Tampere University of Applied Sciences

IoT-Smart Grid Nexus

Home Energy Metering System using Wifi enabled Raspberry Pi, Sensor and User Interface

Saliu Ibrahim Shehu

MASTER’S THESIS
May 2019

Degree Programme
Information Technology

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree Programme
Information Technology

SALIU IBRAHIM SHEHU:
IoT-Smart Grid Nexus
Home Energy Metering System using Wifi enabled Raspberry Pi, Sensor and User Interface

Master's thesis 37 pages, appendices 6 pages
May 2019

A key role of Information Technology (IT) to Sustainable Development (SD) is the promotion of energy consumption awareness that could engender changes in the behavioral pattern of people dependent on energy at far lesser cost than previously known. Protecting the Planet for People continual survival while maximizing Profit had since being a deep concern in the IT world. A solution to use energy efficiently by monitoring; to make informed decisions about the way energy is use and can be manage, formed the theme of this thesis.

This thesis, in part, uncovered the link between Smart Grid technology/meter and IoT technology/energy meter; and further illustrated proposed implementation of a simple IoT-based electricity metering, which can be displayed via an Open Energy Application Programming Interface (API), using amazingly inexpensive hardware components and open source software with none of the complexities associated with present and past similar systems. In other words, proposing an implementation methodology with an open design approach for monitoring and managing energy using IoT-based technology.

This system consists of three main components, namely: A Raspberry Pi (wifi enabled), Sensor (sensing circuitry) and a User Interface (UI). The sensor sensed the pulses as emitted from a modern electricity meter and then the signal or digital information sent to the General-Purpose Input-Output (GPIO) pins of the Raspberry Pi. The Raspberry Pi, on receiving the signal at it programmed input pin, calculates the power/energy value and then sent this data across the internet to the web/app API for easy users' access.

Users can at this point, access the amount of energy consumed in minutes, hours, weeks or even in months via their smartphones, laptops and or tablets. This will equip and enable energy consumers to independently monitor their energy consumption and make better choices to use energy efficiently and help reduce energy wastage while saving cost.

Key words: internet-of-things, smart grid, raspberry pi, sensor, user interface, smart energy meter, iot energy meter, emoncms, people, profit, planet

CONTENTS

ABBREVIATIONS AND TERMS LIST

ADC	Analogue to Digital Converter
AMI	Advanced Metering Infrastructure
AMR	Automated Meter Reading
API	Application Programming Interface
Cr.	credit
CT	Current Transformer
EmonCMS	Energy monitor Content Management System
ETS	Emission Trading Scheme
GND	[Electrical] Ground
GPIO	General Purpose Input Output
GSM	Global System for Mobile [communication]
IBM	Internet Based Metering
ICT	Information and Communication Technology
Imp	Impression or Impulse
IoT	Internet-of-Things
IT	Information Technology
IP	Internet Protocol
kWh	kilo-Watthour
LCD	Liquid Crystal Display
LDR	Light Dependent Resistor
LED	Light Emitting Diode
NOOBS	New Out Of Box Software
OS	Operating System
PLC	Power Line Carrier
RPi 3	Raspberry Pi 3
RPi.GPIO	Raspberry Pi. General Purpose Input Output
SD	Sustainable Development
TAMK	Tampere University of Applied Sciences
UE	User Experience
UI	User Interface
W	Watt
Wi-Fi	Wireless-Fidelity

1 INTRODUCTION

Finland is one of the few countries, in the world, in which private individuals, homes, and offices could fill-out and sign a personalized contract with electrical energy utility companies and be responsible for the bills over a time cycle. Bills payment cost, however, may depend on several fixed and variable factors such as selected energy source mix. Different countries may adopt different billing approach. The one sure variable factor for all approaches is the amount of energy consumed. Energy consumption can be monitored for control and management for different reasons. Two key reasons are reduction in carbon footprint and energy consumption cost.

Efforts to achieve reduction goals in carbon footprint and energy consumption cost for a sustainable future have, in recent years, seen Information and Communication Technology (ICT) helping to transform traditional systems of electrical grids into smart ones. This new grid of electrical network systems worldwide is known as Smart Grid.

The focus of this thesis work is to reveal the nexus, that exist, between IoT and Smart Grid technologies in context of electrical energy metering and to demonstrate the proposed implementation of a personal Home-Office energy consumption metering model that can be monitored via a User Interface (UI) available either as web-based or mobile apps interface, accessible through portable devices such as laptops, tablets and smartphones. The purpose is to provide an unfettered access to energy consumption database that could impact the cost of such energy. It is expected that this solution would provide a seamless and friendly User Experience (UE) that can be extended to all and sundry energy grid-connected consumers using IoT Technologies. It is my hope that this will help create energy consumption awareness and enables customers to make informed decisions about the way they manage and/or control energy use.

The rest of the chapters uniquely address different topics as follows: A brief preview on Smart Grid, IoT and energy metering systems is presented in chapter 2.

Chapter 3 describes the proposed implementation of the IoT energy metering system design. Chapter 4 describes advanced energy metering systems in general, dealing with the nexus that exist between IoT and Smart Grid energy metering. Hardware setup and software configuration form the central theme of chapter 5, included also are the result analysis and its implications. This is followed by a comprehensive discussion on the resulting outcome and possible applications in chapter 6; and finally, a concise conclusion is presented in chapter 7.

2 IOT-SMART GRID AND ENERGY METERING SYSTEMS PREVIEW

2.1 Smart Grid Systems

Smart Grid entails the augmentation and integration of renewable energy sources such as wind, sun, water and biomass into existing fossil sources network and the injection of intelligent decision-making control apparatus (Ekanayake, 2012). For example, it is possible to have a micro grid as an island or isolated grid system consisting of renewable energy generators as wind farms, solar panels, small hydro power generators and bioenergy power generators interconnected to an intelligent control system to connect and disconnect to a national grid with mainly fossil fuel generators.

This kind of setup and arrangement could impact and enable communities to take control of their energy use and significantly reduce their carbon footprint. Basically, what ICT does in this respect is to provide the necessary and required intelligence control system for disconnection and connection in the advent of any undesired events or circumstances. In other words, using digital technology to improve the reliability, security and efficiency of systems of electrical networks (TUT, 2017).

2.2 Internet-of-Things

Internet-of-Things (IoT) is the identification and interconnection of electronic devices and sensors connected to exchange information over and across the internet (Philip et. al., 2017). The introduction of IoT hold the promise to make the way and manner, by which energy generated and consumed, becomes smarter, more measurable and interactive. Steve Collier, IEEE Smart Grid Technology Expert and Director of Smart Grid Strategies of Milsoft Utility Solution, concluded that IoT with its reliability and ability to handle a great magnitude, possibly billions of endpoints, is the solution to our energy need. IoT is a self-regulating process tool that could be key to Energy, Economics and Environmental security and stability.

According to Bob Metcalfe, inventor of Ethernet: IoT is capable of meeting world needs for cheap and clean energy by building the Enernet (Internet-of-Energy Grids). One key role of IoT with regards to energy consumption is it capability to use wireless network to send instantaneous power/energy consumption data to remote server for storage and analysis. Analyzing this data could help identify critical electrical power management related issues (Jadhav & Rajalakshmi, 2017).

2.3 Energy Metering Systems

IoT-based and Smart energy metering systems are a recent switch in trend transitioning from traditional (analog) energy meter involving electromechanical dial and modern (digital) electricity metering involving pulse count. Traditional energy meters are inherently limited in their function and construction giving rise to issues in reading accuracy, reliability and sub-metering.

