**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

Jnanasangama, Macche, Santibastwada Road

Belagavi-590018, Karnataka



**A**

**Project Report**

on

**Battery Condition Prognostic System   
using data acquisition algorithm using Labview in Smart Microgrids**

*Submitted in partial fulfillment of the requirement for the degree of*

**Bachelor of Engineering**

*in*

**Electronics & CommunicationEngineering**

*By*

**1DS16EC010 : AKHILA SP**

**1DS16EC092 : PRAJHWAL RAO**

**1DS16EC110 : SACHIN BS**

**1DS16EC173 : GNANENDRA S**

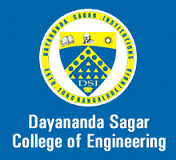
Under the guidance

of

**Dr. H.V. MANJUNATH**

Internal Departmental Project Guide

Designation, ECE Dept., DSCE, Bengaluru



**Department of Electronics & Communication Engineering**

(An Autonomous College affiliated to VTU Belgaum, accredited by NBA & NAAC)

Shavige Malleshwara Hills, Kumaraswamy Layout,

**Certificate**

Certified that the project work entitled “**Battery Condition Prognostic System using data acquisition algorithm using Labview in Smart Microgrids**” carried out by **Akhila S P** (USN-1DS16EC010), **Prajhwal Rao** (USN-1DS16EC092), **Sachin B S** (USN-1DS16EC110), **Gnanendra S** (USN-1DS16EC173) are bonafide students of Dayananda Sagar College of Engineering, Bangalore, Karnataka, India in partial fulfillment for the award of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2019-20. It is certified that all corrections / suggestions indicated for project work have been incorporated in the report deposited to the ECE department, the college central library & to the university. This project report (EC83) has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dept. Project Coordinator Convener Project Guide

Dr. K.N. Pushpalatha H.V. Manjunath

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Head of the Department Dr. C.P.S. Prakash

Dr. T.C.Manjunath Principal

External Project Viva-Voce

Name of the project examiners :

1 : Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 : Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Declaration

Certified that the project work entitled, “**Battery Condition Prognostic System using data acquisition algorithm using Labview in Smart Microgrids**” is a bonafide work that was carried out by ourselves in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engg. of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2019-20. We, the students of the project group/batch no. \_\_\_ hereby declare that the entire project work has been done on our own & we have not copied or duplicated any other’s work. The results embedded in this project report has not been submitted elsewhere for the award of any type of degree.

Ms. Akhila S P USN :1DS16EC010

Sign : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Prajhwal Rao USN : 1DS16EC092

Sign : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Sachin B S USN : 1DS16EC110

Sign : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Gnanendra S USN : 1DS16EC173

Sign : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date : / /

Place : Bengaluru -78

**Acknowledgement**

We would like to express our special thanks of gratitude to our college committee for their continuous support and encouragement: Dr. Hemachandra Sagar, The Chairman, Dr. Premchandra Sagar, The Vice Chairman, Galiswamy, Secretary and Dr C.P.S. Prakash, The Principal. We offer our sincere gratitude for the opportunity of learning provided by the committee. We would also like to thank, Dr T.C. Manjunath, HOD of ECE Department, for his constant support. We are highly indebted to Dr.H.V. Manjunath, our Project Guide, Dr. K.N. Pushpalatha, our Project convener, and would like to extend our gratitude to all our department project coordinators-Dr. Vimala P,Dr. Rajagopal A, Sapna PJ, Deepa NP, and to all my department staff members for giving me such attention and time. I would like to express my gratitude towards my parents and my relatives for their kind cooperation and encouragement which helped me in completion of this internship. My thanks and appreciations also go to my friends for supporting me during this internship. I would also thank the supreme power, the Almighty God who is always the one to guide me to work on the right path of my life.

**Abstract**

Battery Management Systems (BMS) are used in many industrial and commercial systems to make the battery operation more efficient and for the estimation to keep the battery state, as long as possible, away from destructive state, to increase battery life time. For this purpose, many monitoring techniques are used to monitor the battery state of charge, temperature and current. This system evaluates and displays the battery temperature, charging/discharging current and State Of Charge (SOC) for the considered model battery. For monitoring purpose, digital and analog sensors with micro-controllers are used. The battery information and the obtained results explaining the main characteristics of the system are presented by photographs and some experimental results .

The real time Battery Monitoring System (BMS) has been designed using Lab-View . The voltage divider was used for sensing of battery voltage and the ACS712 based on effect hall sensor was used for current sensing. The computer with Ni Lab-View calculated and observed voltage and current of Battery. The result showed that BMS can monitor of voltage and current with real time condition.

**Keywords:** Battery Management System, State of Charge, Micro-controller, Voltage Divider, Hall Effect Sensor, Lab-View.

**Table of Contents**

Title Sheet 1

Certificate 2

Declaration 3

Acknowledgement 4

Abstract 5

Table of Contents 6

List of Figures 7

List of Tables 7

Nomenclature and Acronyms 8

Chapter 1 Introduction 9

1.1 Overview 9

1.2 Literature survey 10

1.3 Objectives / Scope / Aim of the project work 16

1.4 Methodology 16

1.5 Organization of the project report 17

Chapter 2 Block diagram and working principle 18

Chapter 3 Hardware/ Software tools /Description/Interfacing 45

Chapter 4 Algorithm or flowchart 83

Chapter 5 Results and Discussions 86

Chapter 6 Applications, Advantages and Limitations 87

Chapter 7 Conclusions and Future Work 90

References 92

# List of Figures

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Name** | **Pg.No** |
| 1 | IoT definition in its simplest form | 19 |
| 2 | Basic requirements for an IoT solution | 21 |
| 3 | Example of monitoring systems in a network operations center | 22 |
| 4 | IoT levels | 24 |
| 5 | OSI layers and data formats | 25 |
| 6 | “Thing” in IoT: definition view | 30 |
| 7 | “Things” in IoT: IoT level view | 30 |
| 8 | Components of smart sensors | 32 |
| 9 | Block diagram of Battery management system using LabVIEW | 42 |
| 10 | Data Acquisition Algorithm | 89 |
| 11 | Data Networking Algorithm | 90 |

# NOMENCLATURE AND ACRONYMS

|  |  |
| --- | --- |
| IEEE | Institute of Electrical & Electronics Engineers |
| DSCE | Dayananda Sagar College of Engineering |
| ECE | Electronics & Communication Engineering |
| BMS | Battery Monitoring System |
| SOC | State of charge |
| C | charge capacity |
| Ah | Ampere Hours |
| A | Amp |
| V | Voltage |
| I | load current |
| T | Absolute temperature |
| AC | Alternating Current |
| DC | Direct Current |
| EV | Electric Vehicle |
| HEV | Hybrid Electric Vehicle |
| DAA | Data Acquisition Algorithm |
| IOT | Internet of Things |
| SOH | State of Health |
| LabVIEW | Laboratory Virtual Instrument Engineering Workbench |

# CHAPTER 1

# INTRODUCTION

* 1. **Overview**

Today’s electronic devices have higher mobility and are greener than ever before. Battery advancements are fueling this progression in a wide range of products from portable power tools to plug-in hybrid electric vehicles and wireless speakers. In recent years, the efficiency of a battery in terms of how much power it can output with respect to size and weight has dramatically improved. Think about how heavy and bulky a car battery is. Its main purpose is to start the car. With recent advancements, you can purchase a lithium-ion battery to jump start your car, and it only weighs a couple pounds and is the size of your hand. The ongoing transformation of battery technology has prompted many newcomers to learn about designing battery management systems.

Battery is the most essential component of any vehicle. So perfect maintenance of any battery is very much essential for it to function properly. Lead Acid batteries which are more commonly used in the vehicular batteries, need to be efficiently monitored, for it to perform better under all circumstances. So, a more systematic battery management system needs to be implemented so that the performance of the battery can be monitored continuously. When it comes to battery, the two most important parameters are the State Of Charging (SoC) and State of Health (SoH) of the battery. There are several coherent methods to calculate these parameters. But these methods cannot provide correct results, as the battery materials, atmosphere surrounding the battery, the load put on to the battery, will affect these parameters. Overcharging of the battery leads to emission of gases like Hydrogen, Oxygen etc. This Battery Management System (BMS) aims at detecting the emission of these gases from the battery, when it is overcharged, and monitors the other basic parameters such as Voltage, Current, Temperature of the battery using STM controller and sensors. It is also equipped with GPS module, which enables tracking of vehicles. Also these values are displayed in Cloud, which brings the concept of Internet of Things (IoT).

## **Literature** **Survey**

1. **Battery Condition Prognostic System using IoT in Smart Microgrids**

Gayathri M. S., Ravishankar A. N., Kumaravel S., and Ashok S. Department of Electrical Engineering National Institute of Technology Calicut, Kerala, India.

This paper presents an idea to share the battery condition monitoring parameters among the stakeholders like manufacturer, market dealers, users etc. by using latest Internet of Things (IoT) technology. Upon accessing such parameters, corrective actions can be taken to improve the battery life. By placing sensors on the battery, which can send the battery parameter data over internet to cloud database. This database can be accessed by manufacturer, dealers and users to monitor the health condition of battery. This will help to enhance the efficient use of battery and improve the life. Also, it can track the number of batteries in use with usage profile. The hardware prototype of the proposed system has been developed and tested with a lead acid SMF battery. The Mean Square Error (MSE) of the measured quantities are with in acceptable limits.

This paper has presented a novel idea to monitor a battery by the manufacturer, market dealer once it is deployed for its use. The latest IoT technology enables to acquire battery parameter values with less time and cost. This work would facilitates the battery manufacturer to identify batteries that are in inefficient operating conditions and encourages the user for better efficient use through suggestions and recommendations. The storage becomes necessary for any sustainable energy supply system that uses renewable energy technologies. Also, electric vehicles will deeply penetrates into common man’s transportation options. In these two areas, batteries are good option for storage solutions. This makes huge deployment of batteries across the world. Therefore battery management systems that monitors real time trends of parameters and share this knowledge to stake holders becomes highly necessary. This work could demonstrate low cost method to implement the same with the help of emerging IoT technology. The results shows that, the proposed system can be used for designing industrial prototypes.Also it is having an advantage over existing ones, that the parameters can be accesses at any time, any where with the help of IoT which improves the efficiency of corrective actions taken.

1. **Centralized Environment and Battery Monitoring System For Server Rooms**

Haaris Rasool**1**, Aazim Rasool**2**, Urfa Rasool3, Ali Raza**4 ,**Waqar Ahmad1NUCES FAST Pakistan,2NCEPU China,3EME NUST Pakistan,4Air University Pakistan, NUCES FAST Pakistan.

In this paper, the RF transmitter-receiver based Centralized Environmental and Battery Monitoring System (CEBMS) for server room was developed. The

system is capable of monitoring Temperature, Humidity, Smoke, Water Leakage, Battery Status and Battery life of 6 volt lead acid battery (or any other). Day wise data for 5 seconds duration is stored by computer. Graphical User Interface (GUI) is used to display the data in tabular as well as in graphical form. The data is received from sensor unit and transmitted to desired destination by using PIC controller.

Monitoring has always been essential; whether in a server rooms, power plants or in factories where heavy machinery runs. It may also be needed by a small organization handling loads of data. Nothing can work up to the mark without proper and reliable monitoring which ensures flawless performance of the equipment. Environmental conditions can have very bad impact on the equipment and may cause a large scale business loss. It may be in the form of downtime,

equipment loss, failure in machinery operations etc.

The literature shows lot of equipments, devices and techniques of monitoring available in research papers and also by companies in market where monitoring is done in onload condition or monitoring system setup in laboratory in off-load condition. Research groups and leading vendors worked upon it and found out an estimate that losses due to the threats from environmental conditions can be about 50-100 billion dollars to business and industrial organizations. The research results added that in a year, 23% of data centers faced downtime more than 5 times due to environmental issues. 61% of the other data centers experienced downtime 1- 4 times and only 16% data centers showed that environmental issues are not impacting their system reliability.

Environmental threats observed are Temperature, Humidity,Water leakage, Smoke, Dust, Battery condition, Air flow management and Power control etc. Our mission is to design centralized environment and battery monitoring system which is based on accurate measurement and also cost effective.

The developed CEBMS is capable to monitor all the desired parameter. The devices in the market are vary up to the mark in performance efficiency and features, but we needed a simple and cheap solution for our local small business area, organizations and school/universities. Basic environmental monitoring, easy, not very complex, ultimately proving the hardware and software low on cost. Designing is the first and cure important step for every system and equally important for engineers .This project helped us to understand the concept and principles of how we can find simpler and cheap ways to design an existing approach. This device is cheap, it is serving the purpose and resolving the issue. This project helped us learn a lot specially SPI protocol and hardware software designing. All our designed modules are in working conditions and display results. Furthermore, the improvement can be made in system by adding number of sensors and integrating more controllers. this project does not have the control capabilities and opens, so there are many opportunities for future enhancements.

1. **Development of Wireless Battery Monitoring for electric vehicle**

A Wireless Battery Monitoring System (WBMS) for electric vehicle has been developed for monitoring voltage, current and temperature of battery. This system consists of hardwares (sensors, a microcontroller, a bluetooth module, an Android smartphone) and software. It was designed on a low cost microcontroller ATMEGA 328 (Arduino UNO). Voltage, current and temperature data are transfered to microcontroller, then data of battery is transfered using bluetooth communication to display.

1. **Online battery monitoring system based on gprs for electric vehicles**

Electric vehicles which has the advantages of energy saving and environmental protection, are brought to global attention as an energy crisis and climate change. Electric vehicle battery technology is one of the hotspots and difficulties in electric vehicle research in recent years. From a business perspective, capable of grasping the working status of electric vehicle battery in real time, in order to put forward a control strategy for extending the battery life is very necessary.

1. **Implementation and testing of theft and maintenance monitoring of batteries of stand-alone SPV systems**

This paper presents design, implementation and testing of Theft and maintenance monitoring of batteries of Stand-alone SPV Systems. Solar Photovoltaic stand alone power sources are popularly used for outdoor lighting, viz, garden lighting, street lighting etc. Batteries are invariably used to store the solar energy during its availability. Solar power energy sources and the batteries are often located in open fields and remote areas. Two problems are associated with said applications. Firstly, the batteries are more vulnerable to theft. Secondly, the monitoring of dry run or discharge of batteries and maintenance becomes difficult and expensive. To address these problems, a Theft & Maintenance monitoring system is designed and implemented. The implemented system consists of an accelerometer to detect displacement of the battery from its place and temperature sensor to measure the temperature. Microcontrollers and GSM system are used to raise an alarm, display the status.

1. **Development of battery monitoring system in smart microgrid based on Internet of Things (IoT)**

In this work, a historical data based battery management system (BMS) was successfully developed and implemented using an embedded system for condition monitoring of a battery energy storage system in a smart microgrid. The performance was assessed for 28 days of operating time with a one-minute sampling time. The historical data showed that the maximum temperature increment and the maximum temperature difference between the batteries were 4.5 °C and 2.8 °C. One of the batteries had a high voltage rate of change, i.e. above 3.0 V/min, and its temperature rate of change was very sensitive, even at low voltage rate of changes. This phenomenon tends to indicate problems that may deplete the battery energy storage system’s total capacity. The primary findings of this study are that the voltage and temperature rates of change of individual batteries in real operating conditions can be used to diagnose and foresee imminent failure, and in the event of a failure occurring the root cause of the problem can be found by using the historical data based BMS. To ensure further safety and reliability of acceptable practical operating conditions, rate of change limits are proposed based on battery characteristics for temperatures below 0.5 °C/min and voltages below 3.0 V/min.

