Internet of Things 12 (2020) 100292

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Open Software and Data

IoT based real time energy monitoring system using Raspberry Pi

Mani Dheeraj Mudaliar[*], N. Sivakumar

Department of Electrical and Electronics Engineering, Dayananda Sagar College of Engineering, Bangalore 560078, India

ARTICLE INFO

Article history:
Received 1 April 2020
Revised 31 August 2020
Accepted 31 August 2020
Available online 3 September 2020

Keywords:
Raspberry Pi
IoT
Energy monitoring systems
Arduino

ABSTRACT

An IoT based real time energy monitoring system for controlling and monitoring of a switchgear industry is discussed in the present work. Raspberry Pi has been selected for this purpose due to low cost and a highly reliable technology for monitoring the energy consumption of the industry. The Raspberry Pi is used with Node.js programming language to collect the data (various electrical parameters) from the existing energy meters of the industry and to store it locally, for accessing it through Laptop or mobile phone using Grafana. The monitoring system is found be useful to the industry to understand the day-to-day energy pattern, which is essential to facilitate energy conservation measures for minimizing energy consumption.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction: IoT based energy monitoring system

Generally, Industries are keen to adopt various energy conservation measures to minimize the cost of the energy purchase as well as energy wastage [1]. One of the most well-known methods is conducting industrial energy audits to conserve the energy. Energy audits are used to identify the various energy losses in various equipment and heavy machinery which consume more energy. The energy audits provide details of the energy consumption of the industry. By analyzing various energy audit reports, industries consider necessary steps to conserve the energy [2]. Usually, industries are not carried out energy audits on regular basis and hence the conservation of energy is normally arrived in very small percentage in most of the industries. But, energy conservation in industry can be a regular process by continuously monitoring the day to day energy consumption of the industry [3–4].

Energy monitoring devices are being used to monitor the energy consumption in an industry. Conventionally, energy meters and various electrical measuring devices are installed in the industry to monitor the energy consumption [5]. However, meter based energy monitoring systems has become complicated as large number of measuring devices are required. In this regard, the evolution of energy meter to smart energy meter makes the task simpler i.e. when using smart energy meters, separate measuring devices are not required as the smart energy meter itself measures and display various electrical parameters [6]. At the same time, one of the major disadvantages of the smart meter is that the data cannot be retrieved. Further, readings need to be continuously noted manually for data collection which becomes tedious task [7].

If a suitable automatic system is available for monitoring the energy consumption instead of manual data collection, that will not only save the time, also helps to analyze the energy pattern instantly. In this context, the recent Internet of Things (IoT) based devices can perform this task in a better way [8]. Energy meters in association with IoT devices can

[*] Corresponding author.
E-mail addresses: dheerajmudaliar@gmail.com (M.D. Mudaliar), gnsivakumar@yahoo.co.in (N. Sivakumar).

https://doi.org/10.1016/j.iot.2020.100292
2542-6605/© 2020 Elsevier B.V. All rights reserved.

automatically collect and display various industrial data in graphical representation. These meters can be used to collect and store various electrical parameters in industries. The data collected from energy meters which are installed in various nodes of the industry can be stored by using various IoT devices such as Arduino, Raspberry Pi, Intel Edison, Mediatek linkit one, NVIDIA Jetson Nano etc. for further use [9].

The term 'Internet of Things (IoT) technically refers to inter-networking of physical devices and other embedded electronics, sensors, actuators and network connectivity, which enable these objects to collect and exchange data [10]. The IoT devices or objects always stay connected to the real world i.e. physical devices can be connected to the virtual world which can be accessed remotely from anywhere. Because of these remote operation, IoT devices are nowadays used for many real time applications such as smart media, environment monitoring, smart manufacturing, intelligent medical and healthcare, smart buildings, home automation, energy management, transportation, etc. [11,12].

Different kinds of IoT devices available in the market and are used for the above stated applications. The selection of IoT devices depends upon the application and the task to be accomplished. However, Arduino and Raspberry Pi are the most commonly used IoT devices because of simple in operation and easy to handle [13-17]. Further, the cost of Arduino and Raspberry Pi is less as compared with other IoT devices such as Intel Edison, Mediatek linkit one, NVIDIA Jetson Nano, etc. [18–20].

Some applications such as Smart Energy Monitoring, Remote Tracing System for Solar Power Facilities, Cardiograph Measurement Device in Biomedical Instrumentation, Home Security Systems have been developed by using Raspberry Pi due to their flexibility and simple in use [21–26]. Some applications such as Zinc Oxide Nanorods Based Humidity Sensors, Remote Monitoring of Solar Photovoltaic Stand-Alone Systems, Real-Time Monitoring of Solar Home Systems, and Control of Thermal Plants have been developed by using Arduino to perform the same task. In these applications, the Arduino is used as data logger. It is observed that, most of the IoT applications used either Raspberry or Arduino [27–30].

A smart energy monitoring system is developed to monitor the consumer's energy usage by Electricity Board (EB) using Raspberry Pi [31]. The system consists of two main parts i.e. EB Board and the consumer. The EB Board interacts completely through the website and the consumer also has a smart power monitoring app. In this system, all communications carried out via Wi-Fi module of the Raspberry Pi. A system is designed and developed for the remote monitoring and control of complex stand-alone photovoltaic plants [32]. In normal conditions, the system records and periodically reports the overall performances. In case of incorrect behaviors, it immediately informs the operators this tasks re performed by using GSM and microcontroller.

An electricity bill forecasting application by Home Energy Monitoring System is detailed in [33]. In this system, a multifunctional energy meter with Modbus RTU communication port is connected to ESP8266 microcontroller. The system has an IC to convert Modus RTU protocol to Wi-Fi signal. The application is programmed by using Node JS interface. A mobile web energy monitoring system using DFRduino Uno is presented in [34]. In this system, the DFRduino Uno is used to gather energy data from a residential building which will be transmitted and stored to a database through cloud based API (Application Programming Interface) resources. A mobile web application is used for displaying real time and historical energy readings.

It is observed that most of the IoT based energy monitoring systems is using extra modules such as GSM, microcontrollers, etc. based on the case specific applications. The major advantage of the proposed energy monitoring system is, no extra modules are required and only the Raspberry Pi is sufficient to perform all the operations and functions. Hence, it is compact and simple as compared with other systems.

Generally, any kind of IoT devices can be used for industrial energy monitoring system. If an Arduino (a microcontroller with limited capacity of RAM and memory on the chip and less input/output pins) is used to monitor the energy usage or consumption of an industry, the system requires Ethernet shield, WIFI module and necessary sensors. Further, the hardware part becomes quite complex as compared to the same energy monitoring system which uses Raspberry Pi [35].

Whereas, in the case of Raspberry Pi (a mini computer with limited capacity RAM and memory), different boards or versions of Raspberry Pi is available, based on the requirements, the board or version can be selected. Further, number of hardware and sensors required is less and extra Ethernet shield is not required in Raspberry Pi [35]. Various features of Raspberry Pi and Arduino are summarized in Table 1.

Based on architecture and technical parameters, Raspberry Pi and Arduino are quite different boards and both the boards have their own advantages and disadvantages. The major difference among these is the Arduino is a microcontroller board and the Raspberry Pi is a mini computer. Hence, Arduino is just a part of Raspberry Pi as a micro controller. The Raspberry Pi is good for software applications and the Arduino makes hardware projects simple. These two boards can be run by very low power such as battery packs (but battery packs cannot be used in Raspberry Pi due to high current ratings). But, if any power interruption in Raspberry Pi operation, it may cause damage to the software and other connected applications. Hence, Raspberry Pi must be properly shutdown. Whereas, in Arduino, it again restarts if there is any power cut [36].

The Raspberry Pi uses a fully functional Operating System (OS) called Raspbian which has all features similar to a computer such as processor, memory and graphics driver. But, the Raspberry Pi can be used with other operating systems such as Dibian, Chrome OS, Linux, etc. Although Linux is preferred, android can also be installed [3]. Whereas, in Arduino, it does not have any inbuilt operating system and the firmware simply interprets the user defined code. For the boards, input and output pins are used to connect other devices. The Raspberry Pi3 has two packs of input/output pins and the Arduino Uno has twenty (20) pins which allow the board to connect more number of devices. But, the Raspberry Pi is faster than Arduino by 40 times in clock speed and has the RAM 128,000 times more than Arduino [39,40].

Table 1
Difference between Raspberry Pi and Arduino [36–44].

Features	Raspberry Pi	Arduino
General	A minicomputer	A microcontroller
Operating System	Raspbian	Arduino IDE
Program running capability	Multiple programs can be run at a time	Only one program can be run at a time
Power backup	Battery packs cannot be used as it uses high current	Battery packs can be used as the current value is low
Libraries and sensors	Installing libraries and software for interfacing sensors are more complex	Very simple to interface sensors and other electronic components
Cost	Expensive as compared with Arduino	Cheap
Internet	Easy to connect by using Ethernet port and USB Wi-Fi dongles	External hardware is required to connect and the hardware need to be addressed properly by codes
Storage	No on-board storage (separate SD port is available)	On-board storage is available
Ports	4 USB ports to connect various devices	Only one USB port to connect to the computer
Processor	ARM family	AVR family Atmega328P
Power on off functions	Should be properly shutdown to avoid files corruption and software problems	Just a plug and play device, the programs start to run when in power supply and if disconnected it simply stops
Programming language	Python is desirable and C, C++ruby are pre-installed	C and C++

The Arduino has 32 kb storage on board for storing the code which decides the functions of the Arduino. The Raspberry Pi does not have any onboard storage, but it has a micro SD port. Arduino can be expanded by using external hardware or shields such as Wi-Fi, Ethernet, touchscreens, cameras etc. These shields are easily installed for Arduino. But, the Raspberry Pi is a self-constrained board and the shields are pre-installed. Raspberry Pi can also accommodate some parts to add hardware such as Touchscreen, RGB panels etc. Meanwhile, the Pi does not have some options as Arduino board has e.g. Arduino IDE for developing the code. At the same time, Raspberry Pi can use Scratch, IDLE and any other tools which support Linux [33–41].

With the above review of literature, it is well known that the Raspberry Pi is optimally more powerful than Arduino by means of operating speed, onboard ports, storage, RAM, etc. In this context, an energy monitoring system is proposed by using Raspberry Pi 3 model B+ for a switchgear industry to analyze day-to-day energy consumption. The proposed system consists of Raspberry Pi 3 model which has been connected to the energy meters of the industry using RS485. The Raspberry Pi is connected to the available WIFI in the industry to store and display the data in GUI (Graphical User Interface). The proposed system is successfully installed in the industry premises and found to be effective for their future energy conservation plans.

