

RESEARCH ARTICLE

Design And Development of a Remote Monitoring Agent For Energy Service Companies In The Telecommunication Industry [version 1; peer review: 1 approved with reservations]

Sunkanmi Oluwaleye , Francis Idachaba

Electrical and Information Engineering Department, Covenant University, Ota, Ogun State, 112225, Nigeria

V1 First published: 27 Sep 2022, 11:1106
<https://doi.org/10.12688/f1000research.123632.1>
Latest published: 27 Sep 2022, 11:1106
<https://doi.org/10.12688/f1000research.123632.1>

Abstract

Background: Energy Service Companies (ESCOs) for telecommunication sites operate by providing reliable power supply at 100% uptime and billing the mobile operators accommodated on their sites for power usage. To achieve this, there is a need for them to accurately meter the power consumed by connected telecommunication equipment.

Method: This work focused on the design and implementation of a remote monitoring agent (RMA) that will pool both power and environmental data from a telecommunication site. The data pool can be presented as real-time data on the RMA's webpage, it can be downloaded as historical data, and it can be sent to a remote cloud server at regular intervals. The RMA collects both power and environmental data over an RS485 Modbus network and I2C bus respectively. An alternating current (AC) energy meter and a direct current (DC) energy meter were used to harvest the energy data while the environmental data were harvested using a developed Input/Output controller board based on an Atmega328P microcontroller. Raspberry pi was used as the master controller and Node.js was used to build the application running on the master controller.

Result: The result showed how both power and environmental data can be harvested from a telecommunication site and locally presented on the web dashboard for real-time monitoring of the site power system. The data could be saved locally on the RMA and downloaded for future use.

Conclusion: The implementation of this work provided a prototype of the remote monitoring agent (RMA) that can be deployed by Energy Service Companies (ESCOs) in the telecommunication industry to monitor the usage of the power systems on a cell site.

Open Peer Review

Approval Status ?


1

version 1

27 Sep 2022

?

[view](#)

1. **Kenneth Eloghene Okedu** , National University of Science and Technology, Muscat, Oman

Any reports and responses or comments on the article can be found at the end of the article.

Keywords

Monitoring, Raspberry pi, Atmega328P, Node.js, RS485 Modbus, Power, Telecommunication, Programming, Communication.



This article is included in the **Energy** gateway.

Corresponding author: Sunkanmi Oluwaleye (sunkanmiolade@gmail.com)

Author roles: **Oluwaleye S:** Conceptualization, Methodology, Resources, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Idachaba F:** Investigation, Supervision, Validation

Competing interests: No competing interests were disclosed.

Grant information: The author(s) declared that no grants were involved in supporting this work.

Copyright: © 2022 Oluwaleye S and Idachaba F. This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Oluwaleye S and Idachaba F. **Design And Development of a Remote Monitoring Agent For Energy Service Companies In The Telecommunication Industry [version 1; peer review: 1 approved with reservations]** F1000Research 2022, **11**:1106 <https://doi.org/10.12688/f1000research.123632.1>

First published: 27 Sep 2022, **11**:1106 <https://doi.org/10.12688/f1000research.123632.1>

Introduction

The global increase and acceptance of mobile technology has dramatically led to demand for internet penetration by both mobile operator companies and end-users. In order to meet this demand without increasing the environmental pollution the proliferation of masts that individual mobile operators would have caused.^{1,2} The government of countries like Nigeria has made rules to encourage infrastructure sharing among the mobile operators and the tower companies (Towercos) or energy service company (ESCOs) have availed themselves of this opportunity.³ The ESCO provides the platform for multiple mobile operators to share passive infrastructures such as mast, shelter, space, and the required power supply system.⁴ This also reduces the initial cost of building site infrastructure by new entrance.⁵ The ESCOs make revenue from the rent and power consumption fees paid by the mobile operators. However, the ESCOs pay fines when failing to meet up with the 100% power uptime required by the mobile operators' equipment.⁶ Hence, for the ESCOs' operation model to be profitable, one of the important measures that must be ensured is that each tenant's equipment is well metered so that mobile operators can be accurately billed for the power consumed by their equipment.

In this work, a remote monitoring agent (RMA) that will pool both power and environmental data from mobile operators' equipment was developed. The data pool could be presented as real-time data on the RMA's webpage, it can be downloaded as historical data, and it can be sent to a remote cloud server at regular intervals. The remote monitoring agent (RMA) hardware was designed and built using Raspberry Pi 3 B+, IO board built on Atmega328P microcontroller, AC energy meter (Hop energy meter model: DTZ1737), DC energy meter (Hop Energy meter model: DJZ1737), and 3G USB modem as shown in Figure 1. The energy meters were manufactured by Chengdu HOP Telecom Co., Ltd. Node.js (Node.js, OpenJS Foundation) was used to develop the application running on the Raspberry Pi (master controller) and the C programming language was used to program Atmega328P microcontroller using Arduino IDE (Arduino IDE 1.8.19, Arduino).⁷ One RMA was deployed for one telecommunication site in Lagos, Nigeria, to monitor the pattern of power consumed by all the mobile operators accommodated at the site.

Literature review

Providing a reliable and affordable power solution for telecommunication sites has been a major source of concern in the industry.⁸ The power solution architecture generally implemented is shown in Figure 2.⁹ Typically, the power solution will have AC-DC converters (rectifiers) that convert the AC power source coming from either the grid power supply or the AC Genset to DC power that charges the battery and powers the tenant loads.^{6,10} The optional DC-DC converter in the power solution will be an MPPT (maximum power point tracking) solar charge controller for sites implementing hybrid power solutions.^{10,11} The DC-DC converter delivers DC power in an acceptable voltage range to the output power bus of the power equipment. The power from the DC-DC converter can be complemented with the battery bank to reduce Genset run hours, which provides financial savings and a reduction in the emission of carbon dioxide into the atmosphere.

Many works have been done to provide reliable and cost-saving power solutions to telecommunication sites. A study¹² designed and implemented a solar hybrid power solution for off-grid telecommunication sites; a diesel generator was used to support the site whenever there was insufficient energy harvested from the solar system. The system, after implementation, reduces operation costs by 80%. To further avoid the environmental pollution caused by the diesel gensets,^{8,13} presented the use of proton exchange membrane fuel cell (PEMFC) as an optimized and environmentally friendly power solution for the telecommunication site.

A remote monitoring system is just as crucial for ESCOs involved in the telecommunication tower industry as putting in place a reliable power solution for the sites under their management.⁸ Utilize monitoring of power distribution and

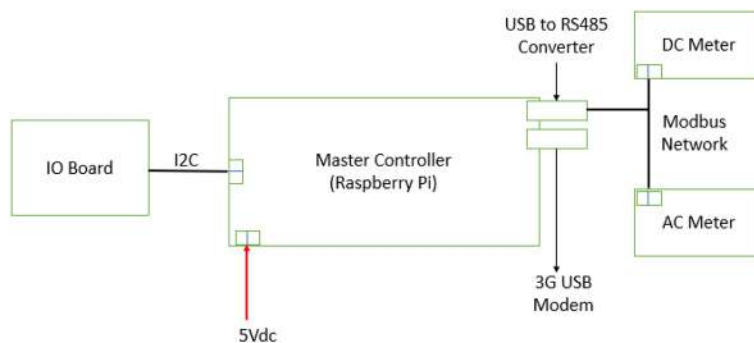


Figure 1. RMA hardware architecture.

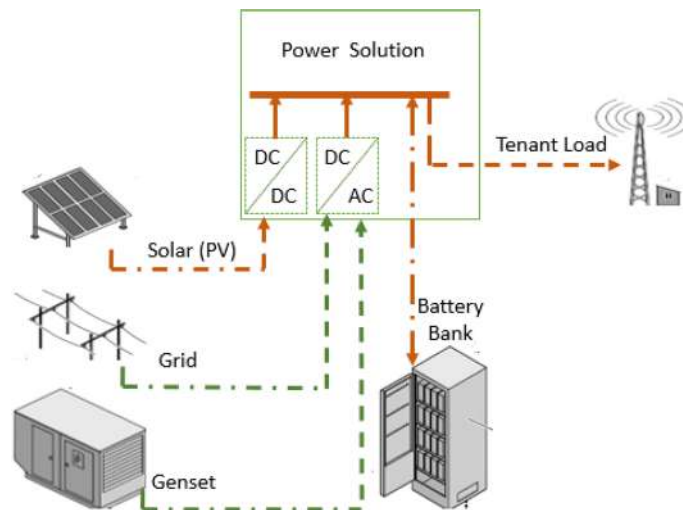


Figure 2. Power solution architecture for telecommunication site. DC: direct current, AC: alternating current, PV: photovoltaic.

consumption to keep tabs on the installed power solution's energy and performance parameters. For the purpose of gathering environmental information, such as internal and external temperatures and relative humidity, a wireless sensor network based on the ZigBee standard was used.¹⁴ Similarly, the power consumption by the telecommunication equipment, cooling devices, and other auxiliary devices were all collected. This data was used to analyze the energy balance of the solution, which revealed how energy was distributed at the site.¹⁵

Another study,¹⁶ focused on building a low-cost remote monitoring system for a solar plant using RS485 Modbus RTU communication with Raspberry Pi. To efficiently monitor the system for preventive maintenance, the raspberry pi pooled operational data from the plant's solar charge controller and uploaded it to a dedicated central server. RS485 RTU communication protocol is frequently used for data transfer between the main controller and other peripheral devices in embedded solution. Its key benefits are long-distance data transfer and great robustness to an electrically noisy environment... Multiple nodes (up to thirty-two) can be connected and communicate over the RS485 RTU bus.^{17,18} demonstrated a master-slave RS485 communication network using a Raspberry Pi as a Modbus master and Arduino Pro minis as slave nodes. The user could edit the programs of the master and slave nodes from the built mobile application using a graphical language. The Raspberry Pi acted as the master and the server for the web application, while the Arduino boards served as the slave nodes.¹⁹ used the I2C communication protocol to integrate legacy equipment to the industrial internet of things (IIoT) infrastructure. I2C communication is also like RS485 Modbus but I2C is a synchronous serial communication protocol that can have multiple masters and slaves on the same bus network of connected devices.²⁰ The use of these communication protocols enables the remote monitoring agent to collect data from multiple sensors installed on the telecommunication.^{21,22}

The aim of this work is to implement most of the technologies reviewed above to develop a RMA that meet the need of ESCOs in the telecommunication industry. The RMA served as data collection unit and was able to present real-time power and environmental data locally over its web interface. The RMA should also send data and alarm events to a cloud server for remote monitoring over the cloud. The unique identification of the telecommunication site can be done on the configuration page in the RMA's local web page.

Methods

System architecture

This work can be categorized into two major developments; the development of the RMA's hardware and the development of the RMA's software. A Raspberry Pi 3 B+ board and Atmega328P microcontroller are the controllers used for the development of the RMA. The Atmega328P was used to develop an IO controller board that interfaced with field devices such as temperature sensors, voltage measurement circuits, digital input sensors, and digital output actuators. The Raspberry Pi was used to pool site power data from RS485 Modbus enabled energy meters, as shown in Figure 3. Arduino IDE (integrated development environment) was used to program the Atmega328P, while the Raspberry Pi was programmed with bash shell scripting and Node.js.

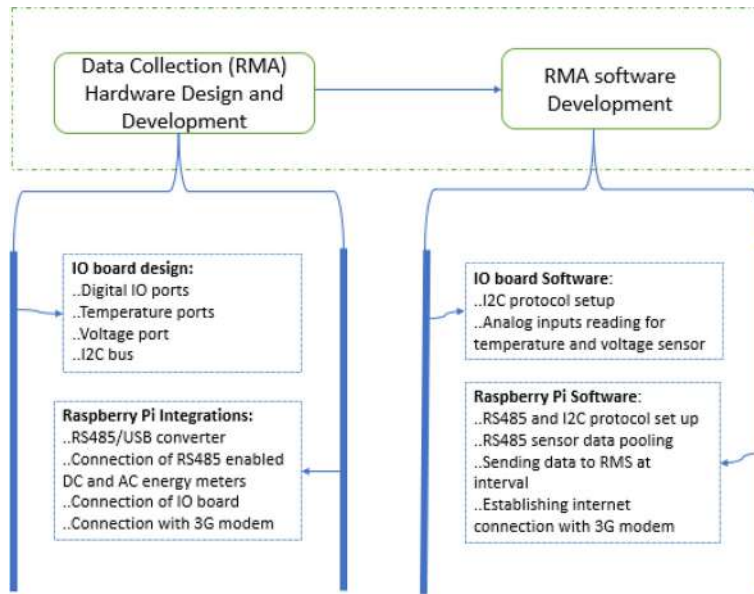


Figure 3. Solution architecture. Remote monitoring agent (RMA), Input/output (IO). DC: direct current, AC: alternating current, I2C: Inter-Integrated Circuit, RS485: ANSI/TIA/EIA-485, USB: universal serial bus.

Design and development of the remote monitoring agent (RMA)

This involves the design and implementation of both the hardware and software required to build the RMA. It included the design and implementation of the data acquisition controller board and programming the Raspberry Pi as the master controller to pool data from the data acquisition controller board and the connected energy meters. The Raspberry Pi locally stored the site data and also upload the data to a cloud server every two minutes. The software tools used to design and develop the data acquisition controller board or input/output (IO) board are EasyEDA and Arduino IDE. The EasyEDA software²³ was used for the electronics design and generating the printed circuit board (PCB) Gerber file format was used for the PCB production. Arduino IDE version 1.8.19 was used to program the Atmega328P for digital and analog input processing that produced the digital output signals that controlled the actuators connected to the RMA. The digital input states and the analog input values captured by the RMA were shared with the master controller (the Raspberry Pi) over I2C serial communication. Windows 10 OS HP intel Core i5 8th generation PC (personal computer) was used as the programming device and for the PCB design.

