

## **Development of a user-friendly, low-cost home energy monitoring and recording system**

James Fletcher and Weeratunge Malalasekera<sup>\*</sup>

Wolfson School of Mechanical and Manufacturing Engineering

Loughborough University, Loughborough, Leics LE11 3TU

UK

\* Corresponding author email: W. [Malalasekera@lboro.ac.uk](mailto:Malalasekera@lboro.ac.uk)

## Abstract

This paper reports research undertaken to develop a user-friendly home energy monitoring system which is capable of collecting, processing and displaying detailed usage data. The system allows users to monitor power usage and switch their electronic appliances remotely, using any web enabled device, including computers, phones and tablets. The system aims to raise awareness of consumer energy use by gathering data about usage habits, and displaying this information to support consumers when selecting energy tariffs or new appliances.

To achieve these aims, bespoke electrical hardware, or '*nodes*', have been designed and built to monitor power usage, switch devices on and off, and communicate via a Wi-Fi connection, with bespoke software, the '*server*'. The *server* hosts a webpage which allows users to see a real-time overview of how power is being used in the home as well as allowing scheduled tasks and triggered tasks (which respond to events) to be programmed. The *system* takes advantage of well standardised networking specifications, such as Wi-Fi and TCP, allowing access from within the home, or remotely through the internet. The *server* runs under Debian Linux on a Raspberry Pi computer and is written in Python, HTML and JavaScript. The *server* includes advanced functionality, such as *device* recognition which allows users to individually monitor several devices that share a single *node*. The openPicus Flyport is used to provide Wi-Fi connectivity and programmable logic control to *nodes*. The Flyport is programmed with code compiled from C.

## 1. Introduction

The management of electrical energy supply to meet the growing demand without contributing more to the green-house gas problem is a major task. To reduce green-house gases, more and more renewable sources of electrical energy are being added to networks. However, these new sources of supplies can be intermittent and variable. For example, analysis of the variability of wind power and how it could be integrated into electrical markets is detailed in [1]. This report discusses the main issues in terms of grid management and technological options required to address the variability of wind and other renewable sources and highlight the importance of developing technical solutions concomitant to the growth of intermittent renewables to ensure electricity system stability. Bass et al [2], using measured data from a gas turbine power generation plant in the UK, have shown how intermittent and variable sources of energy could impact on traditional power generation. Future management of electricity supply networks thus requires energy storage systems, variable tariff systems and cooperation from electricity consumers to limit low-priority power usage in times of high demand on the grid. In this context smart grid technologies are likely to play a major role. An electricity system, that is efficient, reliable, resilient and responsive, is a smarter grid [3]. Such a system would include an advanced metering system with the ability to use electricity more efficiently [3]. Electricity smart grids can be part of an overall smart energy system that include flexible generation systems and fluctuating renewable sources [4]. Home energy management is an important part of the solution to reduce greenhouse gases and incorporation of renewable sources of energy. For the success of the smart grid, home energy management and modelling systems with data recording facilities are also required. A number of recent studies have considered systems, software and hardware for home energy management. In the context of smart grids, a day-ahead real time pricing (DA-RTP) model based on smart meter data has been discussed in [5] and the study shows that a flatter demand curve and lower losses can be achieved by using such a system. The study by Almos et al [6] provides an analysis to illustrate the change of domestic load based on smart meter energy actions and the engagement of consumers is discussed in [6]. The development of an energy management system that incorporates consumer owned smart meters and distributor owned smart meters is described in [7] where applicable hardware and control systems have been demonstrated in a smart grid context. An approach that could be used to automatically analyse smart meter data to identify consumer characteristics has been presented in [8] where house hold characteristics could be used by suppliers to target energy efficiency campaigns. Depuru et al [9] have discussed in detail various features

and technologies that can be integrated with a smart meters with the emphasis on security issues required in smart meter communication networks. Vega et al [10] have reviewed the most relevant literature and summarised infrastructure, communication media – protocols and variables managed by a system. Their study also analysed a large range of energy management models associated hardware and software and emphasised the need to incorporate monitoring, control and supervision of home appliances as well as the need to incorporate communication protocols to ensure the reliability of real time information collection through modern sampling tools and advanced algorithms. In other studies published in the literature, smart meter techniques have been discussed in details. For example, various features and technologies that can be integrated with a smart meters has been discussed by Depuru et al [9]. Vessileva et al [11] have discussed the need the need for providing energy consumers with required information about their energy usage and studied the consumer categories and how it could be used to develop effective demand response measures. Further relevant studies include the work of Beckel [8], Pereira et al [7] and Zhao et al [12]. At a higher level (supplier or distributor), the smart grid concepts appear be a good solution for integrating variable and distributed energy sources. An overview of the emerging smart grid and the potential for the smart grid to act as an enabling technology for renewable energy integration, price-responsive electricity demand, electrified transportation and distributed energy production is discussed in [13]. User mode distributed energy management via smart grids [14] and potential use of information and communication technologies (ICT) with regards to smart grids are discussed in [15]. At a domestic and consumer level smart meters have emerged as the technological solution providing necessary data for the incorporation of renewable energy systems, monitoring and control of metering technologies and demand management (Batista [16]). Availability of reliable, low cost flexible hardware and software solutions capable of metering, data collection, data analysis, storing and transmission are required for the success of the smart grid. Kim et al [17] have discussed the overall structure of a smart grid and the role of Universal Plug and Play (UPnP) devices for Home Energy Management Systems (HEMS). Most other studies relevant to the present study have used IEEE 801.15.4 and ZigBee technologies for device control and energy management (Han and Lim [18], Batista et al [16]). Jang and Healy [19] in their studies have discussed in detail the challenges and obstacles in the implementation of wireless sensor networks in buildings. The main challenges are building structure, reliability degradation, security, battery life time, initial cost and ease of use. Some energy management systems have explored the use of Power Line Communication (PLC) technologies for data collection and recording in

home energy management. These include the work of Han et al [20], Papagiannis et al [21], Al-Mulla and ElSherbini [22]. Further details are available in the review of powerline technologies for smart grid applications by Yigit et al [23].

At consumer level various forms of smart meters have been introduced by electricity suppliers to inform customers of their energy usage profiles and these have enabled the suppliers to obtain useful demand profiles. However some smart meters do not provide a breakdown of which appliances contribute to the overall demand profile. This has been a problem and smart meters are already termed to be 'not so smart meters' [24]. There is great interest in developing useful, secure, low-cost technologies to monitor and record domestic energy usage and allow users to have more control over their energy usage. Perhaps in the future such systems may even be able to automatically switch energy provider on their owners behalf, depending on which companies offer the cheapest, greenest energy. The technology could also be used to deliver targeted advertising to users supported by savings estimates based on the real usage data that the system collects.

None of the existing 'smart meter' solutions offered by energy suppliers to private homes in the UK are able to control devices around the home. Input into the National Grid in the UK is almost entirely comprised of plants which are unable to quickly change their rate of supply, (for example, gas, coal, nuclear) or worse, have an unreliable rate of supply e.g. wind. Since mass produced electricity is not easily stored, the grid must attempt to balance supply with demand by forecasting demand. Demand trends typically follow the day/night, weekday/weekend and summer/winter cycles, but are also affected by social events, such as breaks during soap operas and sport events. If the grid could temporarily enable and disable high-power, non-time critical devices around the home, such as water heaters and air conditioners then they would be more effectively able to balance supply with demand, whilst avoiding resorting to inefficient energy storage or 'brown-outs'.

Appliance monitoring and control systems have the potential to reduce costs, energy consumption and carbon emissions [25]. They can achieve this by providing sophisticated control functionality, and by collecting useful data that can be used to inform appliance purchase decisions and select more economical energy providers. Energy providers can use the data to provide low cost tariffs and in an ideally open market-place for electricity-supply, users should be allowed to switch suppliers at a short-notice to take advantage of low cost tariffs and change their usage profile to minimise their energy costs.

As mentioned above most currently available 'smart meters' are able to collect data but consumers could not identify elements of their energy usage or analyse data to make useful interventions to change their behaviour. In many cases smart meters show a cost at a particular time but do not provide information on which appliances are responsible for the cost or collect item specific data and present them in a useful manner to understand their energy profiles. There is a requirement to develop low-cost smart meters which can collect detailed data, communicate with simple devices such as mobile phones, tablets and computers at home and consumers should be able to see energy usage, broken down by appliances and time of use so that they can make changes in their behaviour to use energy more efficiently or buy new and energy efficient appliances to save energy on long-term. This project aims to develop a low cost secure hardware and software solution, making as much use of existing hardware as possible, which can be used to monitor and control mains electrical *devices*, at the same time collecting information about the way energy is used in the home. The information collected aims to be of sufficient quality to help users make informed decisions about potential appliance purchases and energy tariffs based on the user's actual usage habits.

## 2. The Methodology and System Design

### 2.1 Structure of the system

The main aim of this research is to develop a system that is capable of identifying where energy is used around the home, as well as storing and presenting the data collected in an accessible and visual way using webpages which can be accessed remotely from phones and computers. The system also aims to allow users to switch *devices* around the home on and off through the web interface provided. The system consists of a single *server* and one or more *nodes* which monitor and control one or more attached *devices*. The overall system and its operation is shown in Figure 1.

Figure 1. An illustration of the overall system concept

The *system* consists of a single *server* which is in communication with multiple *nodes* and multiple *clients*. In the context of this research, a *device* is any mains electrical appliance

that can be controlled by this *system*, for example desk lamps, white goods, fans or televisions etc.

A *node* is the hardware which physically switches the mains power to attached *devices* at the request of the *server*. The *node* also monitors the power usage of the *device* and sends this information to the *server*. A home can have many *nodes* in multiple formats, for example a unit that sits between the wall socket and *device* plug, circuitry concealed behind a wall switch or circuitry physically built into an appliance.

The *server* runs permanently in order to receive information from *nodes* and issue commands to *nodes*. The *server* also hosts a web interface capable of serving *clients*, manages scheduled tasks and events, and records power usage and user activity by submitting records to a database.

A *client* allows users to issue commands to the *system* such as turning *devices* on or off. *Clients* can also query usage data and configure rules. Since the *server* hosts a web interface, *clients* can take the form of any device capable of connecting to a wired or wireless network and displaying HTTP or HTTPS webpages. Serving *clients* via a web page as opposed to an application (app) releases any dependency of the *system* on a particular *client* operating system, such as Google Android, Apple iOS or Microsoft Windows. These operating systems would all require dedicated versions (using different languages) of any bespoke *client* software developed for this project, but all support web content by default. Using the display and connectivity features of common-place devices (such as laptops and smartphones etc) eliminates the need to build custom *client* hardware at additional project and end user cost. A detailed system layout which depicts above features is shown in Figure 2.

Figure 2 Detailed system layout

## 2.2 Device Recognition Theory

Often in the home, multiple *devices* share a single plug socket via a multi-socket extension. In such scenarios each *device* would require an individual *node* to monitor the power usage. It is theoretically possible to identify and monitor multiple *devices* attached to a single *node* by analysing and remembering the way *devices* use power.

This is permissible because the electrical hardware within *devices* around the home is made up from different combinations of resistive, inductive or capacitive components. These components can draw current at a different phase to an input Alternating Current (AC) voltage. Those that do not follow the input voltage (resistive) either lag (inductive) or lead (capacitive). When their current draw waveforms are combined it yields a unique overall current draw waveform for that *device*.