Furthermore, convectional energy meter reading is cost and labor intensive. The use of vehicles by field personnel for the purpose of monitoring, reading, and managing electricity meters have the propensity of increasing the surrounding carbon footprint substantially. Modern energy meter with it many advantages over the traditional ones usually involve huge starting capital. They are exclusively setup and controlled by electrical utility providers.

2.3.1 Smart Energy Meters

Smart Energy Meter has been an integral part of smart grid since the inception of smart grid (Anushree & Shanthi, 2016). It is often referred to as Automated Meter Reading (AMR) System (Satish et. al., 2015; Chandra et. al., 2016; Philip et. al., 2017). In fact, since smart grid and smart meter remain inseparable and with growing development to AMR; the term AMI (Advanced Metering Infrastructure) is seldom used instead of AMR (U.S Department of Energy & Office of Electricity Delivery and Energy Reliability, 2016). AMI or smart metering often includes hardware and software systems that collect, measure, and analysis energy usage, and communicate with metering devices either on request or on a schedule (Vignesh & Sathish, 2016). The ensued communication is often a two-way communication between the meter and the utility company network infrastructure which may be wireless, or fixed wired connections such as Power Line Carrier (PLC) (Satish et. al., 2015).

2.3.2 IoT Energy Meters

A new breed of metering system, often referred to as IoT-based metering system, is making energy measurement a lot easier to setup, smarter to measure, overwhelmingly interactive and consumer-focused; such that energy consumption information can be accessed by consumers without the help or intervention of the utility providers (Devagkumar & Chiragkumar, 2016).

This new breed of system is taking a step further in utilizing available wireless communication options as used in AMI, such as cellular communication (e.g. GSM), ZigBee, Wi-Fi, and Bluetooth, by identifying connected devices by their IP addresses; thus adding intelligence to existing infrastructure (Al-Ali & Aburukba, 2015; Satish et. al., 2015; Kurde & Kulkarni, 2016). The overarching advantages of this are in the cost, global reach/accessibility and huge wealth of information made available for further analysis for informed decisions that could positively impact billing cost, energy security against theft and efficient use of energy. Thus, the energy metering provided by smart grid is called AMR, AMI or smart metering; those provided by IoT can be referred to as web/mobile app metering.

3 SYSTEM DESIGN AND DESCRIPTION

3.1 Digital Energy Meter

As can be seen in figure 1 the measured electrical energy is provided directly from the digital energy meter through a flashing or blinking Light Emitting Diode (LED) rate or pulse count. The flashing rate of this LED is proportional to the amount of electrical energy passing through the meter. The unit of electricity that passes through this electronic energy meter is usually measured in impression (or impulse) per kilowatt-hour (i.e. Imp/kWh) (EKM FORUM, 2015). Picture 1 shows a meter with 800 Imp/kWh as indicated by the pointed red arrow. However, most of this kind of digital meters are labelled 1000 Imp/kWh which indicates that the LED flashes 1000 times for each KWh of electricity that passes through it.

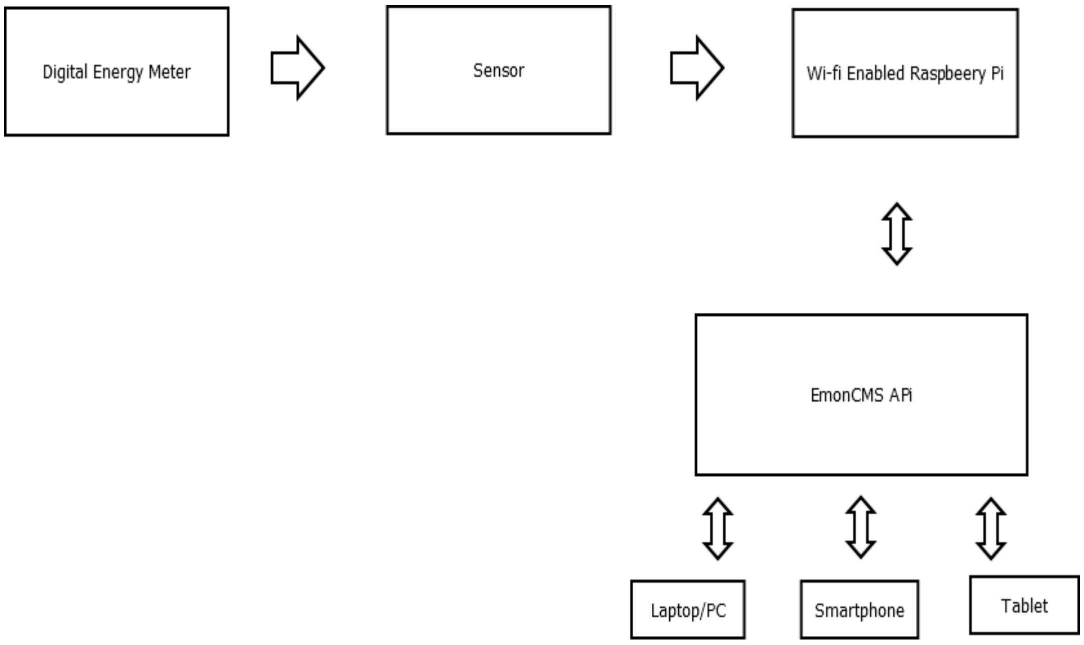


FIGURE 1. System Block Diagram

The underlying key requirements that determine the overall success of this design implementation methodology is the fact that there is an optical pulse counting from the blinking LED and the given tagged or marked number of Imp/kWh (see picture 1). Without these, there is nothing to measure or read from. It is from these parameters that the power was estimated and sent every 60 seconds to the **E**nergy **mon**itor **C**ontent **M**anagement **S**ystem (EmonCMS) API. 1000 Imp/kWh was used.

The practicality of the 1000 Imp/kWh used in this implementation is that 1000 W (1 kW) passes through the meter every hour (3600 seconds) causing the LED to flash 1000 times. The implication is that power is proportionally equals to the number of times LED flashes (i.e. the pulse count). In order to determine the power through same meter every 60 seconds the following deduced formula was used:

$$P = PC * FT, \qquad (1)$$

P is power (instantaneous)
PC is pulse count
FT is flash time (time interval to complete a pulse cycle).

Flash time interval is the set time of 60 seconds divided by the meter's Imp/kWh value. In other words, it is the time taken per flash. For a meter with 1000 flashes:

$$FT = \frac{60}{1000} = 0.060\ seconds$$

For a meter with 800 flashes:

