

Participant Handbook

Sector
Electronics

Sub-Sector
**Semiconductor
& Components**

Occupation
Product Design

Reference ID: **ELE/Q1404, Version 2.0**
NSQF Level 5



**Embedded Full-Stack
IoT Analyst**



Scan this QR Code to access eBook

Published by

Electronics Sector Skill Council of India (ESSCI)

155, 2nd Floor, ESC House, Okhla Industrial Area-Phase 3, New Delhi- 110020, India

Email: info@essc-india.org

Website: www.essc-india.org

Phone: +91 8447738501

All Rights Reserved ©2022

First Edition, August 2022

Copyright © 2022

Electronics Sector Skill Council of India (ESSCI)

155, 2nd Floor, ESC House, Okhla Industrial Area-Phase 3, New Delhi- 110020, India

Email: info@essc-india.org

Website: www.essc-india.org

Phone: +91 8447738501

This book is sponsored by Electronics Sector Skill Council of India (ESSCI)

Under Creative Commons Licence: CC-BY-SA

Attribution-ShareAlike: CC BY-SA



This license lets others remix, tweak, and build upon your work even for commercial purposes, as long as they credit you and license their new creation under the identical terms. This license is often compared to “copyleft” free and open source software licenses. All new works based on yours will carry the same license, so many derivatives will also allow commercial use. This is the license used by Wikipedia and is recommended for materials that would benefit from incorporating content from Wikipedia and similarly licensed projects.

Disclaimer

The information contained herein has been obtained from sources reliable to ESSCI. ESSCI disclaims all warranties to the accuracy, completeness or adequacy of such information. ESSCI shall have no liability for errors, omissions, or inadequacies, in the information contained herein, or for interpretations thereof. Every effort has been made to trace the owners of the copyright material included in the book. The publishers would be grateful for any omissions brought to their notice for acknowledgements in future editions of the book. No entity in ESSCI shall be responsible for any loss whatsoever, sustained by any person who relies on this material. The material in this publication is copyrighted. No parts of this publication may be reproduced, stored or distributed in any form or by any means either on paper or electronic media, unless authorized by the ESSCI.





Shri Narendra Modi
Prime Minister of India

“ Skilling is building a better India.
If we have to move India towards
development then Skill Development
should be our mission. ”



Certificate
**COMPLIANCE TO
QUALIFICATION PACK - NATIONAL OCCUPATIONAL
STANDARDS**

is hereby issued by the
Electronics Sector Skill Council of India
for
SKILLING CONTENT : PARTICIPANT HANDBOOK

Complying to National Occupational Standards of
Job Role/ Qualification Pack: "Embedded Full-Stack IoT 'QP No. ELE/Q1404, NSQF Level 5
Analyst"

Date of Issuance: Sep 9th 2022

Valid up to*: June 2ND 2025

*Valid up to the next review date of the Qualification Pack

Authorised Signatory
Electronics Sector Skill Council of India

Acknowledgements

This participant's handbook meant for Embedded Full-Stack IoT Analyst is a sincere attempt to ensure the availability of all the relevant information to the existing and prospective job holders in this job role. We have compiled the content with inputs from the relevant Subject Matter Experts (SMEs) and industry members to ensure it is the latest and authentic. We express our sincere gratitude to all the SMEs and industry members who have made invaluable contributions to the completion of this participant's handbook. We'd also like to thank all the experts and organizations who have helped us by reviewing the content and providing their feedback to improve its quality.

This handbook will help deliver skill-based training in the field of Embedded Full-Stack IoT assembly and testing. We hope that it will benefit all the stakeholders, such as participants, trainers, and evaluators. We have made all efforts to ensure the publication meets the current quality standards for the successful delivery of QP/NOS-based training programs. We welcome and appreciate any suggestions for future improvements to this handbook.

About this book

This participant handbook has been designed to serve as a guide for participants who aim to obtain the required knowledge and skills to undertake various activities in the role of an Embedded Full-Stack IoT Analyst. Its content has been aligned with the latest Qualification Pack (QP) prepared for the job role. With a qualified trainer's guidance, the participants will be equipped with the following for working efficiently in the job role:

- **Knowledge and Understanding:** The relevant operational knowledge and understanding to perform the required tasks.
- **Performance Criteria:** The essential skills through hands-on training to perform the required operations to the applicable quality standards.
- **Professional Skills:** The Ability to make appropriate operational decisions about the field of work.

The handbook details the relevant activities to be carried out by an Embedded Full-Stack IoT Analyst. After studying this handbook, job holders will be adequately skilled to carry out their duties efficiently according to the applicable quality standards. The handbook is aligned with the following National Occupational Standards (NOS) detailed in the Embedded Full-Stack IoT Analyst QP:

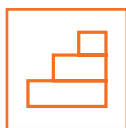
1. **ELE/N1406:** Develop and Test Design for IoT Based System
2. **ELE/N1410:** Build GUI and Applications in a Framework
3. **ELE/N1411:** Test and Troubleshoot the Firmware
4. **ELE/N9905:** Work effectively at the workplace
5. **ELE/N1002:** Apply health and safety practices at the workplace

The handbook has been divided into an appropriate number of units and sub-units based on the content of the relevant QP. We hope it will facilitate easy and structured learning for the participants. We sincerely hope that participants will obtain enhanced knowledge and skills after studying this handbook and make career progress in the relevant and senior job roles.

Symbols Used



Key Learning
Outcomes



Steps



Exercise



Notes



Unit
Objectives

Table of Contents

S. No.	Modules and Units	Page No.
1.	Introduction and Orientation to the Role of an Embedded Full-Stack IoT Analyst	1
	Unit 1.1 – About Electronics Industry	3
	Unit 1.2 – About Role of an Embedded Full-Stack IoT Analyst	9
2.	Process of Developing and Testing Design for IoT Based System (ELE/N1406)	11
	Unit 2.1 – IOT System	13
	Unit 2.2 – IOT System Architecture and Requirements	25
3.	Process of Building GUI and Applications in a Framework (ELE/N1410)	33
	Unit 3.1 – Develop Application for IoT System	35
	Unit 3.2 – Develop GUI/web UI for IoT System	72
4.	Process of Testing and Troubleshooting the Firmware (ELE/N1411)	87
	Unit 4.1 – Testing of Firmware	89
	Unit 4.2 – Debugging of Firmware	95
5.	Work Effectively at the Workplace (ELE/N9905)	117
	Unit 5.1 – Effective Communication and Coordination at Work	119
	Unit 5.2 – Working Effectively and Maintaining Discipline at Work	125
	Unit 5.3 – Maintaining Social Diversity at Work	135
6.	Basic Health and Safety Practices (ELE/N1002)	141
	Unit 6.1 – Workplace Hazards	143
	Unit 6.2 – Fire Safety	155
	Unit 6.3 – First Aid	158
	Unit 6.4 – Waste Management	161





1. Introduction and Orientation to the Role of an Embedded Full-Stack IoT Analyst



Unit 1.1 - About Electronics Industry

Unit 1.2 - About Role of an Embedded Full-Stack IoT Analyst



Key Learning Outcomes



At the end of this module, participants will be able to:

1. Describe about electronics industry
2. List applications of electronics industry
3. Describe trends and challenges in electronics industry
4. List role and responsibilities of an Embedded Full-tack IoT Analyst

Unit 1.1: About Electronics Industry

Unit Objectives



At the end of this unit, participants will be able to:

1. Describe about electronics industry
2. List applications of electronics industry
3. Describe trends and challenges in electronics industry

1.1.1 Introduction

The electronics industry is the economic sector that produces electronic devices. It emerged in the 20th century and is today one of the largest global industries. Contemporary society uses a vast array of electronic devices built-in automated or semi-automated factories operated by the industry.

Electronics industry, the business of creating, designing, producing, and selling devices such as radios, televisions, stereos, computers, semiconductors, transistors, and integrated circuits etc. The electronics industry transformed factories, offices, and homes, emerging as a key economic sector that rivalled the chemical, steel, and auto industries in size.

The electronics sector produces electronic equipment and consumer electronics and manufactures electrical components for a variety of products. Common items in the electronics sector include mobile devices, televisions, and circuit boards. Industries within the electronics sector include telecommunications, networking, electronic components, industrial electronics, and consumer electronics.

Growth in the Electronics Sector

The electronics sector is growing rapidly as a result of increasing demand from emerging market economies. As a result, many countries are increasingly producing more electronics, and investment in the foreign production of electronics has increased dramatically.

Electronics sector growth is accelerated by increased consumer spending around the world. As developing economies grow, consumer demand for electronics also grows. Countries that produce electronics now have strong consumer bases that can afford new electronic products. At the same time, increased competition is driving the costs of electronics production down, making products even cheaper for individuals.

The supportive role of the electronics sector in providing equipment and components for other industries is also a factor of growth as consumers demand more automobiles, energy-efficient homes, and medical technologies.

1.1.2 Application of Electronics in Different Fields

The various electronics applications are:

- **Consumer Electronics:** The devices and equipment meant for daily use are known as customer electronics; this industry is widely applicable to the common people. Some of its applications included office gadgets like computers, scanners, calculators, FAX machines, projectors etc.

It also includes home appliances like washing machines, refrigerators, microwaves, TVs, vacuum cleaners, video games, loudspeakers etc. and some advanced storage devices such as HDD jukebox, DVDs etc.
- **Industrial applications of electronics:** Electronics engineering has a huge impact on the smooth functioning of the industries as it is used in various systems, grids and processing units. For example, smart electric systems collect information from the communication technology department, and several machines use automation and motor control systems using electronics; also, it is used in extracting 3D images from 2D using image processing systems.
- **Robotics and artificial intelligence:** Apart from image processing that involves computer graphics, electronic systems are also used in artificial intelligence and robotics technologies for inspection, navigation and assembly. Virtual reality and face gesture recognition are computer-based, and these developments have been possible because of electronics engineering.
- **Medical applications:** For data recording and physiological analysis, advanced, sophisticated instruments are being developed using the latest technologies and electronics engineering, and these instruments are very useful in diagnosing diseases and for healing purposes.

Electronics play a vital role in the functioning of medical instruments; for instance, the stethoscope is used to listen to the inner sounds of the human or animal body, a glucose metre for checking sugar levels, a pacemaker for dropping and increasing heartbeat count and so on.
- **Defence and Aerospace:** Electronics technology has been used extensively in the defence and aeronautical systems, which include missile launching systems, cockpit controllers, military radars, aircraft systems, rocket launchers for space and many more.
- **Automobiles:** Electronics are widely used in the latest automobile technologies, like anti-collision units, anti-lock braking systems, traction controls, window regulators and several electronic control units.