1. **Design Real Time Battery Monitoring System Using LabVIEW Interface For Arduino (LIFA)**

The real time Battery Monitoring System (BMS) has been designed using LabVIEW Interface for Arduino (LIFA). The voltage devide was used for sensing of battery voltage and the ACS712 based on effect hall sensor was used for current sensing. The computer with Ni LaBVIEW calculated and observed voltage and current of Battery. The result showed that BMS can monitor of voltage and current with real time condition. In this paper, it is reported design for realtime battery monitoring system based on LIFA. Arduino has become a popular open-source, single-board microcontroller among electronic hobbyists, and it is gaining acceptance as a quick prototyping tool for engineering and educational projects also [9]. Lab VIEW is available for all the major platforms and is easily portable across platforms. It is simple and flexible, since it is a graphical approach no need of writing programs of 100 lines like other program languages. In computer interfacing, LabVIEW software is the main display unit for realtime monitoring battery based on LIFA. The aim of these experiments is to demonstrate that current and voltage supplied by a secondary battery during the complete course of the discharge are time-dependent. The secondary Ni-Cd battery has been used to test performance of BMS. The BMS has been succesfully developed based on LabVIEW Interface for Arduino (LIFA). It can be realtime monitoring of voltage and current battery on discharger condition. This system is easy for using and converting data to excel.

**1.3 Objectives / Scope / Aim of the project work**

* To estimate the health of battery by using voltage, current, temperature.
* State of Charge (SoC) of battery is estimated by using predication algorithms.
* Health of battery is also determined by checking on overcharging and depth of discharge.
* Under voltage and over voltage protection circuits are built to prevent destruction and deterioration of battery respectively.
* Database can be accessed by manufacturer, dealers and users to monitor the health condition of battery. This will help to enhance the efficient use of battery and improve the life.
* Real time monitoring of sensor data using lab view interface for Arduino.
* An application is developed to share the monitored parameters to customers, market dealers and manufacturers**.**
  1. **Methodology**

AS we know batteries are essential part of modern technology ,these batteries are needed to be monitored to get more number of life cycles and also for good health condition of the battery. So, we have designed a Battery Monitoring Sysem using Labview i.e, Battery parameters like temperature, current, voltage, SOC and SOH, humidity of the environment are being monitored. Temperature sensor, Voltage circuit, current sensor are used to fetch the data and this data is given to Arduino microcontroller where the data is again sent to an app using wifi-module and also to LabVIEW tool for monitoring. Humidity and Smoke Sensor also provide real time data to arduino which turns ON a buzzer for indication in case of any hazard.

## **1.5** **Organization of the project report**

The project work undertaken by us is organized in the following sequence as follows.

A brief introduction to the work was presented in the introductory chapter in chapter-1.

In Chater-2 a brief theory about Internet of Things and LabVIEW is given.

Chapter-3 gives an in depth idea of Block diagram and flow chart that we have developed to implement our work.Chapter 4 deals with various software and hardware tools used to implement our work and also description of the same. Chapter-5 talks about the various results that we have obtained as the consequence of our algorithm and tools used. Chapter-6 talks about advantages,disadvantages and application of our model.

Finally, the report concludes with the conclusion and future work in chapter –7.

**CHAPTER 2**

**CHAPTER 2.1: Internet of Things**

The Internet of Things (IoT) has gained significant mindshare, let alone attention, in academia and the industry especially over the past few years. The reasons behind this interest are the potential capabilities that IoT promises to offer.

IoT may be considered as a network of physical elements empowered by:

• Sensors: to collect information.

• Identifiers: to identify the source of data (e.g., sensors, devices).

• Software: to analyze data.

• Internet connectivity: to communicate and notify.

Putting it all together, IoT is the network of things, with clear element identification, embedded with software intelligence, sensors, and ubiquitous connectivity to the Internet.

IoT enables things or objects to exchange information with the manufacturer, operator, and/or other connected devices utilizing the telecommunications infrastructure of the Internet. It allows physical objects to be sensed (to provide specific information) and controlled remotely across the Internet, thereby creating opportunities for more direct integration between the physical world and computer based systems and resulting in improved efficiency, accuracy, and economic benefit. Each thing is uniquely identifiable through its embedded computing system and is able to interoperate within the existing Internet infrastructure.

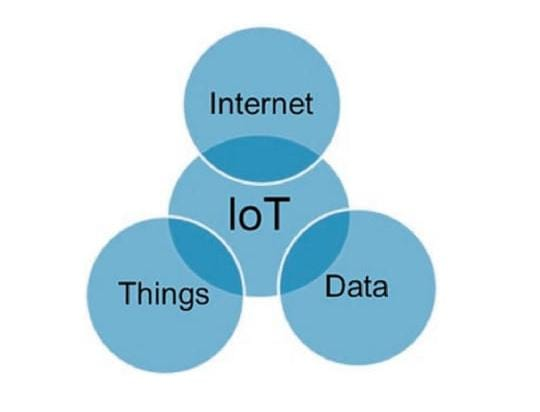


Fig. 2.1.1: IoT definition in its simplest form

The main idea of IoT is to physically connect anything/everything (e.g., sensors, devices, machines, people, animals, trees) and processes over the Internet for monitoring and/or control functionality. Connections are not limited to information sites, they’re actual and physical connections allowing users to reach “things” and take control when needed. Hence, connecting objects together is not an objective by itself, but gathering intelligence from such objects to enrich products and services is.

**How to Monitor and Control Things from Anywhere in the World?**

The basic requirements for IoT are the unique identity per “thing” (e.g., IP address), the ability to communicate between things (e.g., wireless communications), and the ability to sense specific information about the thing (sensors). With these three requirements, one should be able to monitor things from anywhere in the world. Another foundation requirement is a medium to communicate. Such requirement is typically handled by a telecommunications network. Figure 1.3 presents the very basic requirements of an IoT solution.

**Why Do We Want to Monitor and Control Things?**

There are many reasons to monitor and control things remotely over the Internet: monitoring and controlling things by experts (e.g., a patient’s temperature or blood pressure while the patient is at the comfort of his or her own home); learning about things by pointing a smartphone to a thing of interest, for instance; searching for things that search engines (e.g., Google) do not provide today (e.g., where are my car keys); allowing authorities to manage things in smart cities in an optimal manner (e.g., energy, driver licenses, and other documents from Department Motor Vehicle, senior citizen); and, finally, providing more affordable entertainment and games for children and adults. All of these are examples of huge business and service opportunities to boost the economic impact for consumers, businesses, governments, hospitals, and many other entities.

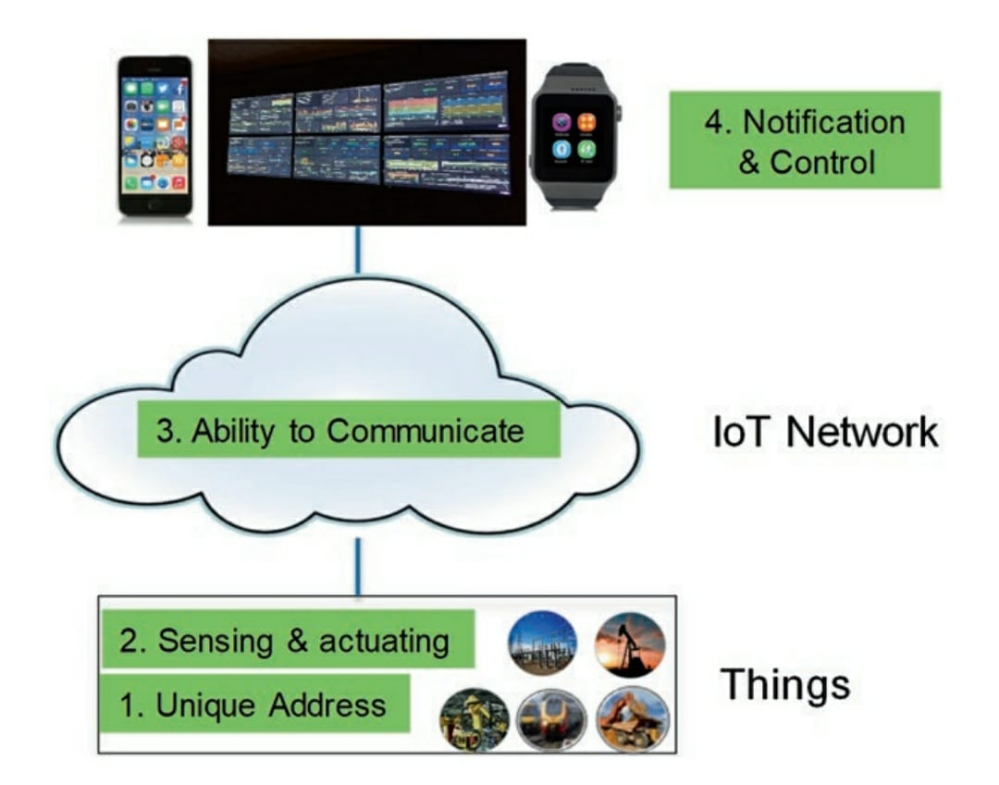
****

Fig. 2.1.2: Basic requirements for an IoT solution

**Who Will Monitor and Control?**

Generally speaking, monitoring and control of IoT services may be done by any person or any machine. For example, a homeowner monitoring his own home on a mobile device based on a security system she or he has installed and configured. The homeowner may also control lights, turn on the air conditioning, shut off the heater, etc. Another example is for a service provider to monitor and control services for its customers in a network operations center (NOC) as shown in fig.

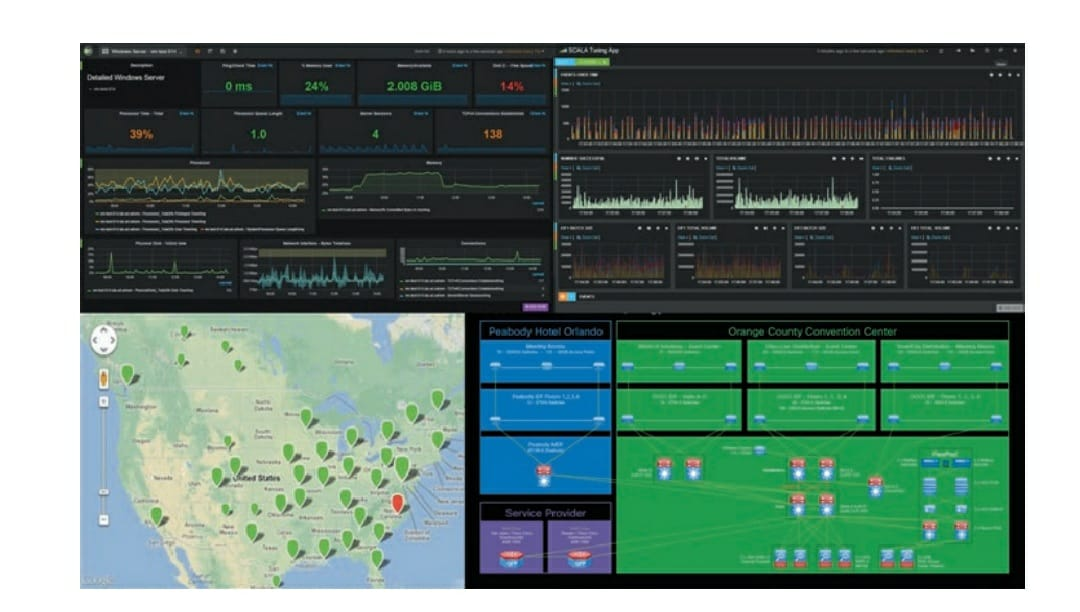
****

Fig. 2.1.3: Example of monitoring systems in a network operations center

Obviously, security is a major concern to prevent access by non-authorized people and, more importantly, prevent a malicious hacker from gaining access to the system and sending old views to the homeowner while a thief is breaking in. The areas of control are far more critical for enterprise-sensitive applications such as healthcare monitoring of patients and banking applications.

**How Is Security Guaranteed?**

Securing IoT is perhaps the biggest opportunity for technology companies and will remain so for some time in the future. Before IoT, information technology security professionals worked in a bubble as they literally owned and controlled their entire networks and secured all devices behind firewalls. With IoT, data will be collected from external, often mobile, sensors that are placed in public sites (e.g., city streets) allowing strangers to send harmful data to any network. Bring your own device (BYOD) is another example where third-party devices and hence noncorporate data sources are allowed to enter the network. IoT areas that are considered to be most vulnerable include:

Accessing data during transport (network and transport security). Data will be transported in IoT networks at all time, for example, from sensors to gateways and from gateways to data centers in enterprises or from sensors to gateways for residential services such as video from home monitoring system to the homeowner’s smartphone while he’s in a coffee shop. This data may be sniffed by the man in the middle unless the transport protocols are fully secure and encrypted. • Having control of IoT devices (control of the APIs) allows unauthorized persons to take full control of entire networks. Examples include shutting down cameras at home and shutting down patient monitoring systems. • Having access to the IoT data itself. Is the data easily accessible? Is it stored encrypted? Shared storage in the cloud is another problem where customer a may log in as customer B and look at his data. Another common problem is spoofing data via Bluetooth. Many companies are adding Bluetooth support to their devices making it more feasible for unauthorized persons to access the device’s data. • Stealing official user or network identity (stealing user or network credentials). Many websites provide default passwords for vendors.

**Overview of the IoT levels**

Advantages of the proposed IoT four-level model include:

• Reduced Complexity: It breaks IoT elements and communication processes into smaller and simpler components, thereby helping IoT component development, design, and troubleshooting.

• Standardized Components and Interfaces: The model standardizes the specific components within each level (e.g., what are the key components for general IoT Services Platform) as well as the interfaces between the various levels. This would allow different vendors to develop joint solutions and common support models.

• Module Engineering: It allows various types of IoT hardware and software systems to communicate with each other.

• Interoperability between vendors by ensuring the various technology building blocks can interwork and interoperate.

• Accelerate Innovation: It allows developers to focus on solving the main problem at hand without worrying about basic functions that can be implemented once across different business verticals.