2. IOT based energy monitoring system for a switchgear company – a case study

A Raspberry Pi based energy monitoring system is designed and implemented for a panel manufacturing company. The company manufactures High Tension (HT) and Low Tension (LT) panels. The overall electrical capacity of the company is 22 kW. The company works for eight (8) hours per day which includes lunch break of 30 min to one hour. Various electrical and mechanical equipment and their energy utilization of the company are shown in Table 2.

The switch gear company is under operation for the past thirty five (35) years. The sanction load for the company from the state electricity board is 10 kW. Approximately, the company manufacturing 325 switch gear panels per year. A typical panel manufacturing details is shown in Table 3. The monthly energy consumption of the company is approximately 2300–3000 Units (kW-hr) per month depending upon the number of panels manufactured in a month, which is approximately

Table 2
Equipment power consumption details.

Name of the Equipment	Specification	Capacity (Watts)	Hours of operation	Power consumption per day (Watts)
Hydraulic press brake machine	7.5 HP	5516	4	22,064
Hydraulic power shearing machine	5 HP	3677	4	14,708
Heavy duty drilling machines	1 HP	735.5	5	14,710
Air and oil cooled welding machine		12,800	5	12,800
Compressor	220 V 15 A	3300	5	16,500
Surface grinding machine	108 W	216	5	1080
Industrial fans	80 W(15)	1200	4	4800
Lightings	60 W(5)	300	6	1800
Total				88,462

Table 3
Number of panels manufactured in the year 2019.

MONTH	Units consumed (kW-hr)	Number of panels
JANUARAY	2717	29
FEBRUARY	2661	27
MARCH	2156	22
APRIL	2470	25
MAY	2165	23
JUNE	2525	26
JULY	2379	24
AUGUST	2332	24
SEPTEMBER	2243	24
OCTOBER	2033	21
NOVEMBER	2217	23
DECEMBER	2557	26

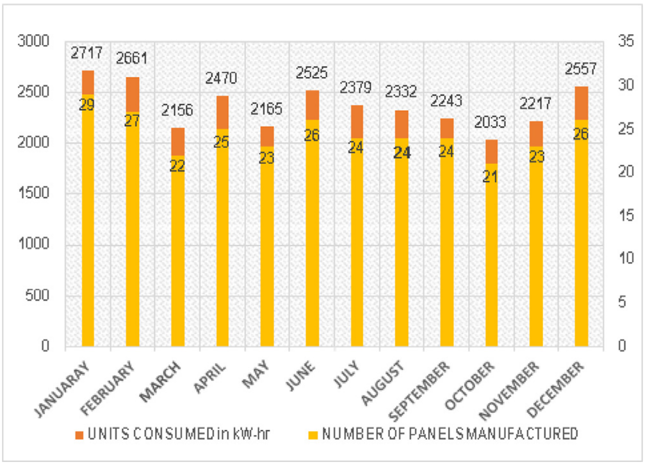


Fig. 1. Units consumed and panel manufactured.

Table 4
Hardware components used in IOT based energy monitoring system.

Sl. No.	Name of the components	Specification	Quantity
1	Raspberry Pi	1.4 GHz 64-bit quad-core ARM Cortex-A53	1
2	Schneider energy meter	EM 6400NG	1
3	Elmeasure energy meter	EN8400	1
4	RS485	Up to 32 unit loads	1

2741 kW of power (considered for an average of 27 panels manufactured in a month). Then, the power utilized for a panel is around 2537 kW. Out of these 2741 kW, actual energy utilized per panel is around 2537 kW i.e. after deducting the miscellaneous utilization (official fans, lighting, etc. excluded from the panel manufacturing process). The details of panels manufactured in a year e.g. 2019 are presented in Table 3 and Fig. 1.

In this context, the company authorities have decided to minimize the specific energy consumption i.e. energy used per panel, to minimize their production cost and to follow the energy conservation codes of the BEE (Bureau of Energy Efficiency - A statutory body of the Government of India under the Ministry of Power). As a first step for energy conservation, it is essential to monitor the present energy scenario and hence it is decided to adopt a suitable energy monitoring system. IoT based system can be adopted for this purpose which will provide real time energy utilization of the industry (Table 4).

3. Methods and material

3.1. System design

The proposed IoT based energy monitoring system consists of the existing energy meters of the industry, Raspberry Pi, Cloud (Data base, Data web service, control application, monitoring application), and visualization systems i.e. display such as a laptop, mobile, desktop, etc. The proposed energy monitoring system is illustrated in Figs. 2 and 3.

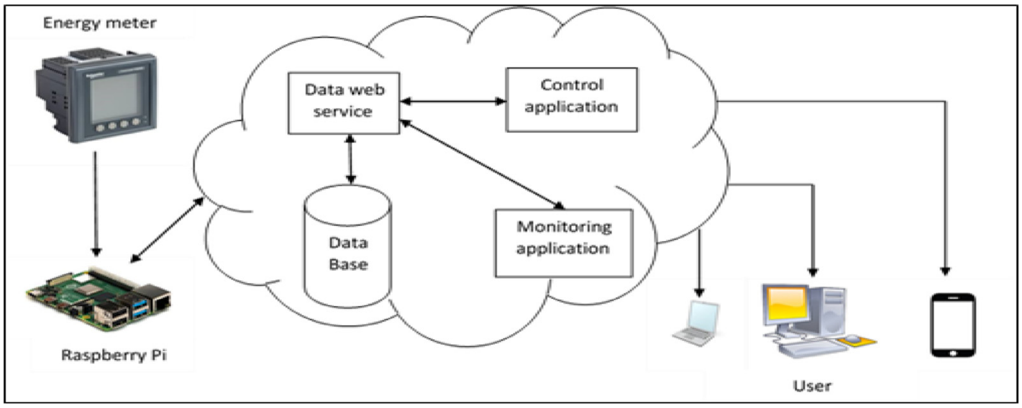


Fig. 2. Proposed concept.

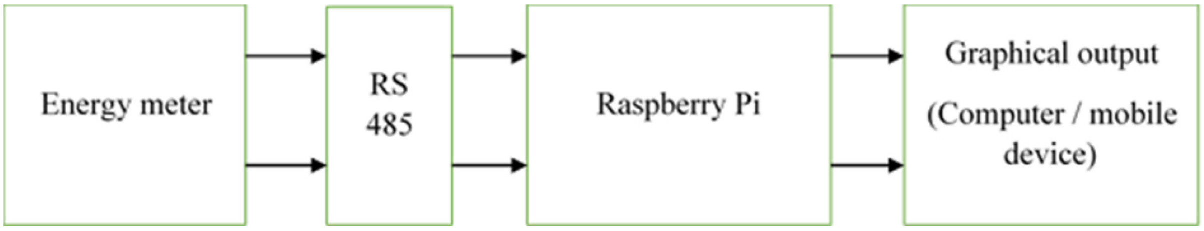


Fig. 3. Block diagram of the energy monitoring system.

The energy meters installed in this industry are Schneider and Elmeasure. These meters are capable to read different electrical parameters such as power, energy, real power reactive power, etc. of the industry (the meters are designed to measure more than 60 electrical parameters).

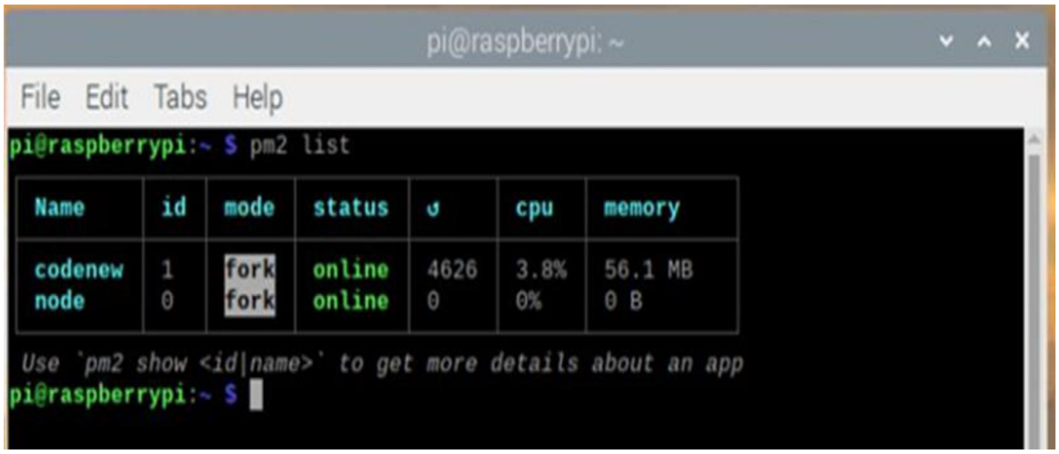
Various electrical parameters from different processes of the industry are normally stored by the existing energy meters and the instantaneous values are displayed. But, the data cannot be remotely accessed when using conventional energy meters and no graphical output available for the stored values. In this regard, RS485 has been fixed as an interface in between the energy meter and Raspberry Pi. The Raspberry Pi has been installed with Node.js and various libraries have been installed. The graphical output of the various parameters have been measured by the energy meter and displayed in the computer or mobile without any interruption. In the proposed system, the existing energy meters installed in industry have been connected to the Raspberry Pi using RS485 and the output can be seen in any of the display device provided, the system should be connected to internet i.e. to display graphical representation of the various electrical parameters measured in the industry remotely.

3.2. Computing system and software

The computing system uses Raspbian (modified Debian), an open source Linux based operating system. The operating system and data are stored on SD card. Micro USB connector is used for power supply – the computer requires only 5 V supply (typically 700 mA) to operate. The Raspberry Pi used in this system which has a flexible computer with ARM 1176JZF-S 700 MHz processor with Raspbian OS. The Raspberry Pi can be compared with other low power devices such as Arduino because it has enough on-board storage and processing power to operate the sensor, provide a web server interface (for data graphics) and operate as a data server including complex data manipulation for immediate interrogation by external clients [14].