1.1.3 Electronic Industry Trends and Challenges

The electronics sector appears to be overgrowing, owing to increased demand from developing countries. Before the virus outbreak, due to increased demand, electronics production skyrocketed, accompanied by a surge in investment.

The global electronic products market is expected to be worth nearly \$1,191.2 billion in 2020, with a Compound Annual Growth Rate (CAGR) of 5.4 percent since 2015. The increase is primarily due to the increasing demand for various electronic products as employees and students have transitioned to online.

Consumer Electronics Market size was valued at over USD 1 trillion in 2020 and is estimated to grow at a CAGR of more than 8% from 2021 to 2027. Rapidly increasing internet penetration across the globe will drive the market growth.

Consumer electronics are electronic equipment for non-commercial use. Consumer electronics include devices that provide one or more functionalities such as computers, laptops, mobile devices, smart wearables, television sets, refrigerators, smartphones, and home appliances.

Continuous investments by market players in R&D for the development of new consumer electronic products with enhanced features will fuel the industry growth of consumer electronics.

Challenges in the electronic industry

Regardless of its merits, the electronic industry faces disruptive forces that will test its business model and ability to survive and thrive.

The global electronic industries are the fastest-growing sector, worth trillions of dollars, and play a critical role in driving consumers to purchase innovative and smart electronic products. The global market for electronic components is expected to grow at a compound annual growth rate (CAGR) of about 4.8 percent from 2020 to 2025.

Electronic industries have always been at the forefront of the most recent technological innovations to reduce costs and improve efficiency with such a large future market potential. Many SMEs have found it challenging to keep up with the trends/changes as technology has advanced faster.

For example, top players such as Apple, Samsung, Microsoft, and Intel, to name a few, are investing heavily in new cutting-edge technology to expand their technological capabilities and remain competitive. They are the leading example of an IR4.0 (industrial Revolution 4.0) Eco-friendly system.

The integration of digital tools and technologies has increased revenue and productivity, improved product quality, reduced waste, and operational costs, and met the most recent customer/global demands.

Electronic Industry Trends

Here are some predictions for the specific trends that are likely to have the most significant impact in 2022. The most important trends in 2022 will likely focus on the convergence of technology trends as tools emerge that let us combine them in new and amazing ways.

1. **The 5G Optimization:** 5G is laying the groundwork for a fully digitalized and connected world. We have seen many new field trials and an increasing number of commercial rollouts over the last two years. Furthermore, we are seeing 5G being adopted in various industries, ranging from manufacturing to healthcare.

With its high output and ultralow latency, 5G can access many high-value areas such as 3D robotic control, virtual reality monitoring, and remote medical control that previous technologies could not. 5G is redefining and accelerating industries like automotive, entertainment, computing, and manufacturing. It will eventually change the way we work and live.

2. **Digitization, data, and virtualization:** Many of us witnessed the virtualization of our offices and workplaces in 2020 and 2021, as remote working arrangements were quickly implemented.

This was simply a crisis-driven acceleration of a much longer-term trend. In 2022, we'll be more familiar with the concept of a “metaverse” - persistent digital worlds that exist alongside the physical world we live in.

3. **Concentrate on Software Quality Standards:** The focus on quality will be the trend for 2022 and beyond. Software solutions will be integrated into our daily lives and the majority of the goods and appliances we use. As a result, software must meet the quality standards of the manufacturing industry.
4. **Teleworking:** Teleworking will continue to grow in 2022, bringing advances in software development. Companies worldwide will need to support hybrid forms of team management and collaboration to increase the productivity of their workforces. As the trend of conducting online meetings and video sales calls continues, this new standard will grow even more in 2022.
5. **Green, Clean, and Lean Energy:** Renewable energy was the only type of energy that saw an increase in use during the pandemic. As industries shut down and people stayed at home, global non-renewable energy consumption decreased, resulting in an 8% reduction in emissions. As a result, increased investment in renewable energy generation is expected in the coming years.

According to the International Energy Agency (IEA), 40% more renewable energy was generated and used in 2020 than the previous year. This trend is expected to continue through 2022. Overall, the cost of generating renewable energy from various sources, such as onshore and offshore wind, solar, and tidal, has decreased by 7 to 16%. This will be highly beneficial to countries and businesses attempting to meet emissions targets such as becoming carbon neutral or even carbon negative.

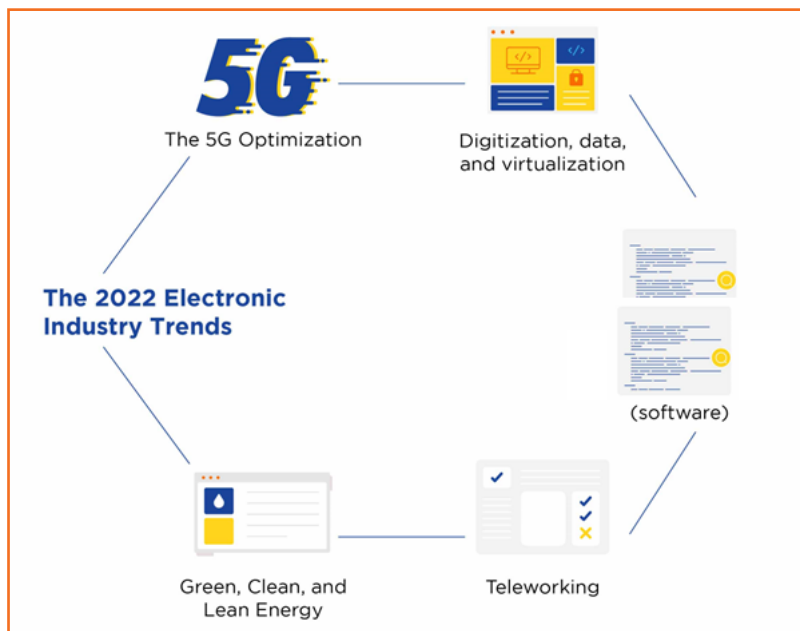


Fig 1.1.1: Trends in electronics industry

Electronics manufacturing trends for 2022

1. **Advanced Materials:** The semiconductor industry has been reliant on silicon for decades, but there is a limit to how far you can etch, lithograph, and pattern a silicon material. As a result,

innovation to increase the performance of integrated circuits is coming from new materials and architectures. Startups and scaleups are developing silicon alternatives and other semiconductor materials or composites for high performance and efficiency.

2. **Organic Electronics:** Organic Electronics offer massive advantages over traditional inorganic electronics. They are cost-effective, flexible, indissoluble, optically transparent, lightweight, and consume low power. In addition, the rise in awareness for sustainable development and eco-friendly manufacturing attracts manufacturers to opt for organic electronics. Designing circuits with microbial components or producing devices with biodegradable and recyclable materials is seen to be the next electronics manufacturing trend.
3. **Artificial Intelligence:** AI-powered solutions are gaining popularity in every sector. AI impacts the growth of semiconductor manufacturing in two ways, one is by building demand for innovative AI-capable electronics components, and two, enhancing the product manufacturing and design processes. The conventional methods have limitations to reshaping product development cycles, improving product design processes, and reducing defects. But the application of AI is solving all these limitations.
4. **Internet of Things:** The rapid growth of the Internet of Things represents an unprecedented opportunity for the electronics manufacturing industry. It re-evaluates the fabrication process and manages practices that are found to be difficult to achieve with conventional approaches. In other ways, the IoT enables electronic manufacturing machines to self-process and store data while being digitally connected. Continuous improvements in the fabrication of sensors are also required since sensors are the key components that enable IoT applications. Further, the transition to 5G-enabled devices requires flawless, innovative chips with more efficient architectures at lower costs.
5. **Embedded Systems:** Embedded systems are an unavoidable part of any electronic device nowadays and it has a crucial role in deciding the speed, security, size, and power of the devices. Since we are in the transition phase of a connected world, there is high demand for embedded systems. So the designing and manufacturing sector of such systems is undergoing numerous innovations to improve performance, security, and connectivity capabilities.
6. **Printed Electronics:** Printing electronics components on a semiconductor substrate is the most effective way to reduce the overall cost of the manufacturing process. So, manufacturers are always trying to tackle this challenge by searching for new technologies and advancements in conventional printing technologies. Unlike traditional semiconductors that use tiny wires as circuits, printed electronics rely on conductive inks and often flexible films. Further, the advancements in printing technologies allow the flexible hybrid electronics field to obtain enough momentum. Therefore, startups and scaleups are developing solutions for advanced printing technologies.
7. **Advanced IC Packaging:** In recent years, chip packaging has become a hot topic along with chip design. The traditional way to scale a device based on Moore's law has limitations nowadays. The other way to get the benefits of scaling is to put multiple complex devices in an advanced package. So, semiconductor manufacturers develop new advanced IC packaging technologies to provide greater silicon integration in increasingly miniaturized packages. This also enables manufacturers to offer customization and improve yields by vertically stacking modular components.

Unit 1.2: About Role of an Embedded Full-Stack IoT Analyst

Unit Objectives



At the end of this unit, participants will be able to:

1. List role and responsibilities of an Embedded Full-Stack IoT Analyst

1.2.1 Role and Responsibilities of Embedded Full-Stack IoT Analyst

An Embedded Full-Stack IoT Analyst creates and implements new embedded OS based on system requirements and/or industry specifications. The individual builds and manages OS drivers for any custom hardware and to enhance existing application with new features.

Responsibilities of an Embedded Full-Stack IoT Analyst

- Develop and test the design for IoT based system.
- Build the GUI and applications in a framework for IoT based system.
- Test and troubleshoot the firmware for proper functioning
- Collect data aggregated from a variety of sources, in a range of formats, and at multiple frequencies.
- Process the data with a wide range of external sources.
- Store the information in a time-series for analysis.
- Analyze the data in multiple ways--with custom analysis systems, with standard SQL queries, or with machine learning analysis techniques. The results can be used to make a wide range of predictions.
- With the information received, build several systems and applications to ease business processes.

2. Process of Developing and Testing Design for IoT Based System



Unit 2.1 - IOT System

Unit 2.2 - IOT System Architecture and Requirements



Key Learning Outcomes



At the end of this module, participants will be able to:

1. Describe IOT
2. List advantages and disadvantages of IOT
3. List components and applications of IOT
4. Describe phases and stages of an IoT system architecture
5. List requirements for building an IoT system

Unit 2.1: IOT System

Unit Objectives

At the end of this unit, participants will be able to:

1. Describe IOT
2. List advantages and disadvantages of IOT
3. List components and applications of IOT

2.1.1 IoT (Internet of Things)

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

IoT systems have applications across industries through their unique flexibility and ability to be suitable in any environment. They enhance data collection, automation, operations, and much more through smart devices and powerful enabling technology.