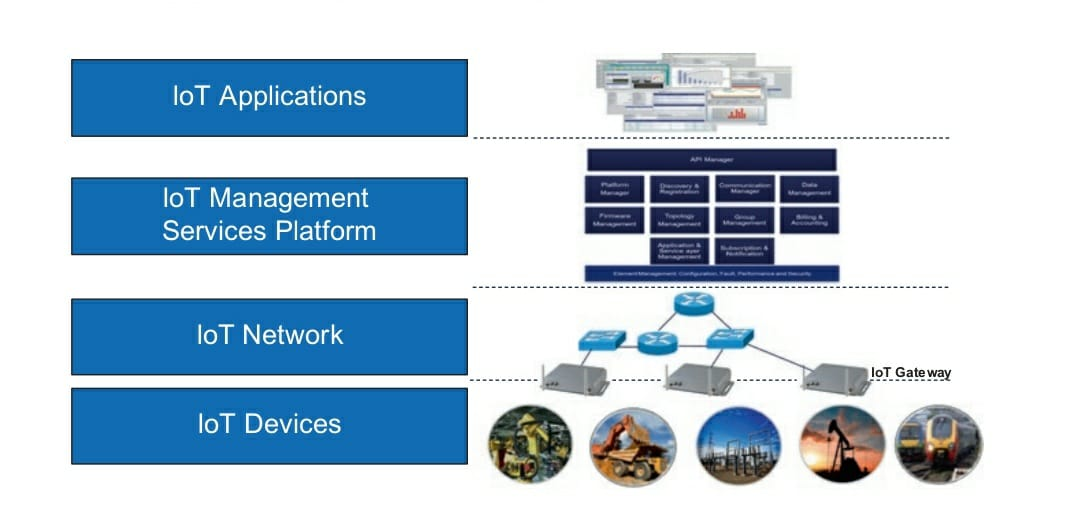
****

Fig. 2.1.4: IoT levels

**The Internet in IoT**

**The Open System Interconnection Model**

The OSI model consists of seven layers as shown in Fig. : Physical (layer 1), Data Link (layer 2), Network (layer 3), Transport (layer 4), Session (layer 5), Presentation (layer 6), and Application (layer 7). Each layer provides some welldefined services to the adjacent layer further up or down the stack, although the distinction can become a bit less defined in layers 6 and 7 with some services overlapping the two layers.

****

Fig. 2.1.5: OSI layers and data formats

• OSI Layer 7 – Application Layer: Starting from the top, the Application Layer is an abstraction layer that specifies the shared protocols and interface methods used by hosts in a communications network. It is where users interact with the network using higher-level protocols such as DNS (Domain Naming System), HTTP (Hypertext Transfer Protocol), Telnet, SSH, FTP (File Transfer Protocol), TFTP (Trivial File Transfer Protocol), SNMP (Simple Network Management Protocol), SMTP (Simple Mail Transfer Protocol), X Windows, RDP (Remote Desktop Protocol), etc.

OSI Layer 6  – Presentation Layer: Underneath the Application Layer is the Presentation Layer. This is where operating system services (e.g., Linux, Unix, Windows, MacOS) reside. The Presentation Layer is responsible for the delivery and formatting of information to the Application Layer for additional processing if required. It ensures that the data can be understood between the sender and receiver. Thus it is tasked with taking care of any issues that might arise where data sent from one system needs to be viewed in a different way by the other system. The Presentation Layer releases the Application Layer of concerns regarding syntactical differences in data representation within the end-user systems. Example of a presentation service would be the conversion of an EBCDICcoded text computer file to an ASCII-coded file and certain types of encryption such as Secure Sockets Layer (SSL) protocol.

• OSI Layer 5 – Session Layer: Below the Presentation Layer is the Session Layer. The Session Layer deals with the communication to create and manage a session (or multiple sessions) between two network elements (e.g., a session between your computer and the server that your computer is getting information from).

• OSI Layer 4 – Transport Layer: The Transport Layer establishes and manages the end-to-end communication between two end points. The Transport Layer breaks the data, it receives from the Session Layer, into smaller units called Segments. It also ensures reliable data delivery (e.g., error detection and retransmission where applicable). It uses the concept of windowing to decide how much information should be sent at a time between end points. Layer 4 main protocols include Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is used for guarantee delivery applications such as FTP and web browsing applications, whereas UDP is used for best effort applications such as IP telephony and video over IP.

• OSI Layer 3 – Network Layer: The Network Layer provides connectivity and path selection (i.e., IP routing) based on logical addresses (i.e., IP addresses). Hence, routers operate at the Network Layer. The Network Layer breaks up the data it receives from the Transport Layer into packets, which are also known as IP datagrams, which contain source and destination IP address information that is used to forward the datagrams between hosts and across networks.1 The Network Layer is also responsible for routing of IP datagrams using IP addresses. A routing protocol specifies how routers communicate with each other, exchanging information that enables them to select routes between any two nodes on a computer network. Routing algorithms determine the specific choice of routes. Each router has a priori knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors and then throughout the network. This way, routers gain knowledge of the topology of the network. The major routing protocol classes in IP networks will be covered in Sect. 2.5. They include interior gateway protocol type 1, interior gateway protocol type 2, and exterior gateway protocols. The latter are routing protocols used on the Internet for exchanging routing information between autonomous systems.

It must be noted that while layers 3 and 4 (Network and Transport Layers) are theoretically separated, they are typically closely related to each other in practice. The well-known Internet Protocol name “TCP/IP” comes from the Transport Layer protocol (TCP) and Network Layer protocol (IP). • Packet switching networks depend upon a connectionless internetwork layer in which a host can send a message without establishing a physical connection with the recipient. In this case, the host simply puts the message onto the network with the destination address and hopes that it arrives. The message data packets may appear in a different order than they were sent in connectionless networks. It is the job of the higher layers, at the destination side, to rearrange out of order packets and deliver them to proper network applications operating at the Application Layer. • OSI Layer 2 – Data Link Layer: The Data Link Layer defines data formats for final transmission. The Data Link Layer breaks up the data it receives into frames. It deals with delivery of frames between devices on the same LAN using Media Access Control (MAC) Addresses. Frames do not cross the boundaries of a local network. Internetwork routing is addressed by layer 3, allowing data link protocols to focus on local delivery, physical addressing, and media arbitration. In this way, the Data Link Layer is analogous to a neighborhood traffic cop; it endeavors to arbitrate between parties contending for access to a medium, without concern for their ultimate destination. The Data Link Layer typically has error detection (e.g., Cyclical Redundancy Check (CRC)). Typical Data Link Layer devices include switches, bridges, and wireless access points (APs). Examples of data link protocols are Ethernet for local area networks (multi-node) and the Point-toPoint Protocol (PPP).

• OSI Layer 1 – Physical Layer: The Physical Layer describes the physical media access and properties. It breaks up the data it receives from the Data Link Layer into bits of zeros and ones (or “off” and “on” signals). The Physical Layer basically defines the electrical or mechanical interface to the physical medium. It consists of the basic networking hardware transmission technologies. It principally deals with wiring and caballing. The Physical Layer defines the ways of transmitting raw bits over a physical link connecting network nodes including copper wires, fiber-optic cables, optical wavelength, and wireless frequencies. The Physical Layer determines how to put a stream of bits from the Data Link Layer on to the pins for a USB printer interface, an optical fiber transmitter, or a radio carrier. The bit stream may be grouped into code words or symbols and converted to a physical signal that is transmitted over a hardware transmission medium. For instance, it uses +5 volts for sending a bit of 1 and 0 volts for a bit of 0.

**End-to-End View of the OSI Model**

Figure 2.1.5 provides an overview of how devices theoretically communicate in the OSI mode. An application (e.g., Microsoft Outlook on a User A’s computer) produces data targeted to another device on the network (e.g., User B’s computer or a server that User A is getting information from). Each layer in the OSI model adds its own information (i.e., headers, trailers) to the front (or both the front and the end) of the data it receives from the layer above it. Such process is called Encapsulation. For instance, the Transport Layer adds a TCP header, the Network Layer adds an IP header, and the Data Link Layer adds Ethernet header and trailer.

Encapsulated data is transmitted in protocol data units (PDUs): Segments on the Transport Layer, Packets on the Network Layer, and Frames on the Data Link Layer and Bits on the Physical Layer, as was illustrated in Fig. 2.2. PDUs are passed down through the stack of layers until they can be transmitted over the Physical Layer. The Physical Layer then slices the PDUs into bits and transmits the bits over the physical connection that may be wireless/radio link, fiber-optic, or copper cable. +5 volts are often used to transmit 1 s and 0 volts are used to transmit 0 s on copper cables. The Physical Layer provides the physical connectivity between hosts over which all communication occurs. The Physical Layer is the wire connecting both computers on the network. The OSI model ensures that both users speak the same language on the same layer allowing sending and receiving layers (e.g., networking layers) to virtually communicate. Data passed upward is decapsulated before being passed further up. Such process is called decapsulation. Thus, the Physical Layer chops up the PDUs and transmits the PDUs over the physical connection.

**The Things in IoT: Sensors and Actuators**

Things are defined as anything and everything stretching from appliances to buildings to cars to people to animals, to trees, to plants, etc.

****

Fig. 2.1.6: “Thing” in IoT: definition view

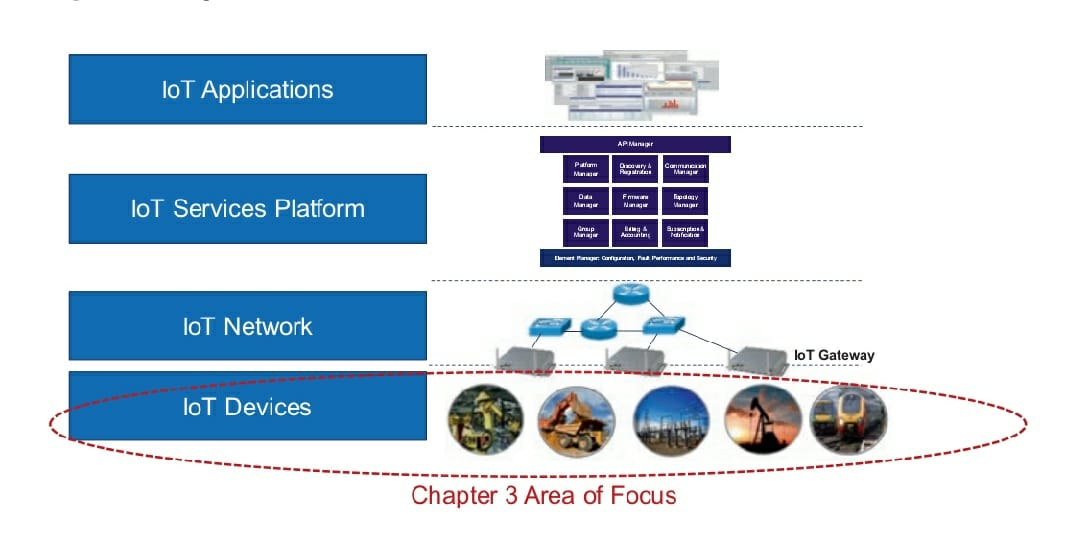
****

Fig. 2.1.7: “Things” in IoT: IoT level view

**IoT Sensors**

**Definition**

A sensor is a device (typically electronic) that detects events or changes in its physical environment (e.g., temperature, sound, heat, pressure, flow, magnetism, motion, chemical and biochemical parameters) and provides a corresponding output. Most sensors take analog inputs and deliver digital, often electrical, outputs. Because the sensing element, on its own, typically produces analog output, an analog-to-digital converter is often required

Sensors are comparable to the human five senses. They form the front end of the IoT devices, i.e., “Things.” Sensors are very crucial in every IoT vertical (e.g., smart cities, smart grid, healthcare, agriculture, security and environment monitoring, and smart parking) as they bridge the world’s physical objects with the Internet. Sensors may be very simple with a core function to collect and transmit data or smart by providing additional functionality to filter duplicate data and only notify the IoT gateway when very specific conditions are met. This requires some programing logic to be present on the sensor itself. In this case, an IoT sensing device requires at least three elements—sensor(s), microcontrollers, and connectivity to send filtered data to IoT gateway or other systems. Figure 3.3 shows the components for smart sensor. Sensors may collect large amounts of data at any time and from any location and transmit it over an IoT network in real time. The data is then analyzed and possibly correlated with other business intelligence databases to provide business insight or enhanced awareness of the environment, bringing onward opportunities and/or gains in efficiency and productivity

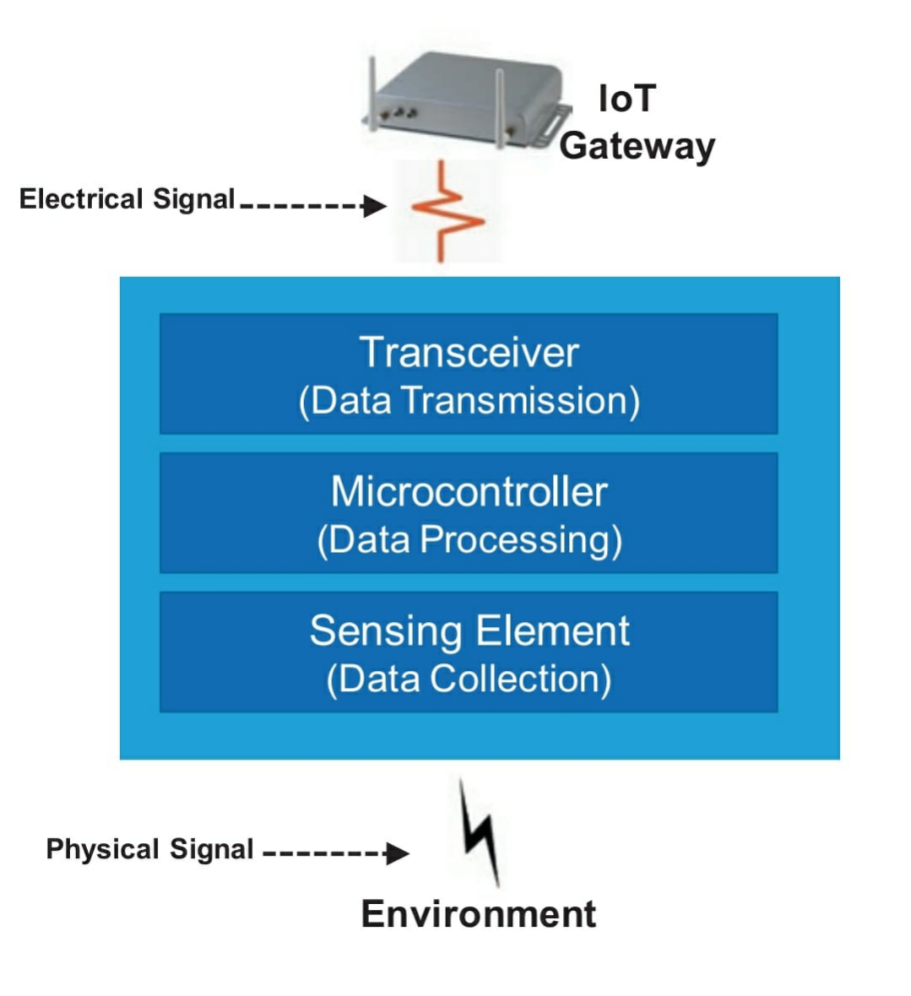
****

Fig. 2.1.7: Components of smart sensors

**Why Sensors**

As we mentioned above, a sensor’s main purpose is collecting data from its surrounding environment and providing output to its adjoining devices (e.g., gateways, actuators) or applications. Sensors typically collect data using physical interfaces (inputs) that sense the environment and then convert input signals into electrical signals (outputs) that are understood by the communication and computing devices. Output signals are then processed by the gateways and/or by applications of the IoT Platform. In some instances, sensors’ outputs are processed directly by a lightweight application.