Figure 3 shows the current waveform from a fan-cooled overhead projector next to that of an assortment of audio/visual equipment including a video tape player, an amplifier, desktop computer and a monitor.

Figure 3. Measured current transducer waveform for two devices

Despite the fact that both *devices* are supplied with the same 230 VAC waveform, their current draw is significantly different. The overhead projector is the simplest *device*, and with no sophisticated control electronics, only appears to show two distant frequency components. The light in the projector is a resistive load (in phase with the mains at 50 Hz), whereas the fan is an inductive load (lagging behind the mains). The audio visual (AV) equipment however, contains many hundreds of resistors, inductors and capacitors and therefore displays a much more erratic waveform, very different from the sinusoidal AC voltage supplied.

To the human eye, these waveforms have an appearance which makes them easily distinguishable. Computer software can differentiate between the two using Fast Fourier Transform (FFT) algorithms [26] FFT is used in this work to identify key frequencies, uniquely characteristic of specific *devices*.

By performing *device* recognition, the *server* software can extract more information about the way power is used around the home, delivering a better user experience. *Device* recognition can work in a variety of configurations. For example, multiple *devices* plugged into a single *node*. In this configuration the number of physical *nodes* required to monitor *devices* individually is reduced. Events relating to individual *devices* can be detected irrespective of what adjacent *devices* are doing. A user may apply this feature to calculate which *device*, on a chain of many *devices*, is using the most power without having to purchase and run separate *nodes* or manually swap the connected *device*.



Theoretically, this configuration could be scaled up to monitor all of the *devices* in a household, from a single *node*. The *node* would monitor AC mains current from a clip-on hall effect sensor attached to the mains supply to the household.

Another possible application for *device* recognition is for predicting failure. In this configuration a single *device* would be plugged into one *node* and the *server* would continually monitor the *device* to ensure its profile matches a known working profile for that *device*. Any deviation from the known profile may indicate a physical deterioration of the device - possibly a precursor to failure.

### 3. Node Hardware

As mentioned earlier in this work, relatively low cost readily available hardware is explored to provide necessary functionality and requirements of the above explained system. In this work the *node* hardware consists of three major components: logic (including communication), is provided by the openPicus Flyport; switching is provided by a Sharp Solid State Relay; and current measurement is carried out by an HX 03-P/SP2 current transducer. These components have been selected for their low cost, availability and suitability for interfacing between mains current and low voltage, DC control circuitry.

#### 3.1 Logic: OpenPicus Flyport Wi-Fi

The Flyport Wi-Fi [27] used is equipped with 18 digital input/outputs which are capable of Pulse Width Modulation (PWM). The Flyport also offers four 10-bit analogue inputs. A 10-bit analogue to digital converter (ADC) can resolve  $2^{10}$  or 1024 divisions which is a relatively poor resolution. The Flyport has an SPI interface which can be used to interact with an external 24-bit ADC giving significantly better resolution. However, as discussed later in section 3.5, a technique exists which can greatly improve the sensitivity of a 10-bit ADC.

#### 3.2 Load Switching: Sharp S202S01F Solid State Relay

Here a SHARP S202S01F solid state relay [28] used to switch the high voltage current. A solid state relay is used as it does not require as much voltage or current as a mechanical relay to hold 'on'. This is important, because the relay is driven by the Flyport which has a digital I/O voltage of 3.3 V and a maximum available current of 18mA [29]. The S202S01F can switch loads of up to 8 A and has an isolation voltage of 4.0 kV. The device is

therefore ideally suited for switching AC mains *devices*. Unfortunately SSRs tend to fail in the ON state which is a potential hazard for use in the home.

### 3.3 Current Measurement: HX 03-P/SP2 Current Transducer

The LEM HX 03-P/SP2 current transducer [30]. The transducer is effectively an analogue Hall Effect sensor which outputs a voltage proportional to the current in the mains circuit. The current transducer has a range of  $\pm 9$  A which corresponds linearly to a DC output voltage of  $2.5 \pm 0.625$  V, where 2.5 V corresponds to a measured current of 0 A.

In addition to these three major components, an LM358N Integrated Circuit [31] is used to shift and amplify (or attenuate) the current transducer signal into the range that the Flyport can register.

### 3.5 10-bit ADC Resolution for Measuring Power

The Flyport (selected to provide the logic and measurement capabilities) only contains a 10-bit analogue to digital converter (ADC). A 10-bit ADC can only resolve 1024 divisions. In any ADC, readings will lock to the nearest integer division. For a design load of 1 kW (where 80% of the Flyport sensitivity represents 1 kW) one division is equal to 13.8 W. Therefore the error is  $\pm 6.9$  W. Such a large error is unacceptable in a *system* that will be used to measure the power consumption of *devices*, such as low energy light bulbs, which may use less than 10 W in total.

One technique to improve the sensitivity of power measurements is to exploit the fluctuating nature of the current transducer output wave, by taking multiple measurements throughout a cycle and calculating the RMS current. This approach works because on average, between divisions, rounding-up errors will be largely offset by rounding down errors.

To illustrate this, a synthesised current draw waveform representing a 100 watt tungsten light bulb powered by 230 VAC was defined in Microsoft Excel. The waveform consisted of 350 samples equally spaced over an 80 ms period. The magnitude of each sample was rounded to the nearest integer as would be the case in the Flyport's ADC. The amplitude of the wave was such that a 1 kW variant would occupy 80% of the Flyports sensitivity, peak-to-peak. The calculated error as a function of ADC resolution is given for the multiple point technique and a single point of measurement as shown in Figure 4.

Figure 4 – Error as a function of ADC resolution

As the figure illustrates, by using multiple sample points, a very high level of sensitivity can be achieved with a reasonably low resolution ADC. By using multiple points, a 10-bit ADC with 1024 divisions has the equivalent sensitivity of a 16-bit ADC with 65536 divisions. For a 10-bit ADC, the error is 0.11%.

#### 4. Server Hardware

The primary function of the *server* is to coordinate communication between *nodes* and *clients*. The *server* also stores and queries data as well as manages scheduled tasks and events. In recent years, various low cost computer hardware capable of running basic Linux operating systems have appeared in the market. In this research we have explored such devices. The Raspberry Pi [32], released in February 2012 is used as the *server* hardware within this project. The Raspberry Pi is a credit card-sized, silent, low energy and inexpensive PC powered by a Broadcom BCM2835 SoC (System on Chip) which contains 512 MB of SDRAM as well as a 700 MHz ARM1176JZF processor [33]. The Raspberry Pi is designed to run variations of Linux. For this project, the Raspberry Pi is configured to run Wheezy Raspbian [34] (build date 9/2/13) which is a free variant of Debian Linux licenced under the GNU General Public License, version 2. This version requires minimal configuration and has USB, network and Python support built in.

#### 5. Node Hardware

Two prototypes were developed for the project. Figure 5 shows the final prototype.

Figure 5 Final prototype with cover removed

Since the *node* is powered by AC mains voltage, the shape of the waveform output from the current transducer is expected to cycle (repeat) at a rate of 50 Hz. In order to collect data at sufficient resolution for analysis the Flyport should aim to collect approximately 100 data points per cycle. To capture 100 points per cycle requires a sample rate of 5000 Hz sustained for at least 20 milliseconds (ideally 80 ms).

The data collected is used by the *system* for two functions: firstly, to calculate the power usage of the attached *devices*; and secondly, to identify attached *devices*. In this prototype a rectifier board (which converts the area enclosed by the sinusoidal current waveform to a flat DC voltage) had been added so that in the event that the Flyport was unable to sample data at a sufficient rate, power could still be measured. However, resorting to this method would make *device* recognition impossible. Fortunately the Flyport sample rate was more than sufficient and therefore the rectifier board was not required.

## 5.1 Amplifier Tuning

The Flyport sensitivity is limited to 1024 divisions, therefore it is important to use as much of the range as possible to increase the resolution of the signal. Tuning also helps to ensure that the signal does not go beyond the Flyport's tolerable parameters i.e. an input voltage between 0 V and 2.048 V, so to prevent damage to the internal ADC. Tuning is achieved using an amplifier circuit capable of applying a fixed gain and offset to the current transducer signal.

Figure 6 shows a simulated output directly from the current transducer when measuring a 60 watt tungsten light bulb. As in this example and the next, a tungsten bulb is used as it has an almost entirely resistive load, and therefore, the current transducer output can be expected to fluctuate perfectly sinusoidally. In the figure, the red lines are the DC cut-off voltages for the Flyport and the green wave is the DC output from the current transducer.

Figure 6 – Current transducer output

As the Figure 10 shows, the peak to peak voltage from the transducer is very small (approximately 0.03 V). This peak to peak range spans fewer than 18 of the 1024 divisions available giving very poor resolution. However, the main problem with this signal is that it is outside of the Flyport's range. The Flyport ADC will output 1024 for every point it captures and could also be damaged from being subjected to 'over-volting'.

Figure 7 shows the same signal after applying an appropriate gain and offset. In this signal the peak to peak height spans 824 of the 1024 divisions, and fits within the Flyport's

tolerable range, with an added margin of safety for noise and drift (due to thermal effects on components and other factors).

Figure 7 – Amplifier output (to Flyport)

This is implemented in the *node* prototype using the offset and gain potentiometers as shown in in Figure 8.

Figure 8 *Node* amplifier board showing important potentiometers

## 5.2 Provisional Calibration

To measure power, the *server* requires that the Flyport samples a waveform from the current transducer using its internal ADC. For any one measurement, the ADC returns an integer value between 0 and 1024, inclusive. The ADC value is linearly proportional to the output from the amplifier, in turn linearly proportional to the instantaneous current draw of the attached *device*. The server calculates an RMS value for the waveform, measured in divisions. To convert this value to amps within the *server* requires calibration parameters. The relationship between the RMS value,  $x$  (divisions) and the current draw of the attached *device* (amps) can be found using eqn, (1).

$$I_{RMS} (Amps) = m x_{RMS}(Divisions) + c \quad (1)$$

To solve for both  $m$  and  $c$  the *node* must be tested under two different loads. Firstly, with no *device* connected; and again with a *device* which draws a stable load. In each of these states, the actual current draw was measured using a Prodigit 2000MU power monitor. The accuracy of the 2000MU when measuring RMS current is claimed to be within 1% and typically around 0.3% (within the operating temperature and humidity) [35]. The test *device* used during calibration of the *system* was a 60 W tungsten light bulb.

The calibration factors  $m$  and  $c$  were found to be  $2.541 \times 10^{-3}$  and  $-1.108 \times 10^{-2}$  respectively. These values are stored in the *server* database.

## 6. Node Firmware

The Flyport is controlled by firmware that runs from its internal non-volatile memory. The firmware is developed in the openPicus IDE and flashed to the Flyport via a USB programmer. The Flyport has 256 KB of internal non-volatile memory of which a very small amount (approximately 1 KB) is occupied by the device's boot loader.

The Flyport firmware performs the following functions;

- Establish and maintain a permanent contact with the *server*
- Periodically send the Flyport's Media Access Control (MAC) address to the *server*, initially for *device* recognition and then as a keep alive signal.
- Respond to a current trace request from the *server* by replying with a burst of 350 samples captured from the current transducer over 80 ms
- Switch the solid state relay on or off at the request of the *server*

## 7 Server Software

In this work the server software is programmed using Python. Python is an open source high level programming language which is almost entirely platform independent. Python supports the essential advanced programming functionality required for this project such as managing TCP sockets, serving web *clients*, advanced mathematics and multi-threading.