IoT - key features

The most important features of IoT include -

1. **AI:** IoT essentially makes virtually anything “**smart**”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks.
2. **Connectivity:** New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
3. **Sensors:** IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
4. **Active Engagement:** Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
5. **Small Devices:** Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

IoT - Advantages

The advantages of IoT are -

- **Improved Customer Engagement:** Current analytics suffer from blind-spots and significant flaws

in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.

- **Technology Optimization:** The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
- **Reduced Waste:** IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
- **Enhanced Data Collection:** Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

IoT - Disadvantages

Though IoT delivers an impressive set of benefits, it also presents a significant set of challenges. It's some major issues are -

- **Security:** IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.
- **Privacy:** The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
- **Complexity:** Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
- **Flexibility:** Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
- **Compliance:** IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

2.1.2 IoT - Technology and Protocols

IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

- **NFC and RFID:** RFID (radio-frequency identification) and NFC (near-field communication) provide simple, low energy, and versatile options for identity and access tokens, connection bootstrapping, and payments.
 - i. RFID technology employs 2-way radio transmitter-receivers to identify and track tags associated with objects.

ii. NFC consists of communication protocols for electronic devices, typically a mobile device and a standard device.

- **Low-Energy Bluetooth:** This technology supports the low-power, long-use need of IoT function while exploiting a standard technology with native support across systems.
- **Low-Energy Wireless:** This technology replaces the most power hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless) must remain in listening mode. Low-energy wireless not only reduces consumption, but also extends the life of the device through less use.
- **Radio Protocols:** ZigBee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but offer high throughput unlike many similar options. This increases the power of small local device networks without the typical costs.
- **LTE-A:** LTE-A, or LTE Advanced, delivers an important upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT a tremendous power through expanding its range, with its most significant applications being vehicle, UAV, and similar communication.
- **WiFi-Direct:** WiFi-Direct eliminates the need for an access point. It allows P2P (peer-to-peer) connections with the speed of WiFi, but with lower latency. WiFi-Direct eliminates an element of a network that often bogs it down, and it does not compromise on speed or throughput.

2.1.3 Components of an IoT System

IoT Hardware includes a wide range of devices such as devices for routing, bridges, sensors etc. These IoT devices manage key tasks and functions such as system activation, security, action specifications, communication, and detection of support-specific goals and actions.

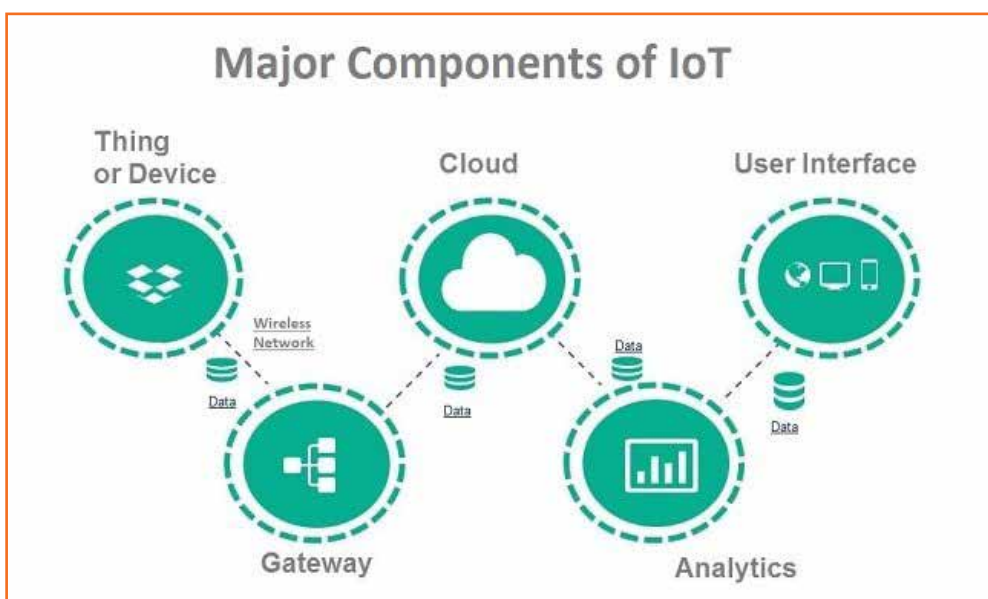


Fig 2.1.1: Components of an IoT system

1. **Smart devices and sensors - Device connectivity:** Devices and sensors are the components of the device connectivity layer. These smart sensors are continuously collecting data from the environment and transmit the information to the next layer.

Latest techniques in the semiconductor technology is capable of producing micro smart sensors for various applications.



Fig 2.1.2: Sensor

Common sensors are:

- Temperature sensors and thermostats
- Pressure sensors
- Humidity / Moisture level
- Light intensity detectors
- Moisture sensors
- Proximity detection
- RFID tags

How the devices are connected?

Most of the modern smart devices and sensors can be connected to low power wireless networks like Wi-Fi, ZigBee, Bluetooth, Z-wave,



Fig 2.1.3: Wi-fi networks

LoRAWAN etc. Each of these wireless technologies has its own pros and cons in terms of power, data transfer rate and overall efficiency.

Developments in the low power, low cost wireless transmitting devices are promising in the area of IoT due to its long battery life and efficiency. Latest protocols like 6LoWPAN- IPv6 over Low Power Wireless Personal Area Networks have been adapted by many companies to implement energy efficient data transmission for IoT networks.

2. **Gateway:** IoT Gateway manages the bidirectional data traffic between different networks and protocols. Another function of gateway is to translate different network protocols and make sure interoperability of the connected devices and sensors.

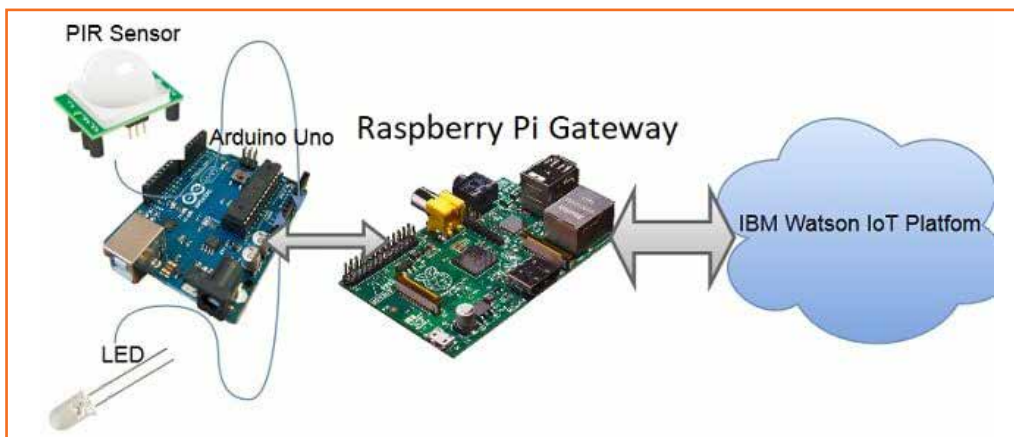


Fig 2.1.4: Gateway platforms

Gateways can be configured to perform pre-processing of the collected data from thousands of sensors locally before transmitting it to the next stage. In some scenarios, it would be necessary due to compatibility of TCP/IP protocol.

IoT gateway offers certain level of security for the network and transmitted data with higher order encryption techniques. It acts as a middle layer between devices and cloud to protect the system from malicious attacks and unauthorized access.

3. **Cloud:** Internet of things creates massive data from devices, applications and users which has to be managed in an efficient way. IoT cloud offers tools to collect, process, manage and store huge amount of data in real time. Industries and services can easily access these data remotely and make critical decisions when necessary.

Basically, IoT cloud is a sophisticated high performance network of servers optimized to perform high speed data processing of billions of devices, traffic management and deliver accurate analytics. Distributed database management systems are one of the most important components of IoT cloud.

4. **Analytics:** Analytics is the process of converting analog data from billions of smart devices and sensors into useful insights which can be interpreted and used for detailed analysis. Smart analytics solutions are inevitable for IoT system for management and improvement of the entire system.



Fig 2.1.5: System analysis

One of the major advantages of an efficient IoT system is real time smart analytics which helps engineers to find out irregularities in the collected data and act fast to prevent an undesired scenario. Service providers can prepare for further steps if the information is collected accurately at the right time.

Big enterprises use the massive data collected from IoT devices and utilize the insights for their future business opportunities.

5. **User interface:** User interfaces are the visible, tangible part of the IoT system which can be accessible by users. Designers will have to make sure a well designed user interface for minimum effort for users and encourage more interactions.

Modern technology offers much interactive design to ease complex tasks into simple touch panels controls. Multicolor touch panels have replaced hard switches in our household appliances and the trend is increasing for almost every smart home devices.

User interface design has higher significance in today's competitive market, it often determines the user whether to choose a particular device or appliance. Users will be interested to buy new devices or smart gadgets if it is very user friendly and compatible with common wireless standards.

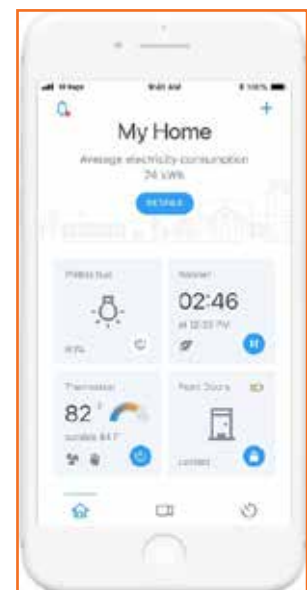


Fig 2.1.6: User interface

2.1.4 Working of an IoT System

First, sensors or devices collect data from their environment. This data could be as simple as a temperature reading or as complex as a full video feed.

