Journal of Engineering Sciences
Vol 12, Issue 7, July/202J
ISSN NO:0377-9254

Internet of Things (IoT) based Energy Tracking and Bill Estimation System

Niharika Tapasi[1], Salman Shaik[2], Sindhupriya Chakali[3], O.Sudhakarbabu[4], Dr.G.Lakshminarayana[5]
Assistant Professor[4], Professor[5],UG Student[123]
DEPT OF ECE
SVR ENGINEERING COLLEGE, NANDYAL

ABSTRACT

Electricity is the most requisite energy in modern times. IoT based energy tracking and bill estimation system discussed in this project has an objective to build awareness among household and industrial consumers about their usage of this energy. It does so by displaying real-time estimated electricity consumption by each load connected to it and real -time estimated bill of total consumption on a monitor unit. To save energy when unused, users can operate the control unit to transmit switching instructions for loads. The proposed system also uses Microcontroller, 4-channel relay module, and Blynk android application. microcontroller fetches average consumption detail of loads from and logs estimated bill to the cloud-hosted database, monitors the duration for which each relay in a 4-channel relay module was switched-on, performs calculations, and transmits real-time results to an IoT cloud interface. 4-channel relay module executes switching instructions on loads sent over the internet via the control unit, and the Blynk android application is the IoT cloud interface with both control unit and monitor unit built-in. This project provides highlights on cloud-hosted database details, hardware design, IoT cloud interface application design, and working principle with mathematical modelling of the proposed system and tested results of this system are discussed, with the cloud-hosted database and IoT cloud interface.

I. INTRODUCTION

The concept of the Internet of Things (IoT) provides interconnection of the system's ability to transmit data over the internet. The energy tracking and bill estimation system discussed in this project utilizes this concept with a microcontroller, binary actuators, cloud-hosted database, and IoT cloud interface to build awareness among consumers about their electricity usage. Deployment of this system using the concept of IoT provided the scope of reading real-time results from anywhere in the world, which only has constraints of having the right credentials and internet access. It also enabled the simple design and implementation of additional features in the proposed system. This project comprises a section of Proposed

Work, which contains system architecture, hardware design, IoT cloud interface application design, working principle, and system implementation and Result Analysis, which consists of tested results of the intended system.

II. POWER SUPPLY

Block Diagram

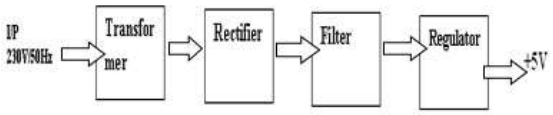


Figure Power Supply

III. HARDWARE

3.1NodeMCU

Introduction to NodeMCU

Introduction
NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.

NodeMCU Development Board/kit v0.9 (Version1)

Since NodeMCU is open source platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 WiFi Module.

There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.

NodeMCU Development Board/kit v1.0 (Version2)

For more information about NodeMCU Boards available in market refer NodeMCU Development Boards

NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc.

Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

How to start with NodeMCU?

NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols.

To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement.

There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement.

To know more about how to build custom NodeMCU firmware online and download it refer Getting started with NodeMCU

How to write codes for NodeMCU?

After setting up ESP8266 with Node-MCU firmware, let's see the IDE (Integrated Development Environment) required for development of NodeMCU.

NodeMCU with ESPlorer IDE

Lua scripts are generally used to code the NodeMCU. Lua is an open source, lightweight, embeddable scripting language built on top of C programming language.

For more information about how to write Lua script for NodeMCU refer Getting started with NodeMCU using ESPlorerIDE

NodeMCU with Arduino IDE

Here is another way of developing NodeMCU with a well-known IDE i.e. Arduino IDE. We can also develop applications on NodeMCU using Arduino development environment. This makes easy for Arduino developers than learning new language and IDE for NodeMCU.

Difference in using ESPlorer and Arduino IDE

Well, there is a programming language difference we can say while developing application for NodeMCU using ESPlorer IDE and Arduino IDE.

We need to code in C\C++ programming language if we are using Arduino IDE for developing NodeMCU applications and Lua language if we are using ESPlorer IDE.

Basically, NodeMCU is Lua Interpreter, so it can understand Lua script easily. When we write Lua scripts for NodeMCU and send/upload it to NodeMCU, then they will get executes sequentially. It will not build binary firmware file of code for NodeMCU to write. It will send Lua script as it is to NodeMCU to get execute.