As shown in Figure 4, the Atmega328P was the principal component of the IO board, the digital input circuitry was built with 12Vdc relays. The digital output circuitry was built with 5Vdc relays, BC547 transistors, 1N4007W diodes, and 1kΩ resistors. The programming device, PC used the programmable universal synchronous/asynchronous receiver/transmitter (USART) port of the microcontroller for program data transfer and serial data monitoring for debugging purposes. The IO board was powered with 5Vdc. The main hardware materials required to build the IO board (Table 1), are Atmega328P microcontroller (with preloaded bootloader), 16MHz crystal, 10kΩ resistor, 1kΩ resistor, 100Ω resistor, 5.1kΩ resistor,

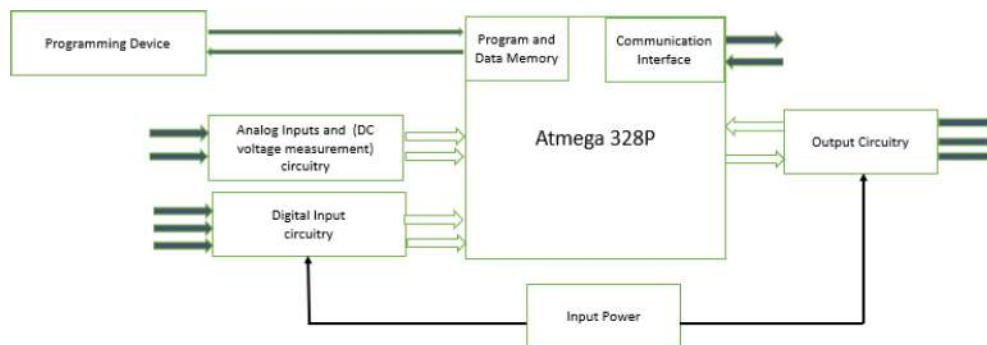


Figure 4. Structure of the input/output (IO) board. DC: direct current.

Table 1. List of input/output (IO) board components. S/N: Item Number, DC: direct current, Conn: connector, PCB, printed circuit board.

IO board components			
S/N	Name	Description	Quantity
1	Conn-Th_2p-P5.00	5vdc-Input,12vdc-input terminal connector	2
2	Srd-5vdc-SI-C	Pcb 5vdc Relay	5
3	Jqc-3ff/12vdc	Pcb 12vdc Relay	5
4	10uf	Electrolytic capacitor C1	1
5	47pf	Ceramic capacitor (C2,C3)	2
6	Terminals/Wj2edgv-5.08-8p	Molex terminal connectors - Cn1,Cn2,Cn3,Cn4	4
7	1n4007w	Diodes - D1,D2,D3,D4, D5	5
8	1n5370bg	Zener diode - D6	1
9	5x20 Blx-Atype-Fuse Holder Xc-7	Fuse holder F1	1
10	5x20 fuse holder	Fuse holder F2	1
11	Header-Male-2.54_1x4	2 Pin header H1, Jp1,Jp2	3
12	Led-0603_R	Red led	5
13	Led_R	Green led5	1
14	Bc547	Transistor - Q1,Q2,Q3,Q4,Q5	5
15	1k	1 kilo ohm resistors - R1,R2,R4,R7,R3, R5	6
16	100k	100 kilo Ohms resistor - R8	1
17	5.1k	5.1 kilo Ohms resistor - R9	1
18	10k	10 kilo Ohms resistor - R6	1
19	49s	16mhz crystal - X1	1
20	Atmega328-Pu	Atmega328p microcontroller - U1	1

5Vdc PCB board relay, 12Vdc PCB board relay, BC547 transistors, 1N4007W diodes, 47pF ceramic capacitor, 10uF electrolytic capacitor, and WJ2EDGV-5.08-8P terminal connectors.

Hardware schematic design using EasyEDA

EasyEDA, an online electronics design and PCB development platform, was used because it does not require local software installation on the PC.²³ The design was done online and saved in the cloud, to be accessed with any PC in the future that has minimum specification of Windows 7 and above, 1 gigahertz (GHz) processor, 1 gigabyte (GB) (32-bit) or 2 GB (64-bit) RAM (random access memory), 16 GB (32-bit) or 20 GB (64-bit) Hard disk space, Microsoft DirectX 9 graphics device with a Windows Display Driver Model graphics card, and internet access. The electronic circuitries of the IO board were designed with EasyEDA using the available electronics components on the platform and component packages were also created for the components that were not available on the platform to create mounting space for it on the PCB board. The most important component of this design was the Atmega328PU microcontroller, and all other circuitry was built around this microcontroller as shown in [Figure 8](#). This microcontroller is based on the Advance Virtual RISC (AVR), developed by Microchip. It supports 8-Bit data processing with 32kB internal flash memory, 1kB electrically erasable programmable read-only memory (EEPROM), and 2kB static random-access memory (SRAM).

The Atmega328PU microcontroller component was first picked on the EasyEDA design sheet. Each component placed on the sheet was linked to other components using the wire tool. To avoid the complexity of the wiring arrangement, a tool called netPort was used to connect one point of a component to another component without drawing wires on the design sheet. The important circuitries to consider in this design are the digital output circuitries, digital input circuitries, and the DC voltage measurement circuitry.

The digital output circuitry is functionally called the relay driver circuit. The circuit consist of a 5Vdc relay which was controlled by the digital output of the Atmega328P to switch relatively larger current and higher voltage on its dry contact. The relay driver circuit consists of a transistor, resistor, and diode. The transistor was used as an amplifier; it amplified the

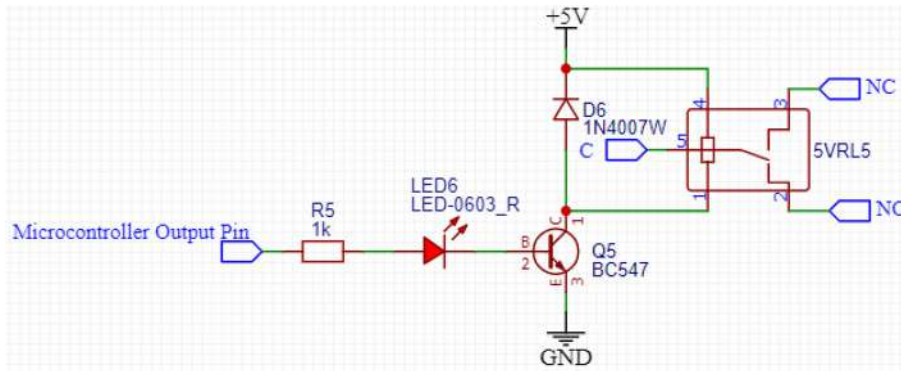


Figure 5. Relay driver schematic. GND: ground, NC: normally close, NO: normally open.

small current flowing from the microcontroller output pin so that full current from the 5Vdc power supply source can energize the coil of the connected relay. The resistor was used to bias the transistor, and the diode was used to prevent reverse current produced on the relay’s inductor coil from damaging the output pin of the microcontroller. Finally, an LED (light emitting diode) was added to show the state of the IO board digital output pin. In operation, the incoming voltage source to be switched is connected to the common of the relay labeled C and the outgoing end is connected to the normally open terminal labeled NO. This drawing schematic for this circuitry is shown in [Figure 5](#).

The digital input circuitry is comprised of a 12Vdc relay. The coil of the relay was driven by the 12Vdc voltage supply, one terminal of the relay coil was already terminated to the positive 12Vdc source on the board while the negative 12Vdc ground (GND) was wired to pass through the digital sensor switch to close the circuit that energized the relay coil when the signal event was sensed. The relay common terminal was connected to the 5Vdc GND, when the relay would be energized the common terminal became connected to the normally open terminal on which the microcontroller digital input pin was connected. This schematic is shown in [Figure 6](#).

The DC voltage measurement circuitry was designed with 100kΩ and 5.1kΩ resistors, and a Zener diode. The resistors were wired to serve as the required voltage divider circuit that ensured that the voltage to the microcontroller analog input pin did not exceed 5V when the DC voltage range of 0–100V was measured. The Zener diode served as a protection for the microcontroller if the supplied input voltage was higher than the acceptable voltage range the circuit could measure. The schematic is shown in [Figure 7](#). The formula to calculate resistor labelled 8 (R8) in the [Figure 7](#) is given in [equation 1](#).

$$R8 = \frac{V_{in} - 5}{0.001} \tag{1}$$

Where V_{in} is the maximum voltage that can be measured. The general formula to find the V_{in} is given in [equation 2](#).

$$V_{in} = \left(1 + \frac{R8}{R9} \right) \tag{2}$$

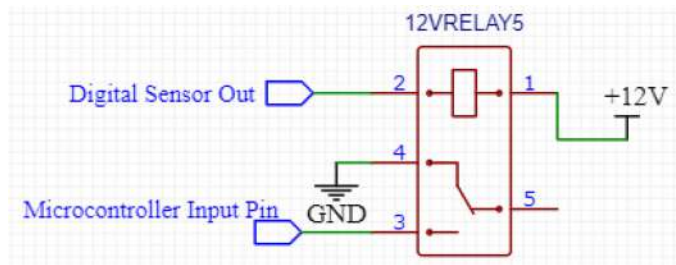


Figure 6. Schematic of the digital input circuitry. GND: ground.

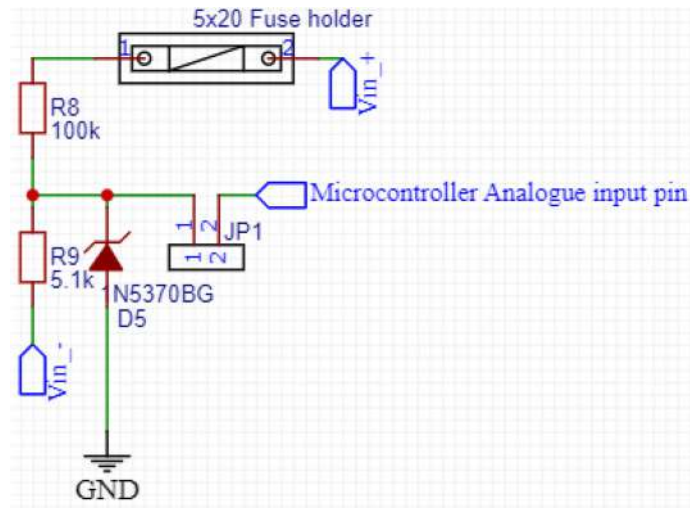


Figure 7. Schematic of the direct current (DC) voltage measurement circuit. Vin: voltage input, GND: ground, R: resistor.

The components listed in Table 1 were locally purchased in an electronics shop and soldered to the printed PCB board according to the design in Figure 8. The soldering was carefully done to avoid possible short circuits that excessive soldering lead could cause. After soldering, the PCB board was properly cleaned to remove unwanted leads that could affect the operation of the board.

Software development: programming of the IO board

The IO board’s microcontroller was programmed with C programming language using Arduino IDE, an open-source IDE that makes it easy to program and upload code to Arduino boards. The software was installed on an Intel Core i5 HP computer. An Arduino Uno revision 2 board had no Atmega328P installed was used as the programming interface for the microcontroller on the IO Board. The program data transfer was done *via* the programmable USART (universal synchronous asynchronous receiver transmitter) port of the Atmega328P microcontroller at Tx (transmitter) and Rx (receiver) pins. From Figure 9, the Arduino Uno board was acting as the USB to transistor-transistor logic (TTL) Converter doing the required level shifting.

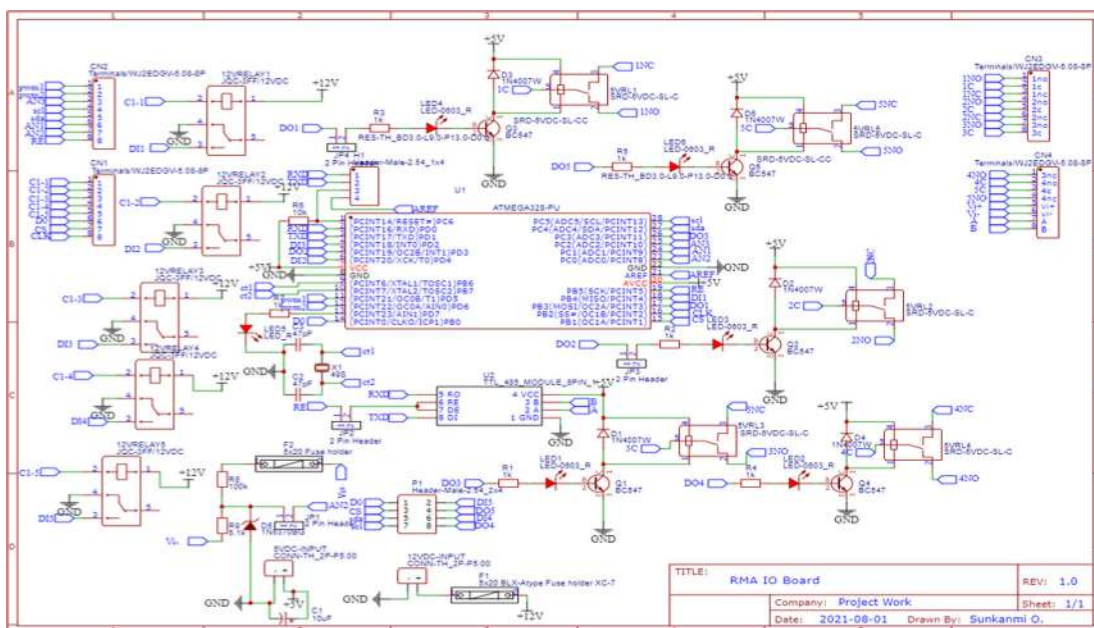


Figure 8. The schematic design of the input/output (IO) board created on the EasyEDA.