$$FT = \frac{60}{800} = 0.075\ seconds$$

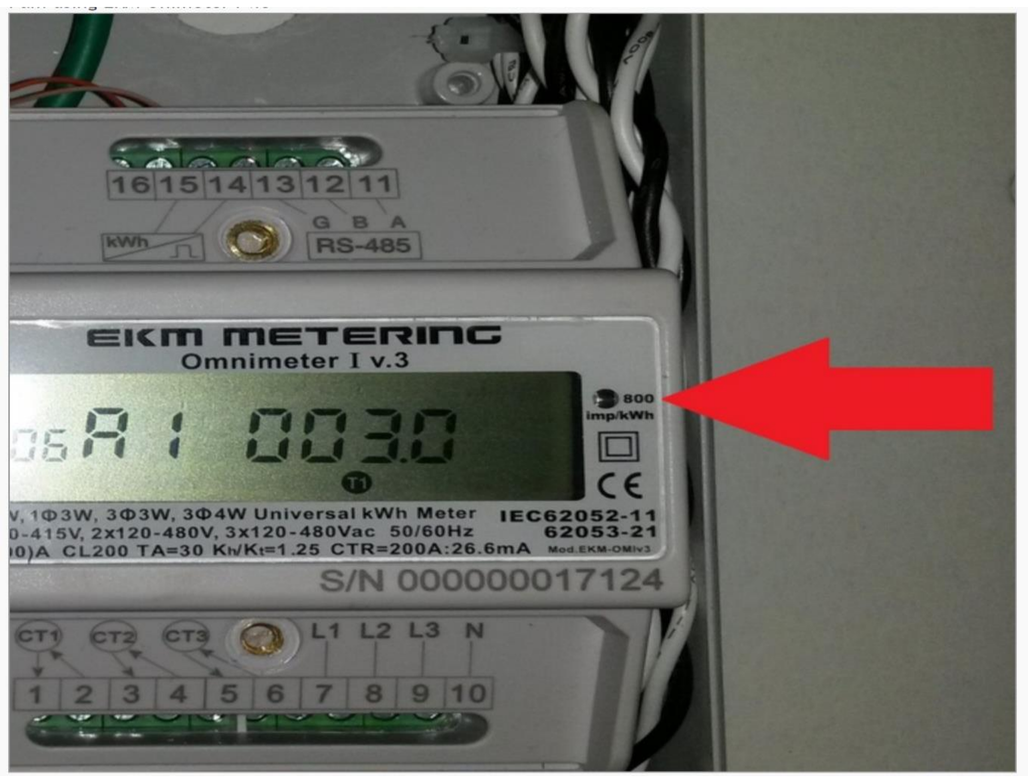
For a meter with 2000 flashes:

$$FT = \frac{60}{2000} = 0.030\ seconds$$

In this case, the LED would flash once every 60ms (i.e. 0.06 seconds). Multiplying this value with the pulse count would indicate the power measured by the meter per flash. Thus, equation 1 becomes:

$$P = PC * 60 \quad W$$

The number of pulses or pulse count is counted by the Raspberry Pi program counter and used in the determination or calculation of power that would be displayed in the EmonCMS API.



PICTURE 1. Imp/kwh mark on digital energy meter (Photo: EKM FORUM, 2015)

3.2 Raspberry Pi

Raspberry Pi is an open source single board computer with its own Microprocessor running on an OS (Linux Distro) unlike Arduino which is considered a Microcontroller unit. Microcontroller devices have no operating system and can only run a single program or sketch at a time, unlike Raspberry Pi that is a fully functional cheap computer specifically designed for a variety of advance applications such as IoT, environmental sensing/monitoring, home automation, gaming, cloud server, media center and security monitoring (Binary Updates, 2017).

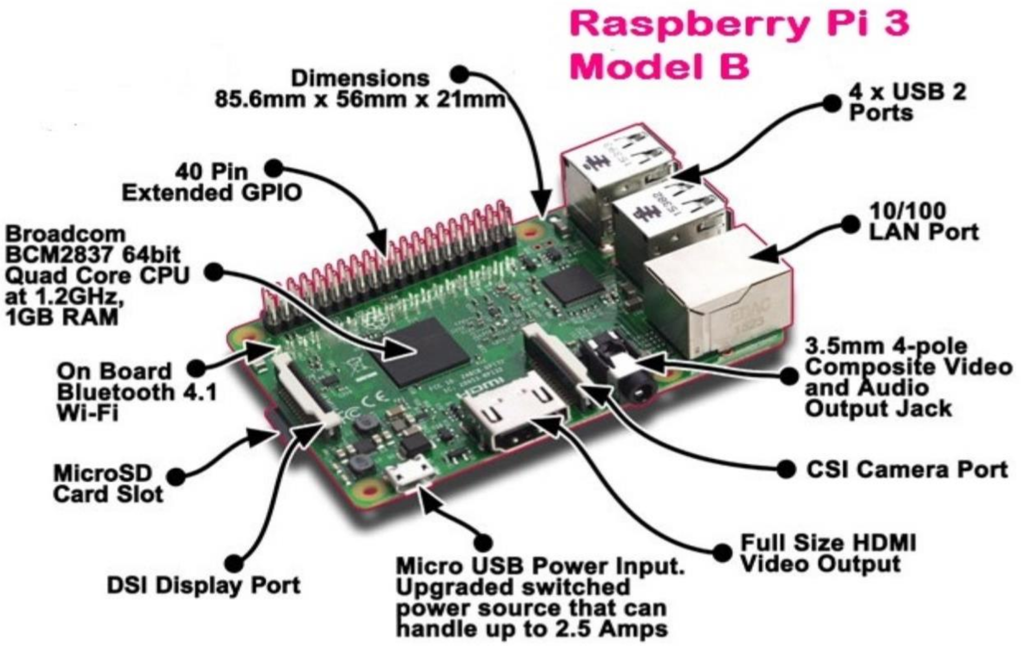


FIGURE 2. Raspberry Pi 3 Model B (Binary Updates 2017, Reprinted)

The Raspberry Pi used in this implementation is a Raspberry Pi 3 Model B (figure 2), with built-in Wifi and Bluetooth. It is accompanied by a 16 GB Class 10 MicroSD Card preloaded with Raspbian OS and NOOBS. It interacts with the external environment via it GPIO. The details of these are presented in the proceeding chapter.

3.3 EmonCMS API

This is an open-source web-app for processing, logging and visualizing energy, temperature and other environmental data. It is part of an OpenEnergyMonitor project intended for open source monitoring for understanding energy and exploring the context of renewable energy and zero carbon (OpenEnergyMonitor, n.d.) Details of the configurations and requirements are available in the proceeding chapter.

In this implementation, this web app was used as a simple home energy monitor (figure 3) for understanding energy consumption remotely; with added advantage of being able to easily access energy consumption data from anywhere on the web without having to open access to local Raspberry Pi.

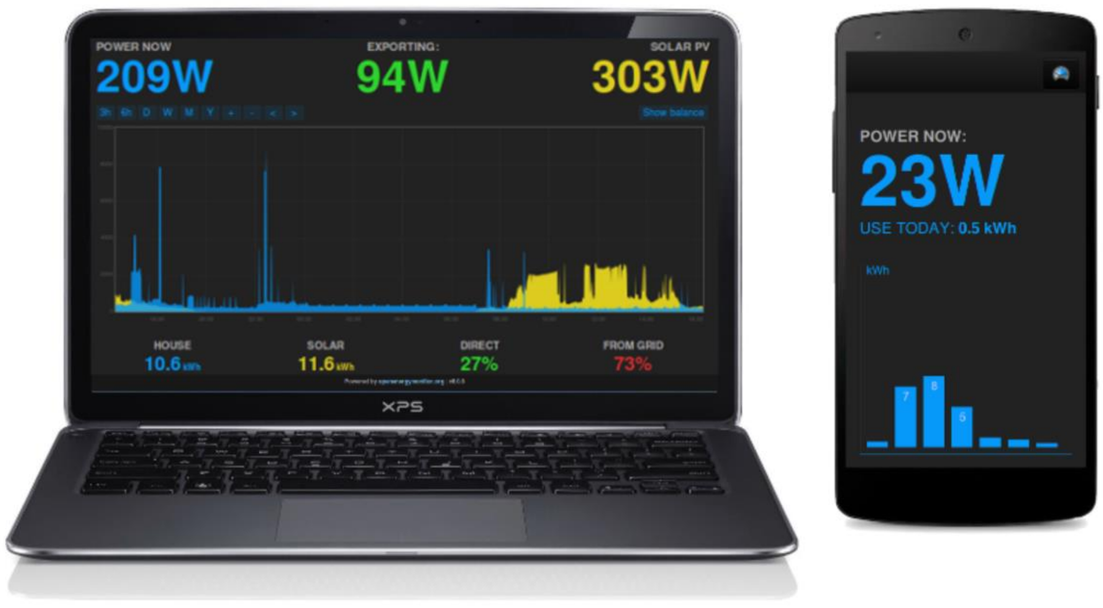


FIGURE 3. EmonCMS API display on laptop and smartphone (OpenEnergyMonitor, n.d.)