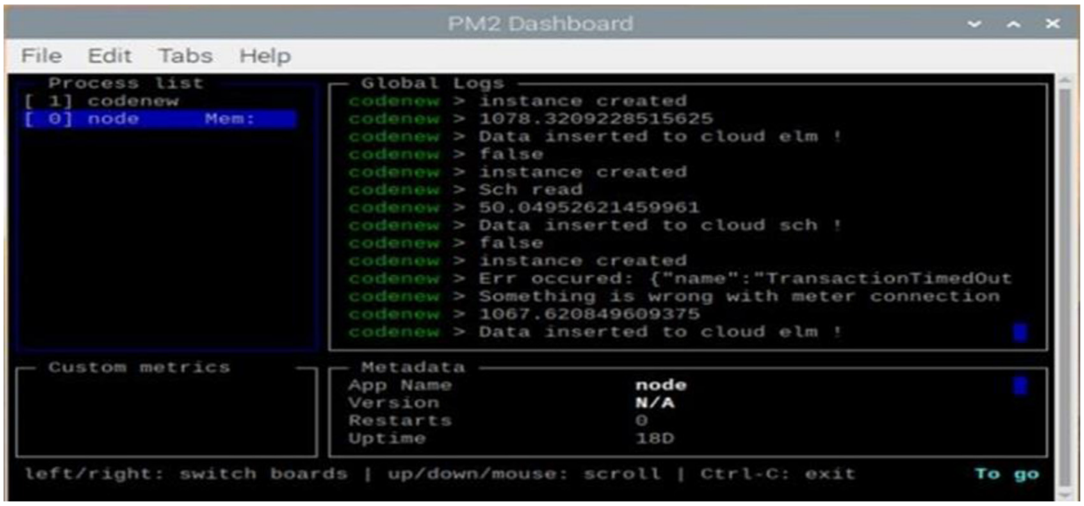
Node.js is installed by default in Raspberry Pi which is a very popular and easily extendable programming language; the proposed system is also uses Node.js. A wealth of open source software is available such as python, java, Node.js etc. to interface wide range of sensors, but most of the systems used Node.js for interfacing [14]. Similarly, this system also uses Node.js, an open-source multi-platform development and run time environment for accessible and fast network applications, especially to support JavaScript-based web servers. Various energy parameters ought to be displayed in terms of graph by using GUI which is provided by Node.js. The data can be monitored remotely by using mobile phone or laptop as long as user connects the device to local-host IP address by using Node.js.

The proposed system consists of InfluxDB which provides both local and cloud storage. At present the data is stored in the local device where the device is installed. The InfluxDB stores the data of various parameters which is read by the energy meters with respect to time. The database of InfluxDB is connected to the Grafana to display the data graphically. The Grafana can be accessed by the user to collect the data.



```
pi@raspberrypi:~ $ pm2 list
┌─────────┬────┬──────┬────────┬──────┬──────┬─────────┐
│ Name    │ id │ mode │ status │ ↺    │ cpu  │ memory  │
├─────────┼────┼──────┼────────┼──────┼──────┼─────────┤
│ codenew │ 1  │ fork │ online │ 4626 │ 3.8% │ 56.1 MB │
│ node    │ 0  │ fork │ online │ 0    │ 0%   │ 0 B     │
└─────────┴────┴──────┴────────┴──────┴──────┴─────────┘
 Use `pm2 show <id|name>` to get more details about an app
pi@raspberrypi:~ $
```

Fig. 4. PM2 list.



```
  Process list                      Global Logs
[ 1] codenew                        codenew > instance created
[ 0] node          Mem:             codenew > 1078.3209228515625
                                    codenew > Data inserted to cloud elm !
                                    codenew > false
                                    codenew > instance created
                                    codenew > Sch read
                                    codenew > 50.04952621459961
                                    codenew > Data inserted to cloud sch !
                                    codenew > false
                                    codenew > instance created
                                    codenew > Err occured: {"name":"TransactionTimedOut
                                    codenew > Something is wrong with meter connection
                                    codenew > 1067.620849609375
                                    codenew > Data inserted to cloud elm !

  Custom metrics                    Metadata
                                    App Name                 node
                                    Version                  N/A
                                    Restarts                 0
                                    Uptime                   18D
left/right: switch boards | up/down/mouse: scroll | Ctrl-C: exit          To go
```

Fig. 5. PM2 monitor.

PM2 is a production process manager for Node.js applications with a built-in load balancer. It keeps the applications alive forever, reloads the programs without downtime and used to facilitate common system admin tasks. PM2 can be started directly which offers a simple and intuitive Command Line Interface (CLI).

The PM2 is used to run the program continuously in the Raspberry Pi, otherwise the program terminates after 15 mins. One of the major advantages of using PM2 is the ability to run the program in the Raspberry Pi during power interruptions without the need of restarting i.e. it starts the program in the Raspberry Pi automatically [20]. In switchgear industries, even though power interruption rarely occurs, however, for safe and uninterrupted operation PM2 is installed. It can be highlighted that the PM2 internal worker (which checks memory) starts in every 30 s which makes the user to wait for a while before the process gets restarted automatically after reaching the memory threshold. Fig. 4 and Fig. 5 show the status of code and number of nodes being utilized by the program.

3.3. Flow chart

The flowchart of the energy monitoring system is shown in Fig. 6. The process starts from checking the connection between energy meter and Raspberry Pi which is connected through RS485. Upon the successful connection between energy meter and Raspberry Pi, the energy meter is read. The energy meter holds the data of several electrical parameters of the industry. The Raspberry Pi collects and stores the data from the energy meter for every 5 s. Subsequently, the stored data will be sent to the cloud as well as InfluxDB. The InfluxDB stores the data in its database along with the timestamp. The InfluxDB enables to assort the data in a readable format along with timestamp. The InfluxDB is installed within the

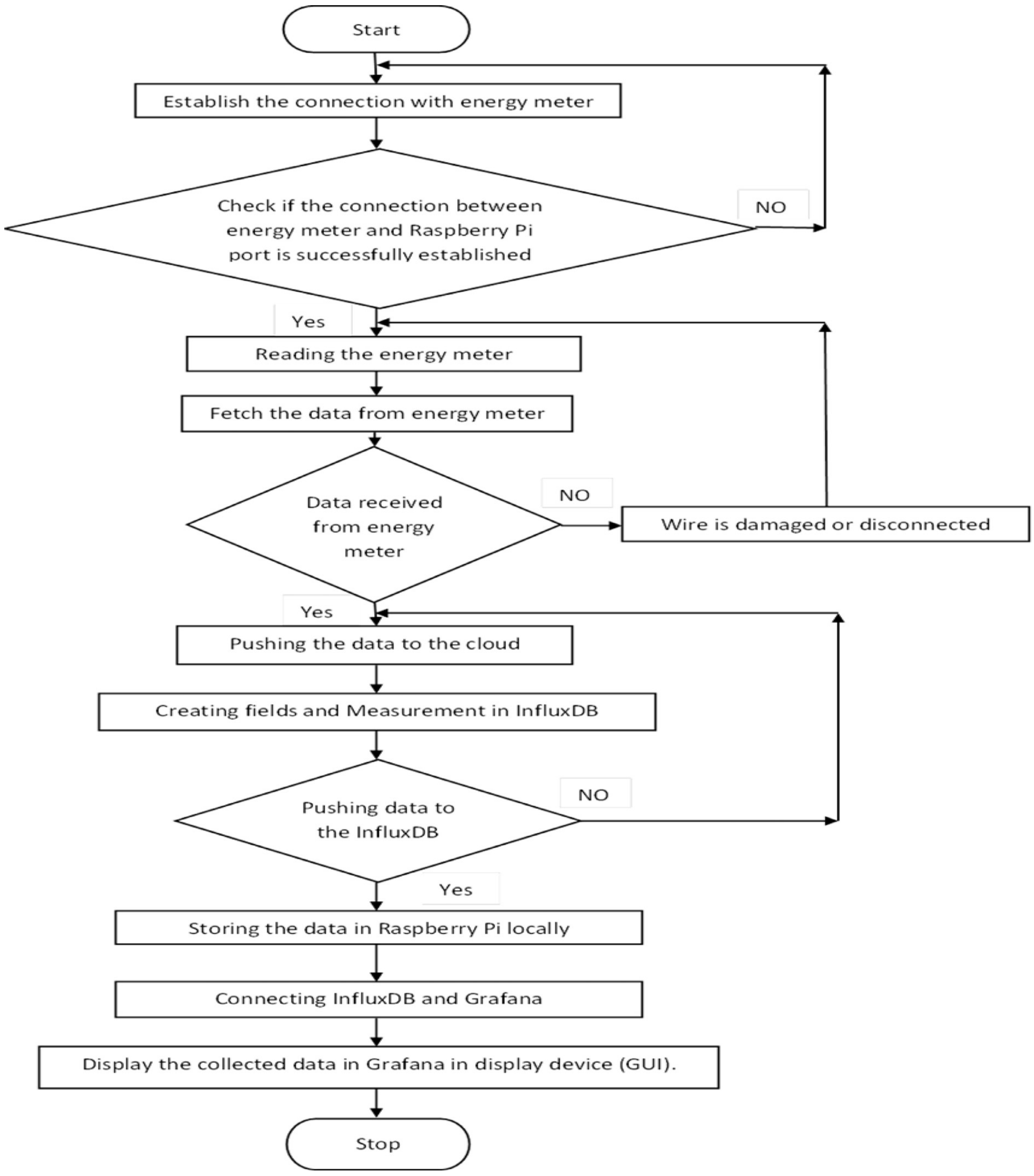

Start
Establish the connection with energy meter
Check if the connection between energy meter and Raspberry Pi port is successfully established
NO
Yes
Reading the energy meter
Fetch the data from energy meter
Data received from energy meter
NO
Wire is damaged or disconnected
Yes
Pushing the data to the cloud
Creating fields and Measurement in InfluxDB
Pushing data to the InfluxDB
NO
Yes
Storing the data in Raspberry Pi locally
Connecting InfluxDB and Grafana
Display the collected data in Grafana in display device (GUI).
Stop

Fig. 6. Flowchart of energy monitoring system.

Raspberry Pi and the database is created locally in the Raspberry Pi. Now, the Grafana is interfaced with the InfluxDB to represent the collected data from energy meter in a graphical representation. The collected data can be accessed in csv format.

3.4. Mapping

The energy meters hold data in its registers and get updated within few milliseconds.

Basically, there are two ways to read energy meters:

1. To read all registers in one shot.
2. By looping over each register one by one.

In addition to that, few considerations ought to be noted before reading the registers:

1. Only 124 registers can be read in one function call (limitation).
2. By combining 2/3/4 registers, one parameter will be retrieved.

