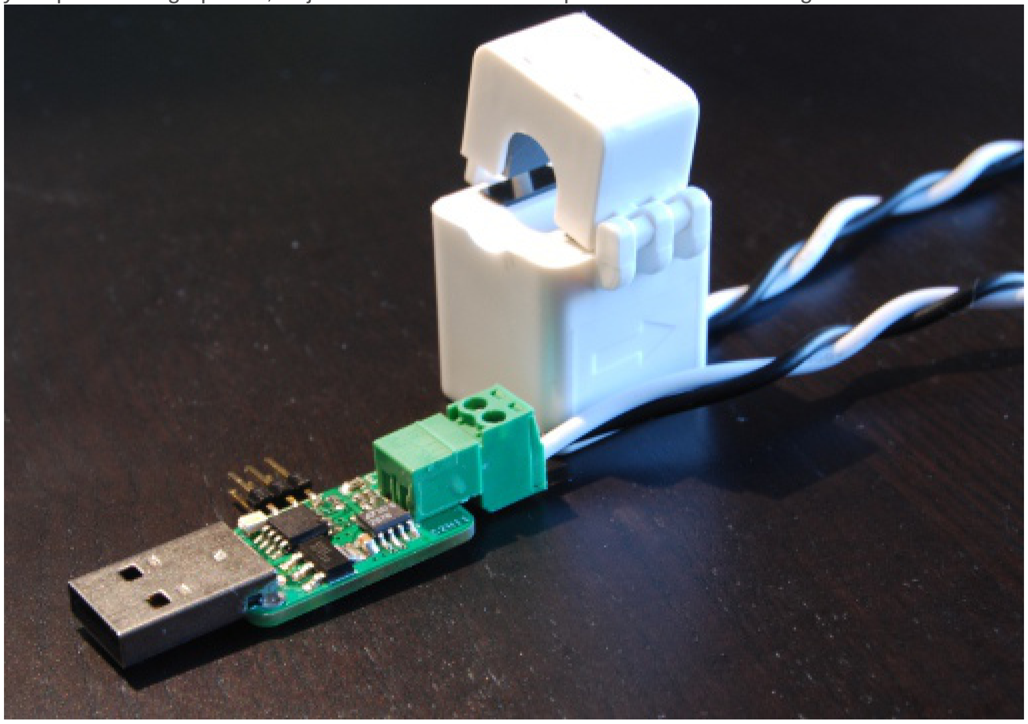
USB CURRENT METER

…or USB power meter?

…or USB KEY AVR Tiny split core interface?

Call it however you want, this project is a small USB key sized circuit to interface an USB system with a single split core current sensor using an ATtiny85.

These non-invasive sensors are widely available on eBay and similar for a reasonable price (around USD$ 30 for the one I used) and let you measure the current flowing through an alternate voltage line, like house mains. This can be used to get a gross measure of instantaneous power consumption, allowing you to make a graph out of it and plot your power usage profile, or just to check how much power a device is using.



A friend of mine is using this one side-by-side with its photovoltaic inverter to upload his data on pvoutput.org, a nice website for photovoltaic plant monitoring.

This also shows a rectifier circuit to acquire data from an alternate voltage source without dual power rails.