3.4 Sensor

A Light Dependent Resistor (LDR) was used to automate the process of counting the LED flashes emanating from the digital meter such that the derived information was logged for use in the Raspberry Pi and then pushed forward to the EmonCMS API for further analysis. The sensor detailed circuitry setup for the Raspberry Pi control is available in the proceeding chapter.

4 ADVANCED ENERGY METERING SYSTEMS

Although, the bifurcation point between smart energy metering and IoT energy metering systems is not completely clear. It is however clear that smart metering support smart grid approach and has become an essential part of it. IoT energy metering, on the other hand, are web-app based and becoming more popular, providing simple upgrades on existing smart grid meters rather than complete replacement (Hudedmani et. al, 2019). Consequently, fewer literature exist now for IoT based energy metering systems compared to smart energy metering systems. Presently, there is an overall dearth in publications with regards to both.

4.1 Existing Systems

Existing energy metering systems, as revealed by studied publications and articles, were quite similar in their approach. The differences were in the hardware used which were based purely on choice, technological advancement, aims and objectives to be achieved. The approach has been to measure voltage and/or electrical current using current transformer (CT) sensor; or electrical energy directly via pulse count using light sensor. These quantities are then manipulated via programming in either a microcontroller or microprocessor to determine the level of energy consumption in near real-time fashion and the final information is displayed on some various display unit or user interface. Additional information could be displayed depending on the goals in mind.

Hardware choice is crucial. The common goal is to establish a link to the measured electrical energy quantity and the final output unit. Depending on the intended use, additional hardware such as GSM module, Liquid Crystal Display (LCD) and theft detection/avoidance circuits could be added. Commonly used Hardware are current sensor (CT) for current/power detection/measurement; light sensor for pulse count detection; microcontroller (e.g. Arduino) or microprocessor (e.g. Raspberry Pi) for energy calculation or manipulation; and connectivity module such as ESP8266: a wifi module for internet or cloud connectivity.

As obvious from studied materials, most systems aims and objectives are to monitor energy consumption remotely for the purpose of controlling users' consumption pattern or level; for proper load management; theft and meter tampering detection and avoidance; submetering; billing mistakes and cost reduction; labor or manpower reduction; avoidance of reading mistakes and cost; and carbon emission reduction.

4.2 Proposed System

The proposed system is aimed at avoiding dealing with complex communication protocols and etiquette programming as well as the avoidance of hardware such as Analog-to-Digital Converter (ADC) and the use of separate wifi module (i.e. ESP2866) to practically achieve the same aims and objectives of most existing systems while preventing overheads.

The proposed system uses LDR to sense or detect light pulses or flashes from a digital electricity meter (Perone, 2012; Chandra et. al., 2016). By the way, in the absence of a modern digital meter, a Laser Diode on Arduino can be used as an emulator for demonstration purposes (See appendix 1 and 2 for wiring schema and Arduino programming code to blink the laser module respectively). The sensed digital information is then absorbed and manipulated by the wifi enabled Raspberry Pi 3 (RPI 3) via the connection link between the LDR and GPIO of the RPI 3. The received information is processed such that the resulting outcome can be visualized locally or globally via data logging request made to remote web server.

The target of this proposed implementation design is the end users, that is, the final entity in the chain, from electricity generation to utilization: the consumer of electricity. The introduction of low-cost microcomputer such as wifi enabled Raspberry Pi has made this achievable via a platform for interconnecting electrical/electronic devices and sensors in homes and small offices over the internet. This arrangement solely put consumers at a vantage position to monitor and manage their energy consumption and it attendant cost as well as positively impacting the carbon footprint of the surrounding environment.

5 HARDWARE SETUP AND SOFTWARE CONFIGURATION

5.1 Hardware and Software Requirements

Below is the unordered list of the essential parts of the model system for the proposed implementation purpose:

- Raspberry Pi
- Raspbian OS
- Python
- C-Language
- Light Dependent Resistor (LDR or Photoresistor)
- 10k Ohm resistor
- 1k Ohm resistors (2)
- BC547B/BC847C (or any general purpose NPN Transistor)
- Cables/Connectors
- Digital Electricity Meter
- EmonCMS

5.2 Photoresistor Circuitry Setup for Raspberry Pi GPIO Pin Control

As can be visualized from figure 4 both LDR and R2 resistor together work as a voltage divider. When the light from the digital meter or laser is off, the LDR resistance remains very high (as high as 2 to 10 kilo ohms), making the voltage that goes to the base of the transistor Q1 very low and not enough to activate the GPIO pin 4 of the Raspberry Pi whose state is at this point HIGH. But when the laser of digital meter is on or blinks, the LDR becomes very low (as low as 200 to 1000 ohms), making the voltage at the base of the transistor high enough to switch on the transistor and activating the GPIO pin 4 of the raspberry Pi allowing current to flow to the ground (GND) thus making the GPIO pin 4 to go LOW.

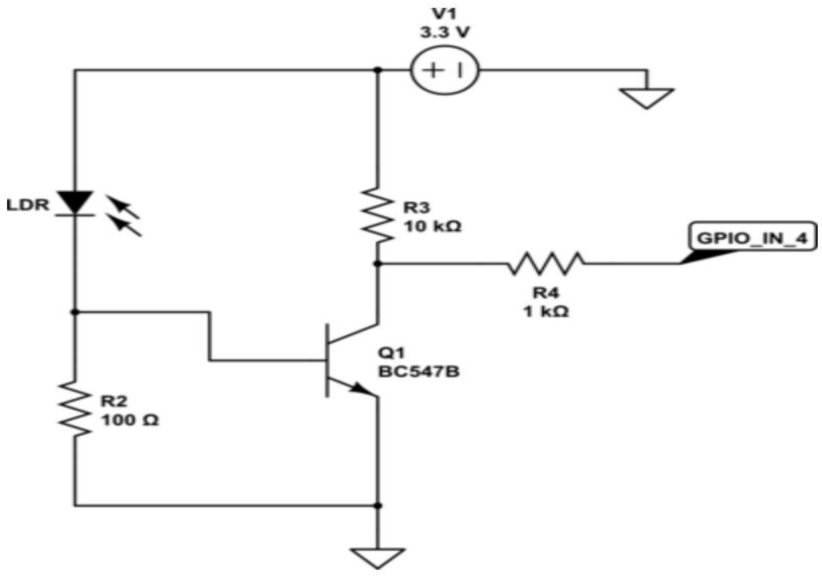


FIGURE 4. LDR circuitry schema showing control input into the pin 4 of Raspberry Pi GPIO (Perone, 2012)

The generated control sigmoid function signal derived from simulation using online circuit lab service can be seen in figure 5.