Next, that data is sent to the cloud, but it needs a way to get there. The sensors/devices are connected to the cloud through a variety of methods including - cellular, satellite, WiFi, Bluetooth, low-power wide-area networks (LPWAN), connecting via a gateway/router or connecting directly to the internet via ethernet

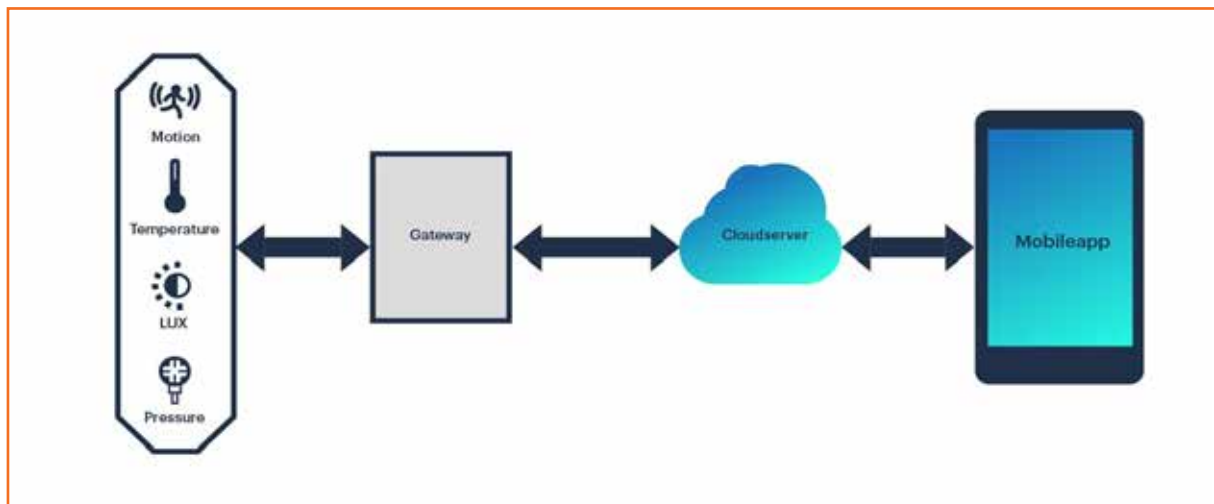


Fig 2.1.7: Working of an IoT system

Once the data gets to the cloud software performs some kind of processing on it. This could be very simple, such as checking that the temperature reading is within an acceptable range or it could also be very complex, such as using computer vision on video to identify objects.

Next, the information is made useful to the end-user in some way. This could be via an alert to the user (email, text, notification, etc). For example, a text alert when the temperature is too high in the company's cold storage.

A user might have an interface that allows them to proactively check in on the system. For example, a user might want to check the video feeds on various properties via a phone app or a web browser.

However, it's not always a one-way street. Depending on the IoT application, the user may also be able to perform an action and affect the system. For example, the user might remotely adjust the temperature in the cold storage via an app on their phone.

2.1.5 Applications of an IoT System

IoT has applications across all industries and markets. It spans user groups from those who want to reduce energy use in their home to large organizations who want to streamline their operations. It proves not just useful, but nearly critical in many industries as technology advances and we move towards the advanced automation imagined in the distant future.

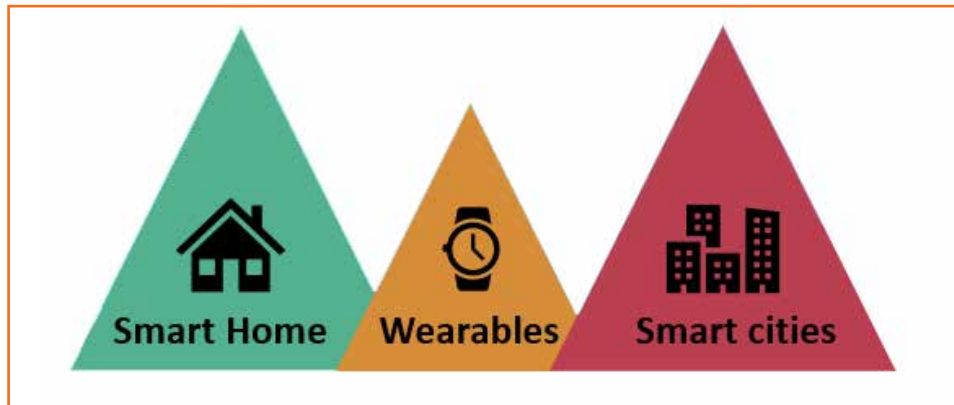


Fig 2.1.8: Applications of an IoT system

1. **Smart Home and Office:** Smart home applications with the use of smart sensors are becoming popular now. Any smart device can be configured and connected to the internet and control using simple mobile application.

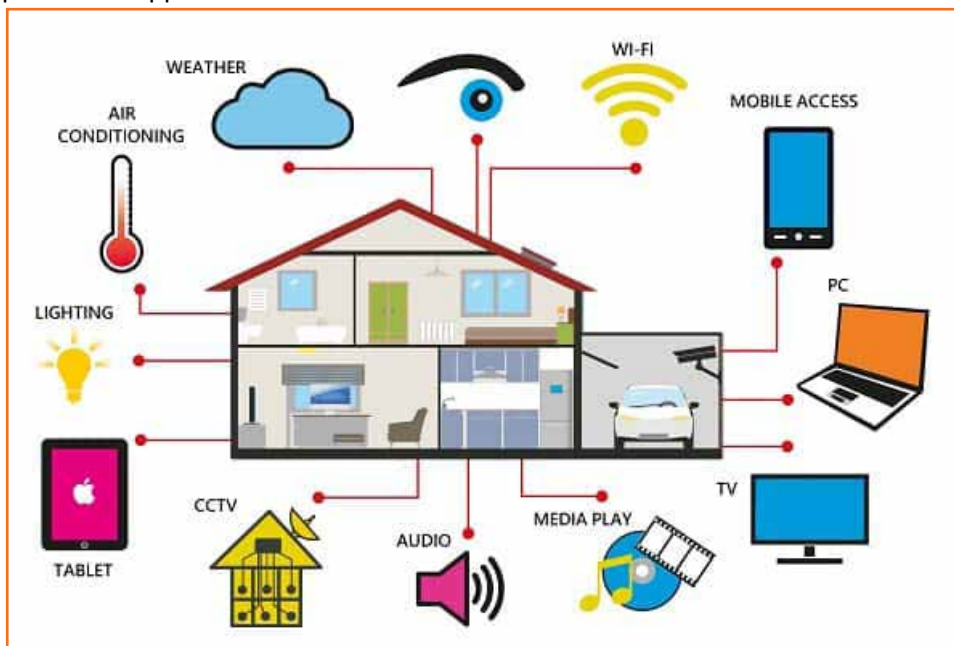


Fig 2.1.9: Applications in smart home and office

Few examples of IoT devices in home are:

- **Smart Door access control system:** Smart locks and door access systems are one of the most popular and cost effective solutions of Internet of Things. Smart locks are easy to implement and control using a web interface or Smartphone application. Integration with RDIF tags, smart door accessing systems can be securely implemented. Users can grant access to the doors using mobile app and lock again once the person leaves the premises.

- **Smart lighting for home and office:** Smart lighting is one of the attractive smart home applications using Internet of Things. In addition to energy saving, it also enables us to manage effectively. Light ambience can be changed using smart hub devices or smart phone app.
 - **Automated Gate and garage:** Using smart sensor technology and Internet of Things, gates and garages can be controlled (operated) conveniently. Once you are about to enter the house or after leaving the premises, you may open or close the gate using mobile devices.
 - **Smart thermostats and humidity controllers:** Smart thermostats are cost-effective and convenient smart home solutions which can be controlled using an Internet connection and smart hub device (or using Smartphone app).
 - **Traffic Management:** Analyzing traffic over a period of time gives an insight of possible trends and patterns that could occur during peak hours. It will help to inform commuters to take alternative routes to avoid congestion and delay.
 - **Pollution monitoring and reporting:** Increasing air pollution is one of the challenges we are facing in every growing city. In order to solve this issue, smart sensors are deployed across the cities to continuously monitor any changes.
 - **Smart Parking Solutions:** Smart sensors installed on parking areas are collecting information about the availability of parking slots and updating it to the database in real time. Once the spot is occupied, it will be updated without any delay.
 - **Water / waste management:** Populations in cities are increasing every year, based on statistics this trend will grow in coming years. Increase in population contributes to an increase in wastes as well.
2. **Wearable Devices:** Wearable smart devices introduced as smart watches around a decade ago and many more functions were added since then. Now our smart watches and wearables are capable of reading text messages, showing notifications of other apps, tracking location, monitoring workout status, reminding schedules and continuously monitoring health conditions.
- With Internet of Things, wearable technology can be used beyond these functions. Major smart wearable manufacturers are developing special operating systems and applications dedicated for smart wearable devices.
3. **Healthcare:** The healthcare industry has been utilizing the possibilities of Internet of Things for life-saving applications. Starting from collecting vital data from bedside devices, real-time diagnosis process, accessing medical records and patient information across multiple departments, the entire system of patient care can be improved with IoT implementation.



Fig 2.1.10: Applications in wearable devices



Fig 2.1.11: Applications in healthcare

IoT will offer convenience for medical practitioners, improve accuracy in the information (helps to reduce error in the data), increase overall efficiency and saves time for each procedures.

4. **Autonomous Driving:** Autonomous driving has been evolving with the use of artificial intelligence and smart sensor technology in Internet of Things. Earlier generation of autonomous vehicle (partial automation) will assists drivers to drive safely, avoid collisions and warn about the conditions of the road and vehicle. Example - cruise control assistance, parking assistance, line changing assistance and efficient fuel /energy management etc.



Fig 2.1.12: Applications in driving

5. **Agriculture and Smart farming:** There are lot of challenges in the agriculture and farming industry to produce more crops and vegetable to feed increasing human population. Internet of Things can assists farmers and researchers in this area to find more optimized and cost effective ways to increase production.



Fig 2.1.13: Applications in farming

Internet of Things is one of the promising solutions to make entire agriculture and farming industry more efficient with less number of workers. Smart sensor technology will help improve each stages of agriculture and automation helps to reduce manual labor.

6. **Industrial IoT for manufacturing:** Manufacturing industry is one of the early adopters of Internet of Things which entirely changed several stages of a product development cycle. Industrial IoT will help optimize various stages of product manufacturing such as



Fig 2.1.13: Applications in industrial manufacturing

- Monitoring of supply chain and inventory management
- Optimization in product development
- Automate mass production processes
- Quality testing and product improvement
- Improves packaging and management
- Process optimization using data collected from huge number of sensor networks
- Cost effective solution for overall management of factories

7. **Disaster management:** Internet of Things with wide range of smart sensors allow engineers to build a more effective emergency response system for factories, schools, hospitals, airports and any other public gathering places. Any emergency situations like fire outbreak or flooding will be automatically detected using sensors and this information is shared to responsible work groups in real time.

During an emergency, fire department, emergency response volunteers, police force, ambulance units and nearby hospitals will receive an alert about the scenario. Automated warning system improves the preparedness and allows authorities to plan and handle any kind of situations immediately.

Some of the common sensors: smoke detector, temperature sensor, humidity sensor, CO2 monitoring sensors and precipitation detector.

8. **Logistic and fleet management:** Smart logistics is a complex task since the goods must be handled with greater care and efficiency. Apart from moving from one location to another location, service providers have to make sure perfect condition is maintained during transportation.