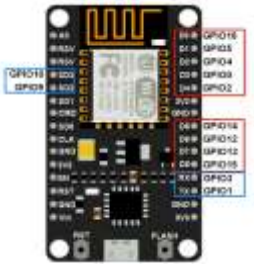
In Arduino IDE when we write and compile code, ESP8266 toolchain in background creates binary firmware file of code we wrote. And when we upload it to NodeMCU then it will flash all NodeMCU firmware with newly generated binary firmware code. In fact, it writes the complete firmware.

That's the reason why NodeMCU not accept further Lua scripts/code after it is getting flashed by Arduino IDE. After getting flashed

by Arduino sketch/code it will be no more Lua interpreter and we got error if we try to upload Lua scripts. To again start with Lua script, we need to flash it with NodeMCU firmware.
Since Arduino IDE compile and upload/writes complete firmware, it takes more time than ESPlorer IDE

NodeMCU GPIO with Arduino IDE

Introduction
General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time.
NodeMCU Development kit provides access to these GPIOs of ESP8266. The only thing to take care is that NodeMCU Dev kit pins are numbered differently than internal GPIO notations of ESP8266 as shown in below figure and table. For example, the D0 pin on the NodeMCU Dev kit is mapped to the internal GPIO pin 16 of ESP8266.



Node MCU DevKit GPIOs

Below table gives NodeMCU Dev Kit IO pins and ESP8266 internal GPIO pins mapping

Pin Names on NodeMCU Development Kit	ESP8266 Internal GPIO Pin number
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD2	GPIO9
D12/SD3	GPIO10

3.2 Liquid Cristal Display

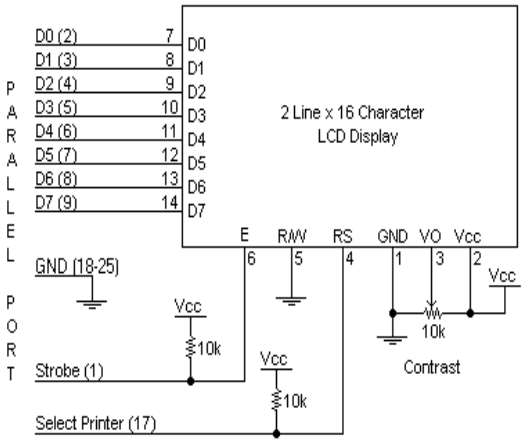
A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

A program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an controller is an LCD display. Some of the most common LCDs connected to the contollers are 16X1, 16x2 and 20x2 displays. This means 16 characters per line by 1 line 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Description Of 16x2:

This is the first interfacing example for the Parallel Port. We will start with something simple. This example doesn't use the Bi-directional feature found on newer ports, thus it should work with most, if no all-Parallel Ports. It however doesn't show the use of the Status Port as an input. So what are we interfacing? A 16 Character x 2 Line LCD Module to the Parallel Port. These LCD Modules are very common these days, and are quite simple to work with, as all the logic required to run them is on board.

3.3 Schematic Diagram



- Above is the quite simple schematic. The LCD panel's *Enable* and *Register Select* is connected to the Control Port. The Control Port is an open collector / open drain output. While most Parallel Ports have internal pull-up resistors, there are a few which don't. Therefore by incorporating the two 10K external pull up resistors, the circuit is more portable for a wider range of computers, some of which may have no internal pull up resistors.
- We make no effort to place the Data bus into reverse direction. Therefore we hard wire the *R/W* line of the LCD panel, into write mode. This will cause no bus conflicts on the data lines. As a result we cannot read back the LCD's internal Busy Flag which tells us if the LCD has accepted and finished processing the last instruction. This problem is overcome by inserting known delays into our program.
- The 10k Potentiometer controls the contrast of the LCD panel. Nothing fancy here. As with all the examples, I've left the power supply out. You can use a bench power supply set to 5v or use a onboard +5 regulator. Remember a few de-coupling capacitors, especially if you have trouble with the circuit working properly.

16 x 2 Alphanumeric LCD Module Features:

- Intelligent, with built-in Hitachi HD44780 compatible LCD controller and RAM providing simple interfacing
- 61 x 15.8 mm viewing area
- 5 x 7 dot matrix format for 2.96 x 5.56 mm characters, plus cursor line
- Can display 224 different symbols
- Low power consumption (1 mA typical)
- Powerful command set and user-produced characters
- TTL and CMOS compatible
- Connector for standard 0.1-pitch pin headers

16 x 2 Alphanumeric LCD Module Specifications:

Pin	Symbol	Level	Function
1	V_{SS}	-	Power, GND
2	V_{DD}	-	Power, 5V
3	Vo	-	Power, for LCD Drive
4	RS	H/L	Register Select Signal H: Data Input L: Instruction Input
5	R/W	H/L	H: Data Read (LCD->MPU) L: Data Write (MPU->LCD)
6	E	H,H->L	Enable
7-14	DB0-DB7	H/L	Data Bus: Software selectable 4- or 8-bit mode
15	NC	-	**NOT CONNECTED**
16	NC	-	**NOT CONNECTED**

FEATURES:

• 5 x 8 dots with cursor
• Built-in controller (KS 0066 or Equivalent)
• + 5V power supply (Also available for + 3V)
• 1/16 duty cycle
• B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
• N.V. optional for + 3V power supply

Data can be placed at any location on the LCD. For 16×1 LCD, the address locations are:

POSITION		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ADDRESS	LINE1	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

Figure Address locations for a 1x16 line LCD

Even limited to character based modules, there is still a wide variety of shapes and sizes available. Line lengths of 8,16,20,24,32 and 40 characters are all standard, in one, two and four line versions.

Several different LC technologies exists. “supertwist” types, for example, offer Improved contrast and viewing angle over the older “twisted nematic” types. Some modules are available with back lighting, so that they can be viewed in dimly-lit conditions. The back lighting may be either “electro-luminescent”, requiring a high voltage inverter circuit, or simple LED illumination.

3.4 PIN DESCRIPTION:

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections).

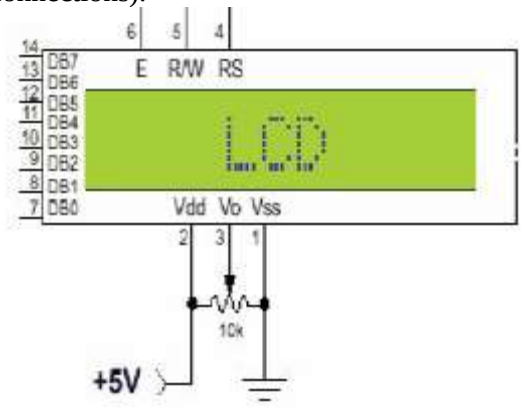


Figure Pin diagram of 1x16 lines LCD

PIN	SYMBOL	FUNCTION
1	Vss	Power Supply(GND)
2	Vdd	Power Supply(+5V)
3	Vo	Contrast Adjust
4	RS	Instruction/Data Register Select
5	R/W	Data Bus Line
6	E	Enable Signal
7-14	DB0-DB7	Data Bus Line
15	A	Power Supply for LED B/L(+)
16	K	Power Supply for LED B/L(-)

Figure Pin specifications

3.5 CONTROL LINES:

EN: Line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

RS: Line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

RW: Line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands, so RW will almost always be low. Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

Logic status on control lines:

• E - 0 Access to LCD disabled
- 1 Access to LCD enabled
• R/W - 0 Writing data to LCD
- 1 Reading data from LCD
• RS - 0 Instructions
- 1 Character

Writing data to the LCD:

1) Set R/W bit to low
2) Set RS bit to logic 0 or 1 (instruction or character)
3) Set data to data lines (if it is writing)
4) Set E line to high
5) Set E line to low

Read data from data lines (if it is reading) on LCD:

1) Set R/W bit to high
2) Set RS bit to logic 0 or 1 (instruction or character)
3) Set data to data lines (if it is writing)
4) Set E line to high
5) Set E line to low

Entering Text:

First, a little tip: it is manually a lot easier to enter characters and commands in hexadecimal rather than binary (although, of course, you will need to translate commands from binary couple of sub-miniature hexadecimal rotary switches is a simple matter, although a little bit into hex so that you know which bits you are setting). Replacing the d.i.l. switch pack with a of re-wiring is necessary.

LCD Commands:

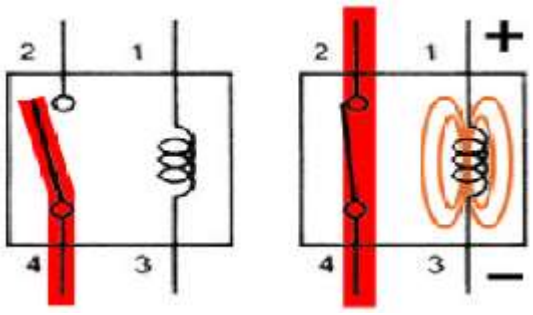
There are some present commands instructions in LCD, which we need to send to LCD through some microcontroller. Some important command instructions are given below:

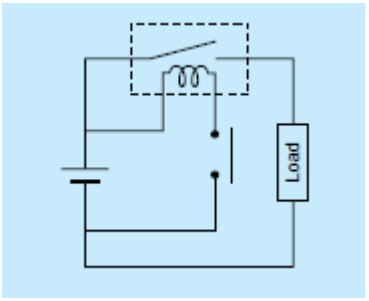
Table Commands

Hex Code	Command to LCD Instruction Register
0F	LCD ON, cursor ON
01	Clear display screen
02	Return home
04	Decrement cursor (shift cursor to left)
06	Increment cursor (shift cursor to right)
05	Shift display right
07	Shift display left
0E	Display ON, cursor blinking
80	Force cursor to beginning of first line
C0	Force cursor to beginning of second line
38	2 lines and 5×7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
OC	Display ON, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2

Relays

A relay is an electrically operated switch. These are remote control electrical switches that are controlled by another switch, such as a horn switch or a computer as in a power train control module, devices in industries, home based applications. Relays allow a small current pin, 4-pin, 5-pin, and 6-pin, single switch or dual switches. Relays are used throughout the automobile. Relays which come in assorted sizes, ratings, and applications, are used as remote control switches. A typical vehicle can have 20 relays or more.





Specification

- Number and type of contacts – normally open, normally closed, (double-throw)
- Contact sequence – "Make before Break" or "Break before Make". For example, the old style telephone exchanges required Make-before-break so that the connection didn't get dropped while dialing the number.
- Rating of contacts – small relays switch a few amperes, large contactors are rated for up to 3000 amperes, alternating or direct current
- Voltage rating of contacts – typical control relays rated 300 VAC or 600 VAC, automotive types to 50 VDC, special high-voltage relays to about 15 000 V
- Coil voltage – machine-tool relays usually 24 VAC, 120 or 250 VAC, relays for switchgear may have 125 V or 250 VDC coils, "sensitive" relays operate on a few milli-amperes

Applications:

Relays are used:

- To control a high-voltage circuit with a low-voltage signal, as in some types of modems,
- To control a high-current circuit with a low-current signal, as in the starter solenoid of an automobile,
- To detect and isolate faults on transmission and distribution lines by opening and closing circuit breakers (protection relays),
- To isolate the controlling circuit from the controlled circuit when the two are at different potentials, for example when controlling a mains-powered device from a low-voltage switch. The latter is often applied to control office lighting as the low voltage wires are

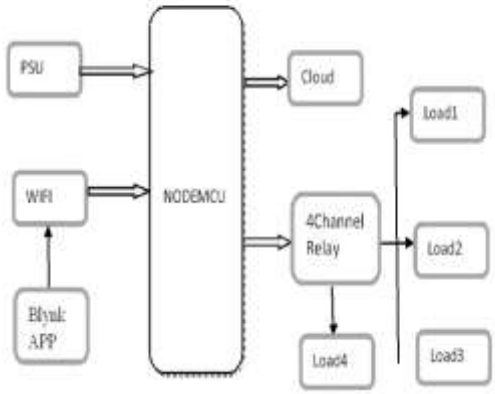
easily installed in partitions, which may be often moved as needs change. They may also be controlled by room occupancy detectors in an effort to conserve energy,

- To perform logic functions. For example, the boolean AND function is realized by connecting relay contacts in series, the OR function by connecting contacts in parallel. Due to the failure modes of a relay compared with a semiconductor, they are widely used in safety critical logic, such as the control panels of radioactive waste handling machinery.
- As oscillators, also called vibrators. The coil is wired in series with the normally closed contacts. When a current is passed through the relay coil, the relay operates and opens the contacts that carry the supply current. This stops the current and causes the contacts to close again. The cycle repeats continuously, causing the relay to open and close rapidly. Vibrators are used to generate pulsed current.
- To generate sound. A vibrator, described above, creates a buzzing sound because of the rapid oscillation of the armature. This is the basis of the electric bell, which consists of a vibrator with a hammer attached to the armature so it can repeatedly strike a bell.
- To perform time delay functions. Relays can be used to act as an mechanical time delay device by controlling the release time by using the effect of residual magnetism by means of a inserting copper disk between the armature and moving blade assembly.