**Sensor Characteristics**

Most IoT applications require smaller and smarter sensors with advanced functionality to collect more data, low-power processors, longer battery life, faster response time, and shorter time to market. Sensors are expected to be dynamic in their natural surroundings with embedded ability to collect real-time data.

In general, sensors can be either self-directed (autonomous) where they work on their own once they are installed or user-controlled where collection conditions are preprogrammed by the user depending on their needs. Finally, sensors should also have the capability to send the collected data (or a subset of it) to the appropriate system via the IoT gateway as we illustrated in Fig. 3.2.

**IoT sensors are expected to have the following characteristics:**

1. **Data Filtering:** A sensor’s core function is the ability to collect and send data to the IoT gateway or other appropriate systems. Sensors are not expected to perform deep analytical functions. However, simple filtering techniques may be required. Onboard data (or signal) processing microcontroller (as shown in Fig. 3.3) makes a smart sensor smarter. The microcontroller filters the data / signals before transmission to the IoT gateway or control network. It basically removes duplicate or unwanted data or noise before transferring the data.

As we mentioned in Sect. 3.2.3, non-autonomous sensors are customprogrammed to produce alerts automatically when certain conditions are met (e.g., temperature is above 70 °F in a data center). They often integrate VLSI technology and MEMS devices to reduce cost and optimize integration.

1. **Minimum Power Consumption:** Several factors are driving the requirements for low-power consumptions in IoT. Sensors for multiple IoT verticals (e.g., smart grid, railways, and roadsides) will be installed in locations that are difficult to reach to replace batteries.
2. **Compact**: Space will also be limited for most IoT verticals. As such, sensors need to fit in small spaces.
3. **Smart Detection:** An important sensing category for the IoT is remote sensing, which consists of acquiring information about an object without making physical contact with it; the object can be nearby or several hundred meters away. Multiple technology options are available for remote sensing, and they can be divided into three broad functions:

* Presence or proximity detection—when just determining the absence or presence of an object is sufficient (e.g., security applications). This is the simplest form of remote sensing.
* Speed measurement—when the exact position of an object is not required, but accurate speed is (e.g., traffic monitoring applications).
* Detection and ranging—when the position of an object relative to the sensor must be determined precisely and accurately (e.g., vehicle collision avoidance).

1. **High** **Sensitivity**: Sensitivity is generally the ratio between a small change in electrical output signal and a small change in physical signal. It may be expressed as the derivative of the transfer function (the functional relationship between input signal and output signal) with respect to physical signal. Sensitivity indicates how much the output of the device changes with unit change in input (quantity to be measured). For example, if the voltage of a temperature sensor changes by 1 mV for every 1 °C change in temperature, then the sensitivity of the sensor is said to be 1 mV/°C.
2. **Linearity**: Linearity is the measure of the extent to which the output is linearly proportional to the output. Nonlinearity is the maximum deviation from a linear transfer function over the specified dynamic range.
3. **Dynamic** **Range**: The range of input signals which may be converted to electrical signals by the sensor. Outside of this range signals cause unsatisfactory accuracy.
4. **Accuracy**: The maximum expected error between measured (actual) and ideal output signals. Manufacturers often provide the accuracy in the datasheet, e.g., highquality thermometers may list accuracy to within 0.01% of full-scale output.
5. **Hysteresis**: When a sensor does not return the same output value when the input stimulus is driven up or down. The width of the expected error in terms of the measured quantity is defined as the hysteresis.
6. **Limited** **Noise**: All sensors produce some level of noise traffic with their output signals. Sensor noise is only an issue if it impacts the performance of the IoT system. Smart sensors must filter out unwanted noise and be programmed to produce alerts on their own when critical limits are reached. Noise is generally distributed across the frequency spectrum. Many common noise sources produce a white noise distribution, which is to say that the spectral noise density is the same at all frequencies.
7. **Wide** **Bandwidth**: Sensors have finite response times to instantaneous changes in physical signal. Also, many sensors have decay times, which represent the time after a step change in input signal for the sensor output to decay to its original value. The bandwidth of a sensor is the frequency range between these two frequencies. When a sensor is utilized to collect measurements, it is recommended to use sensors with the widest possible bandwidth. This ensures that the basic measurement system is capable of responding linearly over the full range of interest. The disadvantage, however, is that wider bandwidth may result in sensor response to unwanted frequency.
8. **High** **Resolution**: The resolution of a sensor is defined as the smallest detectable signal fluctuation. It is the smallest change in the input that the device can detect. The definition of resolution must include some information about the nature of the measurement being carried out.
9. **Minimum Interruption**: Sensors must operate normally at all time with zero or near-zero interruption and be programmed to produce instant alerts on their own when their normal operation is interrupted.
10. **Higher Reliability:** Higher reliability sensor provides the assurance to rely on the accuracy of the output measurements
11. **Ease of Use:** Ease of use is considered the top requirement for any electronic system nowadays. Clear examples we have all experienced are Apple’s iPhone vs. competitor devices with the same functionality. Users are willing to pay more for easy-to-use devices, and sensors are no exceptions. The best user interface is “no user interface” where sensors are expected to work by themselves once they’re connected.

Other characteristics include some data storage and self-warning of anomalous symptoms.

**IoT Actuators**

An actuator is a type of motor that is responsible for controlling or taking action in a system. It takes a source of data or energy (e.g., hydraulic fluid pressure, other sources of power) and converts the data/energy to motion to control a system.

sensors are responsible to sense changes in their surroundings, collect relevant data, and make such data available to monitoring systems. Collecting and displaying data by a monitoring system is useless unless such data is translated into intelligence that can be used to control or govern an environment before a service is impacted. Actuators use sensor-collected and analyzed data as well as other types of data intelligence (see problem 11) to control IoT systems, for example, shutting down gas flow when the measured pressure is below a certain threshold.

**Controlling IoT Devices**

There are two main philosophies to monitor and control IoT devices: local control and global control. The first approach requires an intelligent local controller (e.g., home’s thermostat to control furnace and air conditioning system). The second approach is to move the control onto the cloud and simply embed inexpensive sensors everywhere (e.g., in this case, thermostat is eliminated altogether), and instead put temperature sensors around the house. An extension of this would be to pull the controller boards out of the furnace and air conditioner—connect their inputs and outputs to the Internet as well, so a cloud application can directly read their states and control their subsystems.

Clearly this approach requires many more, much finer-grained connected devices. And it offers the possibility of control strategies that would not be possible for an isolated thermostat. You could use ambient weather conditions, forecasts, and the current locations of the residents as inputs, for example, to determine an optimum strategy for making life comfortable while saving energy.

We believe the right approach is a combination of the two approaches depending on the specific IoT vertical and environment.

**How Things Are Identified in IoT?**

The most convenient way to identify every IoT devices is to assign unique IP address to each sensor and actuator. However, IPv4 addresses are expensive and limited. IPv6 addresses are not widely deployed yet. In addition, many sensors and actuators are not IP enabled. IoT gateways, however, do have unique IP addresses. Hence, non-IP-enabled sensors and actuators may be identified by their associated gateways.

**CHAPTER 2.2: LabVIEW**

Laboratory Virtual Instrument Engineering Workbench (LabVIEW)is a system-design platform and development environment for a visual programming language from National Instruments.

The name LabVIEW is a shortened form of its description: Laboratory Virtual Instrument Engineering Workbench.

LabVIEW is a visual programming language: it is a system-design platform and development environment that was aimed at enabling all forms of system to be developed.

LabVIEW was developed by National Instruments as a workbench for controlling test instrumentation. However its applications have spread well beyond just test instrumentation to the whole field of system design and operation.

LabVIEW was first launched 1986 as a tool for scientists and engineers to facilitate automated measurements - the aim was that it would be a tool that would be as productive for scientists and engineers as spreadsheets were for financial analysts.

Says Jeff Kodowsky of National Instruments who came up with the initial idea and developed it: ”We weren’t seeking to create a language but that’s what we ended up doing because we needed that level of flexibility and control in order to deal with the kinds of IO and processing required.”

In addition to this, Kodowsky had been using an early Apple Mac which utilised graphics more than any other computing system. Kodowsky wanted to be able to utilise this capability to enable quicker programming of the control for instruments.

LabVIEW offers a graphical programming approach that helps you visualize every aspect of your application, including hardware configuration, measurement data, and debugging. This visualization makes it simple to integrate measurement hardware from any vendor, represent complex logic on the diagram, develop data analysis algorithms, and design custom engineering user interfaces.

The programming paradigm used in LabVIEW, sometimes called G, is based on data availability. If there is enough data available to a subVI or function, that subVI or function will execute. Execution flow is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, LabVIEW can execute inherently in parallel.Multi-processing and multi-threading hardware is exploited automatically by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

LabVIEW integrates the creation of user interfaces (termed front panels) into the development cycle. LabVIEW programs-subroutines are termed virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector pane. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators. Controls are inputs: they allow a user to supply information to the VI. Indicators are outputs: they indicate, or display, the results based on the inputs given to the VI. The back panel, which is a block diagram, contains the graphical source code. All of the objects placed on the front panel will appear on the back panel as terminals. The back panel also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the Functions palette and can be placed on the back panel. Collectively controls, indicators, structures, and functions are referred to as nodes. Nodes are connected to one another using wires, e.g., two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus a virtual instrument can be run as either a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows nonprogrammers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for high-quality G programming. For complex algorithms or large-scale code, it is important that a programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems

offer the ability to build stand-alone applications. Furthermore, it is possible to create distributed applications, which communicate by a client–server model, and are thus easier to implement due to the inherently parallel nature of G.

**Chapter 2.3: LabVIEW based BMS**

The battery management system ensures that whether the a battery is being charged or discharged. Committed electronic systems are installed in the car for safety, convenience, comfort and pleasure. These systems have been greatly added to the demands on the battery in modern electrical cars. LabVIEW based battery monitoring system has been developed and ready to use for any battery powered system to adopt safety, convenience and pleasure. The developed LabVIEW based battery management monitors different parameters of the rechargeable battery. The developed system is very much essential for optimum utilization of the rechargeable battery. This system will be useful for manufacturer and user to understand condition and health of the rechargeable battery so that it is easy to take decision of recharging the battery or replace the battery.

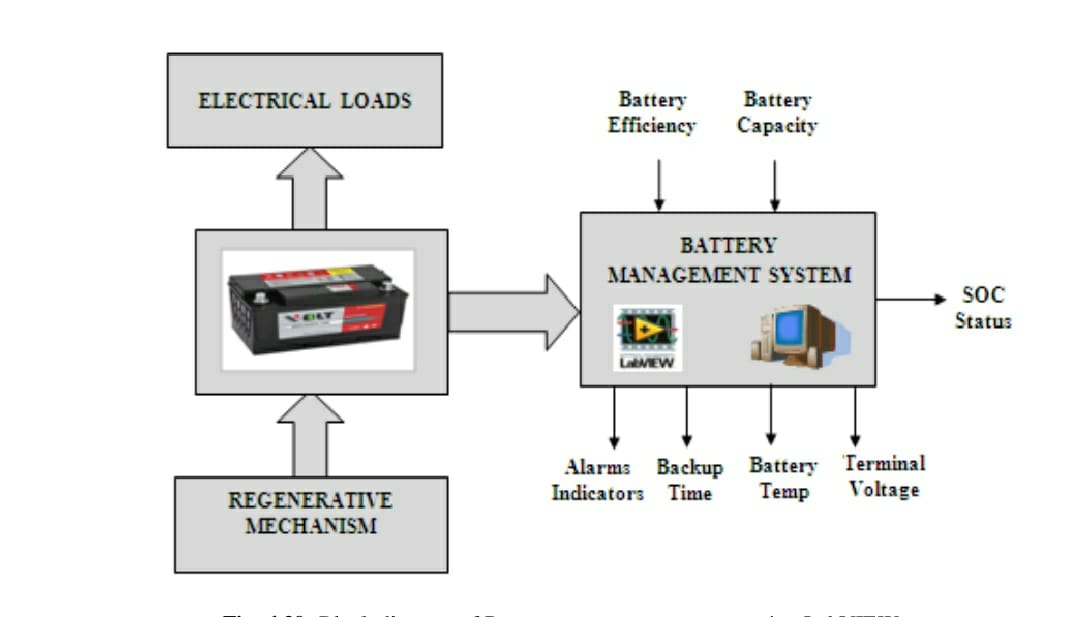
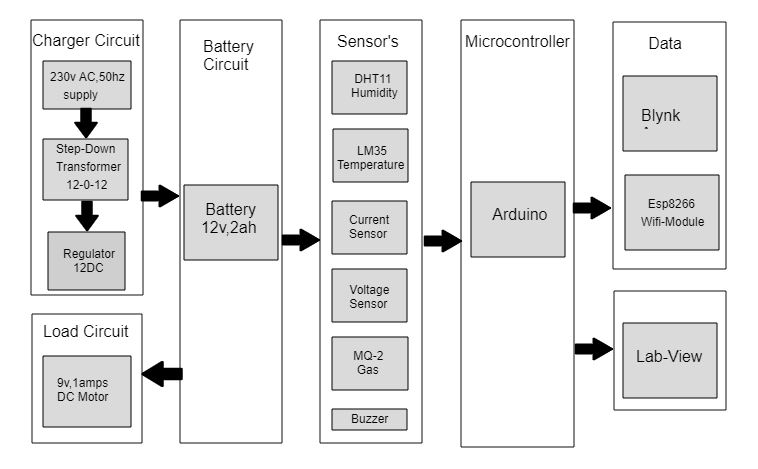


Fig. 2.3.1: Block diagram of Battery management system using LabVIEW

Real-time systems are computer systems that are prone to monitor, respond to, or control external environmental parameters. In this research work of BMS the attempt is made to design a battery management system (BMS) using LabVIEW. This system comprises real time platform and electrical emulation of sensors and actuators for read, process, monitor, control and stores the acquired data for further analysis. This developed system is useful in every battery powered systems and electrical vehicle for fuel status or battery capacity. The data acquisition system acquires battery voltage, battery temperature and discharging current from the battery through developed signal conditioning circuits. This signal conditioning circuits for battery parameters sensing has already experimented in the previous sections of this chapter. The various inputs and outputs sections can be seen in the block diagram of the battery management system. Alarm indicator, backup time, battery temperature, state of charge status and terminal voltage are the outputs of the BMS whereas efficiency and ampere-hour value of the battery are the inputs of the BMS system as shown in figure 4.39. The system user has to feed appropriate information to the system and then system setup will provide proper information to the user.

Hence this regenerative mechanism is controlled by BMS. This regenerative mechanism is not implemented in the current system setup. The developed setup for BMS is given below and different electrical loads are connected to the battery and battery parameters are monitored.

**2.3.Block diagram**

****

2.3.2: BLOCK DIAGRAM OF BMS USING LAB-VIEW

The above diagram shows the basic block diagram of the BMS using Lab-View. Here we are using arduino as the main controller to control the entire system. The main

purpose of this project is to monitor the parameters of the battery to increase its life cycle with less human intervention.

**Chager Circuit:**

A 230v, 50 Hz supply is given as the input to the center tapped step down transformer which generates an 12-0-12v AC signal. The 12v AC signal is given to regulator which converts it into 12v DC signal.

