mA through the red LED and 285mA through the green and blue LEDs. The red LED has a smaller forward voltage drop accounting for the difference. If the 4.7R resistor was decreased it would steepen the gradient of these trendlines in the graph, smaller changes in duty cycle would result in larger changes in current which isn't good.

I've limited the LED duties to keep the maximum current below about 500-600mA. This is the absolute maximum rating specified on the datasheet.



Figure 8 - The lampshade is removable, under it sits some heatsinked LEDs

The software for the [receiver unit is on GitHub](#). A few things to say about it:

- I found [some code on Stackoverflow](#) to convert HSV (Hue, Saturation, Value) to RGB values. This makes it a lot easier to generate the colours. We always want the saturation (S) and brightness (V) to be maximum. This leaves just one variable (H) to adjust. Compare this against an RGB setting. What values of R, G and B do you need to make yellow? How do you gradually adjust these values to transition from orange to red!? The HSV colour system makes things straightforward.

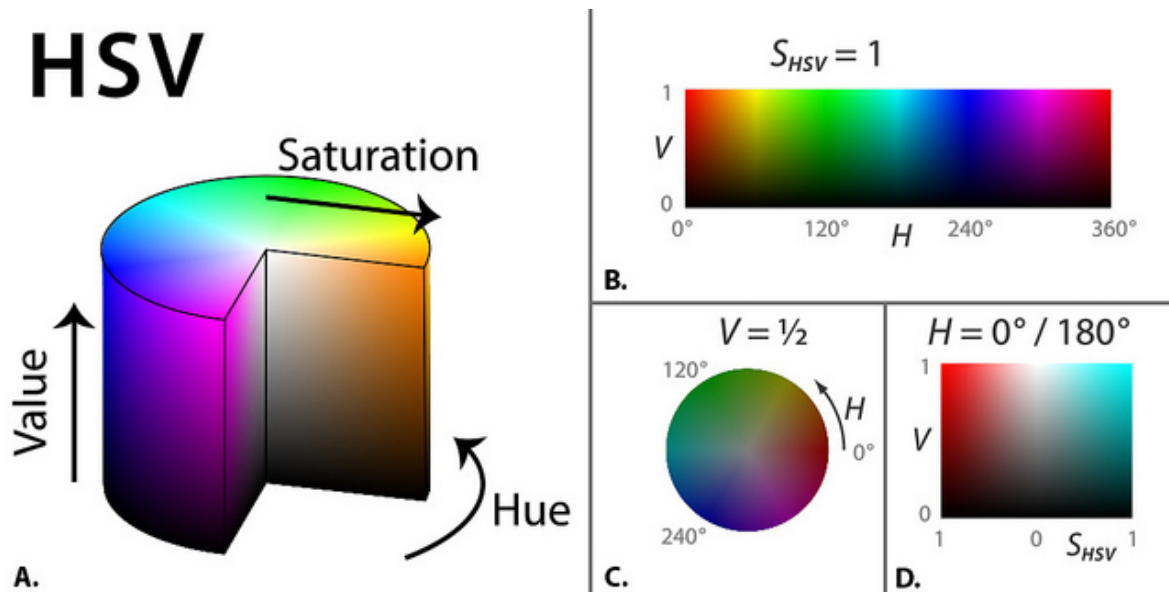


Figure 9 - The HSV colour scale

- What colour do we choose for what power? Initially I tried a linear scaling however I later decided a logarithmic scale would be better. I wanted to see the difference between 50W

and 100W whereas I wasn't so bothered about 5000W and 5050W. I felt like it'd be better to have higher resolution at lower power usages and I'm pleased by the results.