Smart sensors capable of connecting to IoT network continuously monitoring the GPS location, temperature, humidity, shock and tilt angle of the container used for transpiration.

Data collected from these sensors are processed and analyzed in a central cloud system.

9. **Smart grids and energy management:** Smart grid concept is an enhancement of existing power



Fig 2.1.14: Applications in disaster management



Fig 2.1.15: Applications in logistics management

grids with sensors deployed on the transmission lines and individual customer outlets. These sensors help to notify any failure, abnormality in the line, understand the nature of usage and behavior pattern over time.

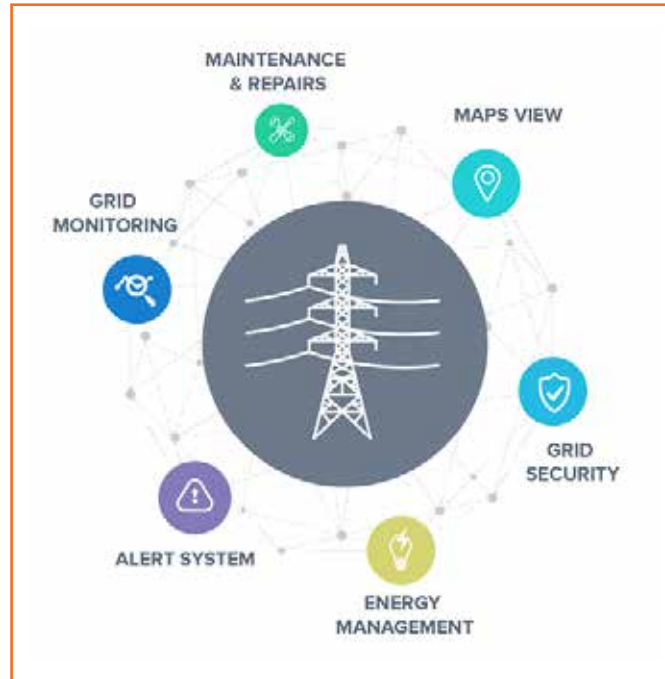


Fig 2.1.16: Applications in energy management

This data can be used to find out areas of improvement, lossy nodes during transmission, and peak time usage statistics with the use of smart meters and sensors. Energy companies can use this information to improve existing grids and implement new changes during upgrade and thus reduce carbon emission.

- 10. Big Data Analytics:** One of the basic components of big data analytics is the data itself; many organizations consider data as most valuable asset to grow their business strategies. The source of data could be from anywhere like machines, environment, plants, peoples or even animals.

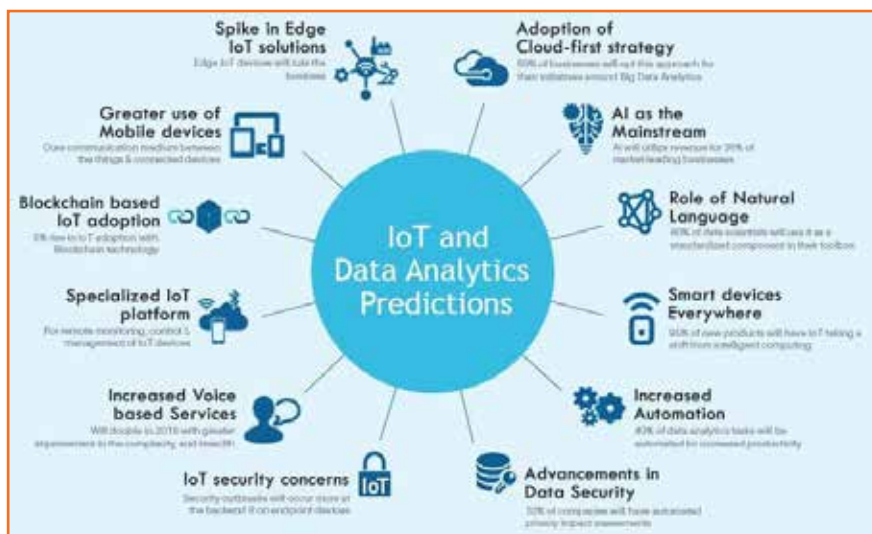


Fig 2.1.17: Applications in Big Data Analytics

Internet of Things uses hundreds of types of sensors designed to collect data from wide range of applications. Huge amount of data from millions of smart sensors will help big data analytics to improve its decision making algorithm using artificial intelligence and machine learning.

Notes



Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=Hl6XAHeX9y0
Introduction to IoT



www.youtube.com/watch?v=Zn4ozz3CkhY
Components of IoT

Unit 2.2: IOT System Architecture and Requirements

Unit Objectives

At the end of this unit, participants will be able to:

1. Describe phases and stages of an IoT system architecture
2. List requirements for building an IoT system

2.2.1 IOT System Architecture

An IoT Architecture is a system of numerous elements such as sensors, actuators, protocols, cloud services, and layers that make up an IoT networking system. It generally consists of differentiated layers that enable administrators to evaluate, monitor, and maintain the system's consistency. As with any system design plan, it also requires a strategy for integration with existing infrastructure and systems.

IoT system architecture is often described as a four-stage process in which data flows from sensors attached to “things” through a network and eventually on to a corporate data center or the cloud for processing, analysis and storage.

Phases of IoT architecture

Four phases in the architecture of IoT –

1. **Networked Devices:** These are the physical devices which include sensors, actuators, and transducers. These are the actual devices that collect and send the data for processing. They are capable of receiving real-time data and they can convert the physical quantities into electrical signals which can be sent through a network.
2. **Data Aggregation:** It is a very important stage as it includes converting the raw data collected by sensors into meaningful data which can be used to take actions. It also includes Data Acquisition Systems and Internet Gateways. It converts the Analog signals provided by sensors into digital signals.
3. **Final Analysis:** This is a stage that includes edge IT analytics and the processing of data to make it more efficient and fully capable of execution. It also includes managing and locating all the devices correctly
4. **Cloud Analysis:** The final data is received here and analysed closely and precisely in data centres. They process and clean the data to make it free from any kind of errors and missing values. After this stage, data is ready to be sent back and executed to perform operations.

Four stages of IoT architecture given below:

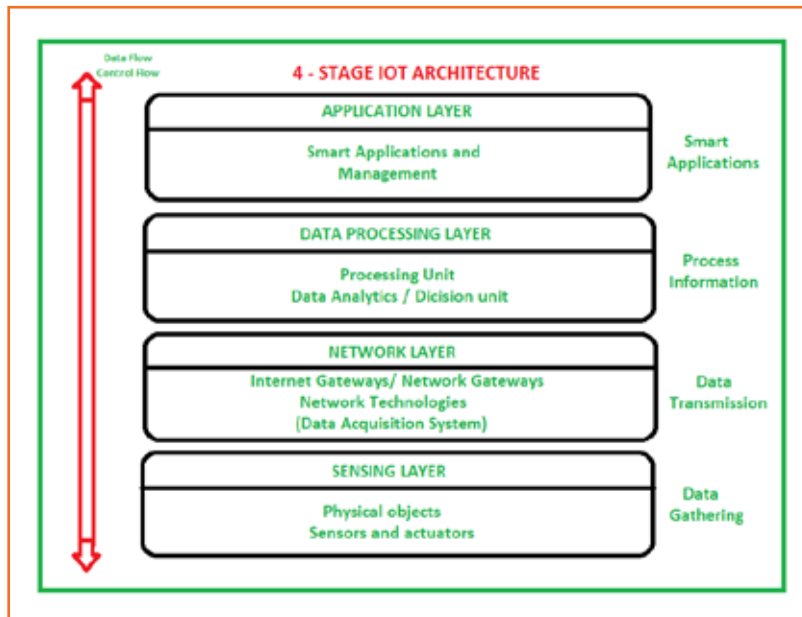


Fig 2.2.1: Four phases of IoT architecture

1. **Sensing Layer (Sensors and Actuators):** The first stage of IoT includes sensors, devices, actuators etc. which collect data from the physical environment, processes it and then sends it over the network.

The process starts with sensors and actuators, the connected devices that monitor (in the case of sensors) or control (in the case of actuators) some “**thing**” or physical process. Sensors capture data regarding the status of a process or an environmental condition, such as temperature, humidity, chemical composition, fluid levels in a tank, fluid flow in a pipe, or the speed of an assembly line as well as much more.

2. **Network Layer (Internet Gateways and Data Acquisition Systems):** The second stage of the IoT consists of Network Gateways and Data Acquisition Systems. A data acquisition system (DAS) collects raw data from the sensors and converts it from analog into digital format. The DAS then aggregates and formats the data before sending it through an Internet gateway via wireless WANs (such as Wi-Fi or Cellular)



Fig 2.2.2: IoT networking

or wired WANs for the next stage of processing. It also performs malware detection and data management.

3. **Data Processing Layer (Pre-processing: Analytics at the Edge):** The third stage of IoT is the most important stage. Here, data is pre-processed on its variety and separated accordingly. After this, it is sent to Data Centres. Here Edge IT comes into use.

Once the IoT data has been digitized and aggregated, it will need processing to further reduce the data volume before it goes to the data center or cloud. The edge device may perform some analytics as part of the pre-processing. Machine learning



Fig 2.2.3: Edge IT

can be very helpful at this stage to provide feedback into the system and improve the process on an ongoing basis, without waiting for instructions to come back from the corporate data center or cloud. Processing of this type will generally take place on a device in a location close to where the sensors reside, such as in an on-site wiring closet.

4. **Application Layer (In-depth Analysis in the Cloud or Data Center):** The fourth stage of IoT consists of Cloud/Data Centres where data is managed and used by applications like agriculture, defence, health care etc.

At Stage 4 in the process, powerful IT systems can be brought to bear to analyze, manage, and securely store the data. This usually takes place in the corporate data center or in the cloud, where data from multiple field sites/sensors can be combined to provide a broader picture of the overall IoT system and deliver actionable insights to both IT and business managers.

Importance of IoT architecture

Administrators use IoT architecture to manage and support IoT devices. IoT devices can be anything from an internet-connected light bulb to pressure safety sensors in a chemical plant.

These devices use small sensors to collect data about their environment and send that data to a server for processing. Servers process this data to create information and insights for businesses. Many times this information is used to automate tasks that improve uptime and efficiency across multiple business systems.

IoT architecture makes this all possible by ensuring data gets where it needs to and is processed correctly. Without proper IoT architecture, networks would become unreliable, defeating the entire purpose of investing in IoT in the first place.

2.2.2 Requirements for Building an IOT Based System

Full-stack IoT development in itself is a set of technology stacks that includes the development of hardware, programming the hardware (Embedded Firmware), Connecting the devices with a communication protocol, and sending data to the central server with IoT data protocols, IoT App development, etc.