**Load Circuit:**

9 V, 1 Amp DC motor is taken as the load to the battery.

**Battery Circuit:**

A 12 V, 2 Ah battery is being monitored.

**Sensors**

**DHT11 Humdity Sensor:** This sensor is used to obtain humidity level around the battery and that data is provided to arduino.

**LM35 Temperature Sensor:** This sensor is used to obtain temperature of the battery and

that data is provided to arduino.

**Current Sensor:** This sensor is used to obtain current which is drawn out of the battery and that data is provided to arduino.

**Voltage Circuit:**  Voltage divider circuit is used to obtain voltage across the terminals of the battery and that data is provided to arduino.

**MQ-2 Gas Sensor:** It is used to detect gas/smoke around the battery and that data is provided to Arduino.

**Buzzer:** Buzzer is used to alert the system when Smoke or High Humidity is detected.

**Microcontroller**

**Arduino:** An arduino microcontroller is used to control the whole system and the

obtained data is given to App and Lab-View.

**Data:**

**ESP8266:** This is a WiFi module obtains the data from arduino and using internet the data is transmitted to an app which can be in a remote place.

**Blynk App:** This app is used to read the live data at any place and it requires internet for obtaining the data from ESP8266.

**Lab-View:** This is a software tool used to monitor the real time data of the BMS and data from arduino is obtained using USB port.

**CHAPTER 3**

**HARDWARE AND SOFTWARE TOOLS**

**3.1.Hardware tools:**

**Humidity Sensors**

Humidity Sensors are the low cost-sensitive electronic devices used to measure the humidity of the air. These are also known as Hygrometers. Humidity can be measured as Relative humidity, Absolute humidity, and Specific humidity. Based on the type of humidity measured by sensor.

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programs in the OTP memory, which are used by the sensor’s internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users’ request.

**Power and Pin**

DHT11’s power supply is 3-5.5V DC. When power is supplied to the sensor, do not send any instruction to the sensor in within one second in order to pass the unstable status. One capacitor valued 100nF can be added between VDD and GND for power filtering.

**Communication Process:**

Serial Interface (Single-Wire Two-Way) Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms. Data consists of decimal and integral parts. A complete data transmission is 40bit, and the sensor sends higher data bit first. Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

### **Working Principle of Humidity Sensor**

Relative humidity sensors usually contain a humidity sensing element along with a thermistor to measure temperature. For a capacitive sensor, the sensing element is a [capacitor](https://www.elprocus.com/capacitors-types-applications/" \t "https://www.elprocus.com/a-memoir-on-humidity-sensor/_blank). Here the change in electrical permittivity of the dielectric material is measured to calculate the relative humidity values.

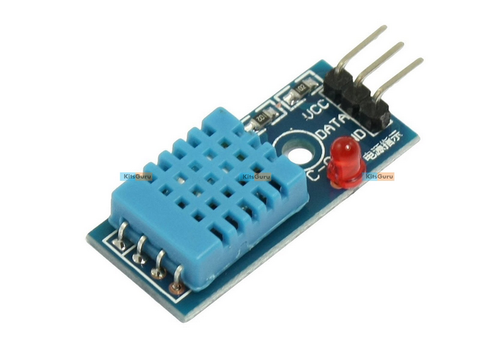
Low resistivity materials are used for the construction of a Resistive sensor. This resistive material is placed on top of two electrodes. Change in the resistivity value of this material is used to measure the change in humidity.

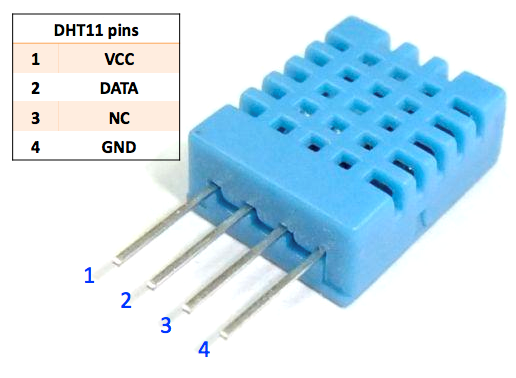
Salt, solid electrolytes and conductive polymers are the examples of resistive material used in Resistive sensor. Thermal conductive sensors measure Absolute humidity values.

**Applications**

The capacitive sensor is used for various applications for measuring humidity in HVAC systems, Printers, Fax machines, Weather stations, automobiles, food processing, refrigerators, etc…

Due to there low cost and small size, resistive sensors are used in residential, industrial and domestic applications. Thermal Conductive sensors are commonly used in pharmaceutical plants, food dehydration, drying machines, etc





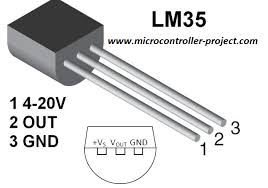
**Specifications:**

|  |  |
| --- | --- |
| Sensor | DHT11 (Temperature & humidity sensor) |
| Operating voltage | 3.3V-5.5V |
| Humility measuring range | 20%-95%（0℃-50℃） |
| Humility measuring error | +-5% |
| Temperature measuring range | 0℃-50℃ |
| Temperature measuring error | +-2℃ |
| Dimensions | 29.0mm\*18.0mm |
| Fixing hole size | 2.0mm |

|  |  |  |
| --- | --- | --- |
| Pin No. | Symbol | Descriptions |
| 1 | DOUT | Communication port |
| 2 | GND | Power ground |
| 3 | VCC | Positive power supply (3.3V-5.5V) |

|  |  |
| --- | --- |
| Port | Arduino pin |
| DOUT | D2 |
| GND | GND |
| VCC | 5V |

**LM 35**



LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of lm35 over thermistor is it does not require any external calibration. The coating also protects it from self-heating. Low cost (approximately $0.95) and greater accuracy make it popular among hobbyists, DIY circuit makers, and students.

**Features**

1• Calibrated Directly in Celsius (Centigrade)

• Linear + 10-mV/°C Scale Factor

• 0.5°C Ensured Accuracy (at 25°C)

• Rated for Full −55°C to 150°C Range

• Suitable for Remote Applications

• Low-Cost Due to Wafer-Level Trimming

• Operates From 4 V to 30 V

• Less Than 60-μA Current Drain

• Low Self-Heating, 0.08°C in Still Air

• Non-Linearity Only ±¼°C Typical

• Low-Impedance Output, 0.1 Ω for 1-mA Load

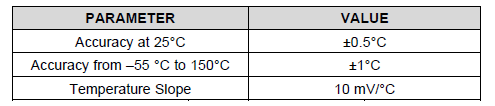
LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of lm35 over thermistor is it does not require any external calibration. The coating also protects it from self-heating. Low cost (approximately $0.95) and greater accuracy make it popular among hobbyists, DIY circuit makers, and students.

#### **LM35 Working Principle (Understanding LM35 Linear Scale Factor)**

[IMG_256](https://b3van8qm1o7ou9d3b48qdhsg-wpengine.netdna-ssl.com/wp-content/uploads/2/2/1/5/22159166/lm35-linear-factor_orig.png)LM35 scale factor

In order to understand the working principle of LM35 temperature sensor we have to understand the linear scale factor. In the features of LM35 it is given to be **+10 mills volt per degree centigrade**. It means that with increase in output of 10 mills volt by the sensor vout pin the temperature value increases by one. For example, if the sensor is outputting 100 mills volt at vout pin the temperature in centigrade will be 10-degree centigrade. The same goes for the negative temperature reading. If the sensor is outputting -100 mills volt the temperature will be -10 degrees Celsius.

**LM35 accuracy level**



**Applications**

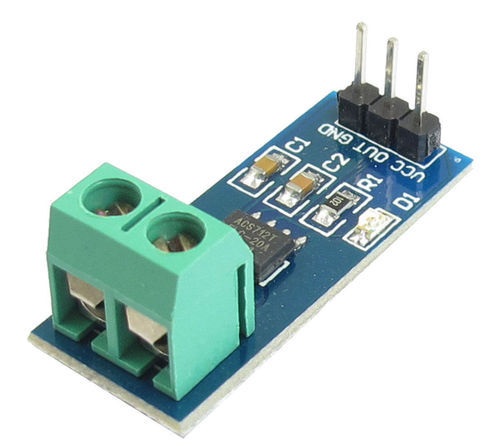
• Power Supplies

• Battery Management

• HVAC

• Appliances

**Current sensor**



Current flowing through a conductor causes a voltage drop. The relation between current and voltage is given by Ohm’s law. In electronic devices, an increase in the amount of current above its requirement leads to overload and can damage the device.

Current flowing through a conductor causes a voltage drop. The relation between current and voltage is given by Ohm’s law. In electronic devices, an increase in the amount of current above its requirement leads to overload and can damage the device.

### **Working Principle**

Current Sensor detects the current in a wire or conductor and generates a signal proportional to the detected current either in the form of analog voltage or digital output. Current Sensor is done in two ways – Direct sensing and Indirect Sensing. In Direct sensing, to detect current, Ohm’s law is used to measure the voltage drop occurred in a wire when current flows through it.  
A current-carrying conductor also gives rise to a magnetic field in its surrounding. In Indirect Sensing, the current is measured by calculating this magnetic field by applying either Faradays law  or Ampere law. Here either a Transformer or Hall effect sensor or fiberoptic current sensor are used to sense the magnetic field.

ACS712 Current Sensor uses Indirect Sensing method to calculate the current. To sense current a liner, low-offset Hall sensor circuit is used in this IC. This sensor is located at the surface of the IC on a copper conduction path. When current flows through this copper conduction path it generates a magnetic field which is sensed by the Hall effect sensor. A voltage proportional to the sensed magnetic field is generated by the Hall sensor, which is used to measure current.

The proximity of the magnetic signal to the Hall sensor decides the accuracy of the device. Nearer the magnetic signal higher the accuracy. ACS712 Current Sensor is available as a small, surface mount SOIC8 package. In this IC current flows from Pin-1 and Pin-2 to Pin-3 and Pin-4. This forms the conduction path where the current is sensed. Implementation of this IC is very easy.

ACS712 can be used in applications requiring electrical isolation as the terminals of the conduction path are electrically isolated from the IC leads. Thus, this IC doesn’t require any other isolation techniques. This IC requires a supply voltage of 5V. Its output voltage is proportional to AC or DC current. ACS712 has a nearly zero magnetic hysteresis.

Where Pin-1 to Pin-4 forms the conduction path, Pin-5 is the signal ground pin. Pin-6 is the FILTER pin that is used by an external capacitor to set the bandwidth. Pin-7 is the analog output pin. Pin-8 is the power supply pin.

**Smoke sensor (MQ06)**



MQ6 Gas Sensor - The MQ6 is a gas sensor module. The module has 4 pins for interfacing of which two pins are VCC and ground, one pin is analog output and one pin is digital pin via a comparator (LM358). The analog output pin of the module is used for detecting concentration level of smoke or gas leakage and interfaced with the A0 analog input pin of the Arduino board. The sensor measures the concentration of leaked gas in ppm according to the following

**Formulae -**

Concen = 1036.5\*R^-2.392 Where

Concen is the concentration of LPG in ppm

R is the ratio of Rs the resistance of sensor to the R0 which is the resistance at 1000ppm at 20 degree Celsius and 65% humidity

The resistance of the sensor Rs is given by the formulae -

Rs = (1024/ADC\_DATA-1)\*RL where

Rs is the resistance of the sensor

ADC\_DATA is digital reading ranging from 0 to 1023

RL is load resistance ranging from 10K to 40K ohms

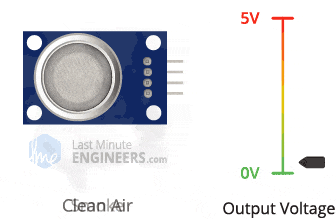
**Arduino Based Smoke Detector Circuit Working**

When the device is powered on, the Arduino initializes the sensor module and the LCD display. It starts reading data from the MQ-06 sensor. The data is read from the analog output pin of the sensor. The read data is in the form of an analog voltage which is digitized using inbuilt ADC channel. The ADC channels on the Arduino board are 10-bit long so the digitized reading varies from 0 to 1023. On calibration, the ADC reading is directly proportional by a factor of one to the concentration of smoke in PPM. The reading is stored in a variable and displayed on the LCD display.

The reading is compared with a calibrated threshold representing the dangerous level of smoke detection. If the concentration of smoke in PPM exceeds the threshold value, the Arduino board activates the alarm indicator in the form of LED  by sending a HIGH pulse at the respective controller pin and activates the DC motor used as exhaust. Otherwise, until the concentration of smoke in PPM is not beyond the dangerous limit,

the LED is kept OFF by passing a LOW at the respective controller pin and motor is kept in stop condition by passing LOW at the pins controlling motor input signals.

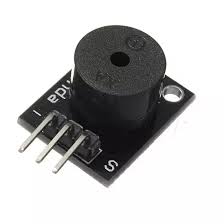
Check out the project code to learn how the portable Arduino board reads data from the MQ-06 sensor, store sensor data, display it on the LCD screen, compare it with a threshold value and activate motor as well as LED indicator on detecting the dangerous level of smoke concentration.



**Specifications**

|  |  |
| --- | --- |
| Operating voltage | 5V |
| Load resistance | 20 KΩ |
| Heater resistance | 33Ω ± 5% |
| Heating consumption | <800mw |
| Sensing Resistance | 10 KΩ – 60 KΩ |
| Concentration Scope | 200 – 10000ppm |
| Preheat Time | Over 24 hour |

**Buzzer**



A piezo buzzer is a type of electronic device that’s used to produce a tone, alarm or sound. It’s lightweight with a simple construction, and it’s typically a low-cost product. Yet at the same time, depending on the piezo ceramic buzzer specifications, it’s also reliable and can be constructed in a wide range of sizes that work across varying frequencies to produce different sound outputs.

The use of the piezo ceramic buzzer was discovered thanks to an inversion of the piezoelectricity principle that was discovered by Jacques and Pierre Curie back in 1880. They found that electricity could be generated when a mechanical pressure was applied to particular materials — and the inverse was true as well.

So when certain piezoelectric materials are subjected to an alternating field of electricity, the piezo buzzer element — often a manmade piezoceramic material — stretches and compresses in sequence with the frequency of the current. As a result, it produces an audible sound.

Unlike magnetic buzzers that have a narrow operating voltage of somewhere between one and 16 volts, piezo buzzers can typically operate anywhere between three and 250 volts. In addition, magnetic buzzers have a higher power consumption of 30 to 100 milliamperes, while piezo buzzers normally consume less than 30 milliamperes — even at higher rate frequencies. And although piezo buzzers require a larger footprint than magnetic buzzers, they produce a higher sound pressure level.

**Specifications**

* Rated **Voltage**: 6V DC.
* Operating **Voltage**: 4-8V DC.
* Rated current: <30mA.
* Sound Type: Continuous Beep.
* Resonant Frequency: ~2300 Hz.