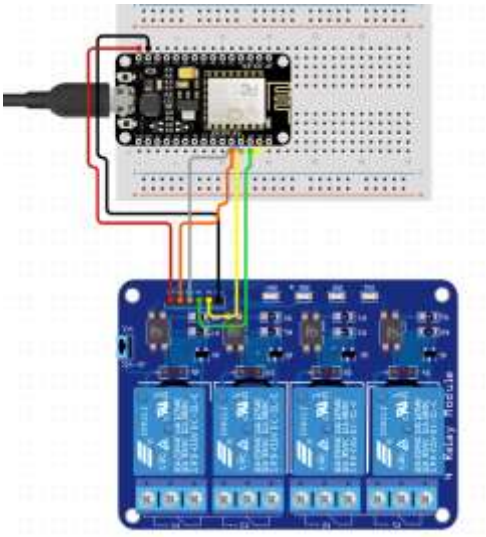
Figure Relay

IV.Result

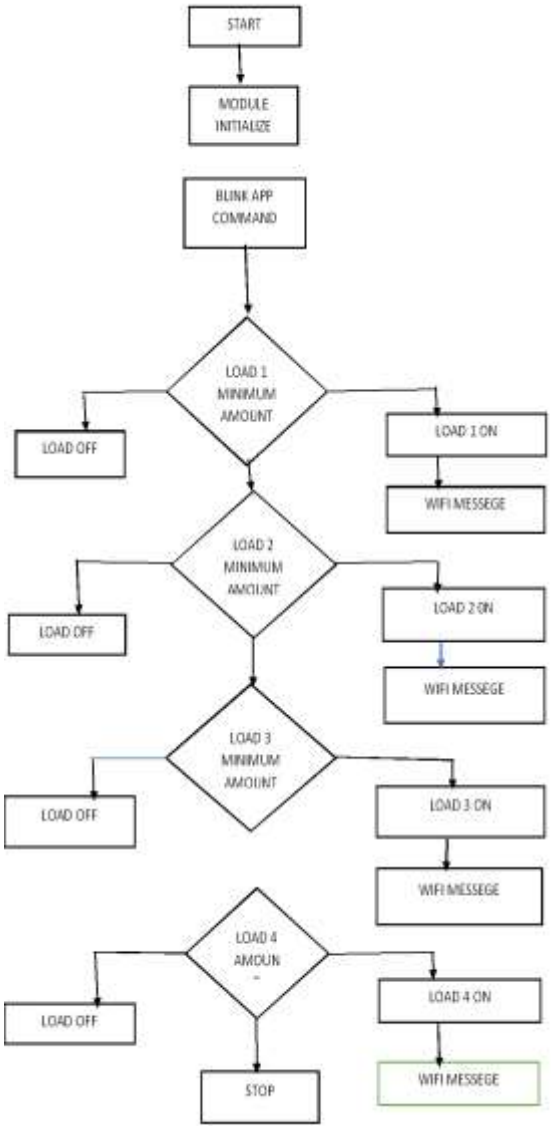
Block Diagram



4.1 Schematic Diagram



4.2FLOWCHART



V. CONCLUSION

IoT based energy tracking and bill estimation system discussed in this paper with various sections is successful in building awareness about electricity usage by displaying realtime estimated electricity consumption by each connected to it and real-time estimated bill of total consumption on monitor unit built-in IoT cloud interface. It has a simple design as it fetches the average consumption detail of loads from a cloudhosted database and not uses any chips or sensors to measure electricity, current, and voltage. It also has additional features of logging the final estimated bill of each month to a cloudhosted database and transmitting switching instructions for loads via a control unit built-in IoT cloud interface. The simplistic design and implementation of extra features were possible by utilizing the concept of the Internet of Things (IoT) in the proposed system.

VI. REFERENCES

[1] Bhalaji, N. "EL DAPP–An Electricity Meter Tracking Decentralized Application." Journal of Electronics 2, no. 01 (2020): 49-71.
[2] A. Y. Devadhanishini, R. K. Malasri, N. Nandinipriya, V. Subashini and P. G. Padma Gowri, "Smart Power Monitoring System Using Iot," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 813- 816.
[3] M. J. Islam Mozumder and S. Ghosh, "IoT Based Automatic Electricity Monitoring and Remote Load Cont rol System Using PIC18F4550," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bangalore, 2018, pp. 1-4.
[4] K. Chooruang and K. Meekul, "Design of an IoT Energy Monitoring System," 2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, 2018, pp. 1-4.
[5] D. Lestari, I. D. Wahyono and I. Fadlika, "IoT based Electrical Energy Consumption Monitoring System Prototype: Case study in G4 Building Universitas Negeri Malang," 2017 International Conference on Sustainable Information Engineering and Technology (SIET), Malang, 2017, pp. 342-347.
[6] A. Chaudhari, B. Rodrigues and S. More, "Automated IOT based system for home automation and prediction of electricity usage and comparative analysis of various electricity providers: SmartPlug," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 390-392.
[7] K. B. Tarase and V. M. Panchade, "Monitoring & Controlling of Substation Using IoT in Distribution Power Grid," 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 2020, pp. 66-70.