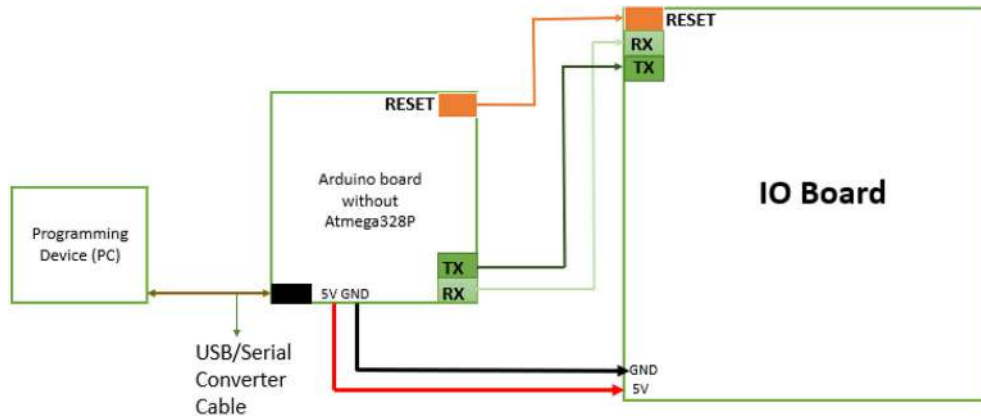


Figure 9. Programming the input/output (IO) Board with empty Arduino Uno revision 2 board. TX: Transmitter, RX: Receiver, USB: universal serial bus, GND: ground.

The C code was written and compiled on the Arduino IDE. The program structure is described by the flow chart algorithm given in Figure 10. The program entry point started from loading all the required libraries. These libraries were rewritten software that was used to extend the functionality of the sketch (Arduino code) written for the IO Board microcontroller. The libraries were as follows:

- Wire library: the library was used to access the SDA (Data line) and SCL (clock line) pins on the microcontroller for I2C communication with other connected I2C devices. In this case, the I2C device to communicate with was the master controller built on Raspberry Pi 3B.
- **MillisDelay** library: a part of the SafeString V3+ library; the library was used to implement non-blocking delay timers in the Arduino IDE sketch. These non-blocking delay timers were used to control when the I2C

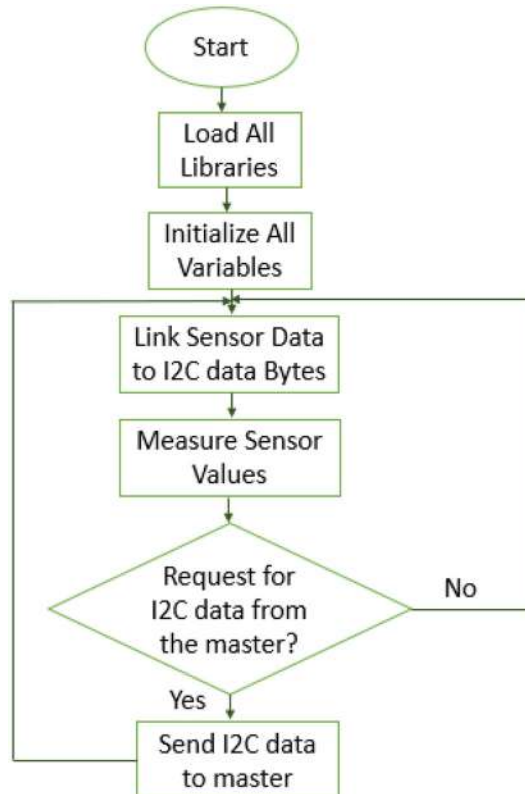


Figure 10. Flow chart of the C Program that runs on the input/output (IO) Board. I2C: inter-integrated circuit.

data byte would be read by the master device and when the sensor values would be captured from the connected sensors

- **TinyDHT** library: from Adafruit; this library was used to read temperature and humidity values from the DHT22 temperature and humidity sensor.

After loading all required libraries, all variables' initializations were done and the program endless loop began as shown in [Figure 10](#).

Integrating AC and DC energy meters

In other to work with power and energy measurement values that can be industrially trusted and acceptable, AC and DC energy meters were purchased from Chengdu HOP Telecom Co., Ltd, a company with certification and necessary compliance to produce energy meters. Both the DC and AC energy meters supported the RS485 RTU communication protocol. The DC meter came with six current sensors for six channels of power metering function using advanced measurement chip technology such as SMT (surface-mount technology). The AC meter was a three-phase four-wire smart meter with a single channel, primarily dedicated to capturing the Genset's energy values. The meter has performance compliance with GB/T 17215.321-2008 (AC measuring special equipment requirements). Also, its communication protocol complied with DL/T 645-2007 (Multi-function Energy Meter Communication Protocol) and YD/T 1363.3-2005 (Communication station power, air conditioner, and environment concentrated monitoring management system). The meter was supplied with three current sensors that measured each phase current of the Genset.

Integrating with AC energy meter

The AC meter had a single channel of three-phase input current measurement for the three-phase output power supplied from the Genset. [Table 2](#) shows the properties of the AC meter; it could measure AC input current up to 400A per phase. The working voltage was within DC 40V~273V or AC 40V~264V. 48VDC was used to power the AC meter since it was deployed in a telecommunication site where the available nominal voltage was 48VDC. The wiring of the AC meter was done as shown in [Figure 17](#). The labels L1, L2, L3, and N, were for the three-phase voltage sampling for the AC meter; the meter measured the phase voltages from this port. Power to the three current sensors and the measured current values from the sensors were respectively connected to ports labeled L1CT, L2CT, and L3CT. The connection points labeled A and B was the port for the RS485 communication with the master controller.

The AC meter shared its measured and calculated values with the master controller (Raspberry Pi) *via* an RS485 communication protocol. The protocol communication parameters for the established RS485 network were given as 9600, 8, N, and 1 are respectively the baud rate, data length, parity, and the stop bit. All the devices communicating over

Table 2. Properties of the alternating current (AC) meter. DC: direct current, A: amperage, RTU: remote terminal unit.

Properties of the AC meter	
Type	Three-phase four-wire
Rated Voltage Un(V)	230/400
Rated Current I n(A)	3 X (400) A
Working voltage:	DC 40V~273V or AC 40V~264V;
Reference frequency	50Hz;
Reference temperature:	23°C±2°C
Reference humidity:	40% ~ 60%
Normal working temperature	-10°C~+45°C
Limited working temperature	-25°C~+55°C
Communication	RS485 Modbus RTU
Current Sensor type	Current transformer (400A primary current/5A Secondary Current)
Current Sensor Dimension	Dimensions (L x W x D): 129mm×127mm×26mm
Bore diameter	φ65±0.5mm
Number of Current Sensors	3

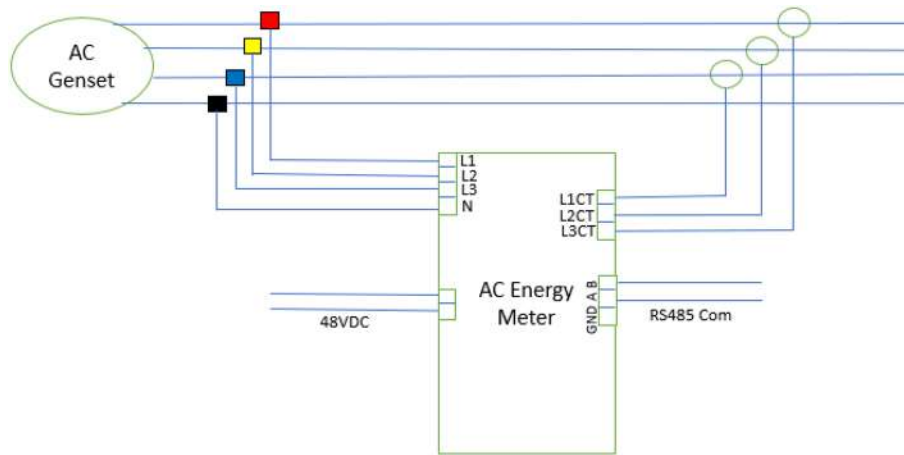


Figure 11. Alternating current (AC) energy meter hardware integration. L1: Line 1, GND: ground, AC: alternating current, L1CT: line 1 current transformer.

the network were set to have this same communication protocol parameters. The AC meter Modbus communication parameter was set using the Modbus register map supplied by the manufacturer manual^{24,25}

Integrating with DC energy meter

The DC meter features are shown in Figure 13. It had six power monitoring channels, and each channel had a current sensor of type hall sensor, which was developed using the open-loop Hall effect. The specification of the current sensor is given in Table 4. The nominal operating voltage of the DC meter was -48V having its positive terminal grounded as required in a typical telecommunication site power design.⁶ However, the DC meter could still operate within the voltage range of -60Vdc to -40Vdc. Both the energy measurement range and power range, as shown in Table 3, are sufficient for the application. Each current sensor was expected to measure the current consumed by the equipment installed by a

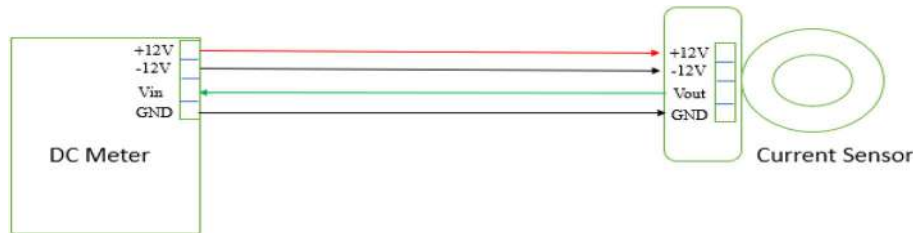


Figure 12. Wiring between hall sensor and meter. Direct current (DC), ground (GND). *Vin*: voltage in, *Vout*: voltage out.

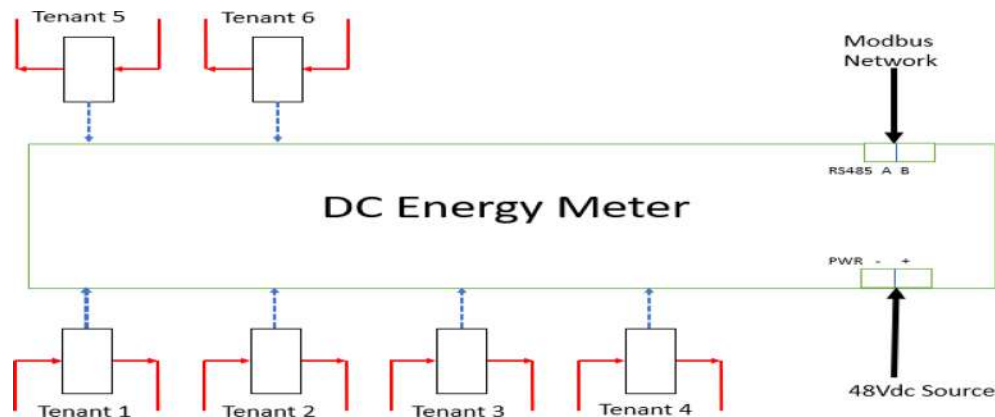


Figure 13. Direct current (DC) meter with current sensor wiring.

Table 3. Direct current (DC) meter specification. LCD: liquid crystal display, RTU: remote terminal unit.

DC meter specification	
Input mode	6 DC power monitoring channels
Rated nominal operating voltage	-48V
Input voltage range	-60V ~ -40V
Measuring range	The specified working voltage 0.8Un to 1.1Un
Extended working voltage	0.6Un to 1.1Un
Input current	400A
Impulse constant	6400imp/kWh
Energy measurement range	0 to 9999999.99kWh
Power range	0 to 99999.9999kW
Accuracy class	1.0/2.0
Clock accuracy	$\leq 0.5s/d$ (23°C), variation with temperature change is better than 0.1s/d
Display type	LCD display
Standby power consumption	$\leq 2W$
Altitude	$\leq 5000m$
Working temperature	-20°C to +60°C
Storage temperature	-40°C to +70°C
Working humidity	$\leq 98\%$ (40°C ± 2), no condensation
Communication	RS4885 RTU
Installation	35mm standard din rail installation
Dimensions	106.5mm x 87.5mm x 59mm
Weight	0.25kg

Table 4. Specification of direct current (DC) meter hall sensor. A: amperage, V: voltage, DC: direct current, Hz: Hertz, Ω : omhs, C: Celsius.

Specification of DC meter hall sensor (current sensor)	
Rated input current	0~400A
Rated output voltage	0~4V
Accuracy	1.0
Auxiliary power supply	DC $\pm 12V$
Offset voltage (Ta=+25°C)	$\pm 10mV$
di/dt follow	>50A/us
Linearity	$\leq 0.2\%FS$
Response time	$\leq 5 us$
Band width	0~20KHz
Load resistance	>10k Ω
Insulation voltage	2.5Kv/50Hz/1min
Insulation resistance	100M Ω
Working temperature	-40°C~85°C
Environment temperature	-25°C~70°C
Power consumption	20mA

particular mobile operator (tenant) on the site. The maximum current the hall sensor (model number: HL01-40-400P104) can measure was 400A, far higher than what a single tenant could consume. The wiring between a single hall sensor and the DC meter is shown in [Figure 12](#). The DC meter powers the hall sensor with 12VDC, the GND of the meter and the hall sensor were connected, and the hall sensor signal output Vout is connected to the DC meter Vin signal input. All the six current sensors were wired as shown in [Figure 13](#), the meter was powered from the 48Vdc source connected to its power supply port. The meter was connected to the Modbus network using its RS485 port.