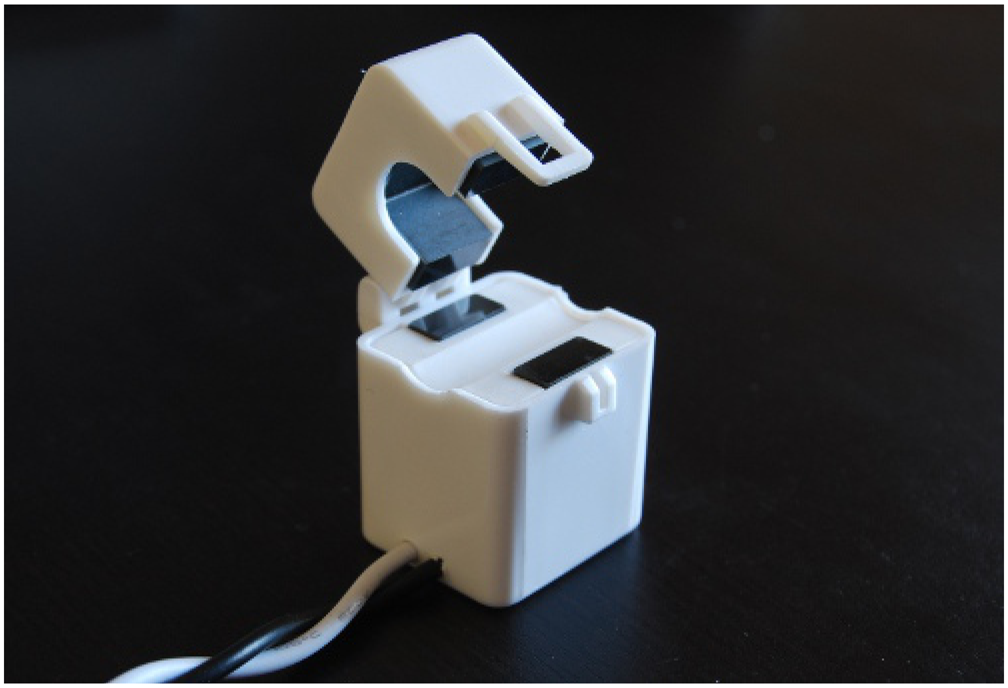
Split Core Sensors

The transducer used for the project is a split-core current sensor. This is basically a current transformer, with a splittable ferrite ring for non-intrusive insertion in a live system (but be careful anyway, ok?), simlarly to a current clamp. Of course the output is isolated from the input.

As the sensor is actually a current transformer, it can only work with AC loads, and output the signal as an AC current. To read the sensor from an ADC, the sensor have to be loaded with an high precision resistor to obtain an alternate voltage output.

The sensor I'm using is similar to this 20A Mini CT from Dent Instruments, available both in the 20A and 50A range with a voltage output of 0.333V at the nominal measured current.

Note that the sensor range is nominal, as the 20A version works with currents from 0.25 to 40 Amperes. Also, this sensor has an integrated load resistor, so that the output is already in volts.



This mini CT sensor is fairly well suited for installation in an Italian (European?) home's electrical system, as the "standard" home contract is 3.3kW at 220V, allowing loads up to 15A. For 110V systems you may want to use a different sensor… just do the math!

Also, keep in mind that you can always do more than one pass through the sensor to get a better accuracy, reducing the range.

Rectifier Circuit

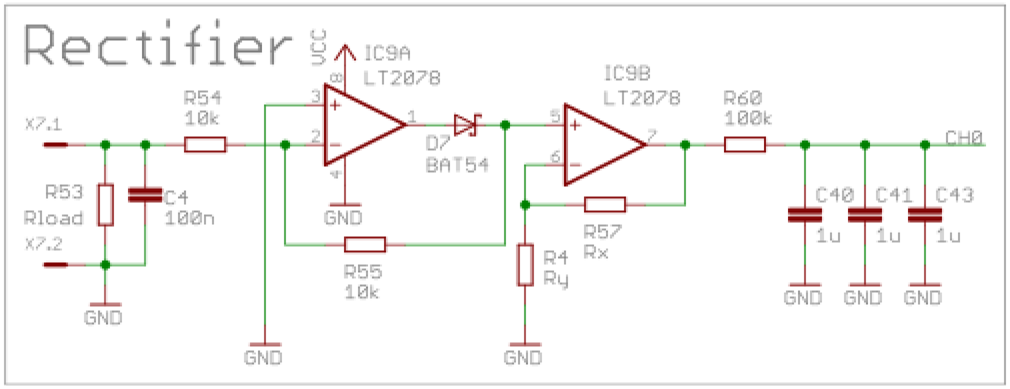
A loaded CT sensor produces an alternate voltage proportional to the input. To measure the signal from a microcontroller there are two possibilities:

- Bias the signal to make it oscillate in the ADC range, acquire the actual waveform and calculate the RMS value using the samples;
- Use a rectifier and filter circuit to get the signal to a more suitable range and measure it directly from the ADC;

While the first solution is nice because also allows to measure the actual frequency and waveform distorsion of the signal, it requires some smarter logic in the firmware and is susceptible to error in the biasing circuit (zero calibration) and acquisition period.