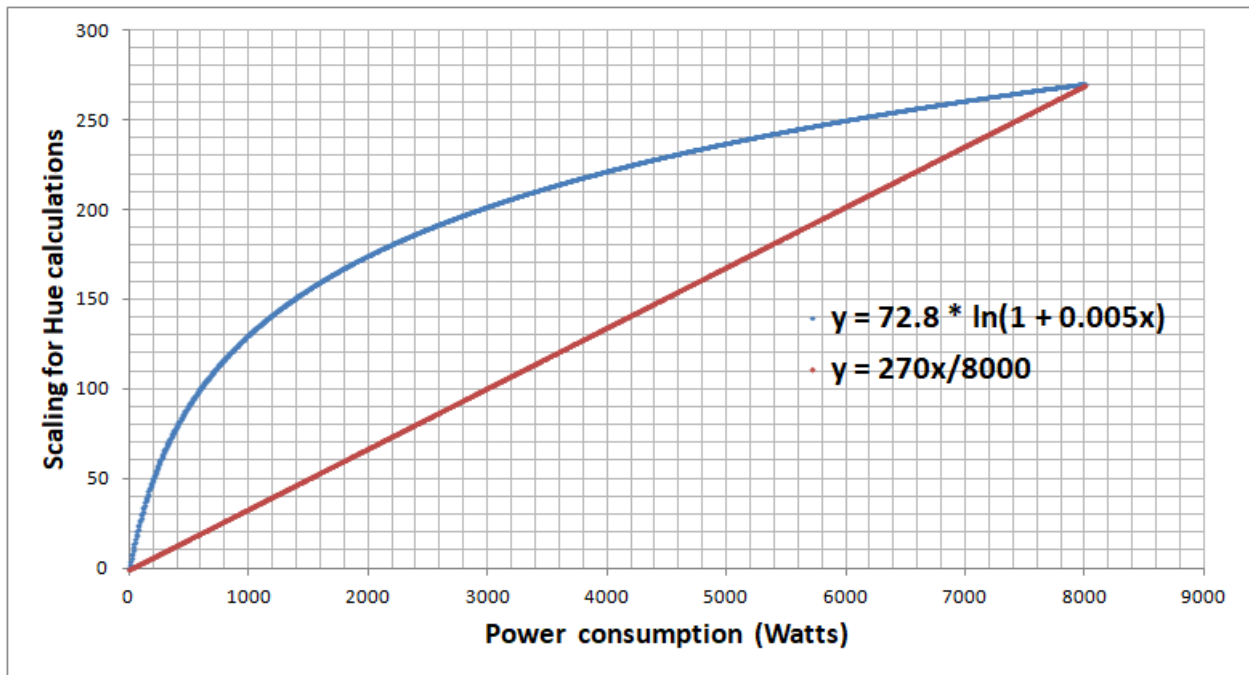


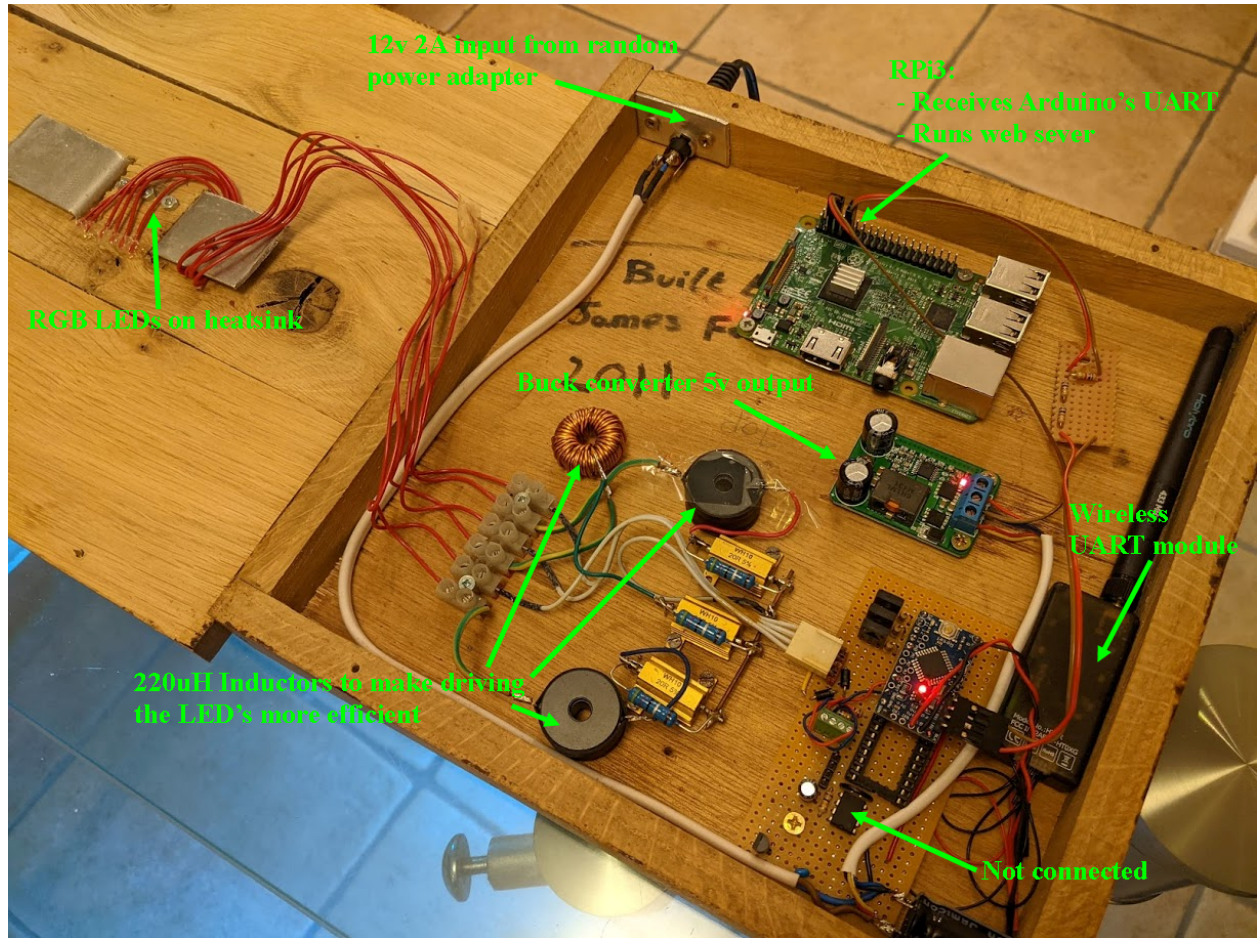
Figure 10 - Linear vs Logarithmic scaling for colour choice