#### **Typical Applications of Buzzer**

* alarms / warning devices / automobile alarms
* pest deterrents
* computer devices
* telephones
* toys / games

**Battery**

A **rechargeable battery**, **storage battery**, or **secondary cell**, (or archaically **accumulator**) is a type of [electrical battery](https://en.wikipedia.org/wiki/Electrical_battery" \o "Electrical battery) which can be charged, discharged into a load, and recharged many times, as opposed to a disposable or [primary battery](https://en.wikipedia.org/wiki/Primary_battery" \o "Primary battery), which is supplied fully charged and discarded after use. It is composed of one or more [electrochemical cells](https://en.wikipedia.org/wiki/Electrochemical_cell" \o "Electrochemical cell). The term "accumulator" is used as it [accumulates](https://en.wikipedia.org/wiki/Accumulator_(energy)" \o "Accumulator (energy)) and [stores energy](https://en.wikipedia.org/wiki/Energy_storage" \o "Energy storage) through a reversible [electrochemical](https://en.wikipedia.org/wiki/Electrochemical" \o "Electrochemical) [reaction](https://en.wikipedia.org/wiki/Chemical_reaction" \o "Chemical reaction). Rechargeable batteries are produced in many different shapes and sizes, ranging from [button cells](https://en.wikipedia.org/wiki/Button_cell" \l "Rechargeable_variants" \o "Button cell) to megawatt systems connected to [stabilize](https://en.wikipedia.org/wiki/Grid_energy_storage" \o "Grid energy storage) an [electrical distribution network](https://en.wikipedia.org/wiki/Electrical_distribution_network" \o "Electrical distribution network).



**Depth of discharge**

Depth of discharge (DOD) is normally stated as a percentage of the nominal ampere-hour capacity; 0% DOD means no discharge. As the usable capacity of a battery system depends on the rate of discharge and the allowable voltage at the end of discharge, the depth of discharge must be qualified to show the way it is to be measured. Due to variations during manufacture and aging, the DOD for complete discharge can change over time or number of [charge cycles](https://en.wikipedia.org/wiki/Charge_cycle" \o "Charge cycle). Generally a rechargeable battery system will tolerate more charge/discharge cycles if the DOD is lower on each cycle.

### **Lifespan and cycle stability**

If batteries are used repeatedly even without mistreatment, they lose capacity as the number of charge cycles increases, until they are eventually considered to have reached the end of their useful life. Different battery systems have differing mechanisms for wearing out. For example, in lead-acid batteries, not all the active material is restored to the plates on each charge/discharge cycle; eventually enough material is lost that the battery capacity is reduced. In lithium-ion types, especially on deep discharge, some reactive lithium metal can be formed on charging, which is no longer available to participate in the next discharge cycle. Sealed batteries may lose moisture from their liquid electrolyte, especially if overcharged or operated at high temperature.

### **Recharging time**

Recharging time is an important parameter to the user of a product powered by rechargeable batteries. Even if the charging power supply provides enough power to operate the device as well as recharge the battery, the device is attached to an external power supply during the charging time. For electric vehicles used industrially, charging during off-shifts may be acceptable.

A rechargeable battery cannot be recharged at an arbitrarily high rate. The internal resistance of the battery will produce heat, and excessive temperature rise will damage or destroy a battery. For some types, the maximum charging rate will be limited by the speed at which active material can diffuse through a liquid electrolyte. High charging rates may produce excess gas in a battery, or may result in damaging side reactions that permanently lower the battery capacity. Very roughly, and with many exceptions and details, restoring a battery's full capacity in one hour or less is considered fast charging.

### **Damage during storage in fully discharged state**

If a multi-cell battery is fully discharged, it will often be damaged due to the cell reversal effect mentioned above. It is possible however to fully discharge a battery without causing cell reversal—either by discharging each cell separately, or by allowing each cell's internal leakage to dissipate its charge over time.

Even if a cell is brought to a fully discharged state without reversal, however, damage may occur over time simply due to remaining in the discharged state. An example of this is the [sulfation that occurs in lead-acid batteries](https://en.wikipedia.org/wiki/Desulfation" \o "Desulfation) that are left sitting on a shelf for long periods. For this reason it is often recommended to charge a battery that is intended to remain in storage, and to maintain its charge level by periodically recharging it. Since damage may also occur if the battery is overcharged, the optimal level of charge during storage is typically around 30% to 70%.

**Charging and Discharging**

During charging, the positive active material is [oxidized](https://en.wikipedia.org/wiki/Oxidized" \o "Oxidized), producing [electrons](https://en.wikipedia.org/wiki/Electron" \o "Electron), and the negative material is [reduced](https://en.wikipedia.org/wiki/Redox" \o "Redox), consuming electrons. These electrons constitute the [current](https://en.wikipedia.org/wiki/Electric_current" \o "Electric current) flow in the external [circuit](https://en.wikipedia.org/wiki/Electrical_network" \o "Electrical network). The [electrolyte](https://en.wikipedia.org/wiki/Electrolyte" \o "Electrolyte) may serve as a simple buffer for internal [ion](https://en.wikipedia.org/wiki/Ion" \o "Ion) flow between the [electrodes](https://en.wikipedia.org/wiki/Electrode" \o "Electrode), as in [lithium-ion](https://en.wikipedia.org/wiki/Lithium-ion_battery" \o "Lithium-ion battery) and [nickel-cadmium](https://en.wikipedia.org/wiki/Nickel-cadmium_battery" \o "Nickel-cadmium battery) cells, or it may be an active participant in the [electrochemical](https://en.wikipedia.org/wiki/Electrochemical" \o "Electrochemical) reaction, as in [lead–acid](https://en.wikipedia.org/wiki/Lead%E2%80%93acid_battery" \o "Lead–acid battery) cells.

The energy used to charge rechargeable batteries usually comes from a [battery charger](https://en.wikipedia.org/wiki/Battery_charger" \o "Battery charger) using AC [mains electricity](https://en.wikipedia.org/wiki/Mains_electricity" \o "Mains electricity), although some are equipped to use a vehicle's 12-volt DC power outlet. The voltage of the source must be higher than that of the battery to force current to flow into it, but not too much higher or the battery may be damaged.

Chargers take from a few minutes to several hours to charge a battery. Slow "dumb" chargers without voltage or temperature-sensing capabilities will charge at a low rate, typically taking 14 hours or more to reach a full charge. Rapid chargers can typically charge cells in two to five hours, depending on the model, with the fastest taking as little as fifteen minutes. Fast chargers must have multiple ways of detecting when a cell reaches full charge (change in terminal voltage, temperature, etc.) to stop charging before harmful overcharging or overheating occurs. The fastest chargers often incorporate cooling fans to keep the cells from overheating. Battery packs intended for rapid charging may include a temperature sensor that the charger uses to protect the pack; the sensor will have one or more additional electrical contacts.

Different battery chemistries require different charging schemes. For example, some battery types can be safely recharged from a constant voltage source. Other types need to be charged with a regulated current source that tapers as the battery reaches fully charged voltage. Charging a battery incorrectly can damage a battery; in extreme cases, batteries can overheat, catch fire, or explosively vent their contents

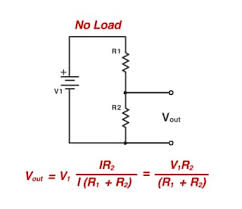
**Regulator**

A **voltage regulator** is a system designed to automatically maintain a constant voltage level. A voltage regulator may use a simple feed-forward design or may include negative feedback. It may use an electromechanical mechanism, or electronic components. Depending on the design, it may be used to regulate one or more AC or DC voltages.

Electronic voltage regulators are found in devices such as [computer power supplies](https://en.wikipedia.org/wiki/Power_supply_unit_(computer)" \o "Power supply unit (computer)) where they stabilize the DC voltages used by the processor and other elements. In [automobile alternators](https://en.wikipedia.org/wiki/Alternator_(automotive)" \o "Alternator (automotive)) and central power station generator plants, voltage regulators control the output of the plant. In an electric power distribution system, voltage regulators may be installed at a substation or along distribution lines so that all customers receive steady voltage independent of how much power is drawn from the line.

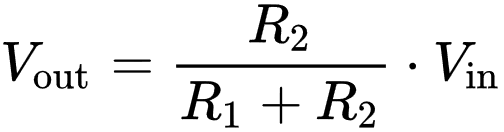
**Voltage divider**

In [electronics](https://en.wikipedia.org/wiki/Electronics" \o "Electronics), a **voltage divider**(also known as a **potential divider**) is a [passive](https://en.wikipedia.org/wiki/Passive_circuit" \o "Passive circuit) [linear circuit](https://en.wikipedia.org/wiki/Linear_circuit" \o "Linear circuit) that produces an output [voltage](https://en.wikipedia.org/wiki/Voltage" \o "Voltage) (*V*out) that is a fraction of its input voltage (*V*in). **Voltage division** is the result of distributing the input voltage among the components of the divider. A simple example of a voltage divider is two [resistors](https://en.wikipedia.org/wiki/Resistor" \o "Resistor) connected in [series](https://en.wikipedia.org/wiki/Series_and_parallel_circuits" \o "Series and parallel circuits), with the input voltage applied across the resistor pair and the output voltage emerging from the connection between them.

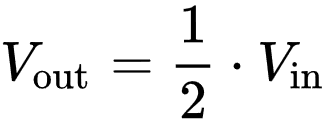


A resistive divider is the case where both impedances, Z1 and Z2, are purely resistive (Figure 2).

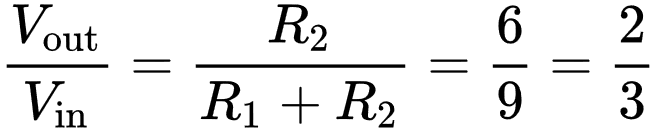
Substituting Z1 = R1 and Z2 = R2 into the previous expression gives:

{\displaystyle V\_{\mathrm {out} }={\frac {R\_{2}}{R\_{1}+R\_{2}}}\cdot V\_{\mathrm {in} }}

If *R*1 = *R*2 then

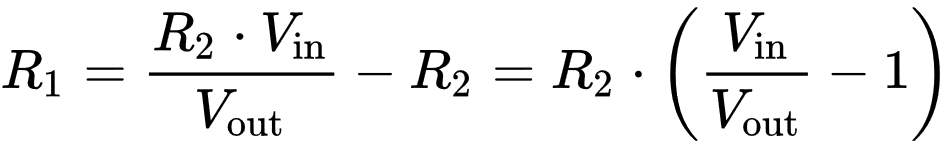
{\displaystyle V\_{\mathrm {out} }={\frac {1}{2}}\cdot V\_{\mathrm {in} }}

If *V*out = 6V and *V*in = 9V (both commonly used voltages), then:

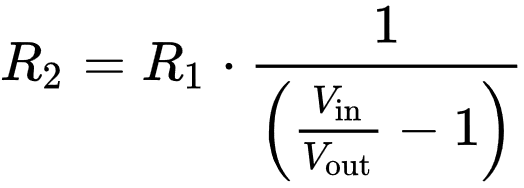
{\displaystyle {\frac {V\_{\mathrm {out} }}{V\_{\mathrm {in} }}}={\frac {R\_{2}}{R\_{1}+R\_{2}}}={\frac {6}{9}}={\frac {2}{3}}}

and by solving using [algebra](https://en.wikipedia.org/wiki/Algebra" \o "Algebra), *R*2 must be twice the value of *R*1.

To solve for R1:

{\displaystyle R\_{1}={\frac {R\_{2}\cdot V\_{\mathrm {in} }}{V\_{\mathrm {out} }}}-R\_{2}=R\_{2}\cdot \left({{\frac {V\_{\mathrm {in} }}{V\_{\mathrm {out} }}}-1}\right)}

To solve for R2:

{\displaystyle R\_{2}=R\_{1}\cdot {\frac {1}{\left({{\frac {V\_{\mathrm {in} }}{V\_{\mathrm {out} }}}-1}\right)}}}

Any ratio *V*out/*V*in greater than 1 is not possible. That is, using resistors alone it is not possible to either invert the voltage or increase *V*out above *V*in

**Applications:**

Voltage dividers are used for adjusting the level of a signal, for bias of active devices in amplifiers, and for measurement of voltages. A [Wheatstone bridge](https://en.wikipedia.org/wiki/Wheatstone_bridge" \o "Wheatstone bridge) and a [multimeter](https://en.wikipedia.org/wiki/Multimeter" \o "Multimeter) both include voltage dividers. A [potentiometer](https://en.wikipedia.org/wiki/Potentiometer" \o "Potentiometer) is used as a variable voltage divider in the volume control of many radios

**Dc motor**

A **DC motor** is any of a class of rotary [electrical motors](https://en.wikipedia.org/wiki/Electrical_motor" \o "Electrical motor) that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The [universal motor](https://en.wikipedia.org/wiki/Universal_motor" \o "Universal motor) can operate on direct current but is a lightweight [brushed](https://en.wikipedia.org/wiki/Brush_(electric)" \o "Brush (electric)) motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of [power electronics](https://en.wikipedia.org/wiki/Power_electronics" \o "Power electronics) has made replacement of DC motors with [AC motors](https://en.wikipedia.org/wiki/AC_motors" \o "AC motors) possible in many applications



**Specifications**

* Standard 130 Type **DC motor**.
* Operating Voltage: 4.5V to 9V.
* Recommended/Rated Voltage: 6V.
* Current at No load: 70mA (max)
* No-load Speed: 9000 rpm.
* Loaded current: 250mA (approx)
* Rated Load: 10g\*cm

Application

The **applications** of electrical **motor** mainly include blowers, fans, machine tools, pumps, turbines, power tools, alternators, compressors, rolling mills, ships, movers, paper mills.

**3.2. Software Tools**

**LabVIEW**

Laboratory Virtual Instrument Engineering Workbench (LabVIEW)is a system-design platform and development environment for a visual programming language from National Instruments.

The name LabVIEW is a shortened form of its description: Laboratory Virtual Instrument Engineering Workbench.

LabVIEW is a visual programming language: it is a system-design platform and development environment that was aimed at enabling all forms of system to be developed.

LabVIEW was developed by National Instruments as a workbench for controlling test instrumentation. However its applications have spread well beyond just test instrumentation to the whole field of system design and operation.

LabVIEW was first launched 1986 as a tool for scientists and engineers to facilitate automated measurements - the aim was that it would be a tool that would be as productive for scientists and engineers as spreadsheets were for financial analysts.

Says Jeff Kodowsky of National Instruments who came up with the initial idea and developed it: ”We weren’t seeking to create a language but that’s what we ended up doing because we needed that level of flexibility and control in order to deal with the kinds of IO and processing required.”

In addition to this, Kodowsky had been using an early Apple Mac which utilised graphics more than any other computing system. Kodowsky wanted to be able to utilise this capability to enable quicker programming of the control for instruments.

LabVIEW offers a graphical programming approach that helps you visualize every aspect of your application, including hardware configuration, measurement data, and debugging. This visualization makes it simple to integrate measurement hardware from any vendor, represent complex logic on the diagram, develop data analysis algorithms, and design custom engineering user interfaces.