The first thing to do in integrating the DC meter was to set the meter's Modbus communication parameter to 9600bps baud rate, 8 data bits, none parity check and 1 stop bit. The procedure used was similar to how the Modbus communication parameter of the AC meter was set. The only differences were that register addresses and acceptable register values were different. Relevant energy data were pooled using the Modbus register map supplied with the meter. With this parameter setting, the DC meter was able to communicate on the Modbus network.^{25,26}

Developing the RMA master controller

The master controller was built with Raspberry Pi B 3+, a mini microcomputer running on the Raspbian operating system developed from the Linux distribution Debian. Raspberry Pi has a lot of hardware peripherals that made it suitable for this project as shown in [Table 5](#); it has four USB ports, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless onboard and 40 GPIO (general-purpose input/output) pins.^{25,27}

The master controller was wired to other sub-units, as shown in [Figure 1](#). The USB to RS485 converter was used to connect the master controller to the Modbus network using one of its USB ports.²⁸ The Modbus network was made of two wires that were looped across the RS485 interface of the three devices (the DC meter, AC meter, and Raspberry Pi) connected to the network. The IO board was connected to the Raspberry Pi *via* the I2C serial communication interface. The I2C interface on SDA1 and SDL1 on pin 3 and 5 of the Pi GPIO were connected to the CN2 terminal, pin 5 and 4, routed on the PCB board to pin 27 and 28 of the Atmega328P microcontroller respectively.

The 3G USB modem was plugged into one of the free USB ports on the Pi to provide internet access, as shown in [Figure 20](#). 5Vdc was supplied to Pi from a 12/5Vdc DC-DC converter to power it.

Master controller software development

The software architecture used for the development of the RMA is shown in [Figure 14](#). The software component can be divided into two parts. The Node.js application and the shell script or Linux-based terminal commands. The shell script was used to configure the Pi's hardware peripheral operations. As shown in [Figure 15](#), the function of the shell script

Table 5. Raspberry Pi Main Features. CPU: central processing unit, SoC: system on chip, RAM: random access memory, GPU: graphics processing unit, USB: universal serial bus, GPIO: general purpose input/output.

Raspberry Pi main features	
Type	R3 B+
CPU	Quad Cortex A53 @ 1.2GHz
SoC	BCM2837
Processor operating voltage	3.3V
Raw input power	5V at 2Amps
Instruction set	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
Storage	Micro-SD
Ethernet	10/100
Wireless	802.11n/Bluetooth 4.0
USB access	4 USB 2.0 Ports
Power	Micro USB power source up to 2.5A (5Vdc)
GPIO	40
Operating temperature	-40 to +85°C

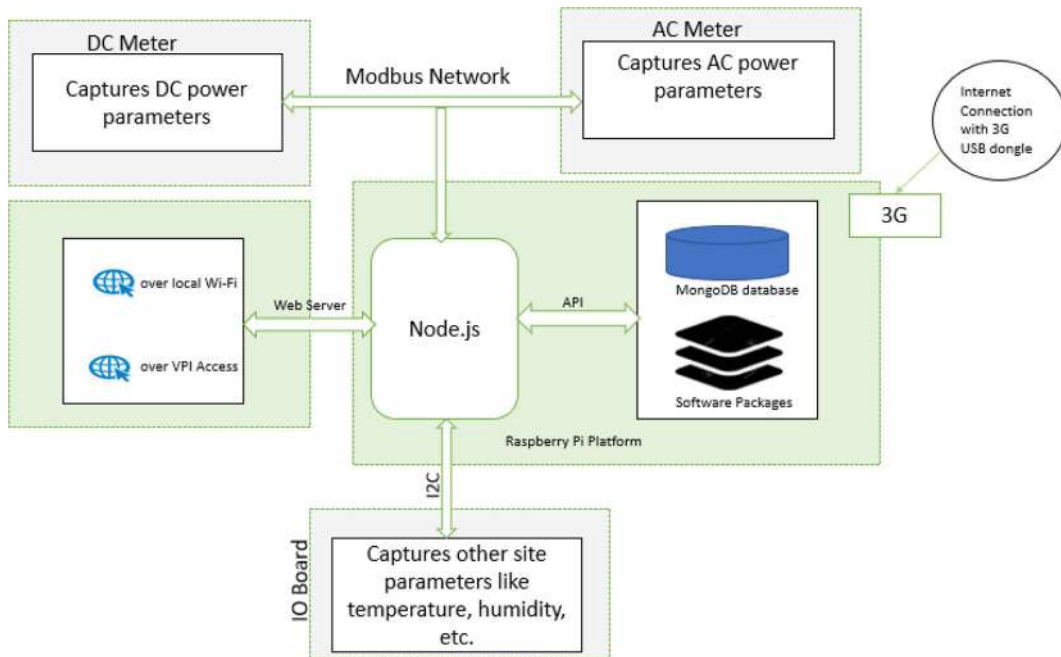


Figure 14. Master controller software architecture. API: application programming interface, DC: direct current, AC: alternating current, IO: input/output, USB: universal serial bus, 3G: third generation, VPI: virtual private internet.

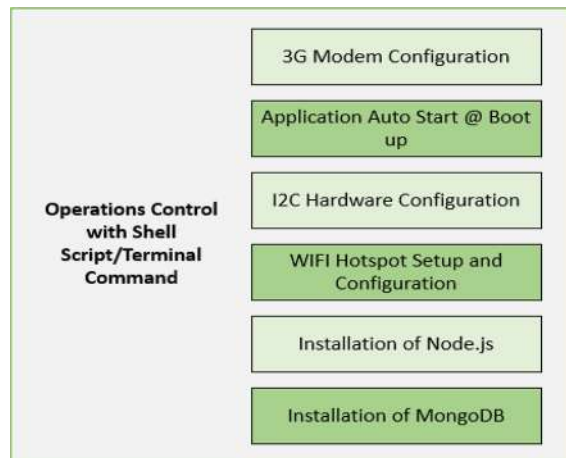


Figure 15. Shell script control operations. I2C: inter-integrated circuit.

programs used in the development could be categorized as 3G USB modem initialization and configuration, node.js application startup scripts, I2C hardware configuration, the configuration of Wi-Fi hardware as Wi-Fi hotspot spot to connect other Wi-Fi devices. Terminal commands were used to install node.js and MongoDB (MongoDB Inc) (RRID: SCR 021224) server.

The software development depended on some already existing libraries in the node.js ecosystem and Linux OS-based libraries compatible with the Raspberry Pi platform. The Linux OS-based libraries were developed using bash shell scripts/programs, while the node.js libraries were developed with JavaScript programming language. Figure 16 shows the development pattern; apt-get command was used to install all the Linux-based dependencies for the development, and node package manager (NPM) was used to install all the dependencies/libraries required for the developed node.js application.²⁹

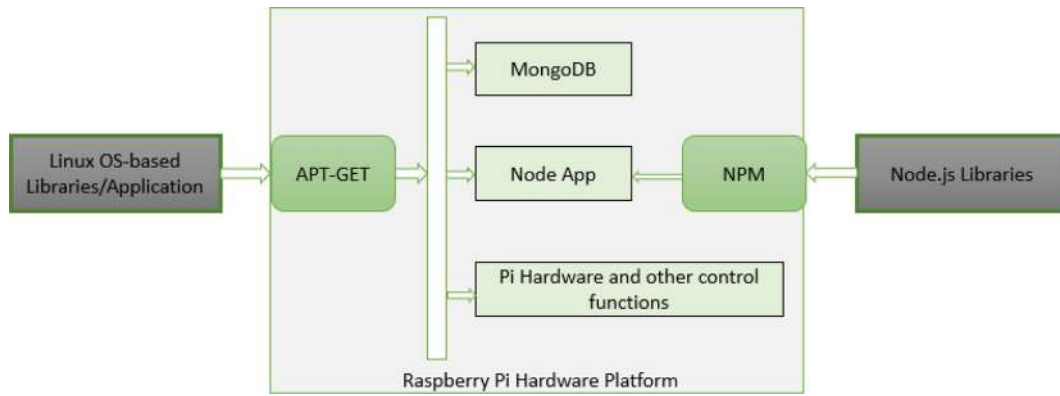


Figure 16. Master controller software development pattern, NPM: node package manager.

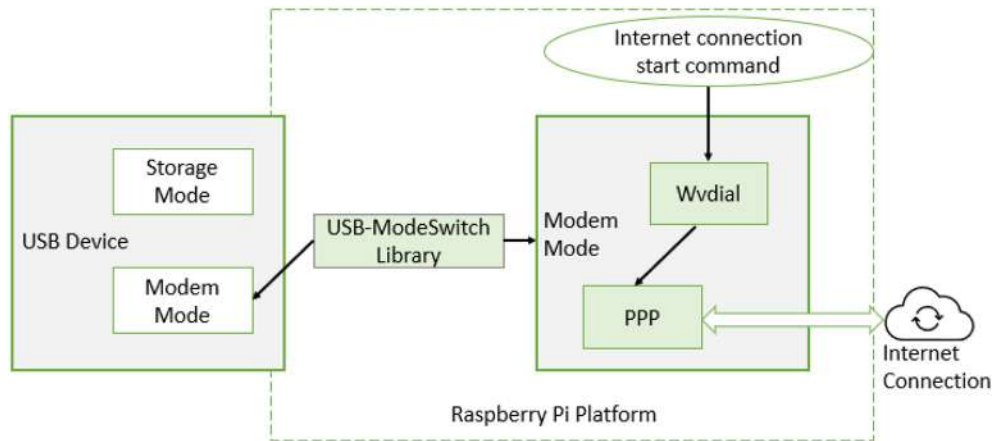


Figure 17. How to universal serial bus (USB) modem to the internet. PPP: point to point protocol, USB: universal serial bus.

Shell scripts in the master controller

In this work, both the executable bash shell script files and the terminal commands used for the development were referred to as shell scripts. This shell script was fundamental to getting the Raspberry Pi to function as expected. Figure 17 shows how the Raspberry Pi was connected to the internet using the USB modem. Three Linux-based libraries were required as follows:

- USB-modeswitch: This library was used to set the 3G USB device to modem mode instead of storage mode.
- PPP (point-to-point protocol): This is a TCP/IP protocol library used to connect one computer system to another over the internet.
- Wvdial; This is a Linux OS-based intelligent point-to-point protocol dialer library used for modem-based internet connection.

When initiating an internet connection, a custom shell script was developed with the algorithm shown in Figure 18. At boot-up or if the internet connection was not detected after 40 minutes, the script initiated the wvdial program that started the PPP to try a modem-based connection to the internet. This connection procedure was possible because the USB-modeswitch had forced the USB modem to the modem mode as shown in Figure 24.

Node.js application in the master controller

The Node.js programming environment was set up by installing both the Node.js and MongoDB with the apt-get command, as shown in Figure 16.^{30,31} Installation of the Node.js package automatically comes with the node package

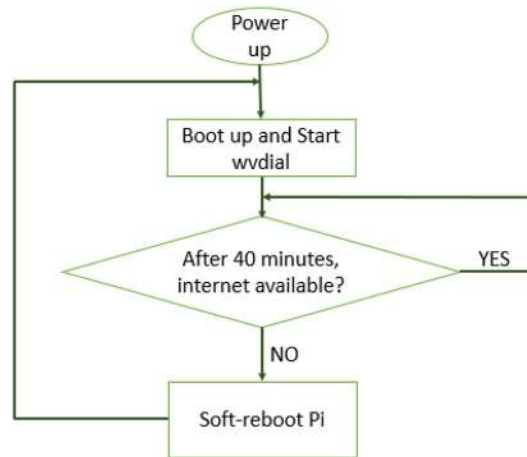


Figure 18. Flow chart for the internet connection shell script.

manager (NPM) software. NPM was used to install all the application dependencies or libraries. The application's information and dependencies were stored in the package.json file. Visual Studio Code (Microsoft, 2015) (vscode) was the code editor used to develop the Node.js program on an intel core i5 HP computer running on Windows 10. Putty was the secure shell (SSH) client software used to connect to the terminal console of the master controller (Raspberry Pi) via the Wi-Fi connection or LAN connection through an ethernet cable. WinSCP, an SSH File Transfer Protocol (SFTP), was used to share programming files between the Windows PC and the Raspberry Pi. All the codes were written in vscode and uploaded to the pi using the WinSCP software.

Node.js application program structure for RMA

The Node.js application building blocks are shown in Figure 19. Express.js was the web framework used; it was integrated with passport.js to support the implemented authentication; it was also integrated with body-parser.js to provide access to the middleware passed data from incoming requests. Finally, express.js handled all HTTP (Hypertext Transfer Protocol) requests and responses that the application covered.