In particular meter documentation, if the parameter's register value is mentioned as "2213" or "40,101", the developer has to find out whether it is required to consider last 3 digits as starting point or 4 digits or all digits from the registers. E.g., In Elmeasure en-8400 m, the registers are mentioned as 40,101, the last 3 digits i.e. 101 should be used. The register values are having even difference such as 101, 103; it means each parameter can be retrieved by combining 2 registers. The Schneider meters can either be read in one shot (max 124 registers in one call) or one by one (by looping through all params).

3.5. Procedure for the complete setup of Raspberry Pi

Step 1: Installation of Raspbian OS in Raspberry Pi, download the required software and files

- ➢ BlenaEtcher software (Flash OS images to SD cards and USB drives) (Available at: https://www.balena.io/etcher/)
- ➢ SD Card Formatter (Available at: https://www.sdcard.org/downloads/formatter_4/)
- ➢ Raspbian OS (Available at: https://www.raspberrypi.org/downloads/raspbian/)
- ➢ Fig. 7 shows the image of Raspbian OS and Raspberry Pi.

Step 2: SD Card and Card Reader fixation

- ➢ A minimum of 8GB class 10 SD card with a card reader is required. Insert the card into the card reader and plug it to the USB port

Step 3: Check the drive in which the SD card is mounted

- ➢ Select my computer or My PC and find the drive name where the SD card is mounted
- ➢ Open SD Card Formatter and select the SD Card
- ➢ Click on format and don't alter any other options
- ➢ Click OK when formating is completed

Step 5: Write the OS on the SD Card using BlenaEtcher

- ➢ Open balenaEtcher and select the Raspberry Pi from hard drive i.e. .img or .zip file to write on the SD card
- ➢ Select the SD card to write image Raspbian OS
- ➢ Review the selections and click 'Flash' to begin data writing to the SD card
- ➢ Insert the SD card into Raspberry Pi B+
- ➢ Insertion of the SD-Card into the Raspberry Pi is shown in Fig. 8.

Step 6: Initial setup

- ➢ Raspberry Pi doesn't have a power switch, connect it to a power outlet

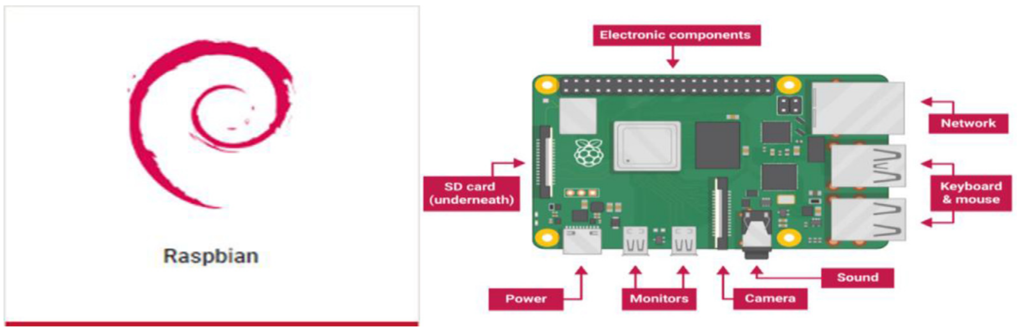


Fig. 7. Raspbian OS and Raspberry Pi 3 B+ model [44].

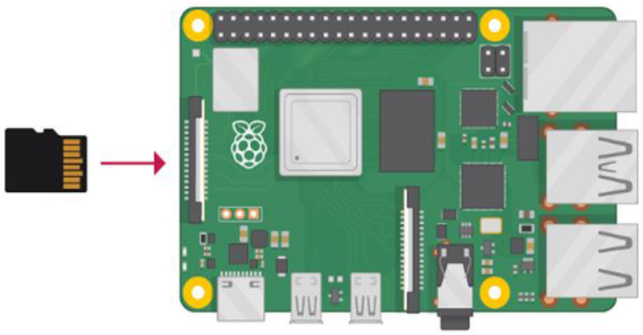


Fig. 8. Insert the SD card into Raspberry Pi [44].

Fig. 9. Raspbian Desktop.

- ➢ A red LED light up on the Raspberry Pi indicates the Raspberry Pi is connected to power
- ➢ As it starts up (booting), the Raspberry Pi logo appears in the top left-hand corner of the screen, after a few seconds the Raspbian Desktop will appear, and then the Welcome to Raspberry Pi application will pop up and will guide the user for initial setup
- ➢ Click Next to start the setup
- ➢ Set Country, Language, and Time zone, then click Next again
- ➢ Enter a new password and click Next
- ➢ Connect to Wi-Fi network by selecting its name, enter password and click Next (if the Raspberry Pi model doesn't have wireless connectivity, the screen can't be seen)
- ➢ Click Next, let the wizard check for updates the Raspbian
- ➢ Click done or Reboot to finish the setup (need to reboot if it is necessary to complete an update). The Raspbian Desktop will appear on the display device as show in the Fig. 9

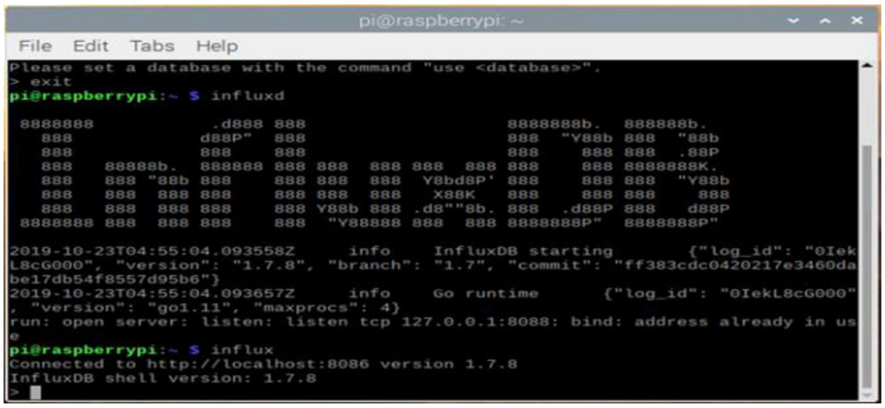
Step 7: Installation of libraries
The following libraries are necessary to run the program in Raspberry Pi

1. npm init –save
2. npm i modbus-serail –save
3. npm install -g
4. npm i express –save
5. npm i request –save
6. npm i request-promise –save

Step 8: Installation of InfluxDB on Raspbian Stretch for Raspberry Pi

- ➢ InfluxDB is an open source time series database developed by InfluxData which runs on 64-bit Windows, macOS or the Raspberry Pi Raspbian distribution
- ➢ Install InfluxDB on Raspbian

Run the following commands:



```
Please set a database with the command "use <database>".
> exit
pi@raspberrypi:~ $ influxd

 8888888           .d888 888                   8888888b.  888888b.
   888            d88P"  888                   888  "Y88b 888  "88b
   888            888    888                   888    888 888  .88P
   888   88888b.  888888 888 888  888 888  888 888    888 8888888K.
   888   888 "88b 888    888 888  888  Y8bd8P' 888    888 888  "Y88b
   888   888  888 888    888 888  888   X88K   888    888 888    888
   888   888  888 888    888 Y88b 888 .d8""8b. 888  .d88P 888   d88P
 8888888 888  888 888    888  "Y88888 888  888 8888888P"  8888888P"

2019-10-23T04:55:04.093558Z     info    InfluxDB starting       {"log_id": "0Iek
L8cG000", "version": "1.7.8", "branch": "1.7", "commit": "ff383cdc0420217e3460da
be17db54f8557d95b6"}
2019-10-23T04:55:04.093657Z     info    Go runtime      {"log_id": "0IekL8cG000"
, "version": "go1.11", "maxprocs": 4}
run: open server: listen: listen tcp 127.0.0.1:8088: bind: address already in us
e
pi@raspberrypi:~ $ influx
Connected to http://localhost:8086 version 1.7.8
InfluxDB shell version: 1.7.8
> 
```

Fig. 10. InfluxDB in Raspberry Pi.

- ➢ curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add
- ➢ sudo apt-get update && apt-get upgrade

These commands will add the apt-transport-https package, add the link to the source of InfluxDB, update the system and finally install InfluxDB [46].

- ➢ sudo apt install apt-transport-httpsecho "deb https://repos.influxdata.com/debian jessie stable" | sudo tee /etc./apt/sources.list.d/influxdb.list
- ➢ sudo apt-get update
- ➢ sudo apt-get install influxdb
- ➢ Intialisation of the InfluxDB after installation is shown in Fig. 10.

Step 9: Link up InfluxDB and Grafana

Thus the complete setup for IoT based Energy monitoring system using Raspberry Pi is executed. Some important steps of the programs used in this system are shown in **Annexure 1**.

4. Results and discussion

The graphical output of collected data in Grafana is shown in Fig. 11. Various electrical parameters such as voltage, current, frequency, power, power factor, energy, etc. have been measured by the existing energy meters i.e. Elmeasure and Schneider are displayed by using Raspberry Pi in Grafana.