The programming paradigm used in LabVIEW, sometimes called G, is based on data availability. If there is enough data available to a subVI or function, that subVI or function will execute. Execution flow is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, LabVIEW can execute inherently in parallel.Multi-processing and multi-threading hardware is exploited automatically by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

LabVIEW integrates the creation of user interfaces (termed front panels) into the development cycle. LabVIEW programs-subroutines are termed virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector pane. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators. Controls are inputs: they allow a user to supply information to the VI. Indicators are outputs: they indicate, or display, the results based on the inputs given to the VI. The back panel, which is a block diagram, contains the graphical source code. All of the objects placed on the front panel will appear on the back panel as terminals. The back panel also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the Functions palette and can be placed on the back panel. Collectively controls, indicators, structures, and functions are referred to as nodes. Nodes are connected to one another using wires, e.g., two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus a virtual instrument can be run as either a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows nonprogrammers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for high-quality G programming. For complex algorithms or large-scale code, it is important that a programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the ability to build stand-alone applications. Furthermore, it is possible to create distributed applications, which communicate by a client–server model, and are thus easier to implement due to the inherently parallel nature of G.

## **LabVIEW key concepts**

Within LabVIEW there are several elements and concepts that are key to the format and operation of the environment. These include:

* *LabVIEW environment:*   The LabVIEW environment consists of LabVIEW VI manager (project explorer), the programming tools, debugging features, templates and ready built sample examples, and an easy interface to the hardware drivers. *Read more about*[LabVIEW](https://www.electronics-notes.com/articles/test-methods/labview/environment.php)[environment](https://www.electronics-notes.com/articles/test-methods/labview/environment.php)*[.](https://www.electronics-notes.com/articles/test-methods/labview/environment.php)*
* *LabVIEW VIs:*   The LabVIEW VI is a “Virtual Instrument” that enables a user interface to be built and it contains the programming code. *Read more about*[LabVIEW](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)[Virtual](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)[Instruments](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)*[,](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)* [VIs](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)*[.](https://www.electronics-notes.com/articles/test-methods/labview/vis-virtual-instruments.php)*
* *LabVIEW G programming:*   This is the graphical programming language where the functional algorithms are built using “drag and drop” techniques. *Read more about*[LabVIEW](https://www.electronics-notes.com/articles/test-methods/labview/g-programming.php)[programming](https://www.electronics-notes.com/articles/test-methods/labview/g-programming.php)*[.](https://www.electronics-notes.com/articles/test-methods/labview/g-programming.php)*
* *LabVIEW dataflow :*   This is the core concept that determines the running order for the programme.

## **LabVIEW environment tools**

Within the LabVIEW environment there is a good selection of tools to assist the development of the overall programmes.

These tools include:

* ***Templates & sample projects:***   This is a tool that features some of the common templates that can be used for starting projects along with some fully formed projects that can be used to see how projects may work, or to be customised for particular applications.
* ***Example finder:***   This has many hundreds of code snippets that illustrate the methodology behind a huge variety of different tasks. These can be used or customised as appropriate.
* ***Debugging tools:***   As with any project, debugging is an essential element of any development. The LabVIEW environment provides some comprehensive tools to assist with the speedy debug of programmes. The tools include elements like probes, breakpoints,, single step, highlighting execution (the ability to slow execution to see how the system operates and thereby isolate issues in a dynamic manner).
* ***Hardware manager(Measurement & Automation Explorer):***   This is a complementary tool to LabVIEW. It enables drivers to be managed – updated and monitored. It also allows the monitoring of validity of the connection to hardware before running the programme.
* ***Source control:***   The LabVIEW environment has several tools to provide configuration management. This enables larger teams to work together on an applications whilst still retaining proper configuration management to be undertaken.
* ***LabVIEW tools network:***   This element of the LabVIEW environment enables the downloading of specialised toolkits for specific applications. Some are provided by National Instruments, whereas others may be provided by NI partners.

## **LabVIEW VIs**

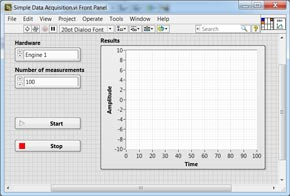
The Virtual Instrument or VI is an integral element of the LabVIEW environment. VIs are individual code modules that make up a complete application.

An application could be as simple as a single VI, but normally many more are included and some applications may utilise hundred of VIs or possibly more dependent upon the particular application.

## **LabVIEW VI basics**

The LabVIEW VI consists of two main elements:

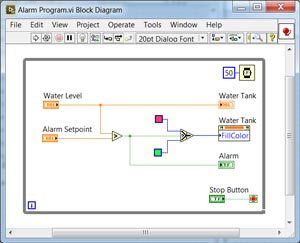
* ***VI Front Panel:***   The LabVIEW front panel is what the user of the completed application will see. It enables them to interact with the VI, inputting controls and also seeing results. It can be likened to the font panel of a test instrument or other piece of equipment.  
    
  The LabVIEW VI front panel can be built up from scratch using the palette of different controls, indicators and data types.



Example of a LabVIEW VI Front Panel

LabVIEW VI front panels can be completely customized. By having a totally customized front panel, it is possible to simplify the operation to provide exactly what is needed without unnecessary controls that may not be required.

* ***VI block diagram:***   The LabVIEW VI block diagram is where the functionality of the VI is programmed in G. The block diagram defines the functionality whilst also providing a visual representation of it. In this way the block diagram is similar to a flow diagram within a programme.



Example of a LabVIEW VI Block Diagram

There is an associated functions palette within the block diagram space where all the elements needed to build the programme can be found. This enables swift accurate programming to be achieved.

LabVIEW programming utilises graphical techniques and in this way, it is much easier to build representations that can be viewed in a similar way to the actual flow of the process.

Like any other language, LabVIEW programming requires a learning process and practice to achieve the maximum effectiveness and speed.

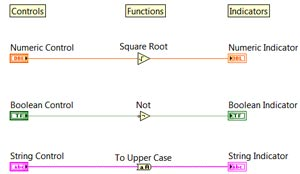
Nevertheless it is possible to achieve quick results with the knowledge of two main concepts: G programming (block diagram elements); and dataflow.

## **LabVIEW programming elements**

LabVIEW programming is undertaken on the block diagram of the Virtual Instrument. G programming is a technical name for the LabVIEW programming language but nowadays the term is largely unused and the name LabVIEW has become to mean the language as well as the software itself.

As with all programming languages, there are inputs, actions, and outputs. In LabVIEW programming these are known as Controls, Functions and Indicators.

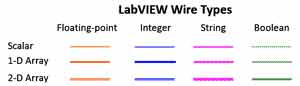
* ***Controls:***   Any LabVIEW control on the front panel will have a corresponding element on the block diagram. The user can input data into the control for use within the overall programme. It is possible to connect the controls to a function to perform a particular action. Controls can come if different data types: single, double, string, etc . . the standard programming data types.  
    
  The Controls palette can be accessed from the front panel window by selecting View » Controls Palette. Alternatively it can be accessed by right clicking on any empty space in the front panel window.  
    
  The Controls palette for LabVIEW programming is broken into various categories – these can be exposed as required to show some or all of these categories to suit the requirements for the application.
* ***Functions:***   LabVIEW functions are taken from the Functions palette on the block diagram and are given inputs and they perform an action on this. LabVIEW has a huge range of different functions ranging from simple mathematics to video processing, spectral analysis and the like.
* ***Indicators:***   LabVIEW indicators are similar to controls, having a Front Panel Counterpart in which they display the output of the block diagram to the user.



LabVIEW programming: controls; indicators; functions

Within the block diagram all the LabVIEW programming elements, i.e. controls, indicators and functions, are connected together. This is achieved using “wires.” The data can be considered to flow along these wires.

There are different wire types which are indicated by the colour and style of the representation.



LabVIEW programming wires

Each wire has a single data source, but it is possible to wire it to many VIs and functions that read the data. Wires are different colors, styles, and thicknesses, depending on their data types as shown above. They may be:

* Numeric integer (Blue)
* Numeric floating point (Orange)
* Boolean (Green)
* String (Pink)

The appearance of the wire indicates whether it is scalar, a 1D array, or a 2D array.

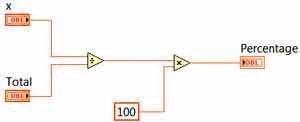
On a LabVIEW screen, a broken wire appears as a dashed black line with a red X in the middle. Broken wires occur for a variety of reasons. One common reason is when wiring two objects with incompatible data types.

## **LabVIEW dataflow & programming**

With text based programming the order of execution is set up by the order of the lines – they are executed in a sequential manner. Examples of these languages include Visual Basic, C++, Java and many others.

With graphical programming it is set by the dataflow within the diagram. Within this concept a function is not able to execute until it has received all its inputs. Once it has all of its inputs, it executes its functions and passes on its output to the next node.

In the diagram below data flows from left to right and this means that the multiplication function cannot execute until the divide function has completed. Therefore the order of execution has been set. It should be noted that the execution follows the actual dataflow and not the position within the window.



LabVIEW programming dataflow example

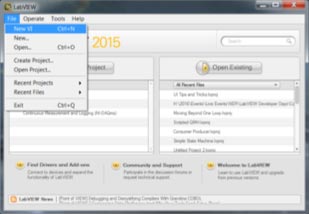
The concept of dataflow within LabVIEW may take those who are more used to text based programmes a little while to master, but once this has been done, it is easy to use as a programming language.

LabVIEW is able to control devices and also receive responses from them. To achieve this, an interface is required enabling LabVIEW to connect to the device or instrument.

These interfaces are called LabVIEW drivers and there is an extensive library of these drivers that are available within the LabVIEW ecosystem.

## **Opening LabVIEW**

With the computer turned on and booted up ready to go, the first stage in the tutorial or demonstration is to open LabVIEW, and then select File >> New VI.



LabVIEW opening page

This will open up a new LabVIEW VI or Virtual Instrument. Unlike traditional instruments, a VI doesn’t have fixed functionality meaning it can adapt as a project expands.

The front panel and block diagram of the VI will open. The front panel has a grey background and is where the user interface of the VI is created. The block diagram is white and is where the code that defines the application is added.

## **Add graph to front panel**

The next step in this tutorial is to put a graph on the front panel that can show the acquired data. To do this, right-click on the front panel to bring up the Controls Palette. Select Silver >> Graph >> Waveform Graph (Silver) and place the item on the front panel.

The Controls Palette has many click and drop controls and indicators that can be used to quickly build a user interface. This is a huge advantage over text-based languages where user interfaces can be very time consuming.



LabVIEW screen showing VI Front Panel Graph

## **Select block diagram**

The next step in this LabVIEW tutorial and demonstration is to select the block diagram. There is now a ‘terminal’ for the waveform graph that allows data to be wired in to show on the front panel. To get data to display in the graph a LabVIEW function is needed to read data from the USB data acquisition (DAQ) device.

To do this, right-click on the block diagram to open the Functions Palette and select Measurement I/O >> NI-DAQmx >> DAQ Assistant to put the function on the block diagram. DAQmx is a hardware driver and there are hundreds of third party drivers available for LabVIEW.



LabVIEW Block Diagram DAQ Assistant

The DAQ assistant is a configuration-based function. This means that a configuration wizard will appear to select the data channels when the function is placed on the block diagram. These configuration-based functions are called Express VIs and they are used to speed up development time for simple programs.

The first configuration window allows the type of measurement to be selected. The second window shows the hardware and channels that can be selected. Simple hardware integration is another advantage of LabVIEW.

**Benefits Of using LabVIEW**

### **Interfacing to devices**

LabVIEW includes extensive support for interfacing to devices, instruments, camera, and other devices. Users interface to hardware by either writing direct bus commands (USB, GPIB, Serial) or using high-level, device-specific, drivers that provide native LabVIEW function nodes for controlling the device.

LabVIEW includes built-in support for NI hardware platforms such as CompactDAQ and CompactRIO, with a large number of device-specific blocks for such hardware, the *Measurement and Automation eXplorer* (MAX) and *Virtual Instrument Software Architecture* (VISA) toolsets.

National Instruments makes thousands of device drivers available for download on the NI Instrument Driver Network (IDNet).

### **Code compiling**

LabVIEW includes a compiler that produces native code for the CPU platform. The graphical code is converted into Dataflow Intermediate Representation, and then translated into chunks of executable machine code by a compiler based on LLVM. Run-time engine calls these chunks, allowing better performance. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single binary file. The execution is controlled by LabVIEW run-time engine, which contains some pre-compiled code to perform common tasks that are defined by the G language. The run-time engine governs execution flow, and provides a consistent interface to various operating systems, graphic systems and hardware components. The use of run-time environment makes the source code files portable across supported platforms. LabVIEW programs are slower than equivalent compiled C code, though like in other languages, program optimization often allows to mitigate issues with execution speed.

### **Large libraries**

Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous for functions such as integration, filters, and other specialized abilities usually associated with data capture from hardware sensors is enormous. In addition, LabVIEW includes a text-based programming component named MathScript with added functions for signal processing, analysis, and mathematics. MathScript can be integrated with graphical programming using *script nodes* and uses a syntax that is compatible generally with MATLAB.

### **Parallel programming**

LabVIEW is an inherently concurrent language, so it is very easy to program multiple tasks that are performed in parallel via multithreading. For example, this is done easily by drawing two or more parallel while loops and connecting them to two separate nodes. This is a great benefit for test system automation, where it is common practice to run processes like test sequencing, data recording, and hardware interfacing in parallel.

### **Ecosystem**

Due to the longevity and popularity of the LabVIEW language, and the ability for users to extend its functions, a large ecosystem of third party add-ons has developed via contributions from the community. This ecosystem is available on the LabVIEW Tools Network, which is a marketplace for both free and paid LabVIEW add-ons.

### **User community**

There is a low-cost LabVIEW Student Edition aimed at educational institutions for learning purposes. There is also an active community of LabVIEW users who communicate through several electronic mailing lists (email groups) and Internet forums.

### **Home Bundle Edition**

National Instruments provides a low cost LabVIEW Home Bundle Edition.

### Free for Non-Commercial use, Edition

National Instruments provides a free for non-commercial use version called LabVIEW Community Edition. This version includes everything in the Professional Editions of LabVIEW, has no watermarks, and includes the LabVIEW NXG Web Module for non-commercial use. These editions may also be used by K-12 schools.

## **LabVIEW applications**

LabVIEW provides a powerful platform for undertaking a wide variety of different applications. It started as an environment for managing test programming, but since its inception, the applications for which it can be used have considerably expanded. It has expanded from being a graphical test management language to become a graphical system design environment.

This means that it can be used for an enormous variety of interesting and diverse applications. Not only can it be used for equipment control (including the control of the large Hadron Collider at CERN) and a variety of data acquisition applications (including car development simulation where Big Data monitoring is undertaken) to the system design arena where it has been used for development of projects from RF circuitry to biomedical equipment, green technology and much more.

## **LabVIEW advantages / disadvantages**

Like any product or platform, LabVIEW has its advantages and disadvantages. These must be carefully considered before starting its use.

**LabVIEW advantages**

* Graphical interface is flexible and simple to use. Most engineers and scientists can learn to use it quickly.
* LabVIEW provides a universal platform for numerous applications in diverse fields.
* LabVIEW can be used with 3rd party hardware: it can be interfaced with C/C++, VB, Fortran etc etc.
* Easy to interface to many hardware items like data acquisition and test equipment products.
* It has excellent customer support and a large active community forum.