The critical part of this application was the data handling; how data were captured from external devices like the energy meters, the IO controller board, and the web application front-end. Also, this includes how data were stored and retrieved

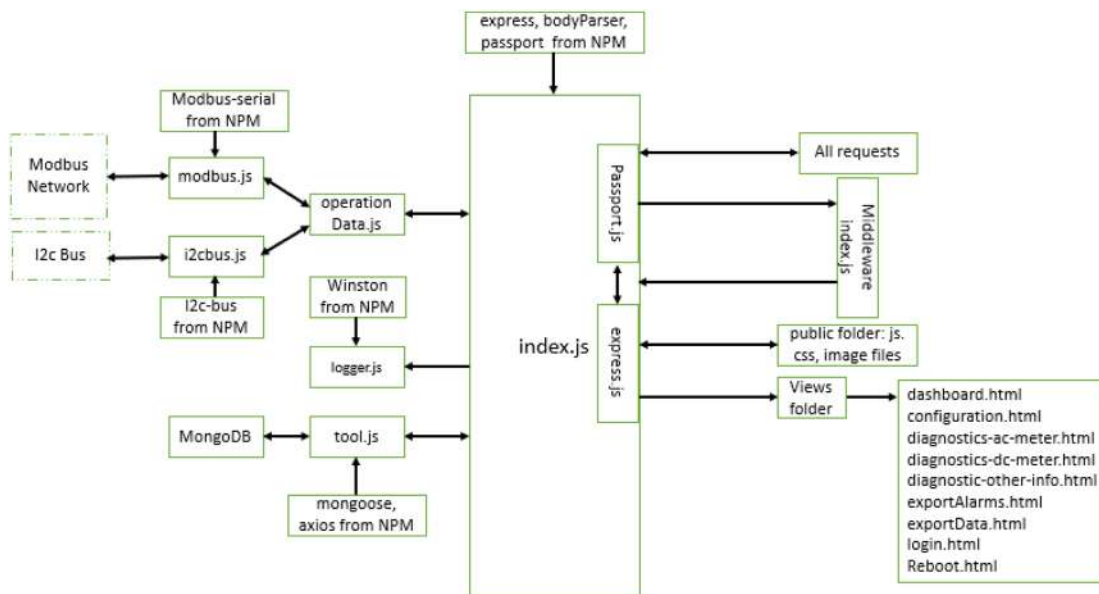


Figure 19. Node.js program blocks for a remote monitoring agent (RMA). NPM: node package manager, I2C: inter-integrated circuit.

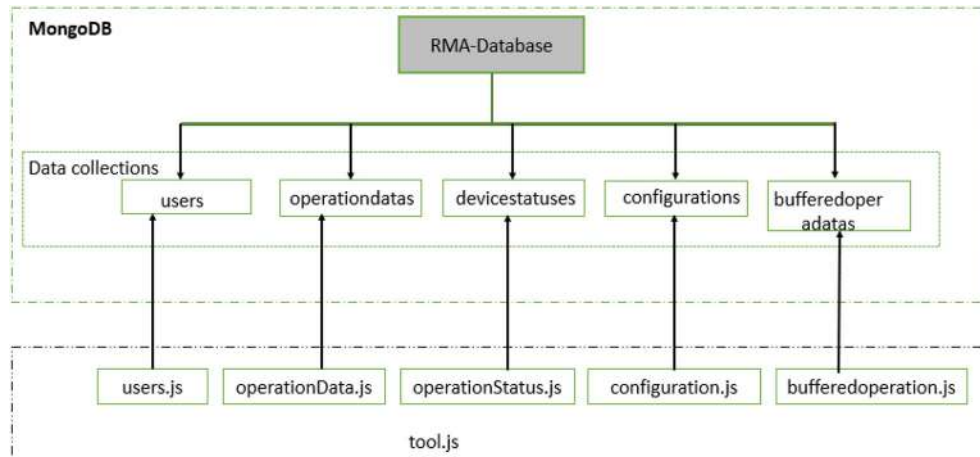


Figure 20. Node application database structure. RMA: remote monitoring agent.

from the database, and how real-time data were shared with the application front-end (user interface) and the application programming interface (API) calls.

The two programming files that contain the algorithm that handled this data management were `tool.js` and `operationData.js`. The `tool.js` imports `mongoose.js`, a library that provided API endpoints to interface with MongoDB database in the node environment.³² `Tool.js` has several functional modules that carry out database-related operations. These functions were exported to the `index.js` where they were to serve requests and make API calls.

The `operationData.js` imported `modbus-serial.js` and `i2c-bus.js` to connect to the RS485 Modbus network and I2C Bus respectively. There were functions on `operationData.js` that were called from `index.js` to retrieve data from the RS485 network and I2C Bus, these real-time data were shared with the user interface every two seconds. Also, these real-time data were uploaded to a cloud application and stored on the local database every two minutes. If the cloud data upload failed, the data would be stored on the buffered data collection in the database until the cloud data upload was successful, then the buffered data collection would be uploaded to the cloud, and the buffered collection data would be deleted from the database.

Figure 20 shows the MongoDB data structure developed for the project. The data collections were created automatically on the MongoDB server when the `tool.js` used the data model created from `user.js`, `operation.js`, `deviceStatus.js`, `configuration.js`, and `bufferedData.js` files in the model's folder. Successful connection to these models from the `tool.js` made it possible to insert data into the collection, read or find data from the collection, update data in the collection and delete data from the collection.

Node.js application program algorithm for RMA

The algorithms that powered the node.js application could be divided into three parts for better understanding as follows³⁶:

- 1) The first part was the collection of codes that runs only once after the program has started. **Figure 21** shows the flow chart of the set of algorithms that ran once when the application started. All the dependencies/libraries were first loaded into the application from the node module folder, also other created program files were loaded from their respective files. Libraries like `passport.js` and `body-parser.js` were initialized and integrated to `express.js` (node.js web application development framework). The next step was to declare the application's global variables, the most important ones are the `currentOperationData` and `deviceStatus` variables. Both are objects, the `currentOperationData` holds all the data pooled from the Modbus network and the I2C bus. The Modbus network comprised the AC meter and DC meter, data from these meters were pooled and temporarily kept in `currentOperationData`. Similarly, data from the I2C bus were pooled from the IO board and also kept in `currentOperationData`. The `deviceStatus` object holds the state of the RMA, such as, the number of items in the buffer, commissioning status, the site ID, server connectivity, internet connectivity, *etc.* Other data kept in the `currentOperationData` were some of the data in the `deviceStatus` object and the controller time captured. The `freshStart` function would be called to retrieve previous `deviceStatus` data saved in the database before the last

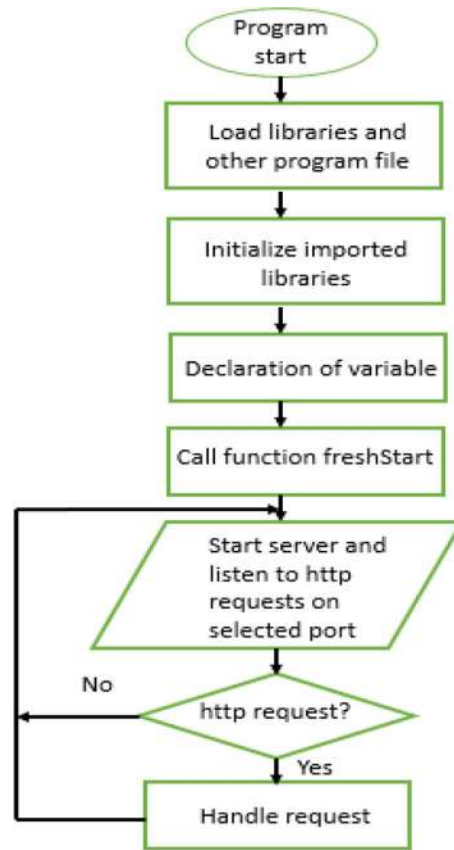


Figure 21. Node.js algorithm that runs once.

shutdown. The data retrieved were passed into the newly created deviceStatus object. The last part of the flow chart for this set of algorithms that run once was starting the server with express.js using the process environment port (if the application is deployed to the cloud) or using port 8080. For the RMA, port 8080 will always be used. The server continuously listens to HTTP requests on this port if the application was running. Now, the RMA's node.js application was ready to serve both GET and POST requests coming from the RMA front-end or coming from RMS (The remote monitoring server installed in the cloud).

- 2) The second set of algorithms were ran every four seconds. The flow chart for this set of algorithms is shown in [Figure 22](#). The primary function of these algorithms was to update the currentOperationData object. The loop starts by calling the handleInitialization function; this function was used to ensure that the version of site configuration details on the RMA was the same as the one in the RMS. If there was a difference, the function that updates the site configuration details was called. The internet connectivity was checked, and its state was updated in the currentOperationData object. Then the function getOperationData was called; this function was imported from the operationData.js file into the index.js file. The function pooled data from the AC meter and DC meter over the Modbus network, the data retrieved was used to update the currentOperationData object. Function handleAlarm was called to update the alarm array and the algorithm checks if the system time has been updated. If the time was not updated, the function getDate4rmMeter would be called, and the date value retrieved from the meter was used to set the Raspberry Pi time. Furthermore, function getI2CData was called to pool data from the IO board over the I2C bus to update the currentOperationData. Finally, the deviceStatus object was updated in the database and the program waited for four seconds for the loop to start again.
- 3) The third set of algorithms is shown in [Figure 23](#), its primary function was to manage how operation data held in the currentOperationData object were uploaded to the cloud and saved locally on the master controller. The first step was to check for internet connectivity, if the internet connection was not available, the currentOperationData data object was stored in the buffered data collection and operation data collection on the database, and the bufferedItem value was incremented. If there was internet connectivity, the program checked if the value of

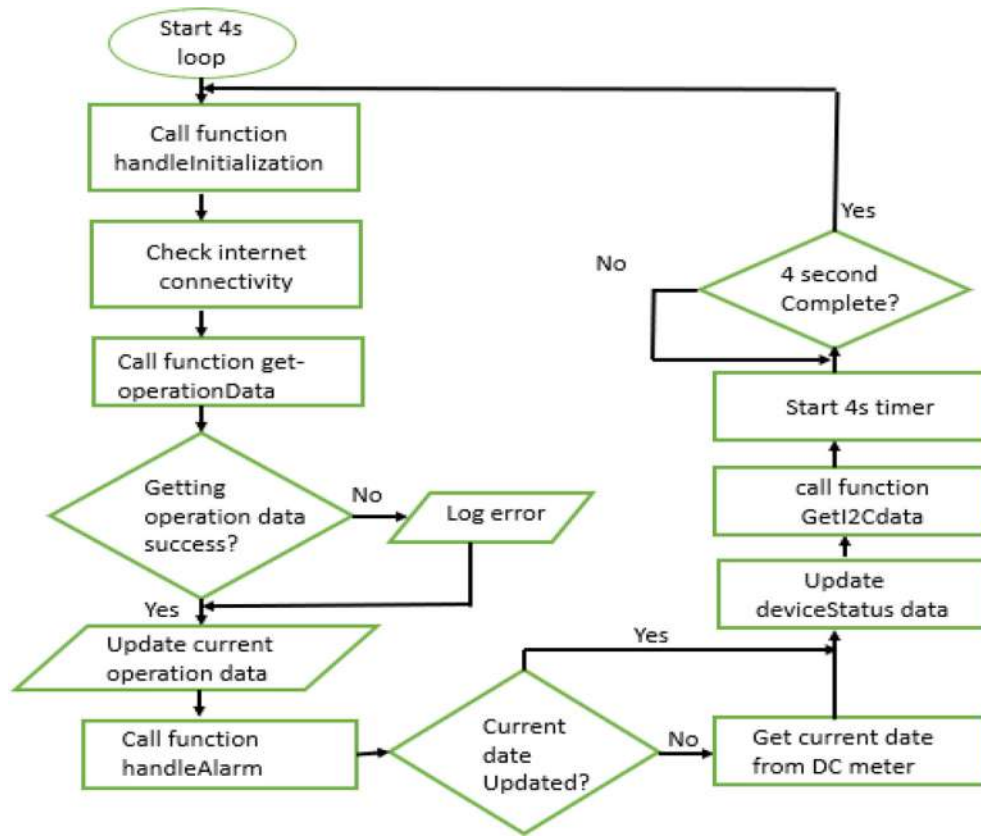


Figure 22. Flow chart for set of algorithms that run every four second.

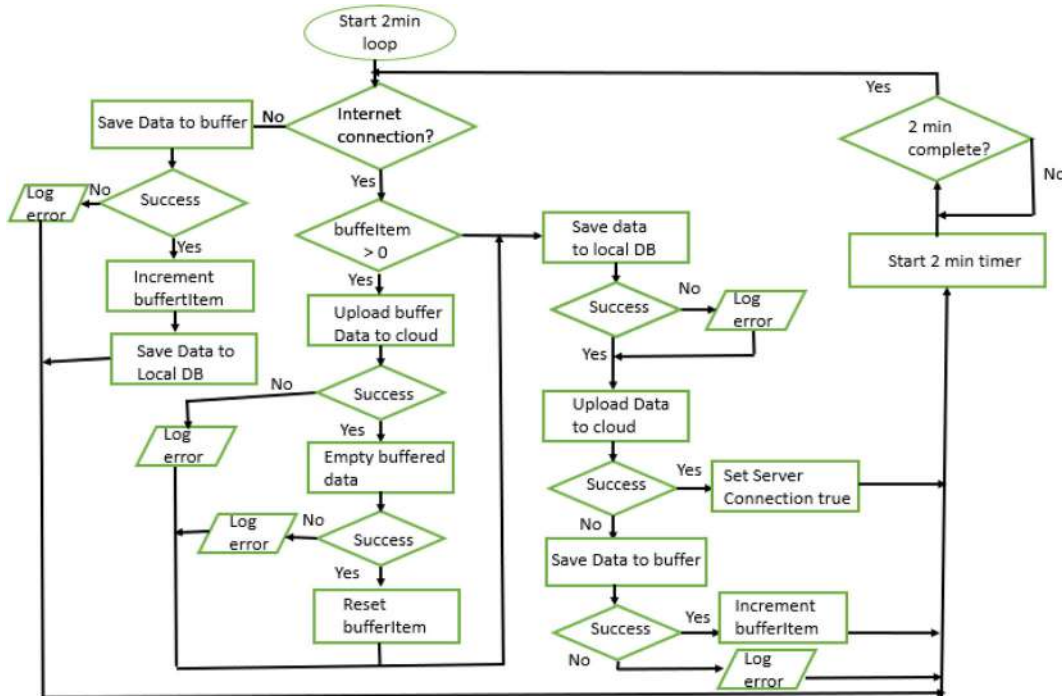


Figure 23. Flow chart for the set of algorithms that run every two minutes.

bufferedItem was greater than 0, if this was true, it uploaded the buffered data to the cloud and empty the buffered data collection if the data upload was successful. If the buffered data collection was successfully emptied, the bufferedItem would be reset to 0 value. Then the program proceeds to save the currentOperationData to the operation data collection on the local database, the program also returned to this same point directly if the value of bufferedItem was not greater than 0. Finally, the currentOperationData object was uploaded to the cloud, and if the data upload failed, the data would be stored in the buffered data collection as shown in Figure 23.