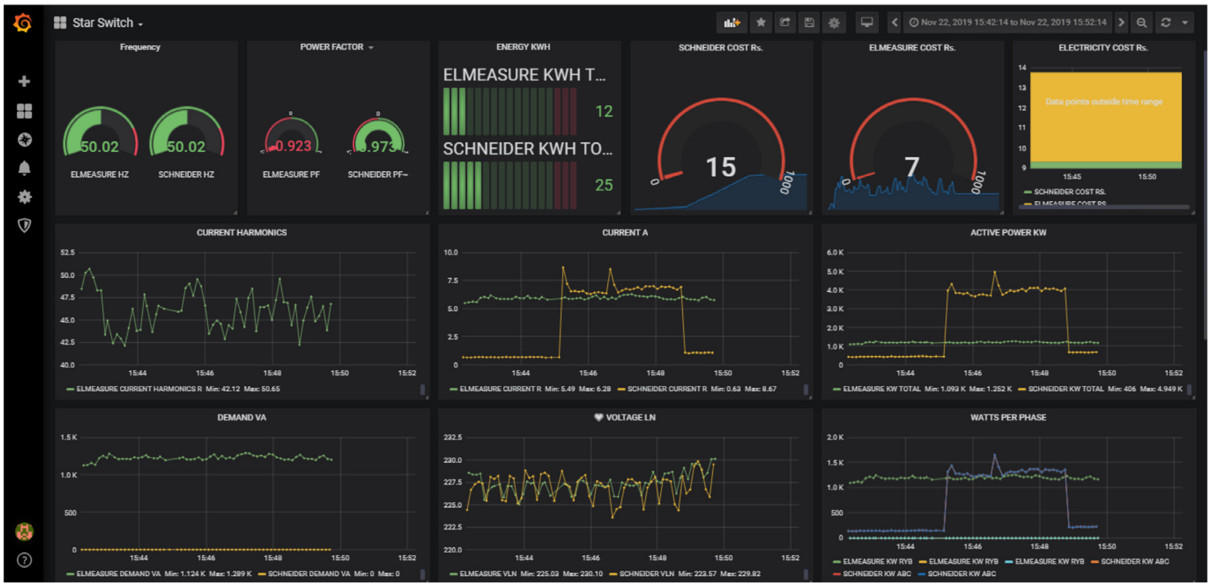

Star Switch
Nov 22, 2019 15:42:14 to Nov 22, 2019 15:52:14
Frequency
50.02
50.02
ELMEASURE HZ
SCHNEIDER HZ
POWER FACTOR
0.923
0.973
ELMEASURE PF
SCHNEIDER PF~
ENERGY KWH
ELMEASURE KWH T...
12
SCHNEIDER KWH TO...
25
SCHNEIDER COST Rs.
15
ELMEASURE COST Rs.
7
ELECTRICITY COST Rs.
Data points outside time range
SCHNEIDER COST RS.
CURRENT HARMONICS
ELMEASURE CURRENT HARMONICS R Min: 42.12 Max: 50.65
CURRENT A
ELMEASURE CURRENT R Min: 5.49 Max: 6.28
SCHNEIDER CURRENT R Min: 0.63 Max: 8.67
ACTIVE POWER KW
ELMEASURE KW TOTAL Min: 1.093 K Max: 1.252 K
SCHNEIDER KW TOTAL Min: 406 Max: 4.949 K
DEMAND VA
ELMEASURE DEMAND VA Min: 1.124 K Max: 1.289 K
SCHNEIDER DEMAND VA Min: 0 Max: 0
VOLTAGE LN
ELMEASURE VLN Min: 225.03 Max: 230.10
SCHNEIDER VLN Min: 223.57 Max: 229.82
WATTS PER PHASE
ELMEASURE KW RYB
ELMEASURE KW RYB
ELMEASURE KW RYB
SCHNEIDER KW ABC
SCHNEIDER KW ABC
SCHNEIDER KW ABC

Fig. 11. Graphical output of collected data in Grafana.

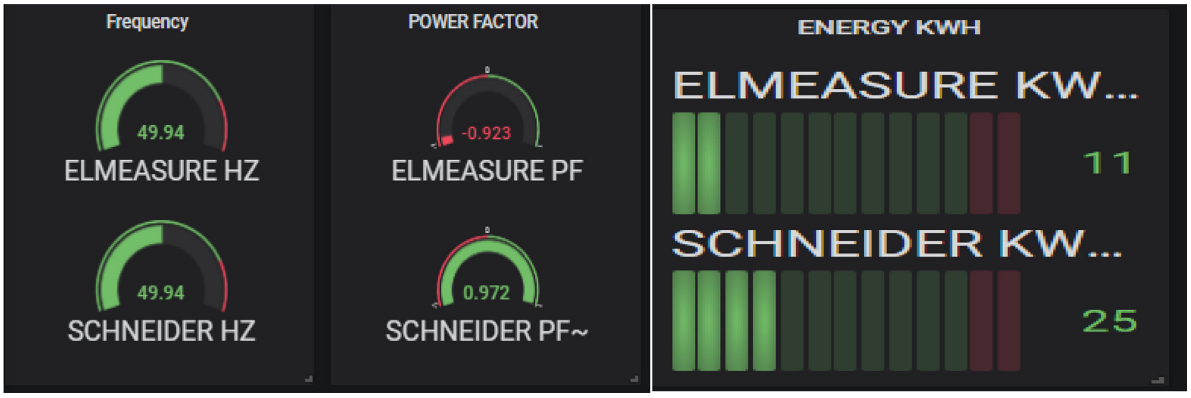


Fig. 12. Grafana output (frequency, power factor and energy measured by Elmeasure and Schneider).

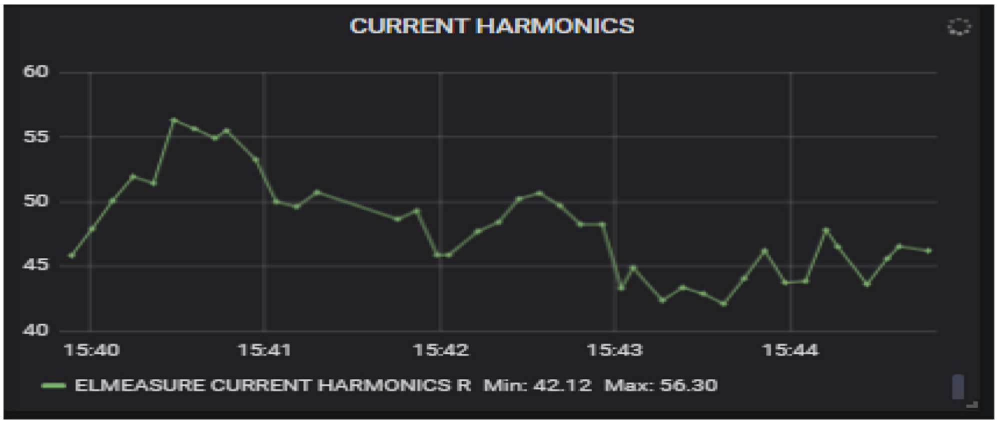


Fig. 13. Grafana output (current harmonicas measured by Elmeasure meter).

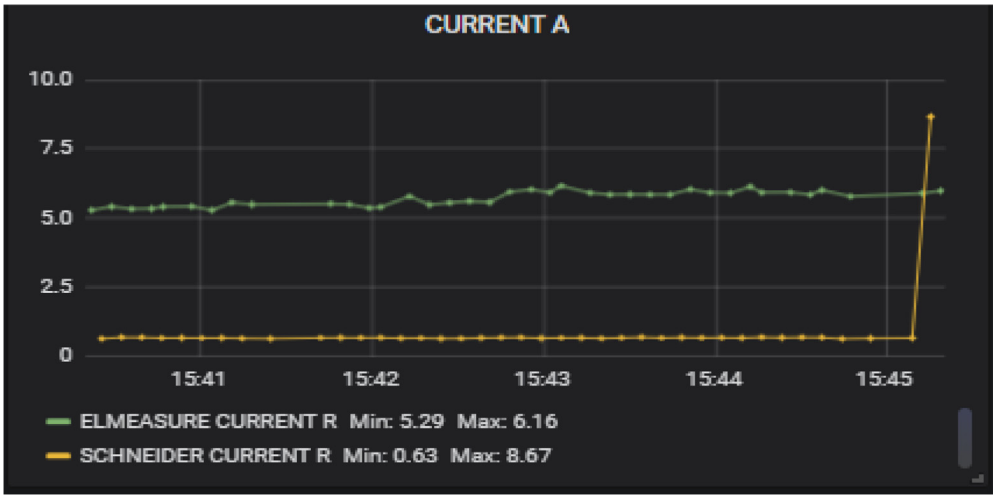


Fig. 14. Garafana output (current measured by Elmeasure and Schneider).

More clear view of the various electrical parameters displayed in Grafana are shown in Figs. 12–20. The existing energy meters of the switch gear industry are designed to measure eighty five (85) electrical parameters, some selected parameters have been shown in **Annexure 2**.

Data collected from the energy meters is stored locally by InfluxDB. The InfluxDB consists of two software processes i.e. Data nodes and Meta nodes. To run an InfluxDB cluster, both the Meta and Data nodes are required. The Meta nodes expose a HTTP API that uses the influxd-ctl command. System operators use this command to perform operations on the cluster

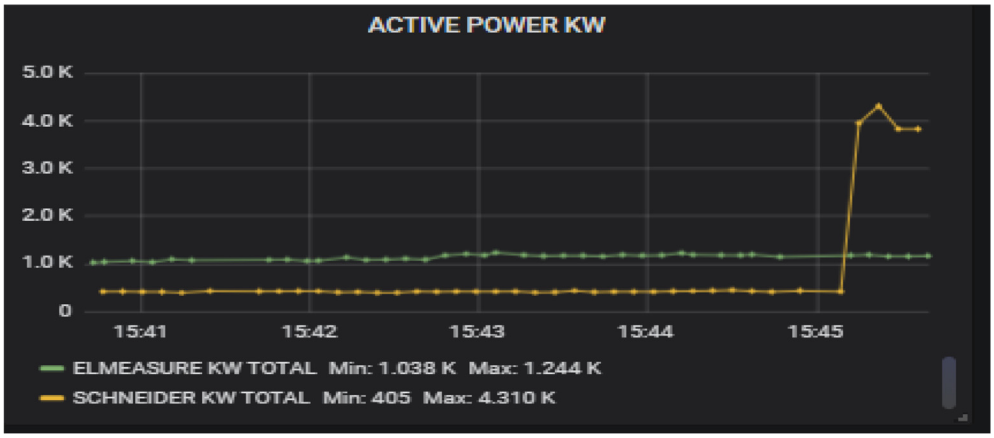


Fig. 15. Grafana output (active power measured by Elmeasure and Schneider).

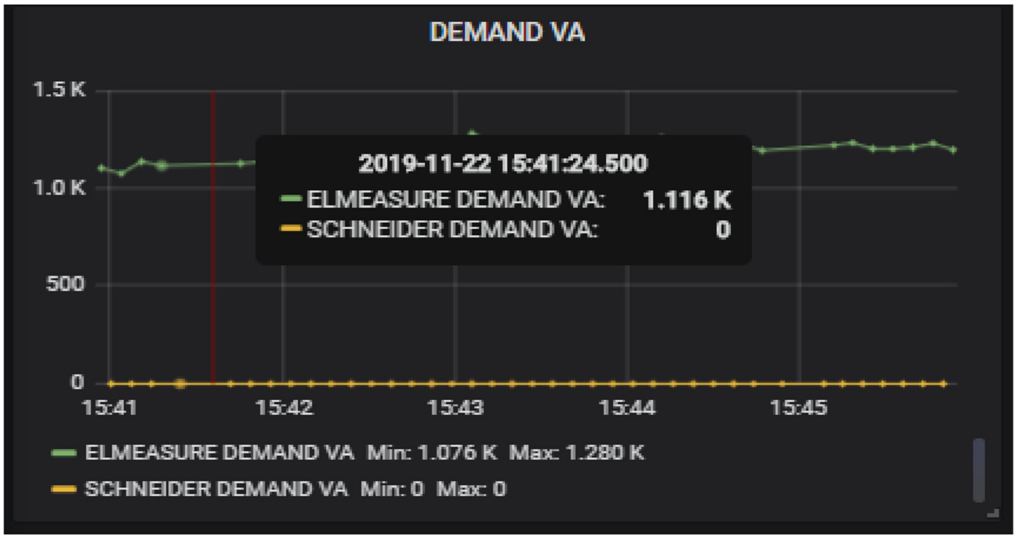


Fig. 16. Grafana output (demand measured by Elmeasure and Schneider).