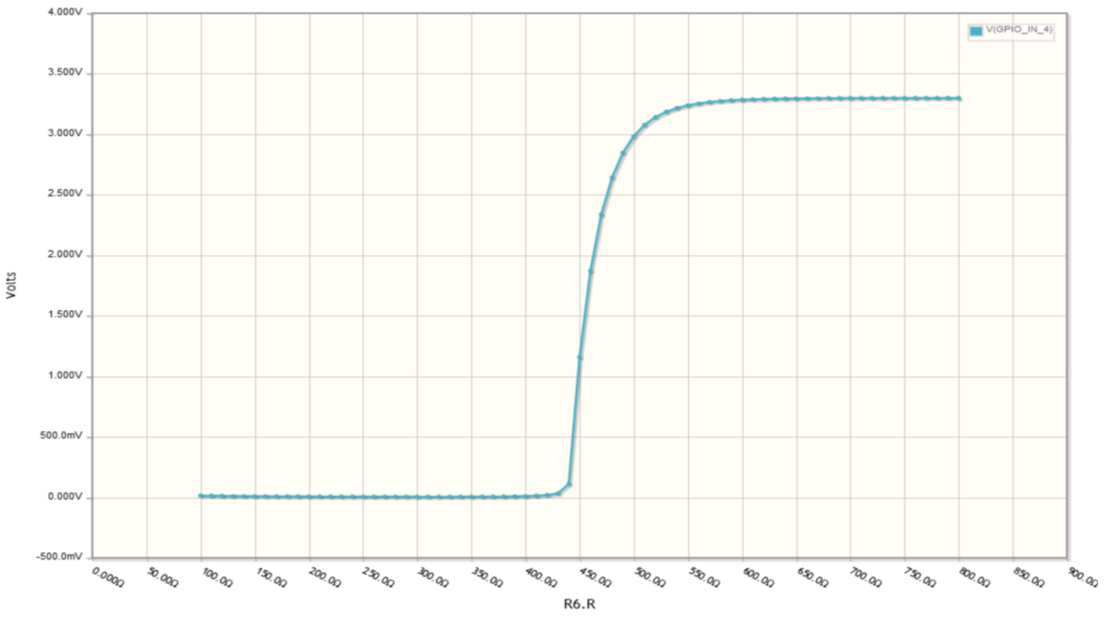


FIGURE 5**.** Graph showing LDR resistance against GPIO pin 4 voltage (Perone, 2012)

The voltage threshold was achieved by setting the voltage divider bottom resistance, R2, to 100 ohms; this can be adjusted in accordance to the designed purpose or use. Calculating the LDR resistance threshold value, we have that: 500 ohms*(0.7v / 3.3v) = 106 ohms (approximately 100 ohms)

5.3 Reading the GPIO State from Raspberry Pi using Python

Below are commands used to *install* and/or *update* the Raspberry Pi-General Purpose Input/Out (i.e. RPi.GPIO) Python module before setting GPIO as INPUT and reading the GPIO pin 4 of the Raspberry Pi. The used code to *read* and *show* the GPIO pin 4 value can be found in appendix 3. The Python module used for this purpose was that created by Ben Croston (Croston, 2013). The RPi.GPIO module is installed by default in Raspbian OS.

To ensure the Python module is at the latest version the following steps may be necessary:

```
$ sudo apt-get update
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

To install the latest development version from the project source code library:

```
$ sudo apt-get install python-dev python3-dev
$ sudo apt-get install mercurial
$ sudo apt-get install python-pip python3-pip
$ sudo apt-get remove python-rpi.gpio python3-rpi.gpio
$ sudo pip install hg+http://hg.code.sf.net/p/raspberry-gpio-
python/code#egg=RPi.GPIO
$ sudo pip-3.2 install hg+http://hg.code.sf.net/p/raspberry-gpio-
python/code#egg=RPi.GPIO
```

To revert back to the default version in Raspbian:

```
$ sudo pip uninstall RPi.GPIO
$ sudo pip-3.2 uninstall RPi.GPIO
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

For other distributions it is recommended that one install RPi.GPIO using the pip utility as superuser (root):

```
# pip install RPi.GPIO
```

5.4 Software Installation

The used application codes and scripts were contained in the **power** folder that was forked from GitHub (Kieranc, 2017). This folder contains essential and related files and source codes required for the proper functioning of the monitoring process.

The file named ***power-monitor.c*** (see appendix 4) automatically starts data logging on boot and stops on shutdown. The Python application, ***monitor.py*** (see appendix 5), counts pulses, and each minute creates power reading in watts which it sends to the EmonCMS API. Yet another simple C app (see appendix 6) is required to listen for an interrupt triggered when LDR detects a pulse. This was necessary due to inability of Python to respond to an interrupt.

The C source code named ***gpio-new.c*** (appendix 6) was adapted and simplified from the isr.c example distributed with wiringPi by Gordon Henderson (Henderson, 2013). This app needs compiling like this:

```
gcc   gpio-new.c   -o  gpio-new  -lwiringPi
```

It is important to make this file accessible and in conformity with the path specified at the bottom of monitor.py (i.e. line 74 of appendix 5).

5.4.1 Command Sequence as used on the Raspberry Pi

The commands below install the advanced Python Scheduler needed by the monitor.py code (appendix 5).

```
sudo  apt-get  install  python-dev  python-pip
sudo  pip  install  apscheduler
```

Following the above commands, the subsequent commands download the required files from GitHub repository, change directory to the **power** folder, copies the power-monitor file to the initialization folder where the logging process can be started and killed, and then changed the mode of access to the power-monitor file now in the initialization folder and finally update this folder to make the power-monitor the default file.

```
sudo  apt-get  install  git
git  clone  https://github.com/kieranc/power.git  &&  cd  power

sudo  cp  power-monitor  /etc/init.d/
sudo  chmod  a+x  /etc/init.d/power-monitor
sudo  update-rc.d  power-monitor  defaults
```

An important step is to check the power-monitor file to ensure the path to the Python application (i.e. monitor.py: appendix 5) matches with the path on the Raspbian OS System.

Once all this is completed including the C app compilation mentioned earlier in the chapter, the data logging process can be started as follows:

```
sudo   /etc/init.d/power-monitor  start
```

5.4.2 EmonCMS and Energy View

With the Raspberry Pi running emoncms app locally, data logging requests are made to remote emoncms web server via an API key as seen in the monitor.py code (line 69 of appendix 5). The configuration of the monitor.py script in relation to emoncms is to submit its output only to the EmonCMS API (a remote version of the local emoncms app is accessible via emoncms website).

Outputs from the monitor.py script are logged as inputs unto the EmonCMS API. These inputs are then processed and visualized, based on configuration settings

(Lea, 2019), and made available for the users to see their up to date energy usage and historical data. Figure 6 is a sample of the energy view showing a moving window with power consumption in the top half and historical daily energy consumption in kWh in the bottom half.