On the other side, the rectifier solution does not allow additional measures, but is easier to use, and does not require a bias circuit on the sensor.

This is the actual rectifier circuit, designed using this appliation note from CR Magnetics.



Note that the circuit does NOT use a dual power supply.

The working principle of this circuit is quite easy:

- the sensor is actually biased at 0V, so that the output oscillates between +Vpeak and -Vpeak, the input load resistor is needeed only for sensor without internal resistor, and the input capaitor can be used for initial filtering.
- the first OPAMP is the actual rectifier – it's output is the same as the input for positive signals (as the diodes is reversed) and an inverted replica of the input signal for nevative inputs (as the diode is in forward). Note that this stage has different input impedance for positive and negative signals.
- the second OPAMP is actually a plain amplifier stage, and has to be tuned to output an average voltage of 1.1V – the actual ADC full-scale – at the desired maximum input current.
- the final stage is a simple RC filter, tuned to with a sufficiently high time-constant for a 50Hz signal.

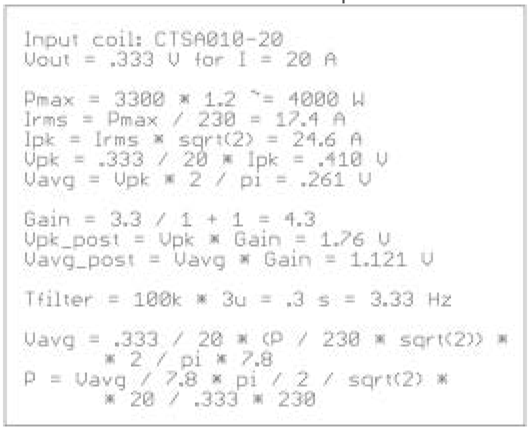
One important aspect of this circuit is that not every OPAMP is capable of working with its input positive terminal grounded, so checkout the OPAMP datasheet and chose a suitable one, such as the LT2078.

Gain and Filtering

Tuning calculations are wrote in the schematics, but this is the basic idea behind it:

- The tuning has to measure input power up to 4kW for a 230V system, so a maximum **RMS** current of 17.4A is archieved;
- counting the sensor, that would give us a maximum **peak** voltage of .410V, and an **average** voltage of .261V;
- the amplifier stage has to be tuned high enough to get an accurate post-filter average measurement, but not too high to saturate at peak values; this tuning uses a gain of 4.3, so that the required 4kW input gives out a 1.121 voltage average output.
- the output filter has to have a time constant much higher than the 50Hz input signal, so an 100kOhm, 3uF RC circuit is used (.3s time constant).

Refer to the schematics for complete calculations.


```
Input coil: CTSA010-20
Vout = .333 V for I = 20 A

Pmax = 3300 * 1.2 ~= 4000 W
Irms = Pmax / 230 = 17.4 A
Ipk = Irms * sqrt(2) = 24.6 A
Vpk = .333 / 20 * Ipk = .410 V
Vavg = Vpk * 2 / pi = .261 V

Gain = 3.3 / 1 + 1 = 4.3
Vpk_post = Vpk * Gain = 1.76 V
Vavg_post = Vavg * Gain = 1.121 V

Tfilter = 100k * 3u = .3 s = 3.33 Hz

Vavg = .333 / 20 * (P / 230 * sqrt(2)) *
       * 2 / pi * 7.8
P = Vavg / 7.8 * pi / 2 / sqrt(2) *
       * 20 / .333 * 230
```

ATtiny USB Interface

Interface with the host system is handled using the V-USB soft-USB library, implemented on an ATtiny85 microcontroller.