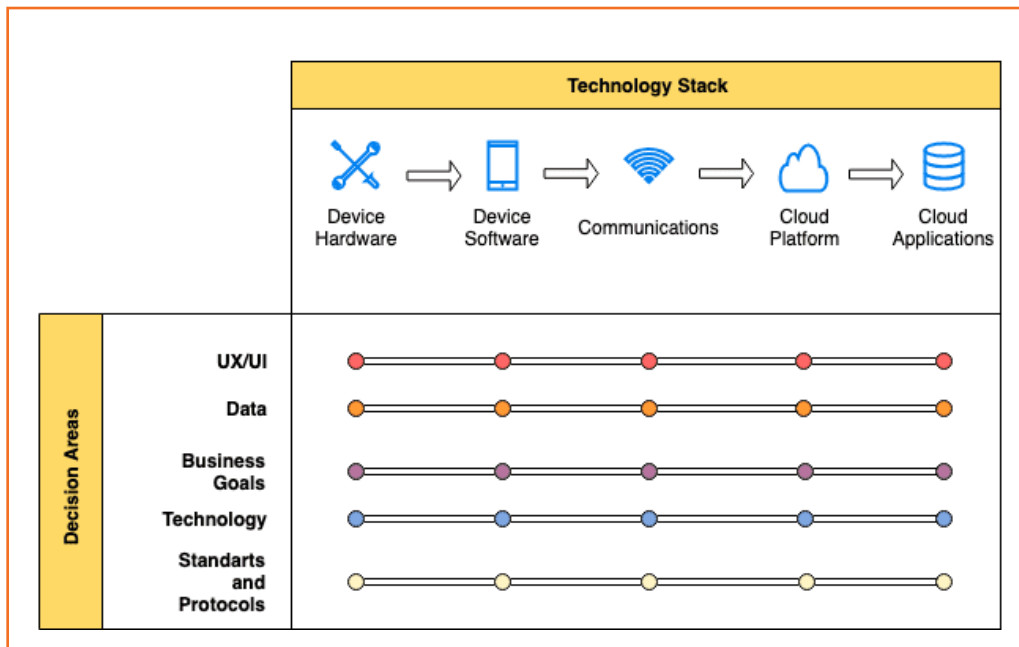


Fig 2.2.4: Requirements for an IoT App

Here are the details of requirements for building an IoT based system:

IOT Hardware (IOT Boards)

An IoT device hardware is a printed circuit board (PCB) that has circuitry and hardware components designed to work with a specific microcontroller. Electronics prototyping is one of the initial steps of IoT hardware development.

Important components of an IoT board

- **Power circuit:** It enables the circuit with the power to run.
- **Programming interface:** It enables you to program the microcontroller from a computer.
- **Basic input:** These are usually buttons
- **Basic output:** These are usually LEDs
- **I/O pins:** For controlling motors, temperature sensors, LEDs, LCD screens, and other peripherals.

Key must-have features for all IoT boards:

- **Processing power:** A microcontroller is quite useful for programming the device as many manufacturers provide the IDE that you require.
- **Wireless capabilities:** It ensures wireless communication without an external transceiver module. Some of the most commonly used wireless technologies include Bluetooth, Zigbee, WiFi, and other LPWAN technologies.

- **Scalability:** With scalability, you can add more functionality to the board. You may want to cross-check if the board communicates via GPIO, UART, SPI, or some other protocol since this will decide how the board interacts with other devices.
- **Memory:** The storage capability depends on the board memory in hardware prominently. To store a lot of data, you need built-in Flash memory. Most IoT boards allow connecting a MiniSD or MicroSD card to increase data storage.

Examples of commonly used IoT boards

- RaspberryPi
- IntelEdison
- Particle
- Arduino

Hardware code or firmware

IoT Firmware development refers to the development of hardware code. The code in the hardware is quite vital to the functioning of the hardware's operation. It interacts with the hardware's electrical and electronic components. For updates in IoT hardware, FOTA should be supported.

Custom Embedded Firmware development requires expertise in programming the hardware along with compatibility with various other peripherals and devices.

IOT cloud platform

IoT cloud platforms leverage the power of IoT devices and cloud computing. In the IoT ecosystem, Edge computing has also been implemented widely. Edge computing and Cloud computing, both are used for processing data but at different places.

The market is replete with a number of robust cloud platforms provided by famous service providers. Here are the most commonly used cloud platforms -

- Google Cloud IoT
- Amazon AWS IoT Core
- Microsoft Azure IoT Hub
- IBM Watson

IoT connections and networks

IoT connection types

- **Device to Device:** direct contact between 2 smart devices;
- **Device to Gateway:** data transfer between sensors and gateways;
- **Gateway to Data systems:** data transfer from the gateway to the data cloud;
- Between Data systems.

IoT network types

- A nanonetwork — a set of small devices (sized a few micrometers at most) that perform very

simple tasks such as sensing, computing, storing, and actuation. Such systems are applied in the biometrical, military, and other nanotechnology areas.

- NFC (Near-Field Communication) — a low-speed network to connect electronic devices at a distance within 4 cm from each other. Possible applications are contactless payment systems, identity documents, and keycards.
- BAN (Body Area Network) — a network to connect wearable computing devices that can be worn either fixed on the body or near the body in different positions, or embedded inside the body (implants).
- PAN (Personal Area Network) — a net to link up devices within a radius of roughly one or a couple of rooms.
- LAN (Local Area Network) — a network covering the area of one building.
- CAN (Campus/Corporate Area Network) — a network that unites smaller local area networks within a limited geographical area (enterprise, university).
- MAN (Metropolitan Area Network) — a big network for a certain metropolitan area powered by microwave transmission technology.
- WAN (Wide Area Network) — a network that exists over a large-scale geographical area and unites different smaller networks, including LANs and MANs.

IoT Protocols

The main protocols that work with IoT dashboards are -

- MQTT (Message Queue Telemetry Transport) is a lightweight protocol that is the most popular for sending simple data flows from sensors to applications and middleware. This protocol functions on top of TCP/IP and includes three components i.e. subscriber, publisher, and a broker. The publisher collects data and sends it to subscribers. MQTT suits small, cheap, low-memory, and low-power devices.
- DDS (Data Distribution Service) is an IoT standard for real-time, scalable, and high-performance machine-to-machine communication. You can deploy DDS both in low-footprint devices and in the cloud.
- AMQP (Advanced Message Queuing Protocol) is an application layer protocol for message-oriented middleware environments. It is approved as an international standard. Its processing chain includes three components that follow certain rules.
- Bluetooth is a short-range communication technology integrated into most smartphones and mobile devices, which is a major advantage for personal products, particularly wearables. Bluetooth is well-known to mobile users.

Right IoT platform for application

An IoT platform is a hardware and software system for managing IoT devices and collecting, storing, visualizing, and analyzing data from those devices. There are many IoT platforms on the market, and their functionality varies enormously. Although all IoT platforms will have a dashboard to display data, some platforms are, in fact, only dashboards, which are only really capable of displaying data from devices.

Note: An IoT Dashboard can be considered a basic IoT platform. A Dashboard can usually display data and control devices.

An IoT platform can usually:

- Collect data from various sources;
- Store data;
- Control Devices;
- Display Data;
- Run Tests;
- Deploy device updates;
- Manage device Inventory.

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=lxTBxZd6jao
IoT architecture



www.youtube.com/watch?v=nw_O23o6Dr0
IoT design methodology

3. Process of Building GUI and Applications in a Framework



Unit 3.1 - Develop Application for IoT System

Unit 3.2 - Develop GUI/web UI for IoT System



Key Learning Outcomes



At the end of this module, participants will be able to:

1. Describe stages of IoT application development
2. List challenges in IoT application development
3. Demonstrate IoT application development process
4. List IoT platform, coding languages and operating system available for IoT application development
5. Demonstrate coding in Python language for application development
6. Describe process of GUI/UI development
7. Describe process of GUI/UI development
8. List various features of GUI/UI
9. Describe various toolkits available for GUI/UI in Python
10. Demonstrate GUI programming

Unit 3.1: Develop Application for IoT System

Unit Objectives



At the end of this unit, participants will be able to:

1. Describe stages of IoT application development
2. List challenges in IoT application development
3. Demonstrate IoT application development process
4. List IoT platform, coding languages and operating system available for IoT application development
5. Demonstrate coding in Python language for application development

3.1.1 IoT Application Development

Internet of Things (IoT) applications are already changing the way we function on a daily basis by relieving us from doing both simple and more complicated tasks. Building IoT applications is also meant to make consumers' lives better, whether it's the concept of a smart home or a wearable band tracking our health rate during different activities.

There are few principles that must be taken into account before creating an application -

- **Ensure the safe collection of data:** The collection of information through special equipment (sensors, etc.) is carried out outside of the usual data transmission networks. Therefore, when developing a custom Internet of Things application, it is very important to think of ways to protect the received information.
- **Organize high-performance data streaming:** The data collection systems consist of hundreds, even thousands of electronic devices. Therefore, for the efficient streaming of such large volumes of information, it is necessary to think over independent mechanisms that are different from traditional packet transfers.
- **Create or select an appropriate IoT platform:** The IoT platform is a set of software tools that collectively help to systematize, store and process data received from electronic devices.
- **Develop an IoT solution in the cloud:** In order to provide fast delivery of processed data to a user device, and also to organize centralized storage, cloud solutions are usually used. Such systems can ensure the efficient operation of the IoT application with minimal operating costs and requirements for carrier networks.
- **Provide for effective data management:** In-memory analysis and data processing systems are most often used. Such solutions ensure the rapid delivery of the processed results to the end user, even in the event of the data collection devices' failure.



Fig 3.1.1: Tips for IoT app design

Stages of developing an IoT application

1. **Plan the Framework of App:** The first stage is to plan the architecture of the IoT app, decide what features will be needed, how the UX will work, how the app will interface with the device system software, and how to transfer data between the devices and IoT app. Attention must be paid to taking control of security and encrypting the data using communication layer security within the app such as TLS or DTLS.
2. **Design the App based on user journeys:** At this stage, we design the app. In other words, you need to plan for future security systems, overloads, and design the app to align with user journeys.

In order to accurately define the user journey(s),

- You need to do some research.
- You need to talk to the end-users or customers and discover exactly what they want from the app.

In cases where there is more than one type of end-user, you should take the time to research what the differences are - maybe one type of user just wants raw data, e.g. an operator wants to plug the data into a CNC machine, whereas another user may want data visualizations and insights, e.g. a factory manager.

It is highly critical that the IoT app has a top-tier UI and UX. To achieve a good UI/UX, you need to make sure that all the different sensor responses and various features and services are presented in a way that is spontaneous and easy to use.