The *server* source code was written using Microsoft Visual Studio Express and the Python Tools for Visual Studio (PTVS) add-on which adds syntax highlighting and other features to Visual Studio for Python. The expansion is available as a free download from CodePlex, Microsoft's free code project hosting website [36].

### 7.1 Configuring the *Server* Environment

Wheezy Raspbian (build 9/2/13), based on Debian Linux (RaspberryPi) is used as the operating system on the Raspberry Pi. Raspbian supports Python by default and is supplied with minimal features, thus reducing unnecessary power consumption.

The operating system is flashed to a Secure Digital (SD) card and inserted directly into the Raspberry Pi. The following significant changes were made from the default Raspbian configuration in order to allow the server software to run;

- IP address fixed to ensure it is always reachable on the network
- MySQL server installed (to store data from the server software)
- Python MySQL [37] libraries installed to allow Python to talk to the MySQL database engine.
- Numpy installed to add some scientific mathematical functions to Python [38].
- SciPi installed to add advanced mathematical functions to Python including FFT [39].
- Samba installed to allow Microsoft Windows based systems to modify files on the Raspberry Pi via a Windows network [40].

## 7.2 Data Storage and Retrieval

Data and settings are stored in a database provided by the free MySQL database engine. The database engine is separate from the *server* but runs beside it on the Raspberry Pi. At start up, the *server* connects to the database to read and manipulate records using Structured Query Language (SQL).

The database schema and other settings are configured remotely via the freely available MySQL Workbench software.

Using a database engine to store data as opposed to storing data in a proprietary format allows third party software to interface with, analyse and manipulate the data. This approach has also helped to implement advanced data storage and retrieval functionality within the *server* software using available software tools.

## 7.3 Web Interface

The web interface is the primary way in which users interact with the *system*. Though the web interface, users can switch *nodes* on or off, monitor power usage, set up scheduled tasks and events and train the *system* to recognise *devices*.

The major advantage of using a web based interface to interact with the *system* is that it does not require bespoke software to be written for specific *client* operating systems. Instead all web enabled devices including PCs, Macs, phones and tablet computers can access the *system* without installation of any additional software, since all of these devices come with web support.

The web interface is served using a proprietary webserver written in Python and integrated into the server software. This proprietary webserver is based on the standard Python

SimpleHTTPServer module which handles incoming HTTP requests. This allows greater control over content delivery than would be permissible trying to link the server software to a third party webserver package (such as Apache).

The website's visual interface is coded in HTML and Cascading Style sheets (CSS) [41] using Notepad++ [42]. The HTML & CSS files are stored in the same directory as the server software allowing for easy customisation. When the web interface is requested, the server software retrieves the relevant files and hands them to a SimpleHTTPServer object which streams them to the client via the HTTP protocol. Live communication between the client and server is achieved using Asynchronous JavaScript and XML (AJAX). Once the user's browser software receives the page in full, an AJAX object is created which connects back to the server periodically and requests updates such as the current system power usage. This technology allows the user to sit back and monitor the state of the system without having to continuously refresh the page.

## 8. Demonstration of the Software and Associated Functionality

### 8.1 The Web Interface

Figure 9 shows the overview screen of the developed web interface. The screen lists basic information about the *system* as a whole, for example the number of *nodes* that are currently connected, the overall *system* power usage, the  $CO_2$  output etc.

Figure 9 – Overview page

Also in the summary table, the overview page also lists all of the *nodes* that are connected. Each entry in the table has a dedicated power control and power usage reading. *Nodes* not currently connected are hidden to keep the user interface clean and easy to navigate.

The overview screen also displays power usage across a time period in graph format as shown in Figure 10. The graphical representation aims to help users associate periods of high consumption with particular activities or events.

Figure 10 – Power usage graph



Graphing is provided using the *Highcharts* 3.0.1 [43] and Javascript API. *Highcharts* can be used to display data in real-time and is compatible with all popular desktop and mobile browsers.

Users can select between a variety of time periods from the menu shown in Figure 11.

Figure 11 – Time period menu

The time period list box governs the date ranges used to calculate energy use, cost and  $CO_2$  output, as well as the date range used on the graph shown previously.

Data on the overview screen are updated automatically without requiring the user to refresh the page or click any buttons. This is achieved using Asynchronous JavaScript and XML (AJAX).

The settings page, shown in Figure 12, is accessed from the menu to the left.

Figure 12 – Settings page

The settings page allows users to set up and configure *nodes*, add locations, set up *device* recognition, test *device* recognition and modify variables such as energy costs and grams of  $CO_2$  per kWh. By default the cost of electricity in terms of money and  $CO_2$  is 13.9p per kWh and 0.55 g  $CO_2$  per kWh respectively [44,45]. Like the overview page, this page communicates with the *server* using AJAX.

## 8.2 Mobile Client Interface

Figure 13 shows the same data accessed via a mobile phone. The server automatically detects mobile clients and serves an alternative touchscreen-optimised version of the overview and settings pages with larger fonts and buttons as shown in Figure 13.

Figure 13 – Mobile version of the overview page

## 8.3 Security

Research for this work has shown that consumers are concerned about the security of similar internet enabled systems, which have the capability to control and / or monitor electronic *devices* within the home. Providing the option to view the web interface via

HTTPS instead of HTTP is one step towards ensuring security. The HTTPS protocol encrypts communication between the *server* and *client* making it impossible for messages to be read or modified in transit by third parties.


When the web interface is loaded on a browser such as Chrome, the user is shown that the page is secured with HTTPS. Chrome indicates this by adding the  `https://` prefix at the start of the *server* address as shown in Figure 14.

Figure 14 – HTTPS connection information in Google Chrome

This figure also shows details about the encryption level. In this case 256-bit encryption is used with a self-signed certificate.

#### 8.4 Node - Server Communication

Messages between individual *nodes* and the *server* are sent via a TCP (Transmission Control Protocol) socket as opposed to a UDP (User Datagram Protocol) socket. Although TCP has a slower response time, it guarantees that information arrives well-formed and in the same order in which it was sent.

With UDP, information is sent as fast as possible. This can lead to more recently sent packets arriving before older packets depending on their route through the network. With UDP it is impossible to know if information has been lost in transmission. TCP response time and throughput is still exceptionally good compared to UDP [46] and more than sufficient for this project.

The *node* TCP socket is opened in an individual thread in order to concurrent connections and implement full asynchronous communication support between the *node* and *server*.

Before a *node* can be used by the *system* it is first 'identified'. Identification matches a node to its associated settings from the database such as name, location and calibration data. Nodes are identified based on their Media Access Control (MAC) address. MAC addresses are used for identification because they are guaranteed to be unique. This allows *nodes* to be identified even if their IP addresses change (for example on a DHCP network) or even when connecting from an external network via the internet.

Figure 15 shows how a user request to turn a *device* on enters the *system* and is communicated to the *node*.

Figure 15 - Power switch input / output path

## 8.5 Signal Processing, FFT and Power Calculation

Raw data arrives at the *server* from the *nodes* as a series of 10-bit ADC divisions vs time offset in milliseconds. The *server* then subjects to signal to the following processing steps.

- Time-zero all points (the first point is not naturally stamped with time = 0 as there is a small delay between the Flyport recording the start time and sampling the first point).
- Artificially resample the signal at constant intervals over  $2^{12}$  (4096) points by interpolating between data points. This step results in some minor loss of quality, but is essential for the FFT analysis.
- Calculate the RMS current in 10-bit ADC divisions using eqn. (2) where  $N$  is the total number of data points and  $I$  is the current at a given data point,  $x$ .

$$I_{RMS}(divisions) = \sqrt{\frac{1}{N} \sum_{x=1}^N I_x^2} \quad (2)$$

- Convert the RMS current in 10-bit ADC divisions to RMS current in amps using eqn. (3) where  $m$  and  $c$  are calibration factors found using a reference *device*:

$$I_{RMS}(amps) = m I_{RMS}(divisions) + c \quad (3)$$

*Permissible because at  $I(amps) = 0$ ,  $I(amps) = I(divisions)$*

- Calculate the Power in watts using equation 4. (AC mains RMS voltage is assumed to be 230 V).

$$Power(watts) = I_{RMS}(amps) \times V_{RMS}(volts) \quad (4)$$

- Extract the frequency spectrum from the signal using the Scipy FFT function.
- Make a list of characteristic frequencies and magnitudes which have a magnitude above a predefined cut off. (This list is used for device recognition).

Figure 16 shows the frequency spectrum generated by a FFT analysis of the waveform of a low energy light bulb. In this graph, a relative magnitude is plotted

against frequency. The horizontal red line marks the position of the cut off threshold.

Figure 16 – Frequency domain of test signal

The two vertical red lines represent the frequency and magnitude of each peak which rises above the cut off frequency.

## 8.6 Device Recognition Implementation

### 8.6.1 Profiling

For a *device* to be recognised, it must first have a 'profile'. *Device* profiles are a list of the key frequencies and magnitudes that were observed by a *node* when powering the *device*. Users instruct the *server* to profile a *device* using the settings page within the web interface. Figure 17 shows the final state of the *device* profiler tool after a *device* has been profiled.

Figure 17 - *Device* profiling tool

This interface allows the user to select the *node* that a *device* is attached to, to name the *device* and to view the frequencies and magnitudes that correspond to the *device* profile. The key frequencies are found as described previously in section 8.5 and are stored in the database.

### 8.6.2 Device Recognition

*Device* recognition follows the process as described by Figure 18.

Figure 18 - *Device* recognition procedure

When *device* recognition is required, key frequencies and magnitudes are calculated from data received from a *node*. This information is passed to the *IdentifyDevices* function internally.

*IdentifyDevices* then loads in all of the *device* profiles (potential *devices*) from the database and tries to eliminate as many *devices* as possible. This is critical for combination testing in the next step. A *device* profile can safely be eliminated if either of the following criteria are true. The potential *device*;

- requires a frequency which is not present in the live data; or
- has a frequency which matches the live data, but at a magnitude which is far in excess of the magnitude for the same frequency in the live data.

Once *devices* have been eliminated, *IdentifyDevices* tries to start combination testing. Combination testing works by superimposing the signals of the potential *devices* in every possible combination and comparing combined frequencies and magnitudes to the live data. Table 1 shows the four possible states that two potential *devices* could be in.

Table 1 – *Device* states

For the software, it is easier to refer to these combinations as bit encoded strings. A ‘string’ is a series of characters and by extension, a bit encoded string is a series of binary characters. The position of the character in the string refers to the *device* and ‘1’ and ‘0’ represent ‘on’ and ‘off’ respectively. This is convenient for iteration as a bit encoded string is effectively a binary number - interchangeable with an integer value.

The software skips the first combination (integer value zero) as this is the equivalent of the software failing to recognise at least one *device* in the live data. If after iterating through all combinations without identifying a *device*, it is assumed that none of the potential *devices* are present and the function returns an empty list.

The number of possible power state combinations that a series of potential *devices* can be in is therefore  $2^N - 1$  where  $N$  is the number of potential *devices*. As mentioned previously, it is therefore very important that as many potential *devices* as possible are eliminated, as each additional *device* doubles the number of possible combinations. More *devices* therefore exponentially increases the (theoretical maximum) time it takes to recognise *devices*.