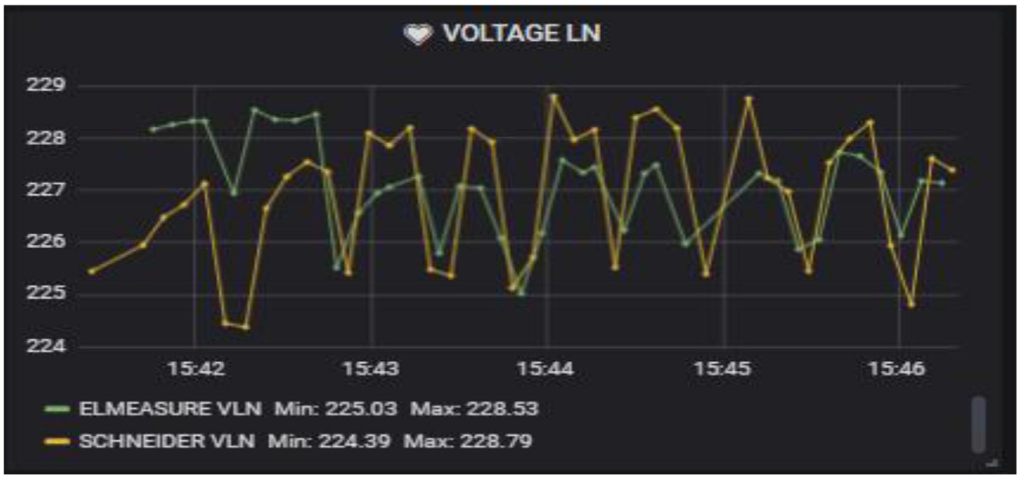


Fig. 17. Grafana output (voltage measured by Elmeasure and Schneider).

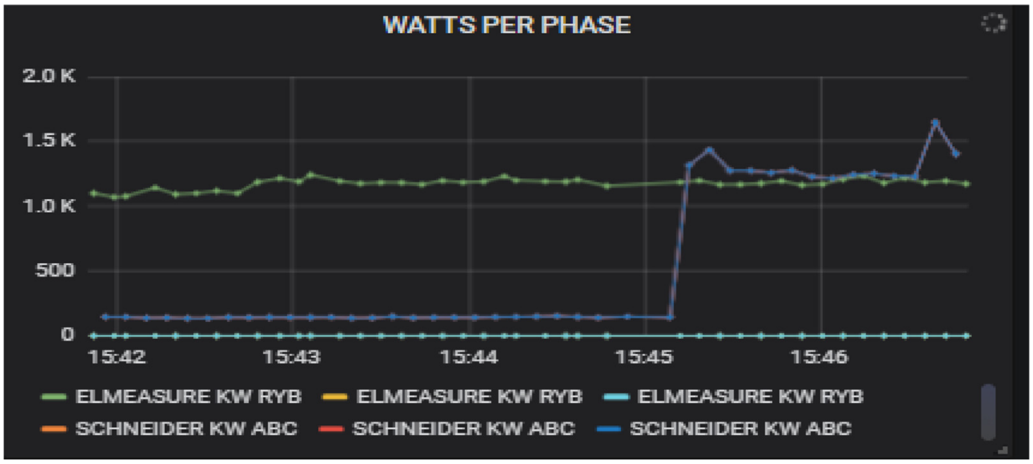


Fig. 18. Grafana output (power/phase - Watts measured by Elmeasure and Schneider).

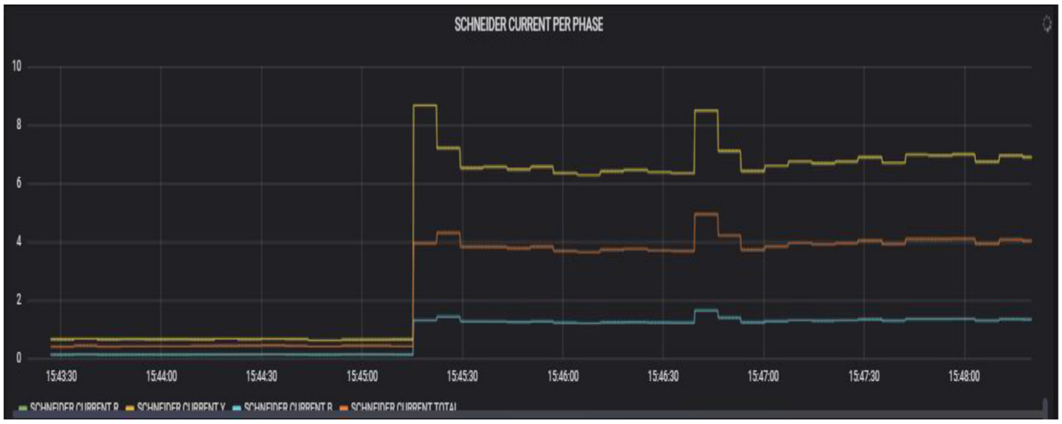


Fig. 19. Grafana output (current/phase measured by Schneider).

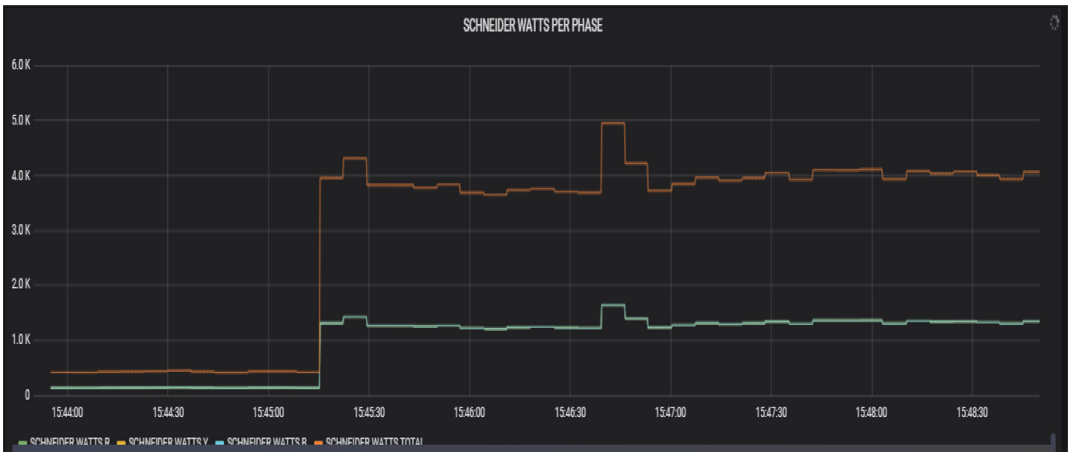


Fig. 20. Grafana output (power/phase - Watts measured by Schneider).

such as adding and removing servers, moving shards (large blocks of data) around a cluster and other administrative tasks. The Meta nodes communicate with each other through a TCP Protobuf protocol. Similarly, the Data nodes communicate with each other through a TCP and Protobuf protocol within a cluster. Data nodes hold the actual time series data, atleast one data node is required to run and it can scale up.

Generally, to run a number of data nodes, this has to be evenly divided by replication factor. However, different replication factors are to be used in different retention policies. The data nodes replicate data and query each other via a Protobuf protocol over TCP. Data nodes are responsible for handling all ‘writes and queries’. Sizing i.e. number of various parameters to be measured is depends on write and query load. If more energy meters need to be added in future for this case, a slight change in program only in some stages and increase the size of local storage would be enough.

The same setup can be replicated in different types of industries and for different types of applications as detailed in chapter 1. For various other applications, the key change will be only on sensors and input sources (E.g. the IoT device is used with energy meters in this switch gear industry).

In this energy monitoring system, a local server is created in the system itself. This has been archived by using a data storage device such as SSD, SD-Card, etc. The SSD or SD-Card is sufficient as it has large data storing capacity for a long period i.e. about 6 months. Further, the retention policy of the data can be changed at any time based on the users’ requirement.

5. Conclusion

In process industries, energy conservation can be achieved by many ways. However, it is essential to analyze the historical and present energy pattern of the industry to adopt a suitable energy conservation measures. In this switch gear industry, it is decided to establish an efficient energy monitoring system to monitor and analyze the day-to-day energy utilization as a preliminary step of energy conservation activities. In this regard, IoT based energy monitoring systems are considered. Many kinds of IoT devices such as Arduino, Raspberry Pi, Intel Edison, Mediatek linkit one, NVIDIA Jetson Nano etc. are available to perform this task. As compared with other data acquisition methods, the Raspberry Pi based systems are found to be the best option to accomplish the task based on their technical criteria.

Hence, an energy monitoring system based on Raspberry Pi is established with the existing energy meters of the switch gear industry. The Raspberry Pi has been programmed for the energy monitoring system and which is operating in an efficient way as detailed in this article. This system would be very helpful to understand the day-to-day energy pattern of the company and it will be useful to implement energy conservation measures in future to operate the industry with less cost and efficient power consumption.

Further, the design of the system, following the IoT paradigm, allows for it to serves in various contexts and use cases with little modification, for example, case specific programs or sensors of the infrastructure could be brought it for the overall system integration (additional sensors or software tools can easily be integrated based on the use case).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge the kind support extended by the authorities of Star Switchgears Industry Ltd, Bangalore, India, Mr. Shakeer Meeran, General Manager for the industrial guidance and providing the operational data utilized in this study, Mr. Arun, Mr. Divakar for the support extended for Node.js programming.