**LabVIEW disadvantages**

* LabVIEW is single sourced and some companies may not like to use a product that is single sourced and not standardised by the industry.
* Cost of ownership – although in line with many other industry products of a similar nature, its cost should be considered before it is introduced.
* For those more accustomed to text programming, graphical programming can take a little familiarisation time.

Like any product, LabVIEW has some disadvantages, but many who use it find it particularly effective. In view of this, LabVIEW should be carefully assessed before its use is introduced.

**Embedded c**

**Embedded C** is a set of language extensions for the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) by the [C Standards Committee](https://en.wikipedia.org/wiki/ISO/IEC_JTC_1/SC_22" \o "ISO/IEC JTC 1/SC 22) to address commonality issues that exist between C extensions for different [embedded systems](https://en.wikipedia.org/wiki/Embedded_system" \o "Embedded system).

Embedded C programming typically requires nonstandard extensions to the C language in order to support enhanced [microprocessor](https://en.wikipedia.org/wiki/Microprocessor" \o "Microprocessor) features such as [fixed-point arithmetic](https://en.wikipedia.org/wiki/Fixed-point_arithmetic" \o "Fixed-point arithmetic), multiple distinct [memory banks](https://en.wikipedia.org/wiki/Memory_bank" \o "Memory bank), and basic [I/O](https://en.wikipedia.org/wiki/I/O" \o "I/O) operations. In 2008, the C Standards Committee extended the C language to address such capabilities by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as fixed-point arithmetic, named address spaces and basic I/O hardware addressing. Embedded C uses most of the syntax and semantics of standard C, e.g., main() function, variable definition, datatype declaration, conditional statements (if, switch case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc

The C Programming Language, developed by Dennis Ritchie in the late 60’s and early 70’s, is the most popular and widely used programming language. The C Programming Language provided low level memory access using an uncomplicated compiler (a software that converts programs to machine code) and achieved efficient mapping to machine instructions.

The C Programming Language became so popular that it is used in a wide range of applications ranging from Embedded Systems to Super Computers.

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces

**Factors for Selecting the Programming Language**

The following are few factors that are to be considered while selecting the Programming Language for the development of Embedded Systems.

* **Size:** The memory that the program occupies is very important as Embedded Processors like Microcontrollers have a very limited amount of ROM.
* **Speed:** The programs must be very fast i.e. they must run as fast as possible. The hardware should not be slowed down due to a slow running software.
* **Portability:** The same program can be compiled for different processors.
* **Ease of Implementation**
* **Ease of Maintenance**
* **Readability**

### **Difference between C and Embedded C**

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc.

C Programming Language is generally used for developing desktop applications whereas Embedded C is used in the development of Microcontroller based applications.

**[Arduino](https://en.wikipedia.org/wiki/Arduino" \o "Arduino) ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment" \o "Integrated development environment))**

The **[Arduino](https://en.wikipedia.org/wiki/Arduino" \o "Arduino) Integrated Development Environment ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment" \o "Integrated development environment))** is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform" \o "Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows" \o "Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS), [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) and [C++](https://en.wikipedia.org/wiki/C++_(programming_language)" \o "C++ (programming language))[[2]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-2). It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.[[3]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-3)

The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License" \o "GNU General Public License), version 2.[[4]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-4) The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) and [C++](https://en.wikipedia.org/wiki/C++" \o "C++) using special rules of code structuring.[[5]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-5) The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library" \o "Software library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)" \o "Wiring (development platform)) project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive" \o "Cyclic executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain" \o "GNU toolchain), also included with the IDE distribution.[[6]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-6) The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.[[7]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-7) By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards[[8]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-8).

|  |  |
| --- | --- |
| **Arduino Pro IDE** | |
| **[Developer(s)](https://en.wikipedia.org/wiki/Software_developer" \o "Software developer)** | Arduino Software |
| **[Preview release](https://en.wikipedia.org/wiki/Software_release_life_cycle" \o "Software release life cycle)** | v0.0.2 / 28 October 2019; 6 months ago[[9]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-9) |
| **[Repository](https://en.wikipedia.org/wiki/Repository_(version_control)" \o "Repository (version control))** | * [github.com/arduino/Arduino](https://github.com/arduino/Arduino)   [IMG_256](https://www.wikidata.org/wiki/Q55080330#P1324) |
| **Written in** | [C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)), [C++](https://en.wikipedia.org/wiki/C++" \o "C++) |
| **[Operating system](https://en.wikipedia.org/wiki/Operating_system" \o "Operating system)** | [Windows](https://en.wikipedia.org/wiki/Windows" \o "Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS), [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux) |
| **[Platform](https://en.wikipedia.org/wiki/Computing_platform" \o "Computing platform)** | [IA-32](https://en.wikipedia.org/wiki/IA-32" \o "IA-32), [x86-64](https://en.wikipedia.org/wiki/X86-64" \o "X86-64), [ARM](https://en.wikipedia.org/wiki/ARM_architecture" \o "ARM architecture) |
| **[Type](https://en.wikipedia.org/wiki/Software_categories" \l "Categorization_approaches" \o "Software categories)** | [Integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment" \o "Integrated development environment) |
| **[License](https://en.wikipedia.org/wiki/Software_license" \o "Software license)** | [LGPL](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License" \o "GNU Lesser General Public License) or [GPL](https://en.wikipedia.org/wiki/GNU_General_Public_License" \o "GNU General Public License) license |
| **Website** | [blog.arduino.cc/2019/10/18/arduino-pro-ide-alpha-preview-with-advanced-features/](https://blog.arduino.cc/2019/10/18/arduino-pro-ide-alpha-preview-with-advanced-features/) |

With the rising popularity of Arduino as a software platform, other vendors started to implement custom open source compilers & tools (cores) that can build and upload sketches to other MCUs that are not supported by Arduino's official line of MCUs.

In October 2019 the [Arduino](https://en.wikipedia.org/wiki/Arduino" \o "Arduino) organization began providing early access to a new Arduino Pro IDE with debugging[[10]](https://en.wikipedia.org/wiki/Arduino_IDE" \l "cite_note-10) and other advanced features

**CHAPTER 4:**

**ALGORITHM OR FLOWCHART**

**4.1.Data Acquisition Algorithm**

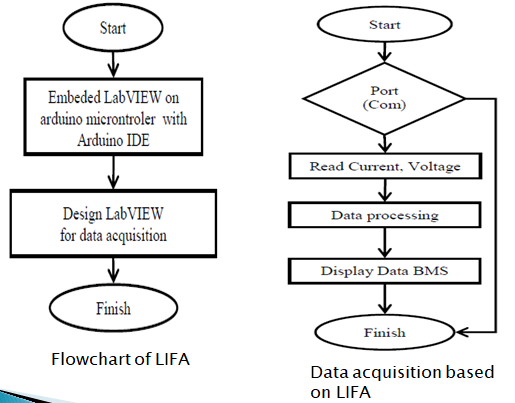


Fig 4.1.Data Acquisition Algorithm

Data acquisition plays an important role in the battery monitoring system. Data acquisition begins with the physical phenomenon or physical property to be measured. The sensor values are continuously monitored at every 1 minute interval. Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer.

Health of the battery is defined by battery current, voltage and temperature. Optimum operating temperature of lead acid battery is 25\*C. Increase in temperature might reduce it’s longevity. Similarly over voltage, under voltage and over current deteriorates battery health. The operating status of battery is having relationship with temperature and hence, the battery status is defined with respect to temperature values.

Hence all these physical parameters are collected via sensors and can be monitored using data acquisition algorithm and necessary measures can be taken to increase battery life.

**4.2. Data Networking Algorithm**

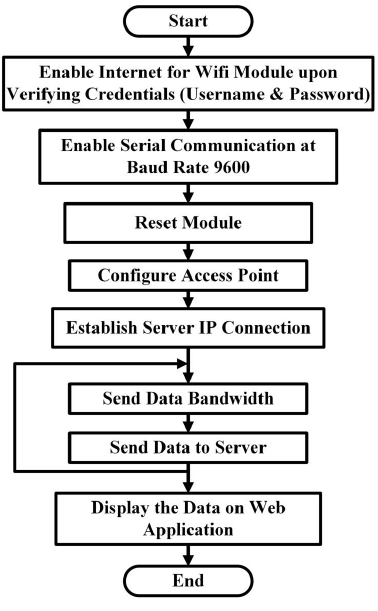


Fig 4.2.Data Networking Algorithm

In order to export data into cloud database, data has to be transferred to data server. The collected data from sensors are transferred to blynk app using ESP8266 and it is configured by using Arduino microcontroller. AT commands are written in embedded C programming to configure ESP8266 and transfer data to server . The algorithm for data transfer is given in Fig. 4.2. A web application is developed to display the monitored battery parameters by stake holders. This will give the data in tabular format as well as in trend fashion to improve the data interpretation efficiency of viewers.

**4.3.State of Charge(SoC) algorithm**

In order to estimate SoC of battery, there are numerous techniques developed which includes coulomb counting method, open circuit voltage method, impedance method, Kalman filter estimation, neural network estimation etc. Out of this, coulomb counting method has its own advantages, since the SoC can be estimated from current. Coulomb Counting will be the basis of your SOC algorithm. You know the capacity(Ah rating) of your battery. If you can accurately measure current and time, you can integrate the current over time to determine the SoC .For example ,if you start at 100% SoC on a 2Ah battery, and you measure a 1A discharge for 30 minutes, your SoC is now

(2Ah-1A\*0.5h)/2Ah=75% SoC .

If you start at 0% SoC on the same battery and measure a 0.1A charge for 5 hours, your SoC is

(0.1A\*5h)/2Ah=25% SoC .

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

**CHAPTER 6**

**APPPLICATIONS,ADVANTAGES AND LIMITATIONS**

**APPLICATIONS:**

Battery monitoring system (BMS) is adaptable to multiple types of battery chemistry, creating cross-platform capabilities for a wealth of sensing needs. In addition to use with electric UAVs, potential applications include electric vehicles (EVs), medical devices, instrumentation, and robotics.

Automotive BMS:



* Hybrid Electric Vehicles (HEVs) and Plug-in Hybrid Electric Vehicles (PHEVs)
* On-board battery charger

Industrial BMS:



* Autonomous vehicles
* Chargers 30kW to 150kW
* Chargers 50kW to 350kW
* Multi-copter and drones
* Power supplies
* Medical devices

Consumer BMS:

* Service robotics
* Wearables
* Home appliances
* Power tools
* Smart speaker/portable audio
* Wireless charger
* Uninterruptible power supply (UPS) systems

**ADVANTAGES:**

• **Improved safety:** Prevents occurrence of catastrophic battery failure by predicting the remaining useful life (RUL) of battery systems, when used in combination with customized algorithms.

• **Powerful:** Processes large amounts of data and reports results wirelessly.

• **Effective:** Monitors the health of multiple batteries simultaneously.

• **Reliable:** Enables a critical, real-time monitoring capability that allows an immediate and controlled response to avoid battery failure.

• **Low-cost:** Using BMS reduces the maintenance and replacement costs.

• **Rugged:** Provides a robust platform for multiple sensing needs, with the potential for miniaturization.

• **Unique:** Features custom-developed software that processes and sends real-time data reports.

**LIMITATIONS:**

The simulated data may indicate some inherent problems, its validity can only be found once performance evaluation begins. Both the mathematical and simulation models derived will always have some degree of inaccuracy due to the unavailability of specific data pertaining to the cells. However the software used gives great insight into how one may expect the lead acid batteries to operate with the given voltage and current loads.

**CHAPTER 7**

**CONCLUSION AND FUTURE SCOPE**

**CONCLUSION:**

A smart BMS is crucial in the realization of the smart grid and the escalation of the EVs industry. The development of battery technology that provides higher energy and power density and reduces cost cannot be fully accomplished without proper BMS circuits and algorithms to monitor and control the battery and guarantee the safety and reliability of the energy storage devices. Although most of the performance requirements for the BMS in laptop and cell phone applications are already provided, there is still much research and development needed to satisfy the standards for EVs and smart grid applications.

The project set up was built for a battery monitoring system Using LabVIEW environment. The whole integrated system was tested for real time monitoring of the parameters of the battery like voltage, current, temperature, humidity and smoke detection. Using these parameters the State of charge(SoC) of the battery is calculated. LabVIEW shell was tested and the measured parameters were monitored at real time rate.

**FUTURE SCOPE:**

We are currently building the system for the electric vehicle batteries and smaller auxiliary battery in internal combustion vehicles. However, by the addition of a cut o↵ period into Solar systems or Home UP systems we can extend the project or the batteries in the aforementioned fields as well. We hope that ease of access to information about the battery status and its life, we increase the trust of the consumer world on battery reliability and hence increase the popularity of stored energy, mostly from renewable sources, such as in Solar photo voltaic systems.

For the future, we expect a continuous improvement of available BMS components. In addition, several interesting approaches regarding new concepts for BMS implementations exist, like fine grained modularization and different means of communication. In addition, regarding robustness and failure detection, there are still new ideas emerging. Another aspect with rising significance might be the integration of BMS functionality into the battery cell. For large battery cells or special operating conditions, this approach could provide advantages by allowing easier assembly of packs, as well as more sophisticated monitoring of cell internal parameters.

**REFERENCES:**

1.“Coulomb counting state-of-charge algorithm for electric vehicles with a physics-based temperature dependent battery model,” (ECCE). IEEE, 2015, pp. 5052– 5059.

2. “Design real time battery monitoring system using lab-view interface for Arduino (LIFA),”. IEEE, 2013, pp. 1–4.

3. “Modified coulomb counting method of SoC estimation for uninterruptible power supply system’s battery management system,” IEEE, 2015, pp. 197–203.

4. Jamaluddin, A., Perdana, F. A., Supriyanto, A., Purwanto, A., Inayati, & Nizam, M. (2014). Development of Wireless Battery Monitoring for electric vehicle. 2014 International Conference on Electrical Engineering and Computer Science (ICEECS). doi:10.1109/iceecs.2014.7045235

5. M. Luo, Y. Xiao, W.-M. Sun, and Z. Wang, “Online battery monitoring system based on gprs for electric vehicles,” in 5th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), vol. 1. IEEE, 2013, pp. 122–125.

6. Jangamshetti, D. S., & Jangamshetti, S. H. (2015). Design, implementation and testing of theft and maintenance monitoring of batteries of stand-alone SPV systems. 2015 International Conference on Renewable Energy Research and Applications (ICRERA). doi:10.1109/icrera.2015.7418443

7. K. Friansa, I. N. Haq, B. M. Santi, D. Kurniadi, E. Leksono, and B. Yuliarto, “Development of battery monitoring system in smart microgrid based on internet of things (iot),” Procedia Engineering, vol. 170, pp. 482–487, 2017.

8. H. M. Khalid, Q. Ahmed, and J. C.-H. Peng, “Health monitoring of lion battery systems: A median expectation diagnosis approach (meda),” IEEE Transactions on Transportation Electrification, vol. 1, no. 1, pp. 94–105, 2015.

9. W. Juang, P. J. Kollmeyer, R. Zhao, T. Jahns, and R. Lorenz, “Coulomb counting state-of-charge algorithm for electric vehicles with a physics-based temperature dependent battery model,” in IEEE EnergyConversion Congress and Exposition (ECCE). IEEE, 2015, pp. 5052–5059