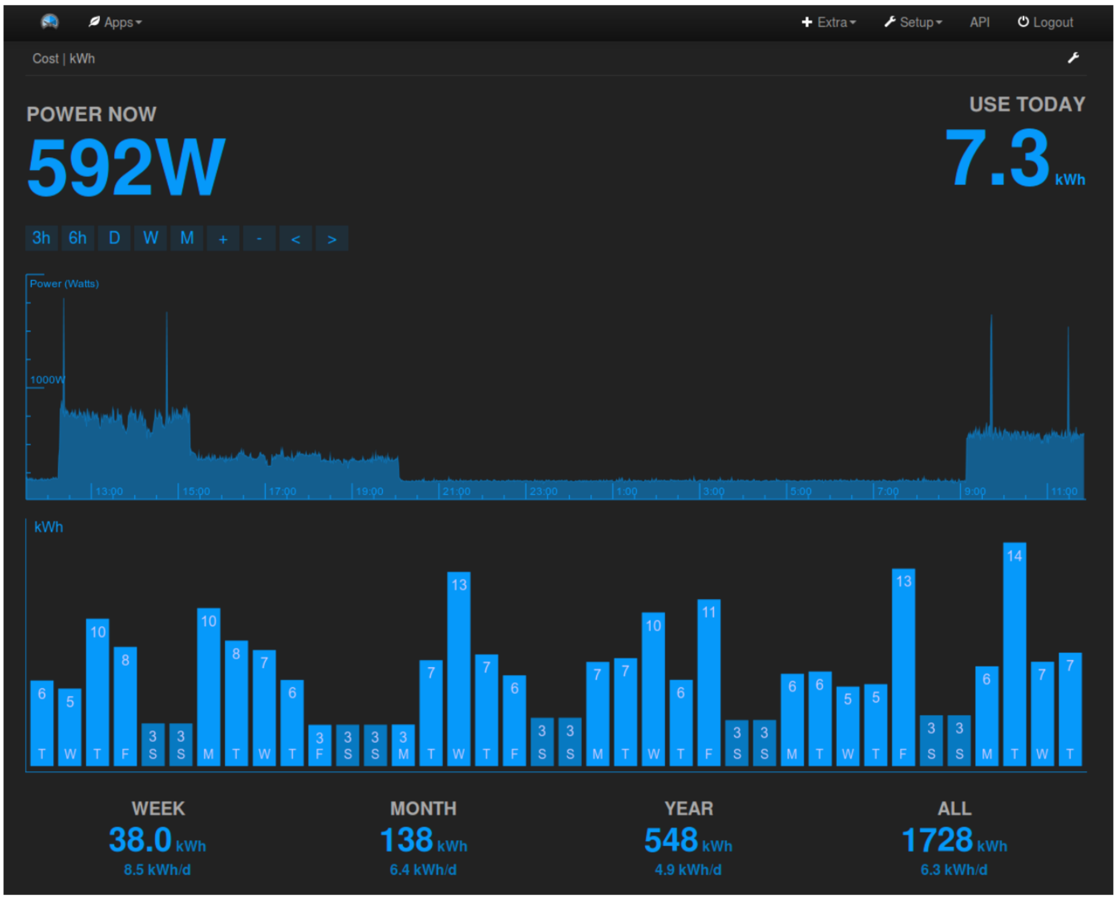


FIGURE 6. Energy view (Reprinted from OpenEnergyMonitor Home Energy Guide)

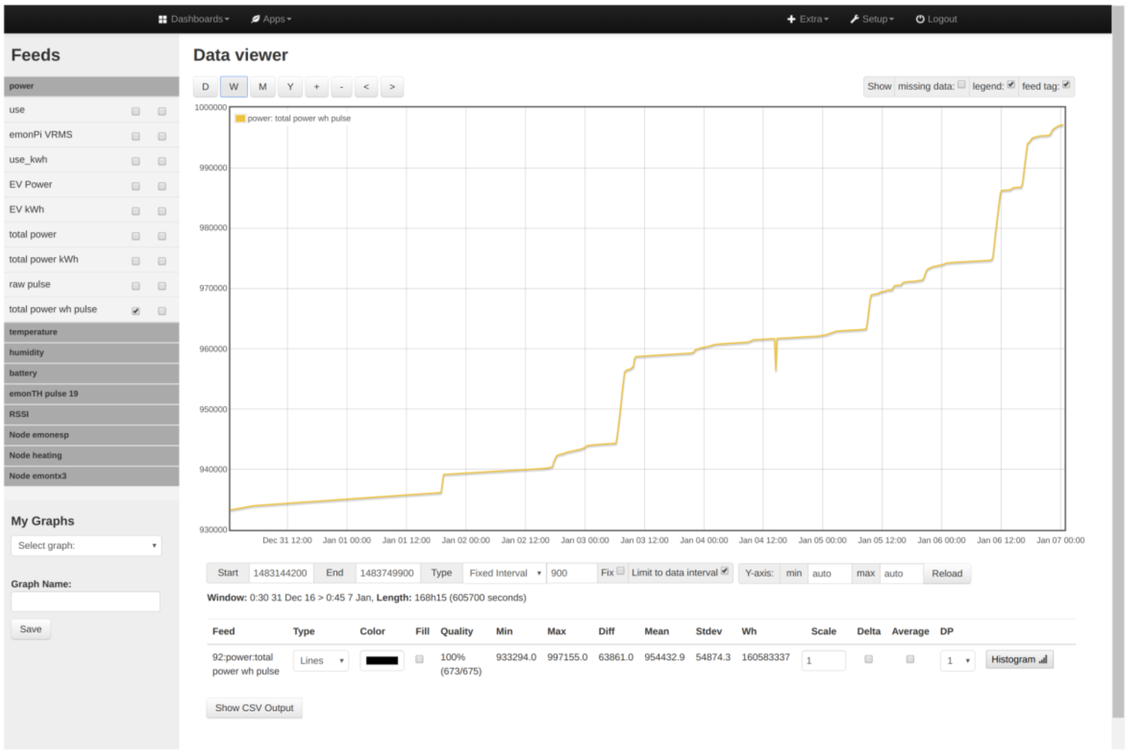
It is important to note that Emoncms installation and configuration setup are outside the scope of this work. This information can be gotten from GitHub and Open Energy Monitor websites: “Install Emoncms on Raspberry Pi.”

5.4.3 Result Analysis

Figure 6 represents an emoncms energy output view showing fed-in power and energy consumption varieties. Uniquely displayed are the hourly, daily, weekly, monthly and yearly energy consumption values with hourly and daily averages.

Depending on the needs or requirements, the emoncms can be configured to provide the least granular information about cumulative energy consumption, a feature rear for most application programming interfaces.

The power value, displayed at the uppermost left-hand corner, represents the instantaneous power output from the Raspberry Pi. The energy value at the uppermost right-hand corner is the energy consumption value calculated over time from the cumulative power input. The emoncms does this through the inbuilt "Power to kWh" input processor. Figure 7 represents the background raw data cumulative power conversion calculation view.



Start 1483144200 End 1483749900 Type Fixed Interval 900 Fix ☐ Limit to data interval ✓ Y-axis: min auto max auto

Window: 0:30 31 Dec 16 > 0:45 7 Jan, **Length:** 168h15 (605700 seconds)

Feed	Type	Color	Fill	Quality	Min	Max	Diff	Mean	Stdev	Wh	Scale	Delta	Average	DP
92:power:total power wh pulse	Lines	■	☐	100% (673/675)	933294.0	997155.0	63861.0	954432.9	54874.3	160583337	1	☐	☐	1

FIGURE 7. Raw data wh accumulator view (Reprinted from OpenEnergyMonitor Home Energy Guide)

Table 1 from similar works suggests that the values derived from IoT enabled power monitoring readings are essentially no different from those gotten from traditional or conventional analogue or digital power meters. This is an indication that IoT based energy meters, potentially, have very good accuracy for power monitoring (Jadhav & Rajalakshmi, 2017). This further confirmed the veracity of the final output power and energy readings that could be generated from this proposed system when implemented.

Table 1. Comparison of IoT meter power readings & Actual values (Adapted from Jadhav & Rajalakshmi, 2017)

Actual Value (watt)	Meter readings (watt)	% Error
20	19.92	0.40
30	30.16	-0.53
40	39.60	1.00
50	49.90	0.20
60	59.90	0.17
70	70.40	-0.57
80	80.20	-0.25
90	89.70	0.33
100	100.10	-0.10
110	111.60	-1.45
120	120.70	-0.58
130	131.00	-0.77
140	141.30	-0.93
150	151.80	-1.20

5.4.4 Result Implications

Not only is the final energy output supported on both mobile and web interfaces; emoncms was specifically designed as a powerful open-source web-app for logging (both locally and remotely), processing and visualising energy. Thus, it is well suited for energy metering capable of energy cost inclusion and available at virtually no financial cost or commitment. Capacity also exist for submetering of every standalone device in homes and offices. Thus, providing information about the least/most energy efficient devices.

Results displayed on emoncms show both real-time and historical values at a glance which are valuable assets and or tools to both electricity utility providers and consumers. This information is required for electricity consumption cost calculations and for electricity consumption decisions making which could positively impact efficient use of energy.