Annexure 1

Major programs used in the energy monitoring system:
i) Variable declaration

The general initialization part used to declare necessary variables of the program to store various electrical parameters in the Raspberry Pi.

```
const fs = require('fs');
var ModbusRTU = require("modbus-serial");
const request = require('request');
const influx = require('influx');
const localInfluxClient = new influx.InfluxDB('http://192.168.0.106:8086');
var sModbusClient = new ModbusRTU();
var data = [];
var singleRegister = null;
var dataToPush = {};
let configuration = {}
```

ii) Program for open a serial port

This part of the program is used to check the connection between Raspberry Pi and energy meters. The program starts with creating instance and return promise. The Raspberry Pi port is configured (dev/ttyUSB0) in this phase. The important point in this phase is that the Raspberry should recognize the RS485 when inserted into the USB port. This program mainly used to ensures connection and configuration. If the connection is successful, the data will be retrieved from the energy meter, otherwise an error message will be displayed.

```
Input: Command to check RS 485 connection with Raspberry Pi
Output: Display the message if any error occurred ("Some error occurred, trying to re-establish connection").
// open connection to a serial port
async function sCreateInstance(c) {
// return new Promise(function (resolve, reject) {
console.log(sModbusClient.isOpen)
// if (!sModbusClient.isOpen) {
if (fs.existsSync('/dev/ttyUSB0')) {
return new Promise(async function (resolve, reject) {
try {
awaitsModbusClient.connectRTUBuffered ("/dev/ttyUSB0", configuration.values[c].con);
//sModbusClient.setID (configuration.values[c].meter_no);
await sModbusClient.setTimeout(3000);
console.log('instance created')
resolve();
}
catch (error) {
console.log("Some error occured, trying to re-establish connection... Hang On !!!");
console.log("******************+Error details start+******************");
console.log(error);
console.log("******************+Error details end+******************");
resolve();
// createInstance();
}
});
}
else {
console.log('check if port is loose or disconnected !');
resolve();
}
//}
//})
}
async function ReadElmeasure(c) {
let ct = 0
return new Promise(async function (resolve, reject) {
sModbusClient.readHoldingRegisters(100, 124, async function (err, res) {
console.log('res',JSON.stringify(res));
if (res) {
data = new Float32Array(new Uint16Array(res.data).buffer);
console.log(data[0])
resolve();
}
else {
// await sModbusClient.close();
console.log('Err occured: ' + JSON.stringify(err));
console.log('Something is wrong with meter connection !');
ct += 1
if (ct == 1)
await ReadElmeasure(c)
resolve();
}
});
});
}
```

iii) Program for writing the data to cloud

In this part, the measurement fields and tags have been created and inserted in the cloud. Data in InfluxDB has been organized by "time series" which contains a measured value such as "Frequency" or "Power factor". The time series can store large number of points i.e. for each discrete sample of the metric. The points consists of time (timestamp), measurement (frequency for example), at least one key-value field (the measured value, e.g. "Value=50 Hz (for Frequency) or 0.9 (for Power factor)), and zero to many key-value tags which contains any metadata.

Conceptually it is considered as a table where the values are stored in which the primary index is always time. Tags and fields are generally columns in the table and which are indexed.

Input: Command for writing Data to cloud
Output: The structured data. Various electrical parameters read from the energy meters will be written to the InfluxDB cloud. If there is no data received from the energy meter due to any
technical reasons by the cloud, an error message will be displayed ("Error saving data to Influx").

```
async function writeDataToCloud(data) {
// console.log(data);
return new Promise(function (resolve, reject) {
// console.log()
if (data != null && data.length > 0) {
var date = new Date();
localInfluxClient.writePoints([
{
measurement: 'elmeasure_en8400',
tags: {
hardware: "energy_meter",
},
fields: {
Watts_Total: data[0] * 1000,
Watts_R_Phase: data [1],
Watts_Y_Phase: data [2],
Watts_B_Phase: data [3],
VAR_Total: data [4],
VAR_R_Phase: data [5],
VAR_Y_Phase: data [6],
VAR_B_Phase: data [7],
PF_Ave_Inst: data [8],
PF_R_Phase: data [9],
PF_Y_Phase: data [10],
PF_B_Phase: data [11],
VA_Total: data [12],
VA_R_Total: data [13],
VA_Y_Total: data [14],
VA_B_Phase: data [15],
VLL_Average: data [16],
Vry_Phase: data [17],
Vyb_Phase: data [18],
Vbr_Phase: data [19],
VLN_Avergae: data [20],
V_R_Phase: data [21],
V_Y_Phase: data [22],
V_B_Phase: data [23],
Current_Total: data [24],
Current_R_Phase: data [25],
Current_Y_Phase: data [26],
Current_B_Phase: data [27],
Frequency: data [28],
Wh_Received: data [29],
VAh_Received: data [30],
VARh_Ind_Received: data [31],
VARh_Cap_Received: data [32],
Wh_Delivered: data [33],
VAh_Delivered: data [34],
VARh_Ind_Delivered: data [35],
VARh_Cap_Delivered: data [36],
PF_average_Received: data [37],
Amps_average_Received: data [38],
PF_average_Delivered: data [39],
Amps_average_Delivered: data [40],
Neutral_Current: data [41],
```

(continued on next page)

```
Voltage_R_Harmonics: data [42],
Voltage_Y_Harmonics: data [43],
Voltage_B_Harmonics: data [44],
Current_R_Harmonics: data [45],
Current_Y_Harmonics: data [46],
Current_B_Harmonics: data [47],
Rising_Demand: data [48],
Forecast_Demand: data [49],
Maximum_Demand: data [50],
Load_Hours_Received: data [51]
},
timestamp: date.getTime()
}],
{
database: 'starswitch',
precision: 'ms',
})
.then(async () => {
// console.log(data)
await sModbusClient.close();
console.log('Data inserted to cloud elm !');
data = []
resolve();
})
.catch(async error => {
await sModbusClient.close();
console.log('Error saving data to InFlux: ' + error);
data = []
resolve();
});
}
else {
console.log('Data is null hence nothing would be stored !');
resolve();
}
});
}
```

Annexure 2

Table a. Readings of Elmeasure energy meter

Meter Name	Time	Average Amps received	Current Harmonics / R	Current (Amps) / R Phase	Total current (Amps)	Frequency (Hz)	Average power factor	Power factor / B	Power factor / R	Power factor / Y	Power factor average received	VAR R / Phase	VAR total	VARh Received
Elmeasure_en8400	2019–09–25T05:21:36.454Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	259.78	259.78	−0.01
Elmeasure_en8400	2019–09–25T05:21:43.185Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	247.33	247.33	−0.01
Elmeasure_en8400	2019–09–25T05:21:50.118Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	241.51	241.51	−0.01
Elmeasure_en8400	2019–09–25T05:21:57.129Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	250.4	250.4	−0.01
Elmeasure_en8400	2019–09–25T05:22:04.126Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	245.02	245.02	−0.01
Elmeasure_en8400	2019–09–25T05:22:11.119Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	249.93	249.93	−0.01
Elmeasure_en8400	2019–09–25T05:22:18.124Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	230.61	230.61	−0.01
Elmeasure_en8400	2019–09–25T05:22:25.115Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	265.27	265.27	−0.01
Elmeasure_en8400	2019–09–25T05:22:32.12Z	3848.28	43.26	3.97	3.97	50	−0.96	1	0.96	1	−0.91	246.67	246.67	−0.01
Elmeasure_en8400	2019–09–25T05:22:39.138Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:22:39.138Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:22:46.174Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:22:53.202Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:23:00.219Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:23:07.18Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:23:14.191Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:23:21.221Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01
Elmeasure_en8400	2019–09–25T05:23:28.225Z	3848.28	43.26	3.97	3.97	50	−0.95	1	0.95	1	−0.91	254.78	254.78	−0.01

Table b. Readings of Schneider energy meter

Name	Time	Active energy delivered (kWh)	Active power (kW) / R	Active power (kW) / Y	Active power (kW) / B	Active power total (kW)	Current (Amps) / R	Current average (Amps) / R	Current (Amps) / Y	Current (Amps) / B	Frequency (Hz)	Voltage (Volts) / RY	Voltage (Volts) / RN	Voltage (Volts) / YB	Voltage (Volts) / YN
Schneider	2019–09–25T05:21:33.04Z	1856.65	0.22	0.22	0.22	0.65	0.99	0.99	0.99	0.99	50	407.61	233.1	405.16	239.5
Schneider	2019–09–25T05:21:39.812Z	1856.65	0.59	0.59	0.59	1.77	1.18	1.18	1.18	1.18	50	407.43	232.85	406.46	239.07
Schneider	2019–09–25T05:21:46.748Z	1856.66	0.98	0.98	0.98	2.94	4.85	4.85	4.85	4.85	50	404.76	230.94	404.81	238.51
Schneider	2019–09–25T05:21:53.752Z	1856.66	0.97	0.97	0.97	2.92	4.7	4.7	4.7	4.7	50	404.39	230.63	404.82	238.58
Schneider	2019–09–25T05:22:00.76Z	1856.67	0.52	0.52	0.52	1.57	2.83	2.83	2.83	2.83	50	405.39	231.27	405.66	239
Schneider	2019–09–25T05:22:07.758Z	1856.67	0.3	0.3	0.3	0.9	1.37	1.37	1.37	1.37	50	407.11	232.59	406.61	239.09
Schneider	2019–09–25T05:22:14.758Z	1856.67	0.5	0.5	0.5	1.5	2.47	2.47	2.47	2.47	50	407.06	232.3	406.45	238.97
Schneider	2019–09–25T05:22:21.757Z	1856.68	0.5	0.5	0.5	1.49	2.2	2.2	2.2	2.2	50	407.41	232.29	405.98	238.75
Schneider	2019–09–25T05:22:28.76Z	1856.68	0.49	0.49	0.49	1.46	2.17	2.17	2.17	2.17	50	406	231.42	405.26	238.65
Schneider	2019–09–25T05:22:35.774Z	1856.68	0.31	0.31	0.31	0.94	1.3	1.3	1.3	1.3	50	406.13	231.75	405.17	238.55
Schneider	2019–09–25T05:22:42.816Z	1856.68	0.29	0.29	0.29	0.87	1.33	1.33	1.33	1.33	50	407.62	232.79	405.61	238.43
Schneider	2019–09–25T05:22:49.841Z	1856.69	1.44	1.44	1.44	4.32	7.41	7.41	7.41	7.41	50	407.47	232.51	405.44	238.42
Schneider	2019–09–25T05:22:56.855Z	1856.7	1.29	1.29	1.29	3.88	6.97	6.83	6.97	6.97	50	406.98	232.23	405.12	238.15
Schneider	2019–09–25T05:23:03.828Z	1856.71	1.23	1.23	1.23	3.69	6.55	6.58	6.55	6.55	50	406.33	231.56	404.69	237.86
Schneider	2019–09–25T05:23:10.825Z	1856.71	1.28	1.28	1.28	3.84	6.79	6.73	6.79	6.79	50	406.49	232.12	404.5	238.12
Schneider	2019–09–25T05:23:17.861Z	1856.72	1.26	1.26	1.26	3.78	6.75	6.68	6.75	6.75	50	405.12	231.2	402.22	237.94
Schneider	2019–09–25T05:23:24.858Z	1856.73	1.64	1.64	1.64	4.93	7.52	8.19	7.52	7.52	50	404.91	230.82	404.1	237.42
Schneider	2019–09–25T05:23:31.858Z	1856.74	2.1	2.1	2.1	6.31	10.3	10.3	10.3	10.3	50	407.41	232.7	404.62	237.17
Schneider	2019–09–25T05:23:38.877Z	1856.75	2.06	2.06	2.06	6.18	10	10.1	10	10.1	50	407.78	232.87	405.58	238.22