The most important thing to consider when developing the interfaces is user-friendliness. A good app is one that can be used with little to no training or explanation. Ideally, users should be able to open the app and get exactly what they need within a few clicks.

Once the user journeys have been identified and mapped out, you need to decide upon the main features and services - dashboard, analytic displays, alerts and notifications, QR or barcode scanner, messaging features, preferences, and controls, etc.

3. **Develop, implement and deploy the App:** The next stage of IoT app development is to actually build an IoT app. It is a stage of project implementation where all the previously alleged concepts of app architecture and its designs are realized. At this stage, you have to develop a functional app and receive the exact interface with all the actionable menus, lists, and forms.

The major challenges while creating the app are its correct merge with all the selected IoT devices, third-party services integrations, and implementation of the security management system and security protocols. The team of developers should ensure all the data is properly transferred and stored.

4. **Testing & Integration:** Once the app has been deployed it needs to be fully tested to make sure that all the features and services work as desired, there are no bugs or glitches, and the user experience is good. Various test cases should be applied and test data examined to make sure that all user journeys perform well.

Once the testing has been completed and any problems ironed out, it's time to integrate the app into the overall IoT system. This usually involves setting up the cloud gateways and networks for the transfer of data and meshing them with the app's communication layer security protocol.

Once connected and integrated, further tests should be carried out to make sure that data integrity and security is maintained throughout the system.

5. **Maintain and Improve:** The final stage for an IoT application is the maintenance phase. Maintenance involves checking and adapting the software, system hardware, and technical specification to improve overall performance.

Challenges of IoT app development

Every development project comes with technology-specific considerations. For IoT app development, many challenges stem from the differences between IoT and the traditional technologies, such as PCs and smartphones, that use web or mobile apps.

- IoT devices have less in-device computing power and storage than these other technologies. App developers must consider these limitations and how they will influence the interaction between applications and the IoT devices. For example, latency may increase because data takes more time to transfer from a remote device to the cloud and then undergo processing.
- Developers need to consider IoT security as an afterthought, which means problems are often discovered after an application is already in use. Such instances could put users or the public at risk. Encryption protocols, user authentication options and thorough testing can help make an IoT application as secure as possible.
- In addition to IoT app security, developers must consider data security. For example, developers should know when an application will collect data, and where that information goes once generated.

- Developers must also strive to make the app user-friendly and stay mindful of its purpose. For example, the user-facing parts of an application would vary depending on if the product is a smart insulin pump or a temperature monitor for an industrial freezer.
- Compatibility with various brands is a concern, primarily if an organization wants to build an app for consumer use. If a business creates a smart home hub application, the developers must consider how someone with an iPhone or Android will connect to the hub and ensure the application still works.

3.1.2 Process of IoT Application Development

For developing an app for IoT, you need to take into account behind-the-scenes but at the same time robust back-end systems.

The back-end (server-side) components of an IoT system collect and process data coming from sensors and actuators installed in smart gadgets, for that you will need a web portal to administer smart devices.

That's where custom web application development for IoT comes into play. That's because a back-end piece of an IoT puzzle and an admin portal for managing "things" are web applications and naturally imply web development.



Fig 3.1.2: Backend system requirements

Let's discuss steps for IoT development:

Step 1: To build from scratch or not?

The first thing you need to decide is whether to custom build a platform or use an off the shelf application ennoblement IoT platform.

1. **Custom build option:** It goes without saying that IoT development is complex and the technical barrier to entry for those looking to build their platform from the ground up is very high. If you select this option, You need to design, build, test, and maintain each part of the IoT stack, including -
 - Server deployment and maintenance
 - Database build and maintenance
 - UX and UI build
 - Developing 3rd party connections and APIs
 - Security considerations
 - Access control
 - Front end build and maintenance

2. **Off the shelf IoT application enabled platform (AEP):** An AEP is a self-contained IoT environment that developers can utilize to build and deploy IoT products and services quickly. Well, known AEPs include -

- Azure IoT
- AWS IoT
- IBM Watson
- Oracle IoT
- Kaa

Benefits of AEP are:

- Many commercial and open-source IoT platforms exist on the market, helping you complete cloud infrastructure for your IoT application quickly.
- These solutions take care of a backend data processing and some offer a toolchain to easily create an admin portal for managing smart devices.

Companies offering off-the-shelf IoT platforms have made many tech decisions for you. Therefore, check them diligently if your product can successfully operate within these technological boundaries.

Step 2: Secure your iot app from the get-go

When you develop IoT applications, it’s rarely a single mobile application to control hardware, rather a mini-ecosystem comprising lots of different elements. That’s why your IoT app must include a multi-layered approach to security from start to end.

That’s why your IoT product must include a multi-layered approach to security.

Here are some key aspects to consider:

User authentication

- Install a strong password policy by disallowing weak passwords
- Enable two-factor authentication with text messages or authentication apps
- On mobile devices, allow users to authenticate quickly with Face ID or Touch ID
- Require users to reauthenticate before providing access to personally identifiable information
- Require customers to authenticate again after they remain inactive for some time
- Avoid hardcoded passwords or duplicate passwords across smart devices



Fig 3.1.3: User authentication

Data encryption

- Ensure that sensitive information is securely encrypted in transit and at rest
- Use well-recommended encryption frameworks to enable strong encryption instead of custom cryptography implementation
- Verify that the same encryption mechanism is implemented across the web and mobile components of your IoT product

Access rights management

- **Enforce the concept of least privilege:** every platform module and the user should have access to the required minimum of data
- Limit access to device debug capabilities only to approved staff, and keep a log of all access instances
- Implement a data wipe feature to efficiently erase all data from a device when it's decommissioned or changes the owner
- General security best practices
- Use the secure HTTPS connection to transfer data between devices and applications
- Verify that your security measures comply with such standards as ISO27001, SOC2 Type 2, and IEC 62304 (if it's a healthcare-related IoT app)



Fig 3.1.3: Access rights management

You need to maintain the balance between these security requirements and the ease of use. In other words, these app protection techniques should not stand in the way of customers interacting with the application.

Step 3: Develop and test all the components of iot product

Once you've decided on a custom vs. off-the-shelf IoT platform and reviewed the security options, now you have to create apps for IoT by following the standard steps -

- Design, prototype, and verify a prototype with customers
- Translate the design into code using appropriate tools
- Test the application in a live environment

There are a couple of things you need to account for because, when building an ecosystem that spans multiple web and mobile platforms.

UX/UI

When designing IoT apps, your emphasis should be on customer-facing components, such as a mobile app and a web interface for visualizing data coming from connected devices.

Some ready-made IoT platforms provide essential building blocks for creating customer-facing applications, so you need to verify your UI/UX assumptions with customers by using interactive prototypes before cementing these user experiences.



Fig 3.1.4: Web interface

You can safely stick with the default UI for an admin portal, as it can be managed internally.

CODE

Now start to create IoT applications by putting together lines of code. There are plenty of frameworks and pluggable components on the market to accelerate the development of mobile and web applications.

Prioritizing the development of a backend, and APIs in particular, pays off in that development of mobile applications can proceed at a steady pace, without setbacks, in parallel with web development.

Because you need to integrate apps with the cloud using real-life APIs instead of relying on stubs and mock data. If they continue to build apps for IoT while the APIs are still being developed, there's a high chance the mobile app's architecture will need a significant overhaul later on.

QA

IoT development implies much testing. A solid QA strategy includes adding automated tests in code during development, besides checking app features from the customer perspective.

At a minimum, include the following QA disciplines in testing strategy:

- Security testing
- Performance testing
- Usability testing
- Compatibility testing
- Scalability testing

You can also step up your testing procedures by employing specific Internet-of-Things testing tools like Wireshark or mPulse.

Step 4: Deploy

After developing and testing IoT product, you are now ready to make it available for customers. You will need to transfer web apps to a live server environment, upload mobile apps to the App Store and Google Play and go through the app verification process.

Step 5: Maintain

After releasing your app, now you need to analyze customer behavior based on data coming from built-in analytics solutions like Google Analytics or Flurry.

You need to keep track of updates to the frameworks employed in the product and watch new mobile OS releases to spot opportunities for enhancing user experience.



Fig 3.1.5: App performance analysis

3.1.3 IoT Platform

IoT is a network of devices which are connected to the internet for transferring and sensing the data without much human intervention, the platform used is termed as the IoT platform. IoT platform serves as a middleware that manages the interactions between a user application and remote devices. These platforms consist all the required capabilities for the cloud support and other needs which is needed to satisfy the IoT technology.

IoT platform is a key part of a large IoT ecosystem, which promotes and links all elements in the scheme. It allows device management, handles communication protocols on software and hardware, collects / analyses information, improves information flow and intelligent apps functionality.

List of ready to use IoT platform

1. **Microsoft Azure IoT:** Azure IoT Suite is an incredibly popular software package from Microsoft, specifically created for the simple integration of information collection devices into a consolidated system for transferring, storing, analyzing and processing data.

This IoT solution is designed for different industry needs. It can be used from manufacturing to transportation to retail. It provides solutions for remote monitoring, predictive maintenance, smart spaces, and connected products.

Features:

- It provides you with an open platform to build a robust application.
- It can be used by beginners as well as experts.



Fig 3.1.6: Azure IoT platform

- There are two solutions to start with, as an IoT SaaS and with open source IoT Templates.
2. **Google Cloud Platform - IoT framework:** Things can be done by Google. Google Cloud is one of the best IoT systems available today with its end-to-end platform. Google stands out from the others because it can process the large quantity of information using Cloud IoT Core. Due Google Cloud provides a multi-layered secure infrastructure. It helps in improving operational efficiency. It provides predictive maintenance for equipment, solutions for smart cities & buildings, and real-time asset tracking.