Once the *system* knows what *devices* are plugged in, it is then possible to convert the key frequencies and magnitudes back into a waveform for each attached *device*. This can then be processed in the same way that the raw data is processed to find out how much power each *device* should be using. However since the key frequencies and magnitudes function ignores noise and small magnitudes, the power of the synthesised waves will be less than the actual total power. As a rough estimate, the *system* can increase each synthesised wave power proportionally until their sum gives the actual measured *system* power. This

technique allows users to see how much each attached *device* is using, separately, with just one *node*.

## 9. Evaluation of the complete system

### 9.1 Testing and evaluation of functionality and response

The performance of the system was evaluated in laboratory conditions. Only one fully functional node prototype was constructed to prove that the Flyport could switch appliances on and off as well as monitor the power of attached devices. This system was tested using two devices, a low energy light bulb and tungsten light bulb connected to a power strip, in turn connected to the node. Performance of the system was good and behaved as expected with regards to both switching and monitoring. These devices were tested intermittently over three weeks and no appreciable drift in power consumption was detected (+/- 8 watts). Power consumption was verified using a Prodigit 2000MU power meter. Some testing was also undertaken using AV equipment and proved successful. During one experiment, the Solid State Relay failed in its 'on' state. This is a potentially dangerous failure mode, especially if such a system were to be used to control a heater. It is therefore recommend to always use a mechanical relay that requires power to hold 'on' in such applications. Due to the limited power of the Raspberry Pi, it was found that an unacceptable lag of up to 3 seconds can exist between issuing a command on the web interface and seeing activation of an attached device. It was found that by disabling the power usage logging features of the system, this lag can be reduced such that it is imperceptible. The primary cause of lag was determined to be the time in which the Flyport takes to receive and act on an instruction to capture power data, followed by the time it takes to send this back to the server and for the server to store it in the MySQL database. During this time, no instructions can be acted upon. Several remedies are suggested including moving the database storage functionality to its own thread and having the Flyport check for further instruction every *n*th cycle of its power usage capturing loop.

An additional Flyport and a node simulator script were used in order to demonstrate that the system can handle multiple nodes and act on user specified rules. Two Flyport units were purchased for the project. The firmware developed for the Flyports makes no attempt to detect whether or not the Flyport is running from a node or an FTDI programming board. Therefore as long as the Flyport is powered, it will attempt to connect to the server and report its state and act on commands. The Flyport has two on-board LEDs, the first of which is programmed to illuminate when connected to the server and second when the

attached device is meant to be activated. Using this feedback it was possible to test the control components of the system on two nodes, despite one node not having the associated monitoring and switching hardware.

In addition to this, a simple Python script was written to simulate the behaviour of nodes. The script was designed such that it connects to the server software and reacts to instructions from the server in the same way a physical node would, but in place of turning a device on or off, the code simply displays a message on the screen explaining what it has been instructed to do. Multiple instances of the node simulator script can be run simultaneously. By combining this infrastructure, it was possible to show that the server is issuing commands to nodes in response to user specified rules such as 'on timer events' and 'on state change' events.

### 9.1 Evaluation of device recognition

Below the *device* setup section on the settings page is the recognition test tool, which can be used to check *device* recognition. Figure 19 shows the *device* recognition test tool after querying the *devices* attached to 'Unnamed Node'.

Figure 19 – *Device* recognition test tool

To use the tool, the user is required to select a node to query and then click the 'check' button. If the *device* recognition function identifies any *devices* attached to the selected node, they appear as a list on the page.

Using this tool, the *system* was shown to successfully identify the Light and Low Energy Light *devices* in all four possible combinations. *Device* recognition applied to these two *devices* proved to be very repeatable and accurate.

*Device* recognition only works for *devices* which draw a static load. This may include for example light bulbs, fans and heaters. *Devices* with irregular duty cycles, such as laptop chargers, computers and washing machines are significantly harder to profile with this implementation of *device* recognition.

### 9.3 Event Handling and Task Scheduling

Event handling and task scheduling are essential elements of the 'Automation' component of this work. Event handling and scheduling are two similar concepts which are both ultimately used trigger a *task*. A *task* performs some kind of action, the available actions

are; sending an email, switching the power state of a *node*, outputting a message to the *server* console, saving the overall system power usage to the database, and running Python code directly.

*Task* scheduling triggers *tasks* at periodic intervals, or at a specific date and time, either repetitively or just once.

An example of a user-driven scheduled *task* might be to turn a light on and then off again at specific times. For example, to make an unoccupied house appear occupied, or to turn decorative exterior lighting off after midnight.

## 10 Conclusions

This research project set out to develop a hardware and software solution to automate, monitor and control electrical/electronic *devices* around the home. The ultimate aim was to develop a low cost, versatile *system* which could raise consumer awareness of energy consumption and also reduce energy consumption, by supporting consumer decisions such as the choice of energy supplier or appliance specification when purchasing new electrical equipment, based on real usage data.

Much of the base level development has been achieved. For example, the production of a working *node* prototype and writing *server* software which supports multiple *node* connections as well as serving a user interface via a web page, performing *device* recognition, calculating power usage, managing scheduled tasks and recording data has been achieved and successfully demonstrated. In the development of the system, low-cost components and open-source software have been used. The system developed is very detailed in terms of data gathering, storing and displaying to the user and could be improved further with additional features required by the home energy market.

However the *system* has several limitations. The *node* prototypes are relatively large and should be packaged in a more discrete enclosure. This being a research project no attention was paid to address this aspect and it could be easily addressed with specialist components. Whilst the vast majority of the core software functionality is in place, there is still plenty of room for improvement. For example, a richer user interface, better task configuration, recording macros and the introduction of a dedicated cost benefit calculator could be added. The calculator would be powered by data collected over the life of the *system* and could be linked to energy supplier websites and domestic appliance manufacturer websites.



The major conclusions from this project are that;

- It is possible to monitor and control multiple *devices* around the home in real time from a web interface using the developed hardware software packages,
- the software developed for this project enables a detailed log of user activity and *device* power usage to be compiled, which can be used to determine consumption trends and inform purchase decisions, and,
- using Fast Fourier Transforms and signal analysis, it is possible to identify the types of *devices* plugged into *nodes* and how much power they use individually, based on the way they use power.

## 11 Acknowledgements

We acknowledge the support from Clker.com, <http://www.clker.com/> for hosting free for commercial use images used in the website component of this project. Including the 'green earth' image that forms part of the HAMC logo. Highsoft Solutions AS, <http://highsoft.com/> for making their JavaScript charting API free for use in projects such as this.

## 12 Nomenclature

Abbreviations

<b>ADC</b>	Analogue to Digital Converter
<b>AJAX</b>	Asynchronous JavaScript and XML
<b>API</b>	Application Programming Interface
<b>ASCII</b>	American Standard Code for Information Interchange
<b>AV</b>	Audio Visual
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>FFT</b>	Fast Fourier Transform
<b>GPIO</b>	General Purpose Input/Output
<b>GSM</b>	Global System for Mobile Communications
<b>HAMC</b>	Home Automation, Monitoring and Control
<b>HDMI</b>	High-Definition Multimedia Interface
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>HTTPS</b>	Hypertext Transfer Protocol Secure
<b>I/O</b>	Input / Output
<b>ID</b>	Identification
<b>IDE</b>	Integrated Development Environment
<b>IP</b>	Internet Protocol
<b>LED</b>	Light-Emitting Diode

<b>MAC</b>	Media Access Control
<b>Mac</b>	Macintosh
<b>PC</b>	Personal Computer
<b>PTVS</b>	Python Tools for Visual Studio
<b>PWM</b>	Pulse Width Modulation
<b>RMS</b>	Root Mean Square
<b>SD</b>	Secure Digital
<b>SDRAM</b>	Synchronous Dynamic Random Access Memory
<b>SoC</b>	System on Chip
<b>SPI</b>	Serial Peripheral Interface
<b>SPST</b>	Single Pole, Single Throw
<b>SQL</b>	Structured Query Language
<b>SSID</b>	Service Set Identification
<b>SSL</b>	Secure Sockets Layer
<b>SSR</b>	Solid State Relay
<b>TCP</b>	Transmission Control Protocol
<b>UART</b>	Universal Asynchronous Receiver/Transmitter
<b>UDP</b>	User Datagram Protocol
<b>USB</b>	Universal Serial Bus
<b>Wi-Fi</b>	Short-range wireless network
<b>XML</b>	Extensible Markup Language

## Glossary

- Client** Any device capable of displaying a webpage (laptop, phone, tablet, computer) that connects to the *server* in order to allow users to interact with the *system*.
- Device** Any mains electrical appliance (light, heater, radio, laptop, etc.) which is monitored and or controlled by a *node*.
- Node** The bespoke hardware developed for this project to monitor and switch *devices*. *Nodes* contain the logic, switching and monitoring circuitry. *Nodes* connect to the *server* via Wi-Fi.
- Server** The entity (mainly software) that coordinates all of the functionality of the *system*. *Nodes* and *clients* connect to the *server*.
- System** The concept of one or more *clients* connecting to a single *server* which in turn controls one or more *nodes* each switching and monitoring one or more *devices*.

## 11 References

- [1] Variability of wind power and other renewables: Management options and strategies  
The international Energy Agency, 2005,  
[http://www.uwig.org/iea\\_report\\_on\\_variability.pdf](http://www.uwig.org/iea_report_on_variability.pdf).

- [2] Bass RJ, Malalasekera W, Willmot P. The impact of variable demand upon the performance of a combined cycle gas turbine (CCGT) power plant, *Energy* 2011; 36(4):1956-59.
- [3] The Smart Grid: an introduction. US Department of Energy publication, 2009, <http://energy.gov/oe/downloads/smart-grid-introduction-0>.
- [4] Lund H, Andersen AN, Østergaard PA, Mathiesen BV, Connolly D, From electricity smart grids to smart energy systems - a market operation based approach and understanding *Energy* 2012;42:96-102.
- [5] Doostizadeh M, Ghasemi H, A day-ahead electricity pricing model based on smart metering and demand-side management. *Energy* 2012;46:221-230.
- [6] Olmos L, Ruester S, Liang S, Glachant J, Energy efficiency actions related to the rollout of smart meters for small consumers, application to the Austrian system. *Energy* 2011;36:4396-4409.
- [7] Pereira R, Figueiredo J, Melicio R, Mendes VMF, Martins J, Quadrado JC, Consumer energy management system with integration of smart meters. *Energy Reports* 2015;1:22-29.
- [8] Beckel C, Sadamori L, Staake T, Santin, S, 2014, Revealing household characteristics from smart meter data. *Energy* 2014;78:397-410.
- [9] Depuru SSSR, Wang L, Devabhaktuni V, Smart meters for power grid challenges issues advantages and status. *Renewable and Sustainable Energy Reviews* 2011;15(6):2736–2742.
- [10] Vega AM, Santamaria F, Rivas E, Modeling for home electric energy management: A review. *Renewable and Sustainable Energy Reviews* 2015;52:948-959.
- [11] Vassileva I, Wallin, F, Dahlquist E, Understanding energy consumption behaviour for future demand response strategy development. *Energy* 2012; 6(1):94-100.
- [12] Zhao L, Zhang Ji-li, Liang R, Development of an energy monitoring system for large public buildings. *Energy and Buildings* 2013;66:41-48.
- [13] Blumsack S, Fernandez A. Ready or not, here comes the smart grid!. *Energy* 2012; 37: 61-68.
- [14] Alagoz BB, Kaygusuz A, Karabiber A, A user-mode distributed energy management architecture for smart grid applications. *Energy* 2012; 44:167-177
- [15] Wissner M, The smart grid - a saucerful of secrets ?. *Appl Energy* 2011; 88: 2509-2518.