The protocol is two simple control requests:

- CUSTOM_RQ_SET_MODE: set the acquisition mode to automatic or single-shot
- CUSTOM_RQ_GET_VALUE: return the measured value, in Watt, normalize for 230V

The actual adc routine exploits the availability of a 20x gain stage in the ATtiny MCU, so that we can get a more precise value for low signals.

That’s the actual routine:

```


#define div_round(a, b) (((a) + b)/2 / (b))
static uint16_t get_power(void)
{
        uint32_t value;
        uint16_t offset;
        uint8_t gain;

        led_a_on();

        offset = 0;
        value  = adc_get(ADC_COIL);

        if (value < AMP_TH) {
                offset = adc_get(ADC_OFFSET_20X);
                value  = adc_get(ADC_COIL_20X);
                gain = 20;
        } else {
                gain = 1;
        }

        if (value < offset)
                value = 0;
        else
                value = div_round((value - offset) * ADC_VREF_mV *
                                  CAL_POWER,
                                  ADC_VREF_BITS * CAL_VOLTAGE *
                                  (uint32_t)gain);

        led_a_off();

        return value;
}
```

As arithmentics is done in fixed point, constants has to be tuned to prevent overflowing, as division is made as the last operation to not loose granularity.

That’s why the actual constant are defined as scaled in `board.h`:

```
#define AMP_TH 45 /* ~ 1024 / 20 */
#define ADC_VREF_mV (1100 / 4)
```

```
#define ADC_VREF_BITS (1024 / 4)

```

Command-line Utility

On the host side, a simple `power` commmandline utility allows to read from the device.

Without arguments, a single reading is requested and the result is wrote in standard output:

```
balto@balto-eee:~/usb-current-meter/commandline$ ./power

132
```

The utility implements some options to change device mode, reset the device and normalize power reading for different mains voltage.
Just run with the `-h` flag to see the help text:

```
balto@balto-eee:~/usb-current-meter/commandline$ ./power -h

Usage: ./power -h

       ./power -R

       ./power [options] divisor

options:

  -h         this help

  -R         reset device

  -b         run in background

  -d delay   delay between updates

  -m         set mode (0=normal, 1=auto sampling)

  -r dbname  rrdtool update mode on dbname db

  -V voltage scale output for real voltage instead of nominal

  divisor    power divisor, correspond to number of turns on the sensor
```

RRDtool Graphs

One nice feature of the commandline utility is that it can be used directly to interface with rrdtools and load data to an RRD database.

First, the DB has to be created with:

```
#!/bin/sh

DB=power.rrd

```

```
# 5 minutes points, 96 hours data
# 30 minutes points, 25 days data
# 2 hours points, 2 months data
# 24 hours points, 2 years data

rrdtool create $DB \
  DS:power:GAUGE:600:U:U  \
  RRA:AVERAGE:0.5:1:1152  \
  RRA:AVERAGE:0.5:6:720   \
  RRA:AVERAGE:0.5:24:720  \
  RRA:AVERAGE:0.5:288:730
```

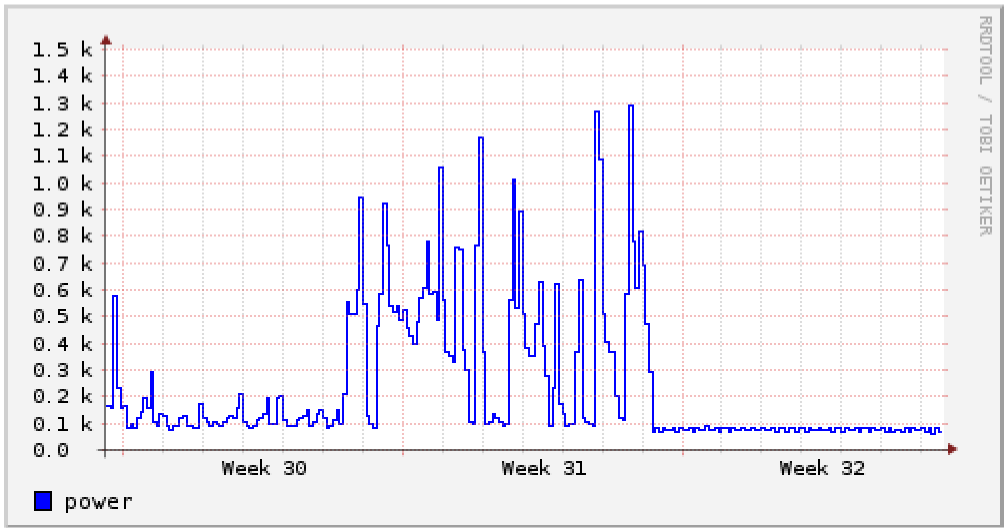
then, just run the `power` utility with the `-r` flag, as in:

```
balto@balto-eee:~/usb-current-meter/commandline$ ./power -r power.rrd
```