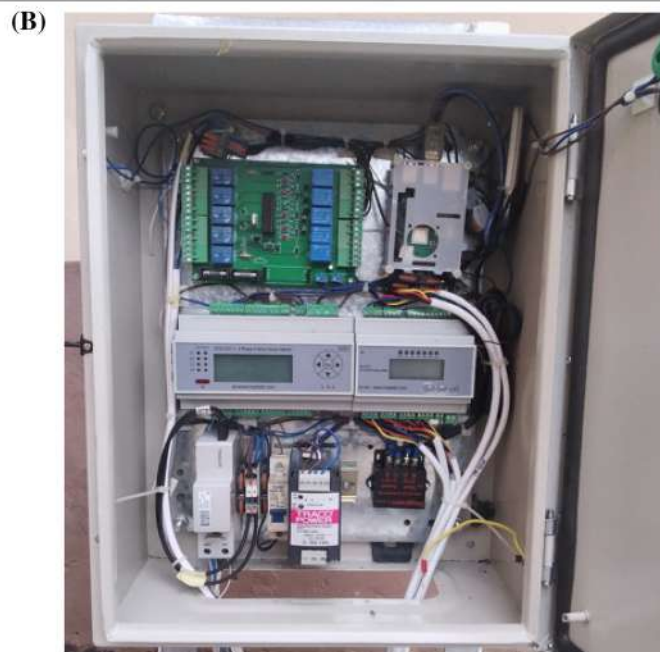
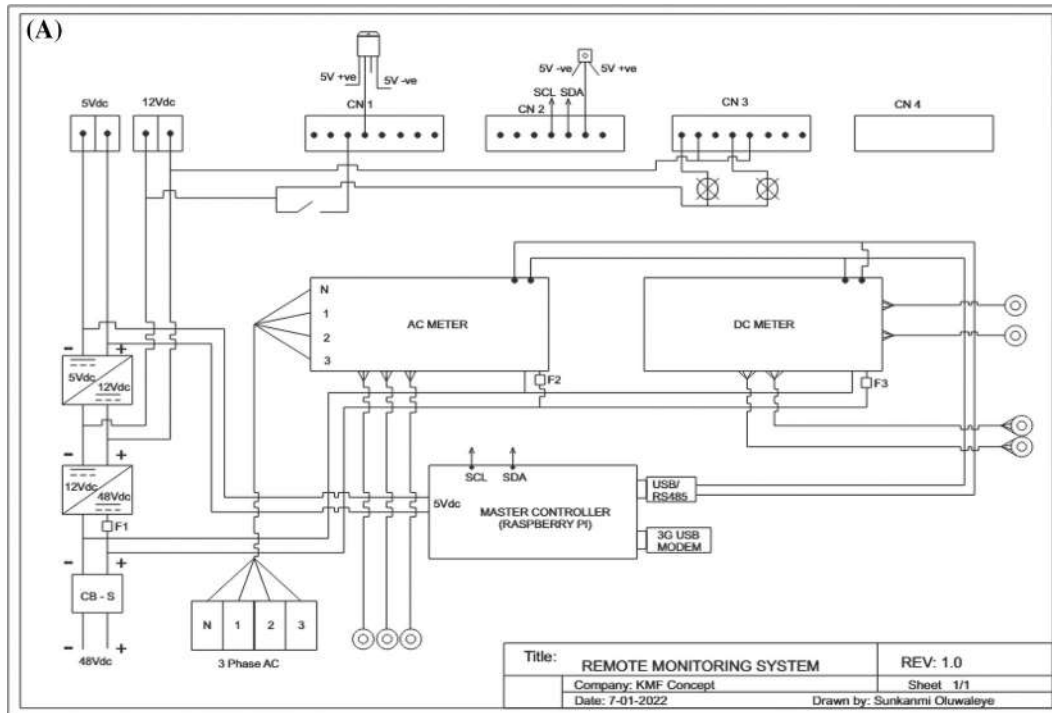


Figure 24. (A) Complete hardware wiring of remote monitoring agent (RMA). (B) Remote monitoring agent (RMA) final panel assembly. CN: connector, SCL: serial clock line, SDA: serial clock line, AC: alternating current, DC: direct current.

RMA hardware wiring and assembly

The electrical components for RMA were wired as shown in [Figure 24](#). The 48VDC from the output of the rectifier installed on the site was connected to the incoming terminal of the surge protection enabled circuit breaker (CB-S). The outgoing of the CB-S was connected to power both AC and DC meters and it was also wired as input to 48/12V DC-DC converter. The output of the 48/12V DC-DC converter was connected as an input to a 12/5V DC-DC converter and wired to power the red and green pilot lamps. The output of the 12/5VDC-DC converter powered both the IO board and master controller (Raspberry Pi).

The 12VDC relay soldered to the IO board needed 12V power for the relay coil to be energized, hence, 12V output from the 48/12V DC-DC converter was also wired to the 12V terminal port of the IO board. The negative of the 12VDC was wired through the door switch to CN1 pin 3 as an input signal that senses the state of the door.

Four current sensors were connected to the DC meter as shown in [Figure 13](#), while three current sensors were connected to the AC meter as shown in [Figure 11](#). The Modbus network connection was made with a 1mm two-core cable, one of the cores looped all the RS485-A terminals of all the Modbus devices together, and the second core looped all the RS485-B terminal of the Modbus devices together as well as shown in [Figure 13](#). All the components were assembled on a 400mm by 300mm by 200mm outdoor panel as shown in [Figure 24B](#) shows the final assembly of RMA.

Results

RMA's hardware assembly and installation

All the hardware was connected and assembled in the panel as shown in [Figure 24B](#). The final assembly picture is shown in [Figure 25B](#), extended from the RMA panel were three current sensors for the genset three-phase power supply. Also, four DC current sensors extended from RMA were for current measurements, met to capture the current consumed by the mobile networks' equipment and the installed battery bank. The cables available for external connections were two core cables that supplied 48Vdc power to the RMA. And a four-core cable that should be connected to the three-phase plus neutral of the AC source for the AC meter's voltage sampling. The humidity and temperature sensor hung below the RMA panel and the battery temperature sensor was dropped inside the battery cabinet. A standing frame was built for the RMA panel as shown in [Figure 26A](#) to make the site installation easy.

The installation permission was requested for the unit to be installed on a live telecommunication site. This was granted after the installation process was reviewed and it was confirmed to be traffic not affecting. The unit was moved to the site and all the necessary cables were connected as shown in [Figure 25](#). The current sensors shown in [Figure 25A](#) could be clamped to the connected equipment cables without disconnecting the cables making the installation seamless without disconnecting tenants' equipment from power during the installation.

RMA's commissioning and local webpage

The RMA was powered with 48Vdc after the site installation, the main controller booted up for about 45 seconds and the RMA Wi-Fi interface was available for connection as shown in [Figure 27](#). The Wi-Fi connection was enabled with WPA2

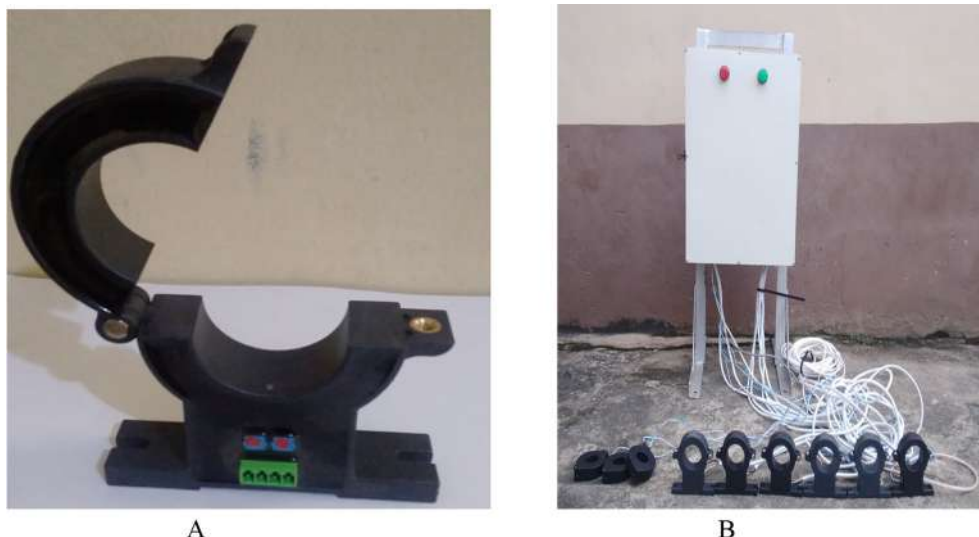


Figure 25. A: Picture of current sensor; B: Final outlook of remote monitoring agent (RMA).



Figure 26. A: Site installation picture view 1; B: Site installation picture view 2.

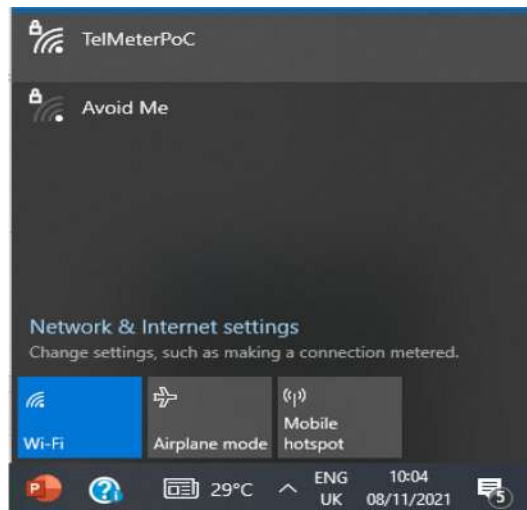


Figure 27. Remote monitoring agent (RMA) WiFi service set identifier (SSID) connection.

(Wi-Fi Protected Access 2) authentication to prevent the connection from an unauthorized user. After a successful WiFi connection was made to the RMA, the user could access the web interface by typing the IP address 192.168.50.10 to the address bar of any web browser. To further prevent unauthorized users and establish access control, the user had to login to the RMA web page as shown in Figure 28. The user could either log in as Admin or Operator username with their respective passwords. The Admin has access to the configuration page, export data page, and robot the main controller button while the Operator user does not have access to these operations but can only view the site details.

After successful login, the user would be redirected to the dashboard shown in Figure 29. The dashboard contains the overview of site operation data and presented the tabs to access other functionalities.

The top of the dashboard gave information about the site status, if the site was already commissioned, the site name and site ID text value will be shown as presented in the dashboard shown in Figure 29. The site name was Lagos Central, and the site ID was Ts_LG_1210A, these values will be “Not Available” if the site has not been commissioned. The value for



Figure 28. Remote monitoring agent (RMA) login page.



Figure 29. Remote monitoring agent (RMA) local dashboard.

the label server connection could be “Connected” or “Not Connected” if the RMA was able to upload site operation data to the cloud server, the value of server connection would be “Connected” and the value would be “Not Connected” otherwise. The label System Voltage gave the value of the DC voltage captured on the site. This voltage was common to the voltage of the battery bank, the tenant’s equipment voltage, rectifier output voltage, and the voltage connected to power the RMA. The local time captured was the time of the main controller, this time was always corrected at boot up using the time of the DC meter pooled over the RS485 network. The DC meter has a complementary metal-oxide semiconductor (CMOS) battery that accurately kept the time at no power. The internet status label displayed “Online” when the RMA was connected to the internet and “Offline” would be displayed if there was no internet connection. There was an algorithm on the main controller that checked the internet connectivity at every four seconds interval.

The genset line voltages, currents, energy consumed, power consumed, and frequency were shown under the genset section of the dashboard. If the site was commissioned, the section for each tenant will display the tenant’s name and the tenant equipment power data such as current, power, and energy consumed by the equipment. The ambient temperature and humidity were shown, and the battery temperature value was also displayed as well. There are navigation tabs listed on the left side of the dashboard, the first one to access during the first installation when the site has not been commissioned was the Configuration tab. This tab is shown in Figure 30, the user can set the site parameters from this page. The fields included the site name, ID, region, longitude, latitude, talents names, and the capacity of the Genset installed. The already configured data could be edited and deleted using the buttons below the configuration form. When this form was submitted, the site configuration collection in the database was updated and the site configuration version value was incremented, the RMA tracked this value to know when to update the site configuration details of the site ID on RMS so that the same site parameters would be on the RMA and the RMS.



Figure 30. Remote monitoring agent (RMA) configuration page.

ID	Parameter Name	Value
01.	ACMeterWorkingStatus	5633
02.	ACcurrenTimeYMM	1804
03.	ACcurrenTimeDDhh	6412
04.	ACcurrenTimeimmss	5187
05.	ACfrequency	51.87
06.	ACphaseAVoltage	233.4
07.	ACphaseBVoltage	233.2
08.	ACphaseCVoltage	232.9
09.	ACfirstAphaseCurrent	11.34
10.	ACfirstBphaseCurrent	10.5
11.	ACfirstCphaseCurrent	11.15
12.	ACfirstTotalApparentPower	16
13.	ACfirstApparentPowerPhaseA	263
14.	ACfirstApparentPowerPhaseB	242
15.	ACfirstApparentPowerPhaseC	257
16.	ACfirstActivePowerPhaseA	1.38
17.	ACfirstActivePowerPhaseB	1.22
18.	ACfirstActivePowerPhaseC	2.62
19.	ACfirstTotalActivePower	5.22
20.	ACfirstTotalReactivePower	15

Figure 31. Alternating current (AC) meter real-time data.