- [16] Batista NC, Melício R, Matias JCO, Catalão JPS, Photovoltaic and wind energy systems monitoring and building/home energy management using ZigBee devices within a smart grid. *Energy* 2013; 49: 306-315,
- [17] Kim H, Lee SK, Kim H, Kim H, Implementing home energy management system with UPnP and mobile applications. *Computer Communications* 2012; 36 (1):51-62.
- [18] Han D-M, Lim J-H, Smart Home Energy Management System using IEEE 802.15.4 and ZigBee. *IEEE Transactions on Consumer Electronics* 2010;10:1403-1409.
- [19] Jang WS, Healy WM, Wireless sensor network performance metrics for building applications. *Energy and Buildings* 2009; 42(6):862-868.
- [20] Han J, Choi C-S, Park W-K, Kim S-H, PLC-based photovoltaic system management for smart home energy management system, *IEEE Transactions on Consumer Electronics* 2014; 60(2):184-189.
- [21] Papagiannis, GK, Papadopoulos, TA, Dovas, CD, Tsiamitros, DA. and Dokopoulos, PS, A PLC based energy consumption management system. Power line performance analysis: Field tests and simulation results, *IEEE Power Technology Conference, 2005 Russia*.
- [22] Al-Mulla A., ElSherbini A., Demand management through centralized control system using power line communication for existing buildings, *Energy Conversion and Management* 2014; 79: 477-486.
- [23] Yigit M, Gungor VC, Tuna G, Rangoussi M, Fadel E, Power line communication technologies for smart grid applications: A review of advances and challenges. *Computer Networks* 2014; Volume 70(9):366-383.
- [24] Gold, S. Not-so-smart-meters, *Network Security*, 2009; 6:9-11.
- [25] Mayers, RJ, Williams ED, Matthews HS. Scoping the potential of monitoring and control technologies to reduce energy use in homes, *Energy and Buildings*, 2010; 42: 563-569.
- [26] Press WH, Teukolsky, SA, Vetterling WT, Flannery, B.P., *Numerical Recipes: The Art of Scientific Computing*, Second Edition, Cambridge University Press, 1992, Chapter 12.
- [27] openPicus. (2013, 06 10). SoftAp Flyport WIFI G. Retrieved 07, 2013, from openPicus Wiki: [http://wiki.openpicus.com/index.php/SoftAp\\_Flyport\\_WIFI\\_G](http://wiki.openpicus.com/index.php/SoftAp_Flyport_WIFI_G).
- [28] SHARP. (2004). S102S01 Series, S202S01 Series. Datasheet, SHARP Corporation.
- [29] Microchip. (2010). PIC24FJ256GA110 Family Data Sheet.

- [30] LEM - HX 03-P/SP2 - Current Transducer. Retrieved from [http://www.lem.com/hq/en/component/option,com\\_catalog/task,displaymodel/id,64.79.06.002.0/](http://www.lem.com/hq/en/component/option,com_catalog/task,displaymodel/id,64.79.06.002.0/) also available at Onecall <http://onecall.farnell.com/lem/hx-03-p-sp2/current-transducer/dp/1617419?Ntt=1617419>.
- [31] Texas Instruments. (July 2010). LM158, LM158A, LM258, LM258A, LM358, LM358A, LM2904, LM2904V, Dual operational amplifiers. Dallas, Texas 75265.
- [32] Raspberry Pi – A credit Card Size Computer, <http://www.raspberrypi.org/>, accessed Feb 2015.
- [33] Broadcom. (2012). BCM2835 ARM Peripherals. Broadcom Europe Ltd. 406 Science Park Milton Road Cambridge CB4 0WW.
- [34] Wheezy Raspbian <http://www.raspbian.org/RaspbianAbout>, accessed Feb 2015.
- [35] Prodigit Electronics Co.LTD. (2009). Model 2000MU Plug-In Power Monitor. Datasheet.
- [36] Microsoft, Python Tools for Visual Studio. Retrieved 23-07-2013, from CodePlex: <http://pytools.codeplex.com/>.
- [37] Python, python.org, MySQL-python, accessed Feb 2013, <https://pypi.python.org/pypi/MySQL-python>
- [38] NumPy.org, accessed Feb. 2013, <http://www.numpy.org/>.
- [39] SciPi.org, accessed Feb. 2013, <http://scipy.org/>
- [40] Samba, Samba.org., Feb. 2013, <https://www.samba.org/samba/>
- [41] CSS Tutorial, W3Schools.com, <http://www.w3schools.com/css/>
- [42] Notepad++, <https://notepad-plus-plus.org/>.
- [43] Highcharts.com, accessed Feb. 2013, <http://www.highcharts.com/>
- [44] Department of Energy & Climate Change. (December 2012). Quarterly Energy Prices. <https://www.gov.uk/government/statistics/quarterly-energy-prices-december-2012>.
- [45] Department of Energy and Climate Change (DECC). (2011). Guidelines to Defra / DECC's GHG Conversion Factors for Company Reporting, [https://www.gov.uk/government/uploads/system/uploads/attachment\\_data/file/69314/pb13625-emission-factor-methodology-paper-110905.pdf](https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/69314/pb13625-emission-factor-methodology-paper-110905.pdf).
- [46] Eric Gamess, Rina Surós. (2008, November). An upper bound model for TCP and UDP throughput in IPv4 and IPv6. Journal of Network and Computer Applications, 31(4), 585-602.



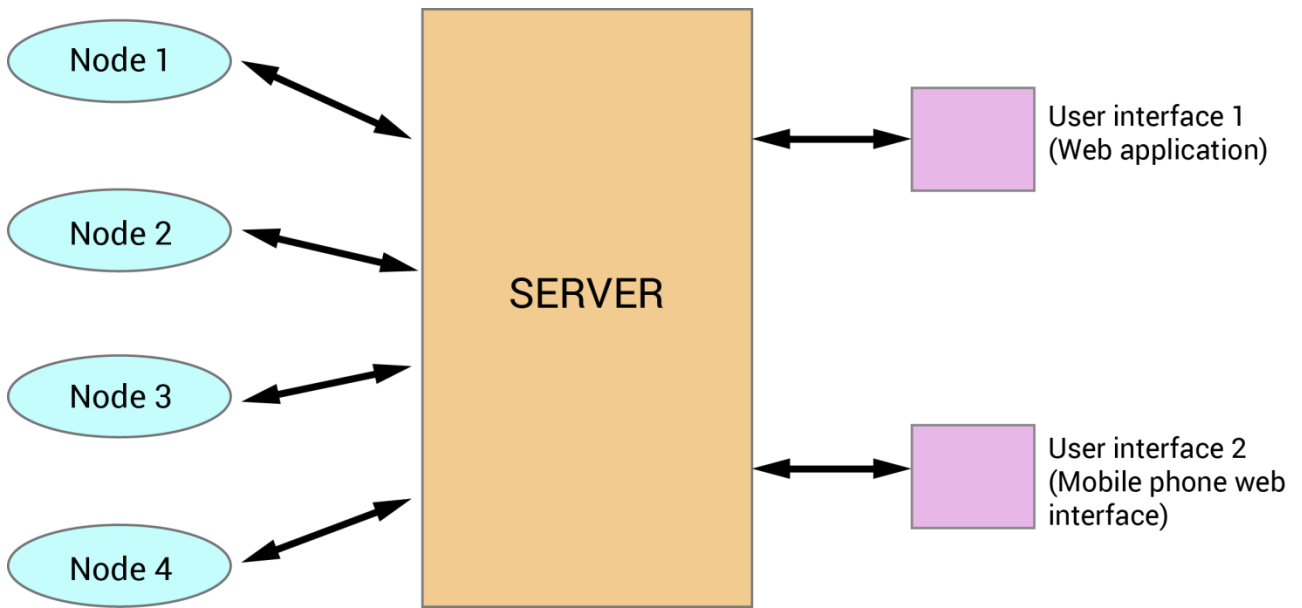


Figure 1. An illustration of the overall system concept

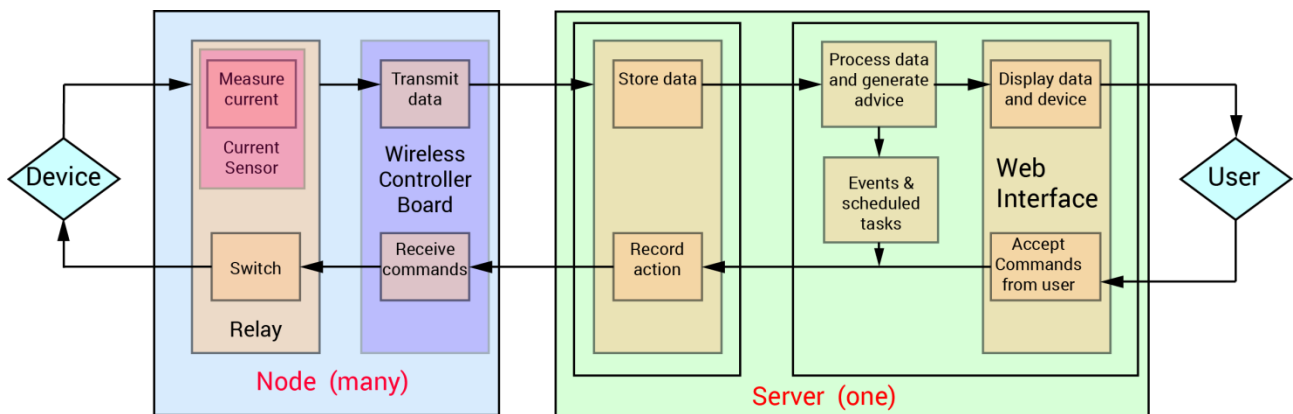
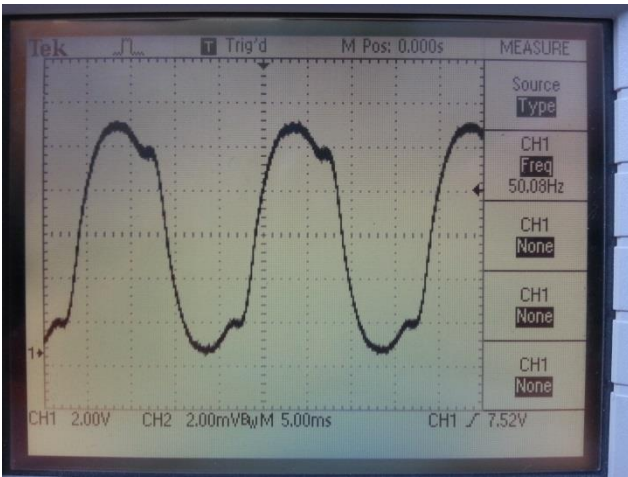
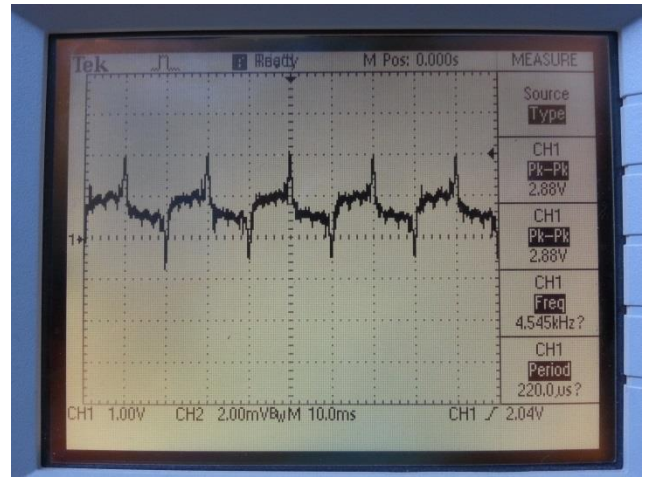