References

[1] X. Chen, C. Li, Y. Tang, Q. Xiao, An Internet of Things based energy efficiency monitoring and management system for machining workshop, J. Clean. Prod. 199 (2018) 957–968.
[2] I. Henao-Hernandez, E.L. Solano-Charris, A. Muñoz-Villamizar, J.S.R. Henriquez-Machado, in: Control and Monitoring For Sustainable Manufacturing in the Industry 4.0: A literature Review, IFAC Papers Online, 2019, pp. 195–200.
[3] K.O. Riellya, J. Jeswieta, The need for better energy monitoring within industry, Procedia CIRP 29 (2015) 74–79.
[4] V. Kolhe, K. Kolhe, S. Choudhari, S. Khedekar, U. Zambare, K. Vadirajacharya, A case study: energy conservation through energy audit at Kandalgaon substation, IEEE International Conference on Energy, Communication, Data Analytics and Soft Computing, 2017.
[5] K. Matsui, Y. Yamagata, S. Kawakubo, Real time sensing in residential area using IoT Technology for finding usage patterns to suggest action plan to conserve energy, Energy Procedia 158 (2019) 6438–6445.
[6] A.J. Lewis, M. Campbell, P. Stavroulakis, Performance evaluation of a cheap, open source, digital environment monitor based on the Raspberry Pi, Measurement 87 (2016) 228–235.
[7] A. Mouapi, H. Mrad, A. Parsad, Implementation of a reliability test protocol for a multi measurement sensor dedicated to industrial applications of the Internet of Things, Measurement (2019).
[8] A.H. Alavi, P. Jiao, W.G. Buttlar, N. Lajnef, Internet of Things- enabled smart cities: state-of-the-art and future trends, Measurement (2018).
[9] L Wilson, Junior Rodrigues, A.S. Fabbio, A.F. Borges, da S. Veloso, Ricardo de, A.L. Rabêlo, Joel, J.P.C. Rodrigues, Low voltage smart meter for monitoring of power quality disturbances applied in smart grid, Measurement 147 (2019) 106890.
[10] Y.H. Tehrani, S.M. Atarodi, Design & implementation of a high precision & high dynamic range power consumption measurement system for smart energy IoT applications, Measurement 146 (2019) 458–466.
[11] M. Gerardo, A. Spinelli, L. Zach, B. Gottesman, A low-cost Arduino-based data logger with cellular modem and FTP communication for irrigation water use monitoring to enable access to Crop Manage, HardwareX 6 (2019) e00066.
[12] M. Ambroz, Raspberry Pi as low-cost data acquisition system for human powered vehicles, Measurement 100 (2017) 7–18.
[13] V.R. Shinde, P.P. Tasgaonkar, R.D. Garg, Environment Monitoring system through Internet of Things(IOT), in: IEEE International Conference on Information, Communication, Engineering and Technology, 2018, pp. 29–31. Aug.
[14] R.I.S. Pereira, I.M. Dupont, P.C.M. Carvalho, S.C.S. Juca, IoT embedded system based on Raspberry Pi applied to read-time cloud monitoring of decentralized photovoltaic plant, Measurement 114 (2018) 286–297.
[15] M. Sridharan, R. Devi, C.S. Dharshini, M. Bhvadarani, IoT based performance monitoring and control in counter flow double pipe heat exchanger. 29 November 2018, IoT 5 (2019) 34–40.
[16] M.O. Agyeman, Z. Al-Waisi, I. Hoxha, Design and implementation of an IoT-based energy monitoring system for managing smart homes, IEEE Fourth International Conference on Fog and Mobile Edge Computing, 2019.
[17] W.-T. Sung, S.-J. Hsiao, The application of thermal comfort control based on smart house system of IoT measurement 149 (2020).
[18] P. Pawar, M. Tarun Kumar, K. Panduranga Vittal, An IoT based Intelligent smart energy management system with accurate forecasting and load strategy for renewable generation, Measurement (2019).
[19] F. Abate, M. Carratù, C. Liguori, V. Paciello, A low cost smart power meter for IoT, Measurement 136 (2019) 59–66.
[20] S.G. Nikhade, Wireless sensor network system using Raspberry Pi and ZigBee for environmental monitoring applications, IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, 6–8, 2015 May.
[21] S. Shapsougha, M. Takrourib, R. Dhaouadic, Imran Zualkernand an IoT-based remote IV tracing system for analysis of city-wide solar power facilities, Sustain. Cities Soc. 57 (2020) 102041.
[22] A. Hafid, S. Benouar, M. Kedir–Talha, F. Abtahi, Mokhtar Attari and Fernando, full impedance cardiography measurement device using Raspberry PI3 and system-on-chip biomedical instrumentation solutions, IEEE J. Biomed. Health Inform. (2017).
[23] R.A Nadafa, S.M. Hatturea, V.M. Bonala, S.P Naik, Home security against human intrusion using Raspberry Pi, Procedia Comput. Sci. 167 (2020) 1811–1820.
[24] S.E. Oltean, Mobile robot platform with Arduino Uno and Raspberry Pi for autonomous navigation procedia manufacturing 32 (2019) 572–577.
[25] H.H.M. Yusof, S.W. Harun, K. Dimyati, T. Bora, K. Sterckx, W.S. Mohammed, J. Dutta, Low-cost integrated zinc oxide nanorods based humidity sensors for arduino platform, IEEE Sens. J. (2018).
[26] O. Oussalem, M. Kourchi, A. Rachdy, M. Ajaamoum, H. Idadoub, S. Jenkal, A low cost controller of PV system based on arduino board and INC algorithm, in: Materials Today: Proceedings, 2019.
[27] A. López-Vargas, M. Fuentes, M. Vivar, F.J. Muñoz-Rodríguez, Low-Cost Data logger Intended For Remote Monitoring Of Solar Photovoltaic Stand-Alone Systems Based On Arduino, IEEE Sensors, 2018.
[28] A. López-Vargas, M. Fuentes, M. Vivar, IoT application for real-time monitoring of Solar Home systems based on arduino with 3G connectivity, IEEE Sens. (2018).
[29] C. Ulloa, J.M. Nufiez, A. Suarez, C. Lin, Design and development of a PV-T test bench based on arduino, Energy Procedia 141 (2017) 71–75.
[30] M. Huba, P. Bistak, T. Huba, Filtered PI and PID control of an arduino based thermal plant, IFAC-Papers Online 49-25 (2016) 336–341.
[31] J. Belwyn Samson, R. Catherine Joy, K. Alwin Fredrick, W. Joel Wesley, M. Nithin Sathiya, S. Stanley Samuel, Smart Energy Monitoring Using Raspberry Pi, Third International Conference on Computing Methodologies and Communciation IEEE, 2019.
[32] M. Gagliarducci, D.A. Lampasi, L. Podesta, GSM-based monitoring and control of photovoltaic power generation, Measurement 40 (2007).
[33] C. Chupong, B. Plangklang, Electricity Bill Forecasting Application by Home Energy Monitoring System, 5th International Electrical Engineering Congress, IEEE, 2017.
[34] K.M.E. Galera, O.E. Llantos, Mobile web energy monitoring system using DFRduino Uno, Procedia Comput. Sci. 124 (2017).
[35] N.A. Othman, M.R. Zainodin, N. Anuar, N.S. Damanhuri, Remote monitoring system development via Raspberry- Pi for small scale standalone PV Plant, in: 7th IEEE International Conference on Control System, Computing and Engineering, 24–26, 2017, pp. 360–365. November.
[36] F. Stradolini, A. Tuoheti, T. Kilie, D. Demarchi, S. Carrara, Raspberry-Pi based system for protocol monitoring, Integration, VLSI J. 63 (2013) 213–219.
[37] V. Subramanian, A. Perumal, B. Krishnamoorthy, S.K. Rai, Implementation of effective and low-cost building monitoring system(BMS) using Raspberry Pi, Energy Procedia 143 (2017) 179–185.
[38] E.B. Priyanka, C. Maheswari, S. Thangavel, IoT based field parameters monitoring and control in press shop assembly, IoT 3-4 (2018) 1–11.
[39] C.-S. Choi, J.-D. Jeong, J. Han, W.-K. Park, I.-W. Lee, Implementation of IoT based PV monitoring system with message queuing telemetry transfer protocol and smart utility network, IEEE Information and Communication Technology Convergence, 2017.
[40] V.V. Garbhapu, S. Gopalan, IoT based low cost single sensor node remote health monitoring system, Procedia Comput. Sci. 113 (2017) 404–415.
[41] I. Gonzalez, A.J. Calderon, Integration of open source hardware Arduino platform in automation, Sustain. Energy Technol. Assess. (2019) 100557.
[42] B. Lahfaoui, S. Zouggar, B. Mohammed, M.L. Elhafyani, Real time study of P&O MPPT control for small wind PMSG turbine systems using Arduino microcontroller, Energy Procedia 111 (2017) 1000–1009.
[43] Techdata: Raspberry Pi Foundation Raspberry Pi3 B. Available at: <https://openwrt.org/toh/hwdata/raspberrypifoundation/raspberrypi3b> (Accessed January 15, 2020).
[44] BCM2711 - Raspberry Pi Documentation. Available at: <http://www.raspberrypi.org> (Accessed January 17, 2020).
[45] Report: Arduino – Arduino Board Uno SMD. Available at: <http://www.arduino.cc> (Accessed January 16, 2020).
[46] Web reference: <https://www.influxdata.com>(Accessed January 19, 2020).
[47] Web reference: <https://grafana.com>(Accessed January 19, 2020).