Consumers’ direct and unfettered access to this information could substantially influence energy consumption pattern which could lead to cost reduction in electricity bills and effectively end the era of estimated bills which are often not in favour of the consumers as well as afford the consumers a high level of control and oversight over their energy consumption and billing prerogative.

Government and or energy provider could take advantage of the historic and real-time data using machine learning and data mining tools prediction models to predict future energy demand trends for ensuring uninterrupted power supply of electricity and achieve significant reduction in unexpected load impact on electricity grids that could lead to faults and or failures. Taking these steps could lead to the installation of grids that are more resilience and tolerant to any kind of negative load impact. This, clearly, is a pathway to energy sustainability and security.

Obvious from the results is the fact that billing cycles should not be constraint only to months but could also be calculated based on shorter intervals such as hours and days which could be based on daily and or hourly averages. This presents a more flexible billing system for businesses responsible for billings.

Potential exist to eliminate paper billing system since all the necessary information is accessible online. This could help reduce or eliminate carbon emission during paper/pulp production in addition to preventing the use of fossil fuel vehicles to places for energy meter reading purpose by personnel. All this could sum up to increase any business/company Emission Trading Scheme (ETS) credits (Haavisto, 2011) in Europe and its equivalents in the rest of the world.

Overall, in this case, results are achieved with the least minimum overhead in terms of hardware and software used. The implications are that to achieve same accurate results as other similar systems, complex protocols and etiquette programming have been avoided as well as outright elimination of unnecessary hardware. This made the whole system relatively easy to setup, configure and install at low cost compared to other similar ones. Besides, it is energy user/consumer friendly and inclusive.

6 DISCUSSION

Although, IoT has far more applications than just energy consumption metering, its application to electrical network system is similar and complementary to that of Smart Grid injection into the same system. IoT and Smart Grid are two different technologies that enable, support and complement one another. It appears both Smart Grid and IoT are two different sides of the same ‘Gold Coin’ in this regard. Both side of the coin makes the coin valuable. The values that can be derived from storing and analyzing the data as typified in this work may be invaluable for a host of other purposes aside fulfilling the aims and objectives of this work.

Getting stored data displayed remotely on a web UI via some analytical tools, as demonstrated, would encourage unfettered access and independent assessment of individual electrical energy consumption which before now has been in the exclusive domain of electrical utility providers. This could also provide a platform for providers to adopt a business mode that aligns to their own benefits and those of their clients.

A future work on this system could be to interface a GSM module with the Raspberry Pi to send energy usage information to customers via SMS. This could help cover the population of those without internet access but are in possession of a phone.

Yet another futuristic work on this system would be to include cost calculations that could be adjusted according to the billings or tariffs in the different regions.

7 CONCLUSION

This thesis work has revealed a link, in meaning and function, between Smart Grid meter (i.e. Smart energy meter that is an integral part of Smart Grid) and IoT based energy meter and further demonstrated the proposed implementation of a simple and portable IoT-based electricity metering system using inexpensive hardware components and open source codes forked from GitHub.

The simplicity of this system and its cost effectiveness are sterling features to be desired for future enhancement and development as well as the avoidance of complex communication protocols, software installation and bogus hardware additions.

The granular information about energy consumption derived from this system would not only help consumers become aware of their energy consumption pattern but also aid them in making informed decisions to minimize energy use and use energy efficiently. This has an overall tendencies of energy security, economic stability and environmental protection at a scale beyond the individual consumer imagination. Indeed, possibilities exist for sustainable energy use, as well as economy and environmental improvement which could begin with change in the individual energy consumption behavioral pattern.

REFERENCES

Al-Ali, A. R & Aburukba, R. 2015. Role of Internet of Things in the Smart Grid Technology. Journal of Computer and Communications, Vol. 3 pp. 229-233. http://dx.doi.org/10.4236/jcc.2015.35029

Anushree, S.V. & Shanthi, T. 2016. IoT Based Smart Energy Meter Monitoring and Theft Detection Using ATMEGA. International Journal of Innovative Research in Computer and Communication Engineering. Vol. 4 (11). Read on 24.4.2019. DOI: 10.15680/IJIRCCE.2016.0411205

Binary Updates. 2017. Introduction of Raspberry Pi 3 Model B. read 3.3.2019. https://binaryupdates.com/introduction-of-raspberry-pi-3-model-b/

Chandra, P. A. et. al. 2016. Automated Energy Meter Using Wifi Enabled Raspberry Pi. IEEE International Conference on Recent Trends in Electronics Information Communication Technology (RTEICT), India. ISBN: 978-1-5090-0775-2.

Croston, B. 2013. raspberry-gpio-python: A Python module to control the GPIO on a Raspberry Pi. Read on 3.3.2019. https://sourceforge.net/p/raspberry-gpio-python/wiki/install/

Devagkumar, S. U & Chiragkumar, P. B. 2016. IoT Enabled Smart Grid. International Journal of Scientific Research and Development (IJSRD) National Conference on ICT & IoT.

Ekanayake, J. et. al. 2012. Smart Grid: Technology and Applications. John Wiley & Sons. Chapter 1 pp. 1-14.

EKM FORUM. 2015. Blinking red diode. EKM Metering Systems. Read on 12.3.20 19. https://forum.ekmmetering.com/viewtopic.php?t=3433

Haavisto, H. 2011. The European Union Emission Trading Scheme-development of volumes and prices. Working Paper. Aalto University School of Engineering Department of Energy Technology.

Henderson, G. 2013. Gordons Projects: WiringPi. Read on 5.3.2019. https://projects.drogon.net/raspberry-pi/wiringpi/

Hudedmani, M. G. et. al. 2019. IoT Based Smart Energy Meter for Smart Grid Applications. International Journal of Advanced Science and Engineering. Research Article Volume 5 (3) 982-987. E-ISSN: 2349 5359; P-ISSN: 2454-9967. Read on 24.4.2019. https://doi.org/10.29294/IJASE.5.3.2019.982-987

Jadhav, A. R. & Rajalakshmi, P. 2017. IoT Enabled Smart and Secure Power Monitor. Conference paper. IEEE Region 10 Symposium (TENSYMP). Electronic ISBN: 978-1-5090-6255-3. Read on 23.2.2019. https://ieeexplore.ieee.org/document/8070096

Kieranc. 2017. Simple electricity meter monitoring for Raspberry Pi. GitHub Inc. Read on 3.2.2019. https://github.com/kieranc/power

Kurde, A. & Kulkarni, V. S. 2016. IOT Based Smart Power Metering. International Journal of Scientific and Research Publication 6 (9), ISSN 2250-3153

Lea, T. 2019a. Emoncms. Read on 3.4.2019. https://github.com/emoncms/emoncms

Lea, T. 2019b. Install Emoncms on Raspberry Pi (Raspbian Stretch). Read on 2.4.2019. https://github.com/emoncms/emoncms/blob/master/docs/RaspberryPi/readme.md

OpenEnergyMonitor. n.d. Guide: Home Energy. Read on 4.2.2019. https://guide.openenergymonitor.org/applications/home-energy/

Perone, C. S. 2012. Raspberry Pi & Arduino: a laser pointer communication and a LDR voltage sigmoid. Read on 3.3.2019. http://blog.christianperone.com/2012/08/raspberry-pi-arduino-a-laser-pointer-communication-and-a-ldr-voltage-sigmoid/

Philip, M. et. al. 2017. IoT Based Energy Meter (AMMP). International Journal of Internet of Things 6 (2), 88-90.