Figure 2 Detailed system layout



Overhead Projector (Fan and Light)



AV Equipment

Figure 3. Measured current transducer waveform for two devices

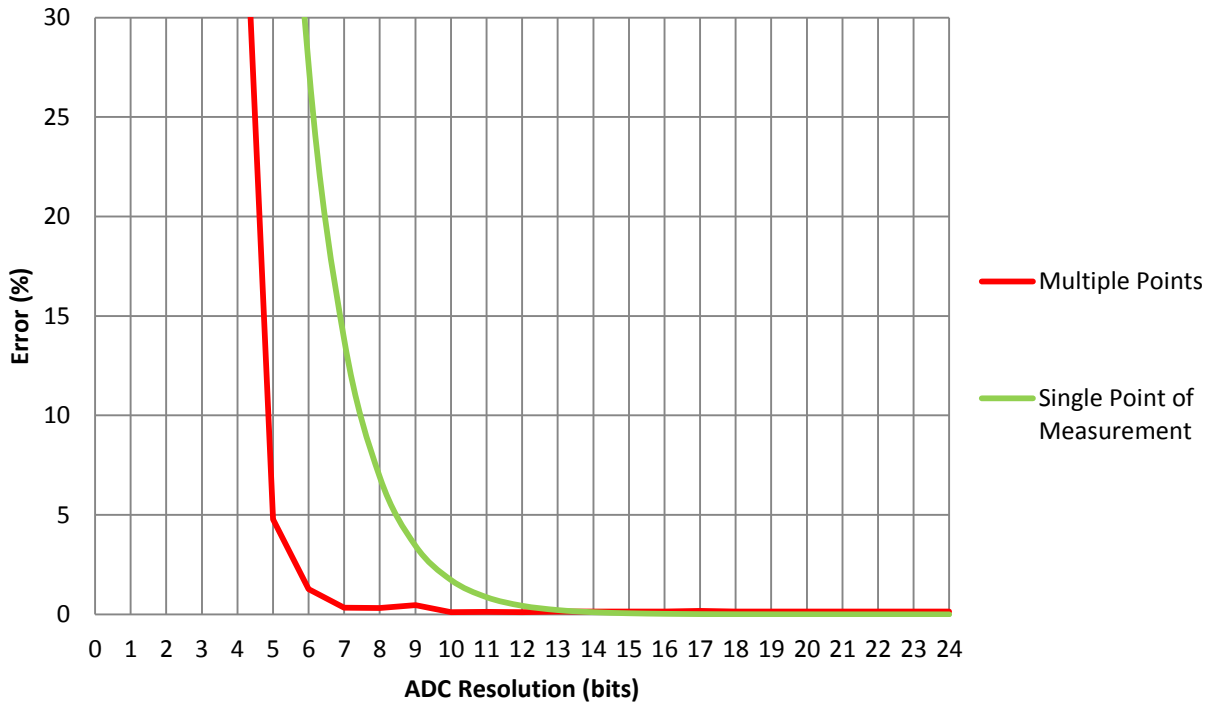


Figure 4 – Error as a function of ADC resolution



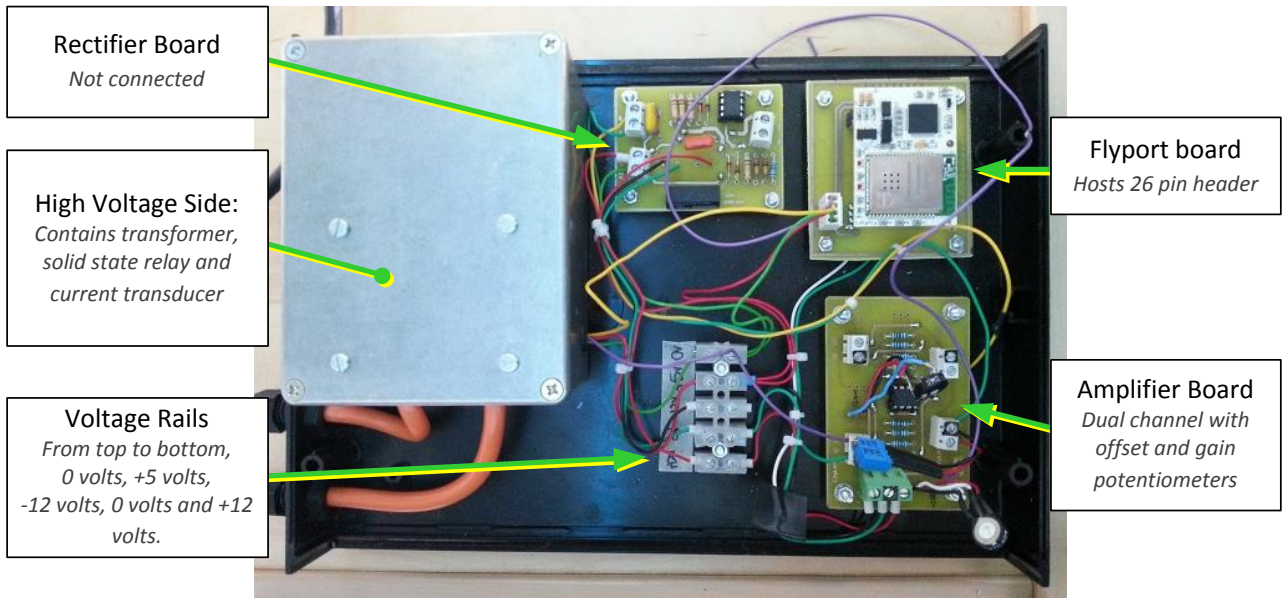


Figure 5 Final prototype with cover removed

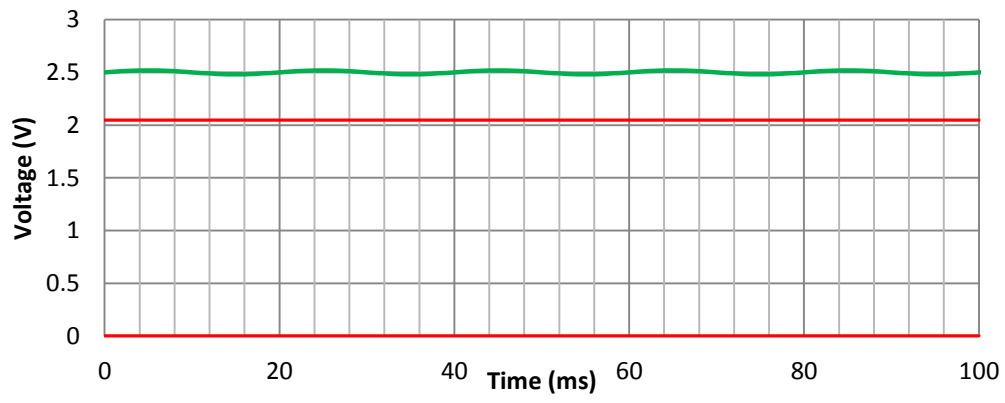


Figure 6 – Current transducer output

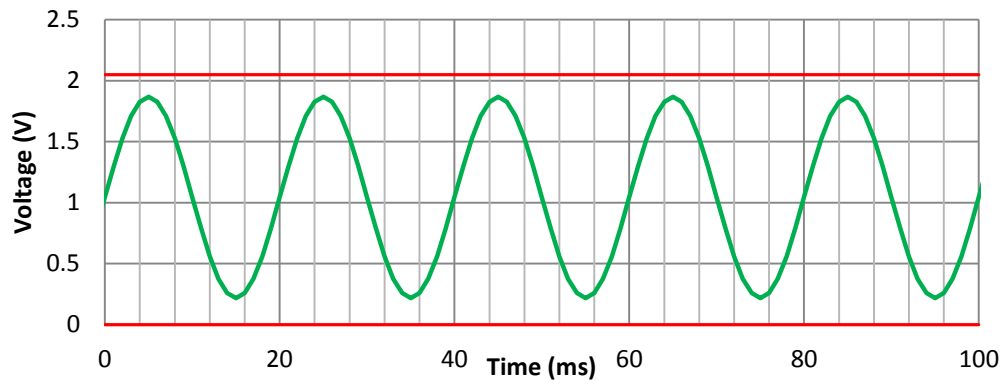


Figure 7 – Amplifier output (to Flyport)

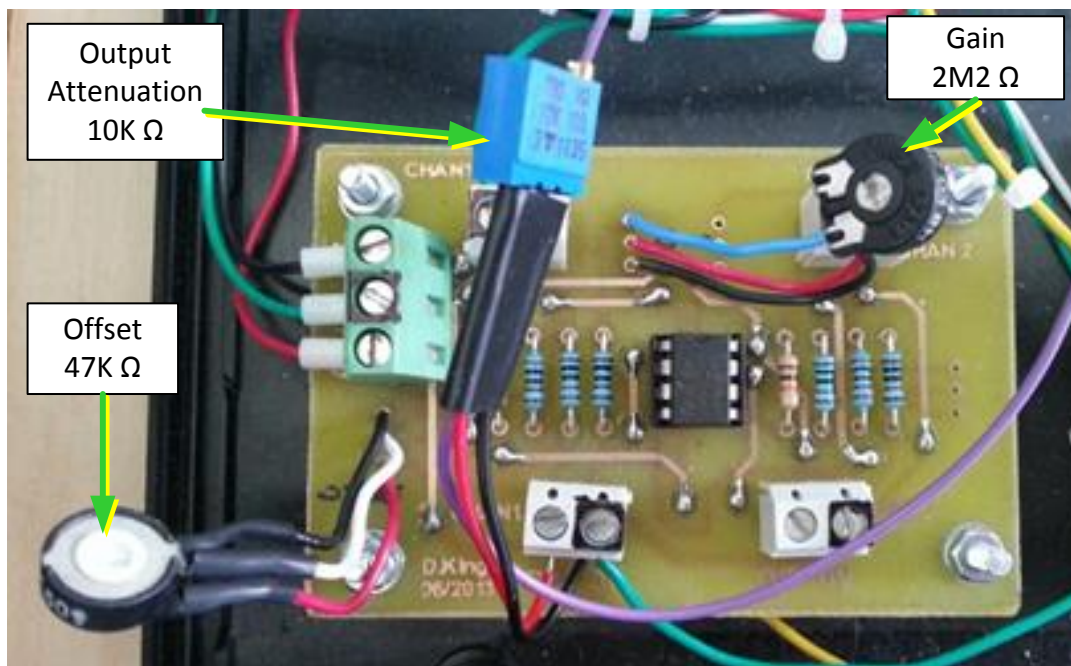


Figure 8 Node amplifier board showing important potentiometers

Automation Server

192.168.0.17

HOME AUTOMATION MONITORING CONTROL

A Mechanical Engineering MSc major project with Loughborough University

### Overview

System Power Usage Now	64.48 watts
Number of Nodes Connected	1
Control	<input type="button" value="All On"/> <input type="button" value="All Off"/>