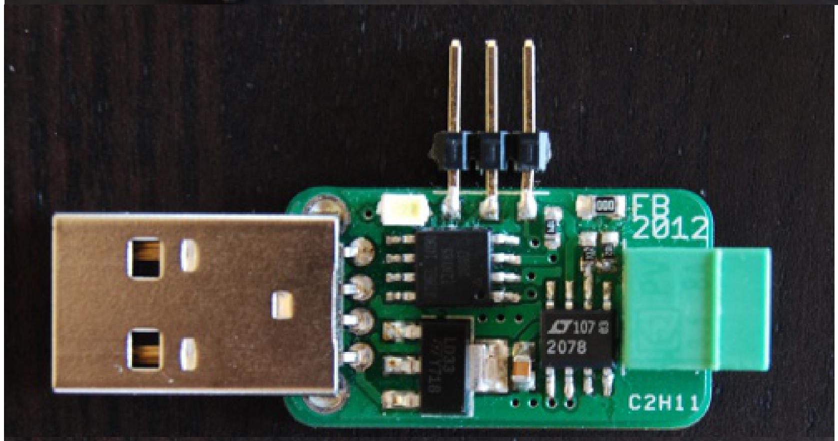
That’s going to update the DB automatically at about 10Hz.

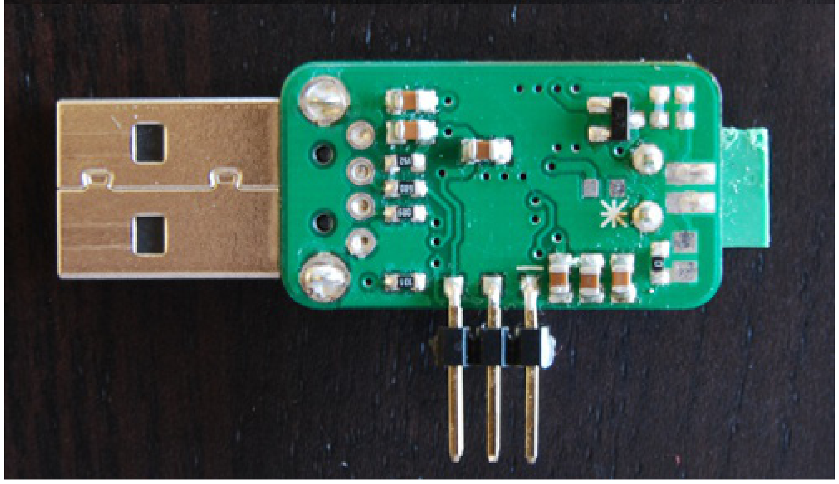
Once the DB is populated with data, a graph can be generated using normal rrdtool commands, as an example this is the power consumption of my house for 4 weeks.

```
#!/bin/sh

DB=$HOME/power/power.rrd
RRDTOOL=rrdtool
OUT="/opt/lighttpd/www/htdocs/rrd"
OPTS="-w 419 -h 200"

$RRDTOOL graph $OUT/power.png $OPTS -l 0 --start -3w \
  DEF:power=$DB:power:AVERAGE \
  LINE1:power#0000ff:power
```



From the graph you can observe a standby power of about 100W, where the on-off pattern of the last week was just the fridge turning on an off.





Source : http://fabiobaltieri.com/2012/08/09/usb-current-meter/