Future Ideas

I'd like to incorporate a Raspberry Pi in the lamp to upload the electricity usage throughout the day to a web server. Then it'd be possible to record the usage and display it as a graph. That wouldn't be too tricky...

Update March 2021 - Raspberry Pi

I enjoyed working on this update. I have configured a RPi3 to receive the flash counts from the Arduino over UART. I used a simple resistor divider to level shift the arduino's 5v signal to a 3.3v one suitable for the Pi. There's also a buck converter to produce 5v for the Pi from our 12v input.



The Pi logs the energy used during each minute of the day to a .txt file. It runs apache and some PHP to serve up the data in the format of a graph.

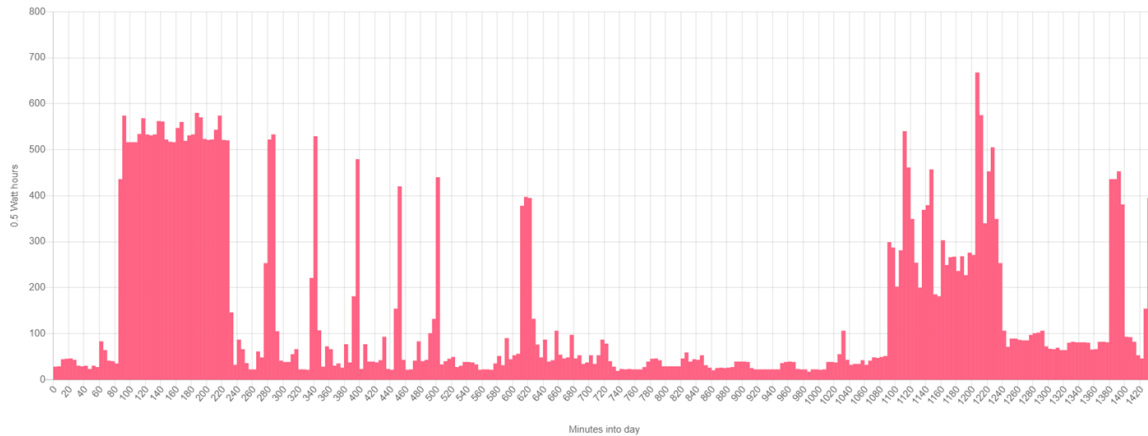
Flat 159 Energy Monitoring

Day

<

>

Fri Apr 09 2021



Last 5 minute average: 2358 Watts, Today's Total: 21.57 kWh, Cost: £3.68

Figure 11 - The webpage used to display the energy used throughout the day

I'm always so rusty when it comes to writing web pages using HTML/CCS/Javascript/PHP so I rely heavily on examples, Stack Overflow and www.w3schools.com. I thought I'd try to document how it all works. I don't claim it to be a masterpiece - I'm no web developer.

- There's a [python daemon](#) that is configured to automatically run on Pi startup. I did this using cron by following [this handy guide](#).
 - The python daemon is very simple. It waits for serial input from the arduino and writes the flash count (ie. energy used) each minute to a date stamped text file. A new file is created each day and will gradually fill with $60 \times 24 = 1440$ entries.
- Stored in the `/var/www/html/` directory are the web server files:
 - [Index.html](#)
 - Renders the page using the CSS in `style.css` and adds the javascript
 - [style.css](#)
 - [Z1.js](#)
 - This uses `XMLHttpRequest()` to grab data from the server for the day of interest. It then uses [chart.js](#) to plot the data nicely
 - [Data_parser.php](#)
 - When data is requested by the client for a certain day, this reads the respective log (created by the python daemon) and packages the data ready for transmission

Final bits and bobs

I had a smart meter installed and it has 2 IMP lights next to each other. One 2000/kWhours and one for 2000/kvarh. Initially I had some strange behaviour. I then realised my photoresistor setup was too sensitive and capturing flashes from both IMP lights because they were close to one another. I switched the pull down resistor in the Tx unit for a 47K one and that fixed the “cross talk” issue.