Select Time Period	Today (from midnight)
Energy Use	47.82 watt hours
Cost	£0.01
CO <sub>2</sub>	27 g

Node ID	Name	Current State	Switch	Power Usage
2	Unnamed Node	On	<input type="button" value="On"/> <input type="button" value="Off"/>	64.48 watts

Figure 9 – Overview page

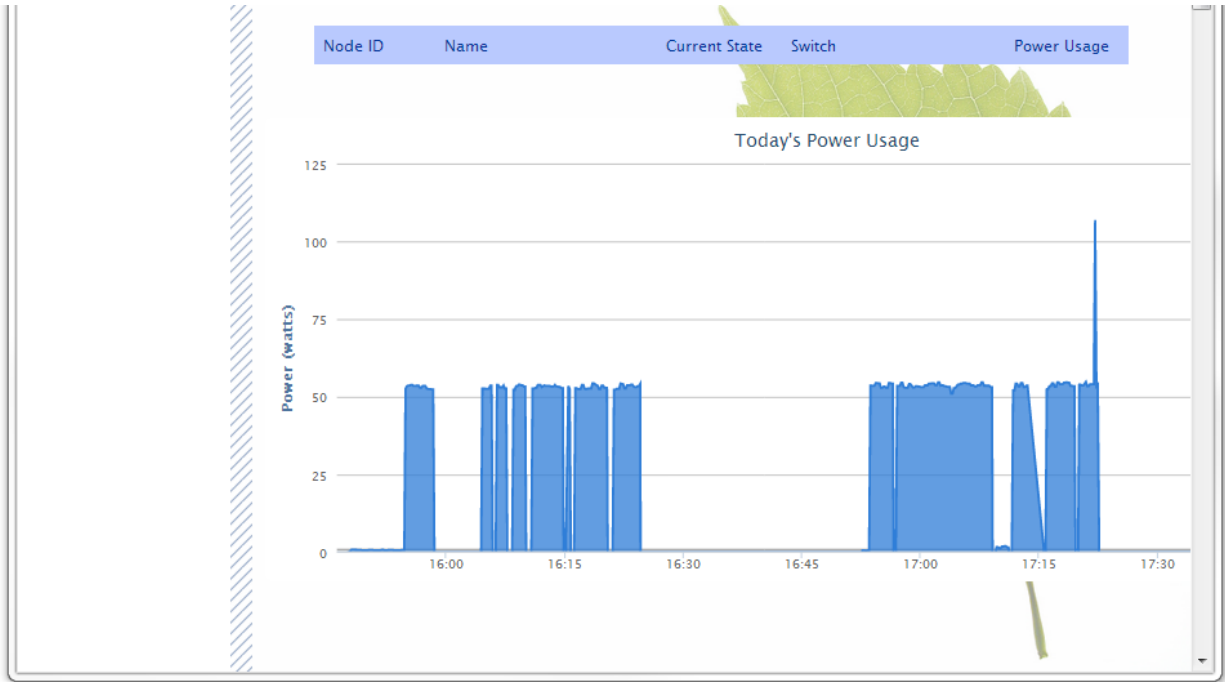


Figure 10 – Power usage graph

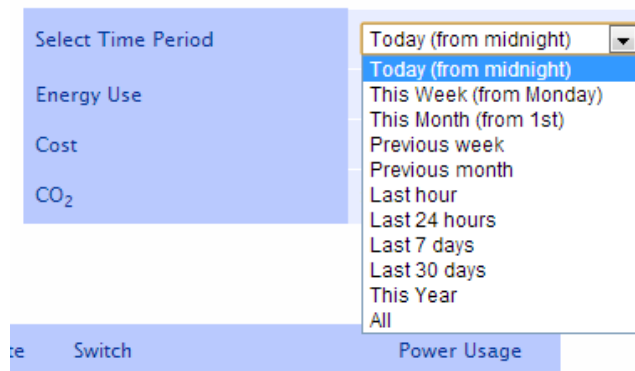


Figure 11 – Time period menu

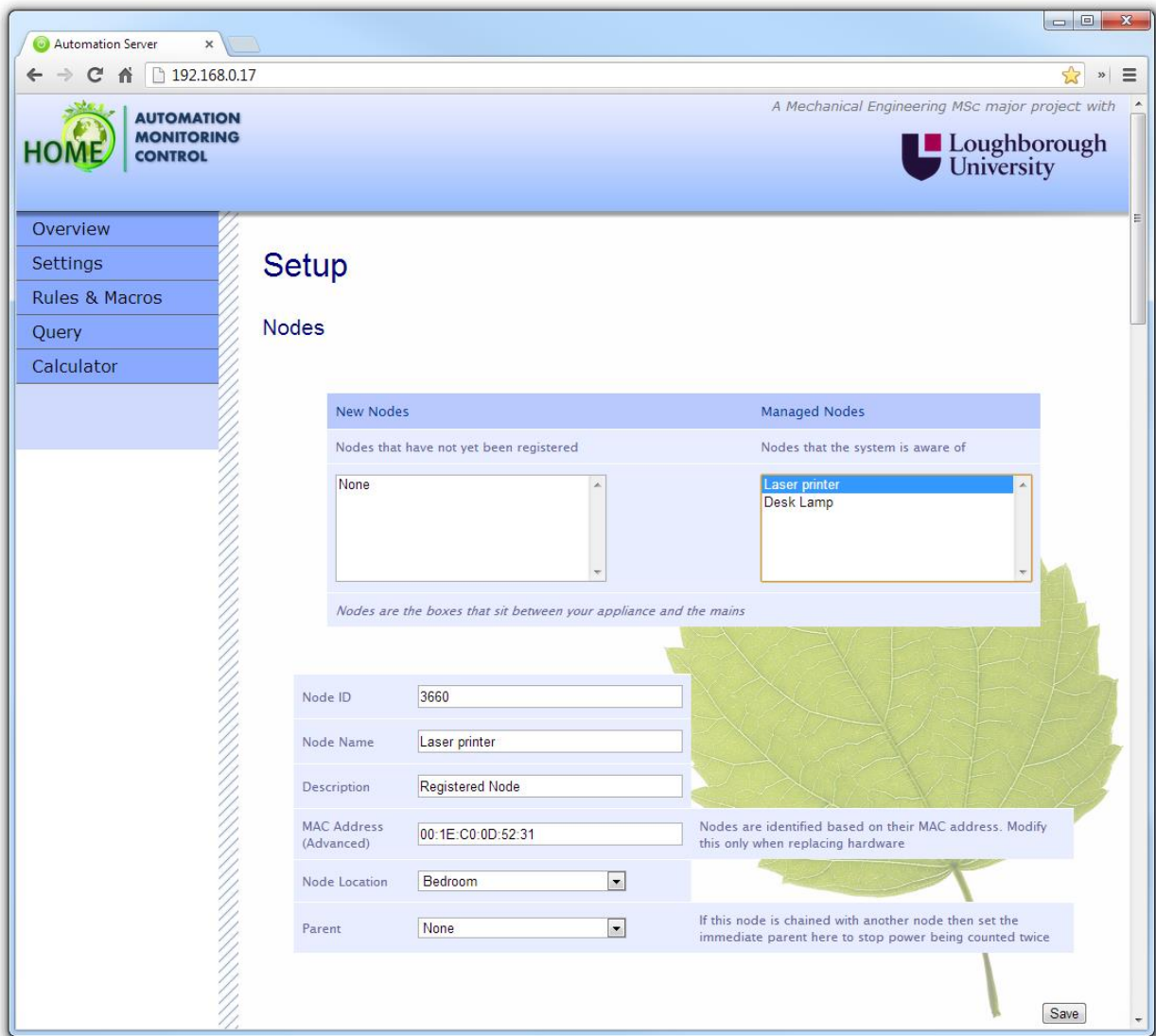


Figure 12 – Settings page

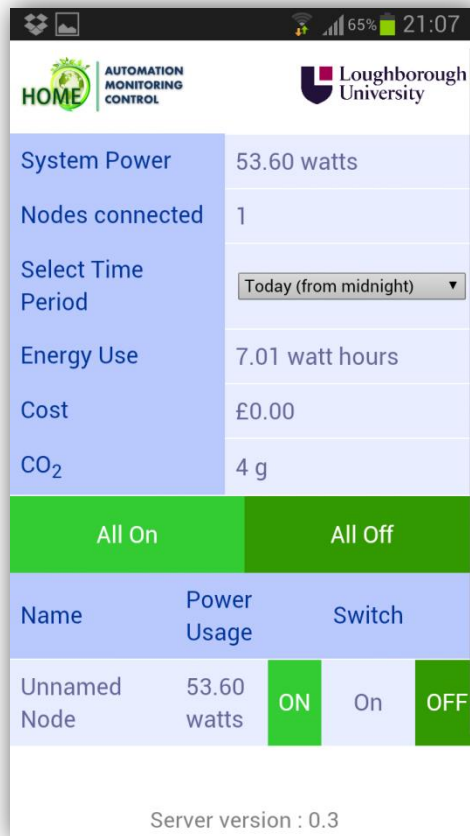


Figure 13 – Mobile version of the overview page

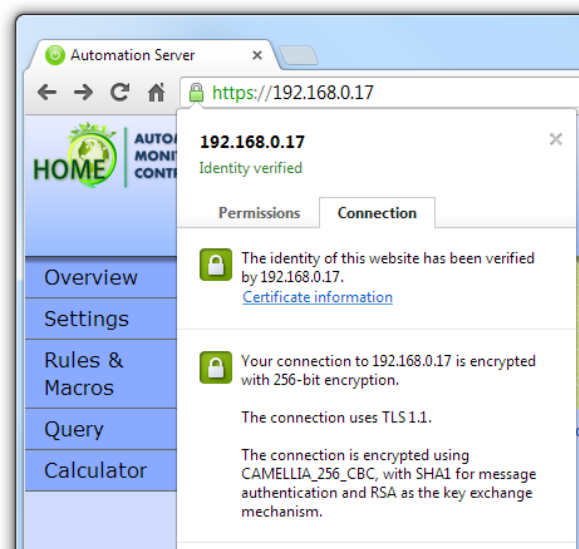


Figure 14 – HTTPS connection information in Google Chrome

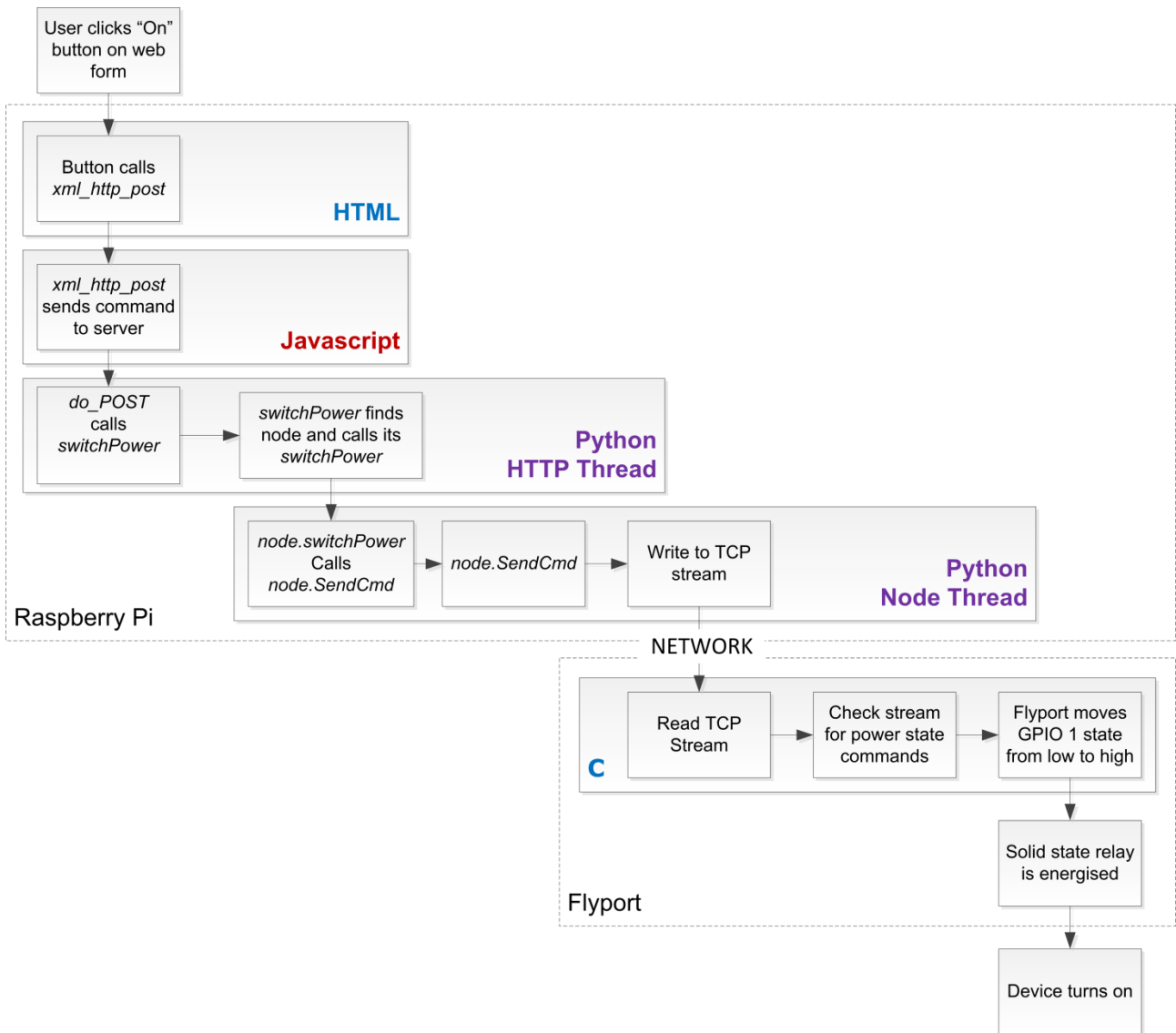


Figure 15 - Power switch input / output path

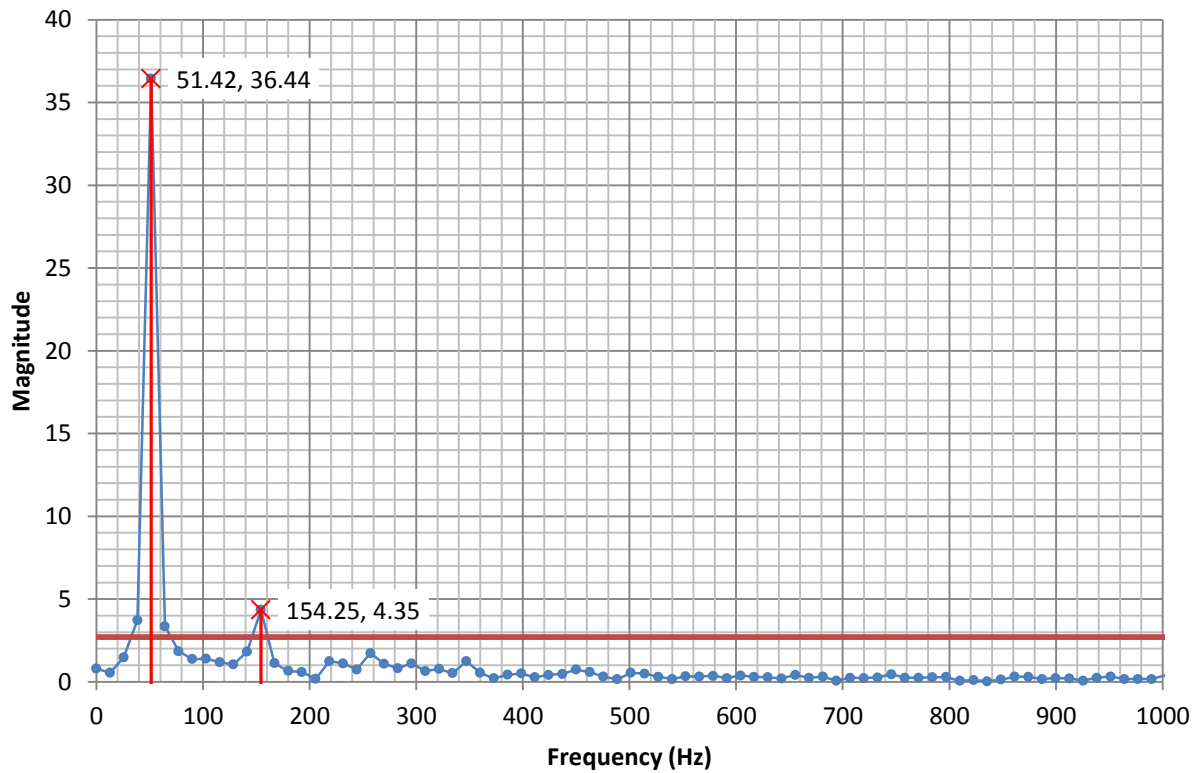


Figure 16 – Frequency domain of test signal

## Devices

Lights, Fans, AV Equipment etc

This system can try to identify connected devices based on the way they use power.  