The diagnostics tab has three sub-tabs for the AC meter, DC meter, and other information. These three tabs categorized the real-time data pooled from the AC meter, DC meter, and the IO board respectively. The information on this tab was met for fault troubleshooting. The AC meter real-time data pooled over RS485 Modbus is shown in Figure 31. Similarly, the DC meter real-time data pooled over RS485 Modbus is shown in Figure 32. The IO board real-time data were pooled over the I2C bus and other operational data like internet connectivity, the number of items that were buffered, site and server configuration version values, and many more were shown when the Other Info tab was clicked as shown in Figure 33. These data were continuously pooled at four seconds intervals from the RMA backend application and updated on this page. The same data was saved to the RMA database every two minutes and uploaded to the RMS every two minutes as well. Only the Admin user has access to this page.

For quick monitoring of historical data, the Trendline tab shows the AC power chart as shown in Figure 34. Operation data from the last twenty-four hours were queried from the database and only the latest 100 points (number chosen arbitrarily) of these data were sent to this page to be plotted on these charts. The first chart displayed the AC power source line voltage, current, and power. The second chart displayed the total dc power consumed by all the tenant’s equipment installed on the site and the DC voltage. The last chart displayed the battery temperature, ambient temperature, and ambient humidity.

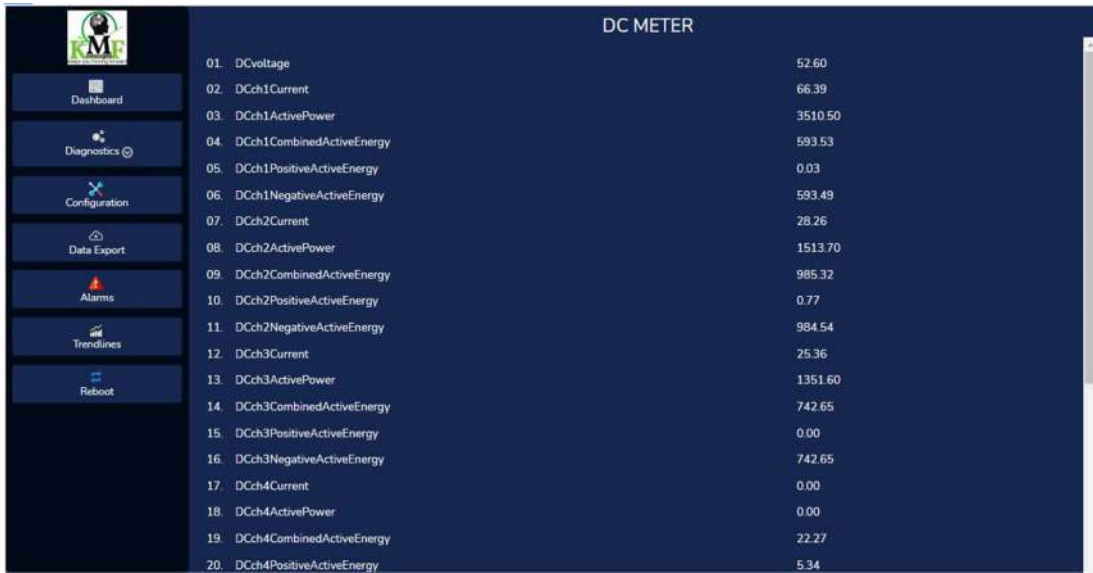


Figure 32. Direct current DC meter real-time data.

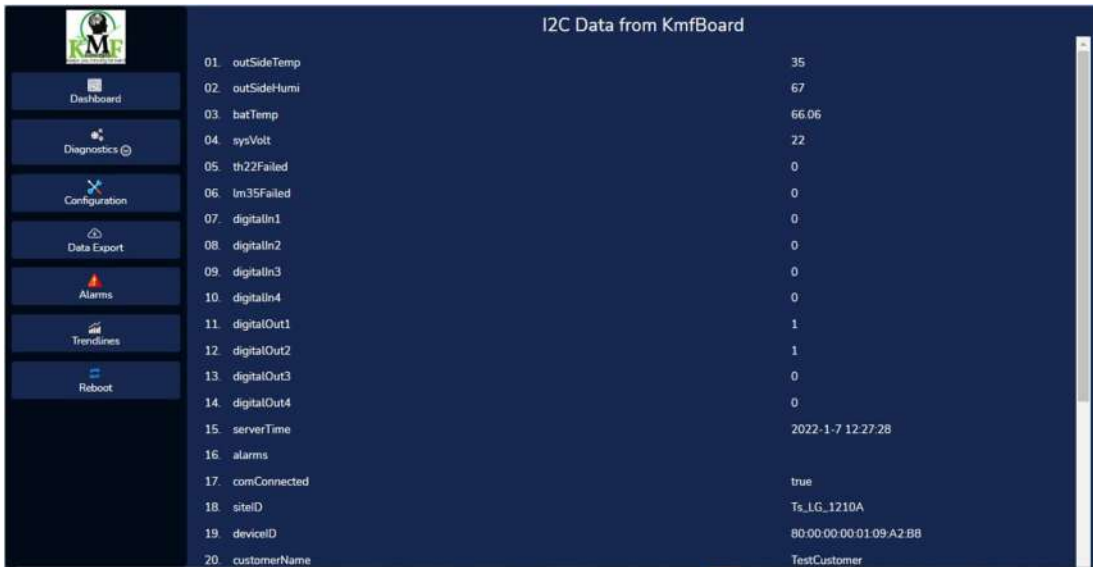


Figure 33. Input/output (IO) board real-time data. I2C: inter-integrated circuit.

Locally from the site, when connected to the RMA, the user can download the site operational data as shown in Figure 35. The user could select the start date and the end date for the required data, clicking on the “Get Data” button queried the operational data from the database using the date selected as the query selector. The data within the date range were retrieved and sent to this page and displayed as the table shown in Figure 35. This data could be downloaded in excel format for further analysis.³⁵

Validation of current and voltage measurement

The accuracy of the current and voltage measurement captured by the RMA was benchmarked by taking a manual reading with UNI-T UT203+ digital multimeter. The corresponding currents and voltages manually measured for the captured values on the RMA’s webpage in Figure 29 are shown in Tables 6 and 7. The percentage error for the current and voltage measurements was calculated with equation 3 and the values are shown in column %Error in Tables 6 and 7 respectively. Equations 4 and 5 were used to calculate the total error for the current and voltage measurements respectively. TE_c was 1.97 and TE_v was 0.54.

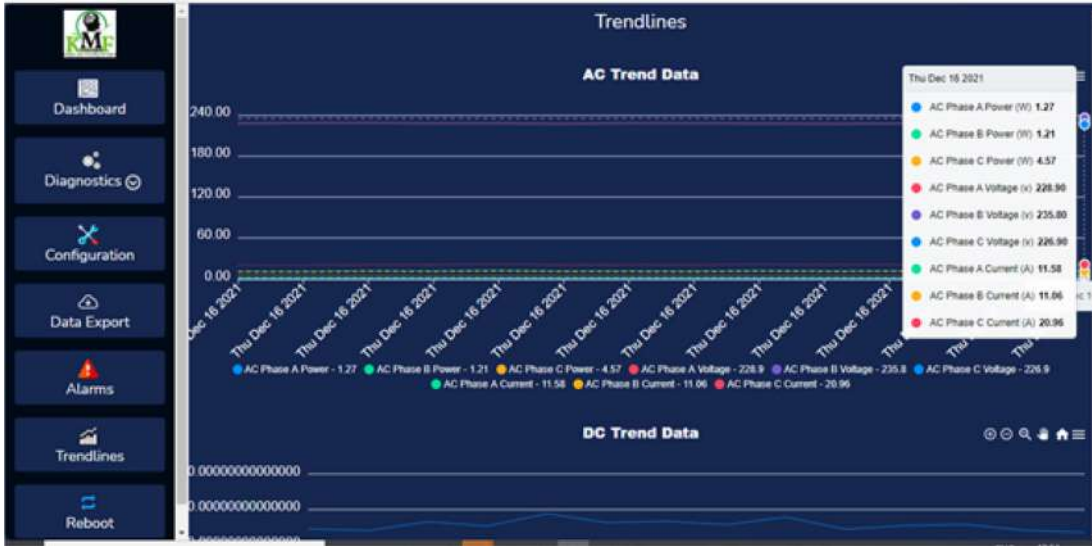


Figure 34. RMA trendline chart. AC: alternating current, DC: direct current.

Site ID	Server Time	Timestamp	Com connected	AC First A Phase power factor	AC First B Phase power factor	AC First C Phase power factor
		2021-11-03T19:53:01.280Z	100	100	100	0
		2021-11-03T19:55:01.280Z	100	100	100	0
		2021-11-03T19:57:01.280Z	100	100	100	0
		2021-11-03T19:59:01.281Z	100	100	100	0
		2021-11-04T09:33:42.476Z	100	100	100	0
		2021-11-04T09:35:42.365Z	100	100	100	0
		2021-11-04T09:37:42.365Z	100	100	100	0
		2021-11-04T09:39:42.373Z	100	100	100	0
		2021-11-04T09:41:42.363Z	100	100	100	0
		2021-11-04T09:43:42.351Z	100	100	100	0
		2021-11-04T09:55:17.550Z	100	100	100	0
		2021-11-04T09:57:17.548Z	100	100	100	0

Figure 35. Remote monitoring agent (RMA) data export page. AC: alternating current.

$$\%Error = \frac{|M - R| \times 100}{M} \quad (3)$$

Where M is the manually measured value using the digital multimeter and R is the captured value from the RMA webpage.

$$TE_c = \frac{\sum_{i=1}^6 \left(\frac{|MC_i - RC_i| \times 100}{MC_i} \right)}{6} \quad (4)$$

TE_c is the total error for the measured current values, MC_i is the manually measured current at a point, and RC_i is the current value captured on the RMA webpage for the point.

$$TE_v = \frac{\sum_{i=1}^6 \left(\frac{|MV_i - RV_i| \times 100}{MV_i} \right)}{6} \quad (5)$$

Table 6. Comparison of current measurements. RMA: Remote Monitoring Agent, DC: Direct Current, AC: Alternating Current, L1: Line 1, L2: Line 2, L3: Line 3.

Point of measurement	Manual measurement (A)	RMA reading (A)	%Error
MTN (DC)	71.5	70.5	1.40
Airtel (DC)	30.1	31.68	5.25
Spectranet (DC)	25	24.55	1.80
Genset L1 (AC)	11.51	11.42	0.78
Genset L2 (AC)	11.01	10.91	0.91
Genset L3 (AC)	20.6	20.25	1.70

Table 7. Comparison of voltage measurement. RMA: Remote Monitoring Agent, DC: Direct Current, AC: Alternating Current, L1: Line 1, L2: Line 2, L3: Line 3.

Point of Measurement	Manual Measurement (A)	RMA Reading (A)	%Error
MTN (DC)	53.1	52.6	0.94
Airtel (DC)	53.1	52.6	0.94
Spectranet (DC)	53.1	52.6	0.94
Genset L1 (AC)	228.9	229.1	0.09
Genset L2 (AC)	234.9	235.5	0.26
Genset L3 (AC)	227	227.1	0.04

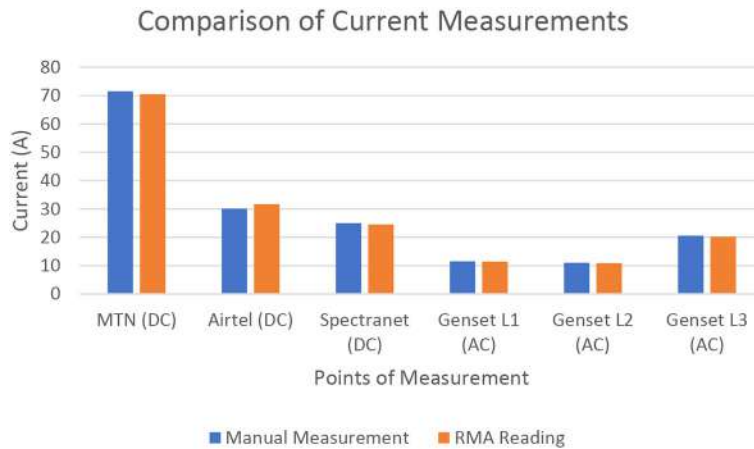


Figure 36. Chart of comparison of current measurements. AC: alternating current, DC: direct current, RMA: remote monitoring agent, L1: line 1, L2: line 2, L3: line 3.

Where TE_v is the total error for the measured voltage values, MV_i is the manually measured voltage at a point, and RV_i is the voltage valued captured on the RMA webpage for that point.

Figures 36 and 37 show the charts that compare the values of currents and voltages measured manually and the corresponding values captured on the RMA’s webpage.