Pocero, L. et al./HardwareX 1. 2017. Open source IoT meter devices for smart and energy-efficient school buildings. ScienceDirect, HardwareX (1) 54-67. Read on 27.03.2018. http://dx.doi.org/10.1016/j.ohx.2017.02.002

Satish, P et. al. 2015. Automated Reading System – A Study. International Journal of Computer Applications (0975 – 8887). Volume 116 (18)

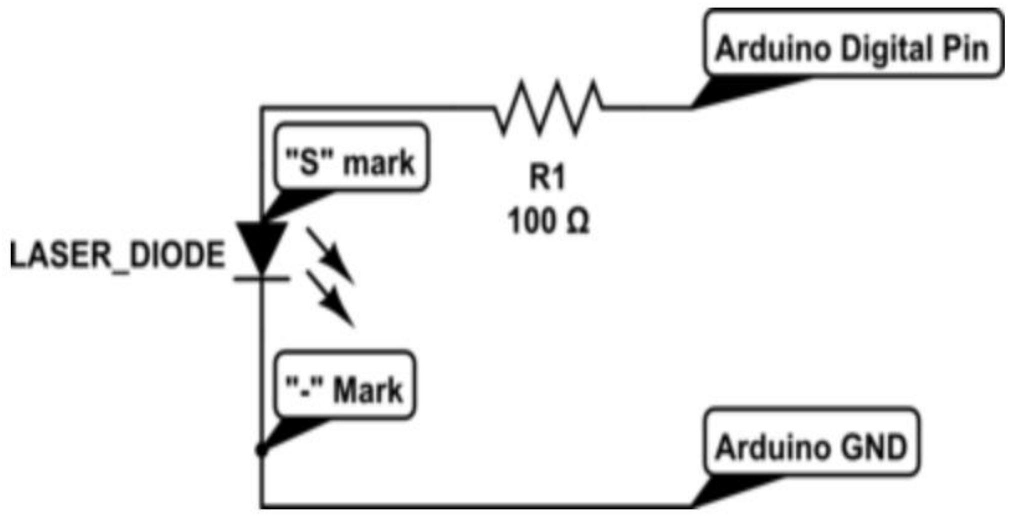
TUT. 2017. Course Slide: Overview of Smart Grid concepts and visions. Introduction to Smart Grids and Renewable Energy.

U.S. Department of Energy & Office of Electricity Delivery and Energy Reliability, 2016. AMI and Customer Systems: Results from the SGIG Program.

Vignesh, D. & Sathish S. 2016. Raspberry Pi Based Control and Monitoring of Smart Grid under an embedded System using WSN and Internet-of-Things. International Journal of Science, Engineering and Technology Research. IJSETR 5 (5), ISSN: 2278-7798

APPENDICES

Appendix 1. Wiring schema for Laser module (Perone, 2012)



Appendix 2. Arduino C code to blink Laser module (Perone, 2012)

```
int laserPin = 12;

void setup()
{
    pinMode(laserPin, OUTPUT);
    digitalWrite(laserPin, HIGH);
}

void loop()
{
    digitalWrite(laserPin, HIGH);
    delay(2000);
    digitalWrite(laserPin, LOW);
    delay(2000);
}
```

Appendix 3. C code to read and show the GPIO pin 4 value (Perone, 2012)

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)

while True:
input_value = GPIO.input(4)
print "Input Value (PIN 4):", input_value
time.sleep(0.1)
```

Appendix 4. power-monitor.c code (Kieranc, 2017)

```
#!/bin/sh
# /etc/init.d/power-monitor

case "$1" in
  start)
          echo "Starting power monitor..."
          gpio edge 1 falling
          python /root/power/monitor.py &
          ;;
  stop)
          echo "Stopping power monitor..."
          kill `ps -ef | grep monitor.py | grep -v grep | awk '{print $2}'`
          killall gpio-new
          ;;
  *)
          echo "Usage: /etc/init.d/power-monitor (start|stop)"
          exit 1
          ;;
esac

exit 0
```

Appendix 5. monitor.py code (Kieranc, 2017)

```
#!/usr/bin/python

"""
    Modified by KieranC to submit pulse count to Open Energy Monitor EmonCMS API

    Power Monitor
    Logs power consumption to an SQLite database, based on the number
    of pulses of a light on an electricity meter.

    Copyright (c) 2012 Edward O'Regan

    Permission is hereby granted, free of charge, to any person obtaining a copy of
    this software and associated documentation files (the "Software"), to deal in
    the Software without restriction, including without limitation the rights to
    use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
    of the Software, and to permit persons to whom the Software is furnished to do
    so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE.
"""

import time, os, subprocess, httplib, datetime
from apscheduler.scheduler import Scheduler

# The next 2 lines enable logging for the scheduler. Uncomment for debugging.
#import logging
#logging.basicConfig()

```

```
pulsecount=0
power=0

# Start the scheduler
sched = Scheduler()
sched.start()


# This function monitors the output from gpio-irq C app
# Code from vartec @ http://stackoverflow.com/questions/4760215/running-shell-command-from-python-and-capturing-the-output
def runProcess(exe):
    p = subprocess.Popen(exe, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    while(True):
      retcode = p.poll() #returns None while subprocess is running
      line = p.stdout.readline()
      yield line
      if(retcode is not None):
        break


# Every minute this function converts the number of pulses over the last minute into a power value and sends it to EmonCMS
@sched.interval_schedule(minutes=1)
def SendPulses():
  global pulsecount
  global power
# print ("Pulses: %i") % pulsecount # Uncomment for debugging.
  # The next line calculates a power value in watts from the number of pulses, my meter is 1000 pulses per kWh, you'll need to modify this if yours is different.
  power = pulsecount * 60
# print ("Power: %iW") % power # Uncomment for debugging.
  pulsecount = 0;
  timenow = time.strftime('%s')
        url = ("/emoncms/input/post.json?time=%s&node=1&json={power:%i}&apikey=<insert API key here>") % (timenow, power) # You'll need to put in your API key here from EmonCMS
        connection = httplib.HTTPConnection("localhost")
        connection.request("GET", url)


for line in runProcess(["/usr/local/bin/gpio-new"]):
    pulsecount += 1
```

Appendix 6. gpio-new.c code (Kieranc, 2017)

```c
/*
 * This file was adapted and simplified from the example isr.c
 * distributed with wiringPi by Gordon Henderson
 *
 * It waits for an interrupt on GPIO 1 and prints 'Interrupt' to stdout
 * It is used with a python script to monitor pulses from a power meter
 * and report the usage to EmonCMS
 *
 * See here: http://github.com/kieranc/power/
 *
 * Copyright (c) 2013 Gordon Henderson.
 ***********************************************************************
 * This file is part of wiringPi:
 * https://projects.drogon.net/raspberry-pi/wiringpi/
 *
 *    wiringPi is free software: you can redistribute it and/or modify
 *    it under the terms of the GNU Lesser General Public License as published by
 *    the Free Software Foundation, either version 3 of the License, or
 *    (at your option) any later version.
 *
 *    wiringPi is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *    GNU Lesser General Public License for more details.
 *
 *    You should have received a copy of the GNU Lesser General Public License
 *    along with wiringPi.  If not, see <http://www.gnu.org/licenses/>.
 ***********************************************************************
 */


#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <limits.h>
#include <wiringPi.h>

void myInterrupt() {
        printf ("Interrupt\n") ;
        fflush (stdout) ;
}

/*
 *********************************************************************************
 * main
 *******************************************************************************
 */

int main (void)
{
  wiringPiSetup () ;

  wiringPiISR (1, INT_EDGE_FALLING, &myInterrupt) ;

  for (;;) {
        sleep(UINT_MAX);
    }
    return 0;
}
```