(Device identification only works with devices that draw a static load.)

Device is currently connected to Node	Unnamed Node
Device Name	Low Energy Light
Please plug the device into the node, ensure it is switched on (node will turn on automatically) and press Start	<input type="button" value="Start"/>
Progress	<div style="width: 100%; height: 10px; background-color: green;"></div>
Frequencies	<div style="border: 1px solid gray; padding: 5px;">           49.98Hz @ 8.37            159.92Hz @ 4.32         </div> <div style="margin-top: 5px;"> <p>These are the major frequencies your device induces</p> </div>

Figure 17 - Device profiling tool



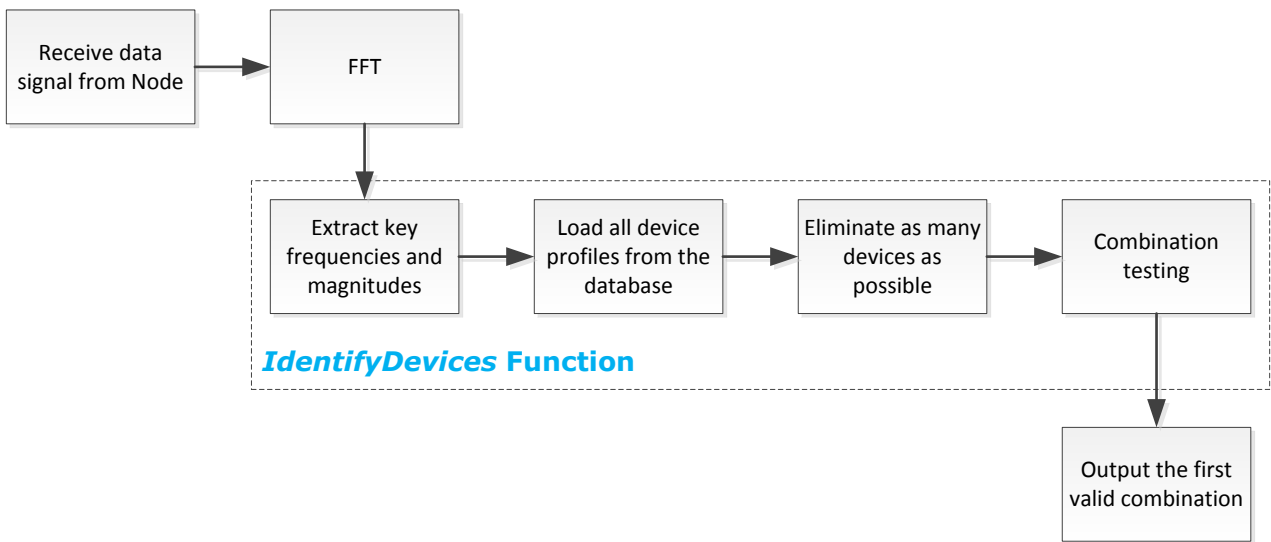


Figure 18 - Device recognition procedure

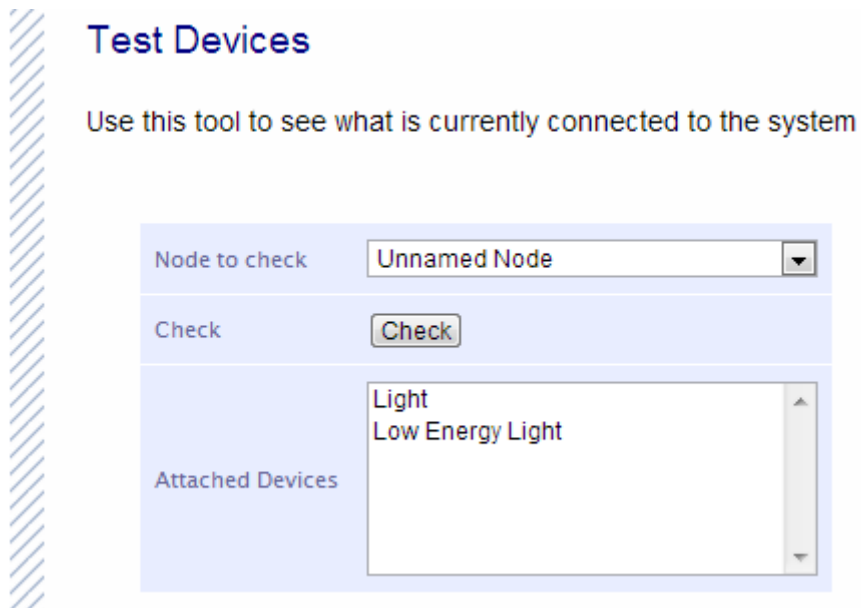


Figure 19 – Device recognition test tool

Potential <i>Device</i> 1	Potential <i>Device</i> 2	Bit encoded string (binary)	Integer Value of bit encoded string
Off	Off	00	0
Off	On	01	1
On	Off	10	2
On	On	11	3

Table 1 – *Device* states