Discussion

The RMA master controller developed with Raspberry Pi 3 controller has proven that this microcomputer is a powerful credit-card size alternative for a personal computer with the ability to integrate effectively with peripheral hardware as shown by.³³ The possibility of setting up the RS485 Modbus communication network on Raspberry Pi provided the flexibility required to simultaneously integrate the microcomputer with up to thirty-two industrial devices on the same

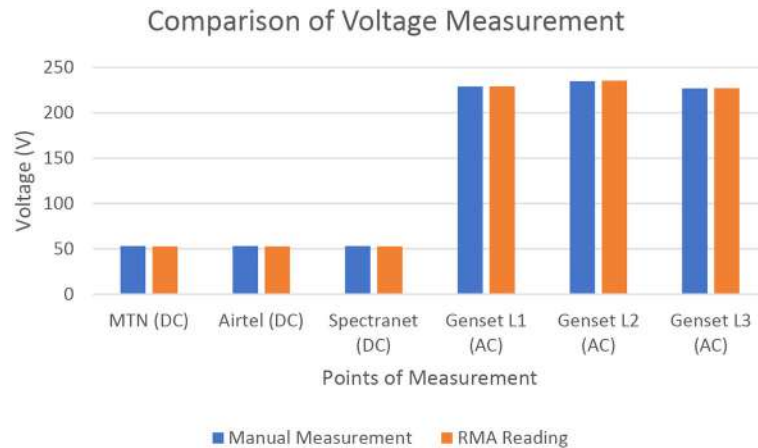


Figure 37. Chart of comparison of voltage measurement. AC: alternating current, DC: direct current, RMA: remote monitoring agent, L1: line 1, L2: line 2, L3s: line 3.

network using just two wires as mentioned under the literature review section. This has made it possible to integrate the Raspberry Pi microcomputer (the master controller) with the two energy meters, and with the developed IO board over the I2C communication protocol. This work extended was done in the cited literature papers by using raspberry pi to implement both synchronous and asynchronous serial communication in a single application. With the energy meters and their connected current sensors, the master controller could pool real-time data of all power/energy data captured from the connected mobile operators' equipment and the genset. This made the RMA suitable for energy service companies (ESCOs) in telecommunication industry to monitor individual power/energy profile of mobile operators connected to an example of power solution developed by.¹²

Furthermore, since the main controller was a computer with an onboard WiFi chip, it was possible to build and host a web application on it that could locally serve the connected users through the web application frontend and also send data to a cloud server.³⁴ Similar to what was done by¹⁶ for a solar plant. Data captured by the main controller were saved in its database which could be retrieved to serve locally connected users or the cloud server. The internet connection to the main controller was provided using a 3G USB model, the RMA had been online since it was installed and the custom-built shell script on the master controller ensured internet connectivity by restarting the modem whenever there is no internet connection. Having the master controller connected to the internet was critical for the RMA to serve its purpose. The web application hosted on the master controller started up whenever the main controller (Raspberry Pi) booted up successfully. Both the WiFi hotspot and the internet modem started up as well during the controller boot-up session. The automatic startups of these applications were handled by the shell scripts running on the Raspberry Pi.

Conclusions

The implementation of this work provided a platform for Energy Service Companies (ESCOs) in the telecommunication industry to accurately meter the power consumed by the accommodated mobile operators' equipment on their site. The web application on the RMA provided access to site information. Harvested data can be locally stored and uploaded to a cloud server for further information processing. This study could be extended to capture other important data on the telecommunication site, such as rectifier information, fuel level measurement and site surveillance. Also, it can be extended to take commands from the remote server to start/stop the generator set remotely or control the whole power system.

The solution developed in this work looked a bit bulky, future research could perform similar tasks while being much more smaller and cost effective if the input/output (IO) board can be enabled to measure both AC and DC currents and voltages.

Data availability

Underlying data

Zenodo: Design and development remote monitoring agent for energy service companies in the telecommunication industry. <https://doi.org/10.5281/zenodo.6829361>.³⁵

This project contains the following underlying data:

- rma data.csv. (Data set downloaded from the remote monitoring agent (RMA) over its local webpage using WiFi connection)

Data are available under the terms of the [Creative Commons Attribution 4.0 International license \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/).

Software availability

Source code available from: <https://github.com/skcareer/Software-for-RMA>

Archived source code at time of publication: <https://doi.org/10.5281/zenodo.6828291>³⁶

License: [GNU Affero General Public License v3.0](https://creativecommons.org/licenses/by/4.0/).

References

- Nomamidobo Amadasun K, *et al.*: **Transitioning to Society 5.0 in Africa: Tools to Support ICT Infrastructure Sharing**. *Data*. Jun. 2021; **6**(7): 69.
[Publisher Full Text](#)
- Adebayo EA, Adeeyo AO, Ogundiran MA, *et al.*: **Bio-physical effects of radiofrequency electromagnetic radiation (RF-EMR) on blood parameters, spermatozoa, liver, kidney and heart of albino rats**. *J. King Saud Univ. - Sci.* 2019; **31**(4): 813–821.
[Publisher Full Text](#)
- Francis P, Idachaba E: **Telecommunication Cost Reduction in Nigeria through Infrastructure Sharing between Operators**. *Pac. J. Sci. Technol.* 2010; **11**.
- Mamushiane L, Lysko AA, Dlamini S: **SDN-enabled Infrastructure Sharing in Emerging Markets: CapEx/OpEx Savings Overview and Quantification**. 2018.
- A. A. SAmeh KTOOJ: **Significant Factors Causing Cost Overruns in Telecommunication Projects in Nigeria**. *J. Constr. Dev. Ctries.* 2010; **15**(2).
- Ferraro M, Brunaccini G, Sergi F, *et al.*: **From Uninterruptible Power Supply to resilient smart micro grid: The case of a battery storage at telecommunication station**. *J. Energy Storage*. 2020; **28**(April 2019): 101207.
[Publisher Full Text](#)
- Kondaveeti HK, Kumaravelu NK, Vanambathina SD, *et al.*: **A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations**. *Comput. Sci. Rev.* 2021; **40**: 100364.
[Publisher Full Text](#)
- Li H, Li K, Zafetti N, *et al.*: **Improvement of energy supply configuration for telecommunication system in remote area s based on improved chaotic world cup optimization algorithm**. *Energy*. 2020; **192**: 116614.
[Publisher Full Text](#)
- Kaldellis JK, Ninou I: **Energy balance analysis of combined photovoltaic-diesel powered telecommunication stations**. *Int. J. Electr. Power Energy Syst.* 2011; **33**(10): 1739–1749.
[Publisher Full Text](#)
- Okundamiya MS, Emagbetere JO, Ogujor EA: **Design and control strategy for a hybrid green energy system for mobile telecommunication sites**. *J. Power Sources*. 2014; **257**: 335–343.
[Publisher Full Text](#)
- Wibowo WW, Astuti YDRW, Hudaya C: **Solar-Powered Base Transceiver Station**. *Proc. - 2018 2nd Int. Conf. Green Energy Appl. ICGEA 2018*. 2018; pp. 108–112.
[Publisher Full Text](#)
- Salihodhin MABM, Muhtazaruddin MNB, Bani NA, *et al.*: **UTM Razak School of Engineering and Advanced Technology, Universiti Teknologi Malaysia, Kuala Lumpur, "Hybrid Power Supply System for Telecommunication Base Station**.
[Publisher Full Text](#)
- Zhai Z, Martínez JF, Beltran V, *et al.*: **Decision support systems for agriculture 4.0: Survey and challenges**. *Comput. Electron. Agric.* 2020; **170**(January): 105256.
[Publisher Full Text](#)
- Ompal VMM, Kumar A: **FPGA integrated IEEE 802.15.4 ZigBee wireless sensor nodes performance for industrial plant monitoring and automation**. *Nucl. Eng. Technol.* 2022; **54**(7): 2444–2452.
[Publisher Full Text](#)
- D'Aniello F, Sorrentino M, Rizzo G, *et al.*: **Introducing innovative energy performance metrics for high-level monitoring and diagnosis of telecommunication sites**. *Appl. Therm. Eng.* 2018; **137**(February): 277–287.
[Publisher Full Text](#)
- Siva M, Krishna R, Dinesh K, *et al.*: **Low Cost Remote Monitoring of Solar Plant through RS485 Communication**. *Int. J. Innov. Technol. Explor. Eng.* 2019; **8**(9): 3034–3037.
[Publisher Full Text](#)
- Jian YW: **Application of RS-485 Bus in Warehouse Communication Technology**. 2021.
[Publisher Full Text](#)
- Hung PD, Van Chin V, Chinh NT, *et al.*: **A flexible platform for industrial applications based on RS485 networks**. *J. Commun.* 2020; **15**(3): 245–255.
[Publisher Full Text](#)
- Nguyen V, Dugenske A: **An I2C based architecture for monitoring legacy manufacturing equipment**. *Manuf. Lett.* 2018; **15**: 67–70.
[Publisher Full Text](#)
- Łukasz P, Marcin O, Dariusz K, *et al.*: **I2C interface design for hardware master devices**. *IFAC Proc. Vol.* 2012; **45**(PART 1): 163–168.
[Publisher Full Text](#)
- McBride WJ, Courter JR: **Using Raspberry Pi microcomputers to remotely monitor birds and collect environmental data**. *Ecol. Inform.* 2019; **54**(October): 101016.
[Publisher Full Text](#)
- Nadafa RA, Hatturea SM, Bonala VM, *et al.*: **Home Security against Human Intrusion using Raspberry Pi**. *Procedia Comput. Sci.* 2020; **167**(Iccids 2019): 1811–1820.
[Publisher Full Text](#)
- EasyEDA: **Online electronics design software tools**.
[Reference Source](#)
- Rholam O, Tabaa M, Monteiro F, *et al.*: **Smart device for multi-band industrial IoT communications**. *Procedia Comput. Sci.* 2019; **155**: 660–665.
[Publisher Full Text](#)
- Hsiao CH, Lee WP: **OPIIoT: Design and Implementation of an Open Communication Protocol Platform for Industrial Internet of Things**. *Internet of Things (Netherlands)*. 2021; **16**(August): 100441.
[Publisher Full Text](#)
- Huang Y, Li C: **Real-time monitoring system for paddy environmental information based on DC powerline communication technology**. *Comput. Electron. Agric.* 2017; **134**: 51–62.
[Publisher Full Text](#)
- Pereira RIS, Dupont JM, Carvalho PCM, *et al.*: **IoT embedded linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant**. *Meas. J. Int.*

- Meas. Confed.* 2018; **114**(August 2017): 286–297.
[Publisher Full Text](#)
28. Mudaliar MD, Sivakumar N: **IoT based real time energy monitoring system using Raspberry Pi.** *Internet of Things.* 2020; **12**: 100292.
[Publisher Full Text](#)
 29. Besada-Portas E, Bermúdez-Ortega J, de la Torre L, *et al.*: **Lightweight Node.js & EJS-based Web Server for Remote Control Laboratories.** *IFAC-PapersOnLine.* 2016; **49**(6): 127–132.
[Publisher Full Text](#)
 30. Wisnusenna S, Yonatan MS, Wibisurya A, *et al.*: **Model of Web Based Application to Control Bridge Traveler Using Raspberry Pi.** *Procedia Comput. Sci.* 2018; **135**: 518–525.
[Publisher Full Text](#)
 31. Zhou K, Yuan Y: **A smart ammunition library management system based on raspberry pie.** *Procedia Comput. Sci.* 2020; **166**: 165–169.
[Publisher Full Text](#)
 32. Kumar JV, Moeed SA, Kumar CM, *et al.*: **Integration patterns of MongoDB GridFS for advanced data science and big data processing.** *Mater. Today Proc.* 2021. no. xxxx.
[Publisher Full Text](#)
 33. Oguntosin V, Oluwaleye S, Idachaba F: **Conceptual design of smart multi-farm produce dehydrator using a low-cost programmable logic controller and raspberry pi.** *F1000Res.* 2021; **10**: 810–825.
[PubMed Abstract](#) | [Publisher Full Text](#)
 34. Vujović V, Maksimović M: **Raspberry Pi as a Sensor Web node for home automation.** *Comput. Electr. Eng.* 2015; **44**: 153–171.
[Publisher Full Text](#)
 35. Oluwaleye S, Idachaba F: **Design and development remote monitoring agent for energy service companies in the telecommunication industry [Data set].** Zenodo. 2022.
[Publisher Full Text](#)
 36. Oluwaleye S, Idachaba F: **Design and development remote monitoring agent for energy service companies in the telecommunication industry (To develop a software for a remote monitoring unit for telecommunication site.). [Source code]** Zenodo. 2022.
[Publisher Full Text](#)

Open Peer Review

Current Peer Review Status: ?

Version 1

Reviewer Report 29 September 2022

<https://doi.org/10.5256/f1000research.135761.r151658>

© 2022 Okedu K. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

? **Kenneth Eloghene Okedu** 

Department of Electrical and Communication Engineering, National University of Science and Technology, Muscat, Oman

The paper presents the implementation of energy service companies in the telecommunication industry to accurately carry out the monitoring and metering of the consumed power by the equipment of mobile operators in a site. Lagos state in western Nigeria, was used as the site in this study. The concept of the paper is interesting, however, there are some shortcomings. The following comments would help the authors in improving the readership of this work.

1. The paper should be checked for grammatical and typo errors; e.g., "In other..." instead of in order. Also, some statements are not clear in the introduction, as it is difficult to comprehend them.
2. In order to boost the theory of this work, more mathematical formulations of the model and its design are required.
3. Figure 8 is not readable.
4. A detailed model system of the topology of the study should be shown before the description of each component. Readers would like to have a holistic view or a bigger picture of the energy service companies in the telecommunication industry and the link with remote monitoring agent for proper monitoring, and the variables being monitored, with detailed parameters of the components. This is required in order to bring out the contribution of the paper.
5. Figure 17 should be properly captioned.
6. The work is basically describing only the hardware techniques employed in this research. A simulation result section is required and it should be robust. A more rigorous simulation study showing graphically the various variables of the system over time is imperative. The responses or dynamic behaviors of the voltages, currents, temperature, humidity, energy

consumed, power consumed and frequency of the proposed model system earlier requested should be given in this paper.

7. The data in Figure 35 are constant and seem not to change over time.

8. The paper is too long. It should be concise.

Is the work clearly and accurately presented and does it cite the current literature?

Partly

Is the study design appropriate and is the work technically sound?

Partly

Are sufficient details of methods and analysis provided to allow replication by others?

Partly

If applicable, is the statistical analysis and its interpretation appropriate?

No

Are all the source data underlying the results available to ensure full reproducibility?

No

Are the conclusions drawn adequately supported by the results?

Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Energy Systems.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research