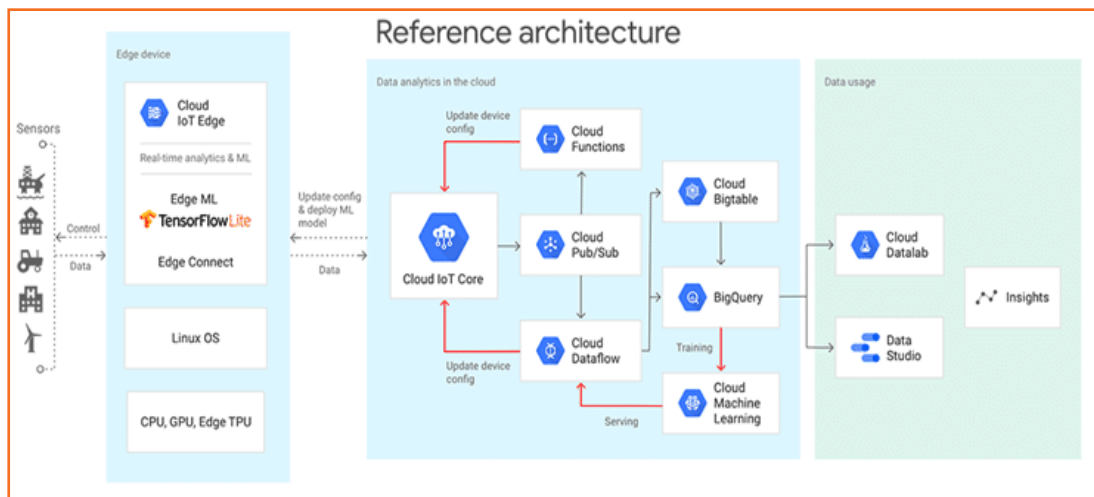


Fig 3.1.7: Google cloud IoT platform

Features:

- Machine learning capabilities for any IoT need.
 - Real-time business insights for globally dispersed devices.
 - AI capabilities.
 - Provides support for a wide range of embedded operating systems.
 - Location intelligence.
3. **IBM Watson - IoT framework:** IBM Watson is very popular among the internet of thing platform among developers. The artificial intelligence based IBM Watson software implements support for a reliable relationship between information collection devices, servers and user parts of the developed applications. The Bluemix hybrid cloud-supported Watson IoT platform allows developers to use IoT-applications easily. IBM Watson manages the secure communication and also data storage. Real-time data exchange also is done by IBM Watson.

This platform will help you to capture and investigate the data for devices, machines, equipment and find out the understandings for better decisions.

This platform will allow you to optimize operations and resource. By providing the correct business insights and bidirectional communication facility, it will help in increasing the revenue to a great extent.

Features:

- AI and Analytics.
 - Domain expertise.
 - Provides flexible solutions.
 - Provides security.
 - Captures real-time data.
 - Provides analytics service as an add-on.
4. **Amazon Web Services (AWS) IoT framework:** Amazon Web Services (AWS) is an IoT platform provided by Amazon. This IoT platform provides cloud computing, database, and security services through the AWS Console. There are so many other services such as Regions, Availability Zones, and Virtual Private Clouds (VPCs).

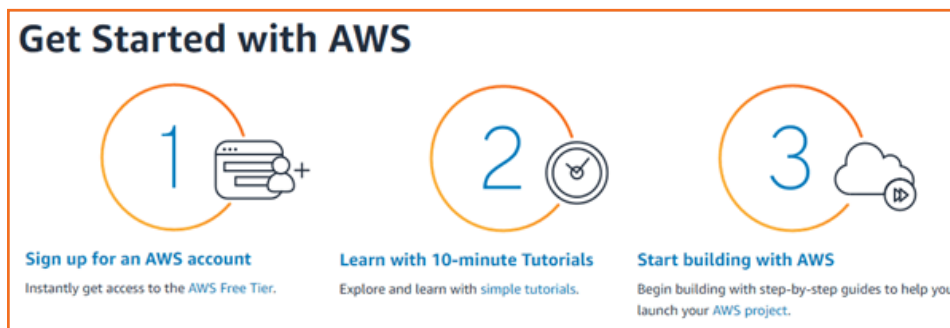


Fig 3.1.8: AWS IoT platform

It is a managed cloud service. AWS IoT Core will allow devices to connect with the cloud and interact with the other devices and cloud applications. It provides support for HTTP, lightweight communication protocol, and MQTT. It provides Registry for recognizing devices, Secure Device Gateway, Compatible Software Development Kit for devices which AWS partnered with HW manufacturers like Intel, Texas Instruments, Broadcom and Qualcomm.

Features:

- It can process a huge amount of messages.
 - It is a reliable and secure platform to route the messages to AWS endpoints and other devices.
 - Your applications will track and communicate even when not connected.
 - You will be able to use other AWS services like AWS Lambda, Amazon Kinesis, and Amazon QuickSight etc.
 - It allows secure access to your devices
5. **Oracle IoT:** Oracle IoT is one of the leading software solutions for the development of Internet of Things applications, built over one of the most flexible programming environments - Oracle. Based on cloud computing technologies, applications created with Oracle IoT have a whole host of advanced capabilities, including device virtualization, high-speed messaging, endpoint management, stream processing, data enrichment, event storage, REST API support, and enterprise connectivity.

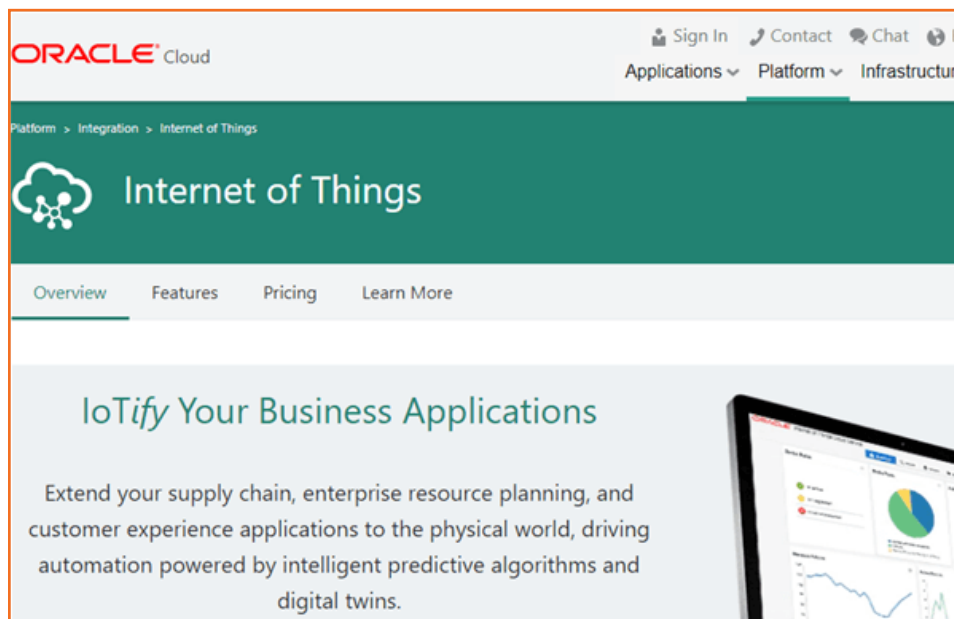


Fig 3.1.9: Oracle IoT platform

With the help of Oracle IoT cloud, you can connect your devices to the cloud, perform analysis of data from these devices in real time, and perform integration of data with enterprise applications or web services. It supports integration with Oracle and non-oracle applications and IoT devices using REST API.

Features:

- It will allow you to create an IoT application and connect a device to JavaScript, Android, iOS, Java, and C POSIX.
- It will help you to extend the supply chain, ERP, HR, and customer experience applications.
- Operational efficiency and worker productivity will be improved.
- It provides features like device virtualization, high-speed messaging, and endpoint management to connect.
- To analyze the data, it provides features like stream processing and data enrichment.
- Using REST API, integration can be done with Oracle and non-oracle applications and IoT devices.

6. **Cisco IoT Cloud Connect:** Cisco IoT Cloud Connect provides robust, automated, and highly secure connectivity for the enterprise. IoT data management is done by the Cisco Kinetic IoT platform to extract, move and compute the data.

Cisco IoT cloud connect is a mobility cloud-based software suite. This IoT solution is for mobile operators. It will fully optimize and utilize the network. Cisco provides IoT solutions for networking, security, and data management.

Features:

- Granular and real-time visibility.

- It provides updates for every level of the network.
 - For IoT security, it provides benefits of protecting the control system from human errors & attacks, increased visibility & control by defending malware and intrusion, and centralized security controls.
7. **Salesforce IoT:** Salesforce is power by thunder. Thunder allows companies to unlock earlier unseen ideas and allows anyone to take proactive, personalized activities from any device to bring their clients closer than ever. Many businesses now develop their apps or migrate to Salesforce on the Salesforce platform. This has raised demand for developers and administrators from Salesforce.

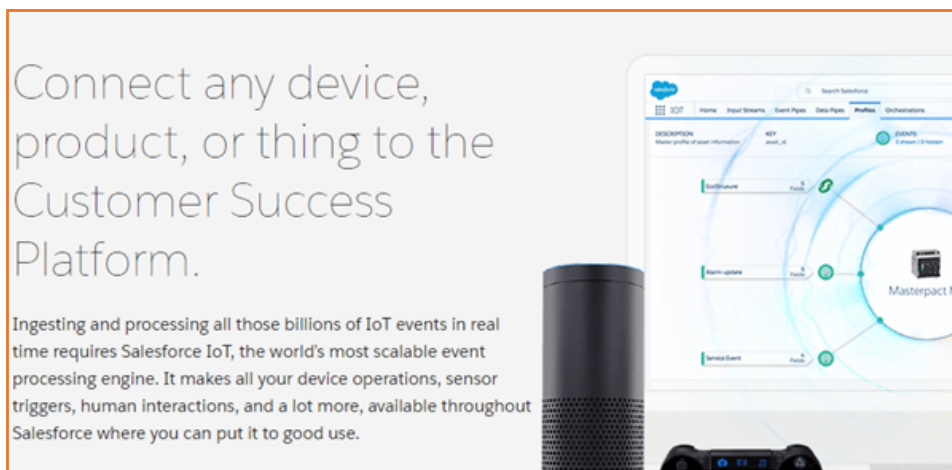


Fig 3.1.10: Salesforce IoT platform

Sales force IoT cloud will help you to transform all data which is generated by the customers, partners, devices, and sensors into relevant actions. It has partner connectors like AWS, Cisco Systems, etc.

Features:

- It allows you to test business ideas without programming.
 - It will provide you the real data about the product usage and performance.
 - It can work with the data from any device.
 - You can create device profiles for customer context data in CRM and for streaming data from the connected devices.
 - Using RESTful API, you can import data from any source.
 - No need of CS degree while creating and managing orchestration rules.
 - Real-time traffic view.
8. **SAP IoT:** The SAP Internet of Things cloud platform has everything you need to build and handle an IoT application. The SAP platform provides a convenient environment to remotely manage and monitor all connected devices of your IoT system. In the SAP Platform, a remote-devices we can connect directly or through cloud service. Obviously, SAP can use IoT information to create machine learning and artificial intelligence applications while maintaining recent technological trends.

Some prominent features of the SAP IoT cloud platform include:

- Building AI apps using IoT data,
 - Reliable message processing,
 - Secure data management,
 - Easy integration of IoT protocols with legacy systems, etc.
9. **KAA IoT:** Kaa IoT is one of the most effective and rich Open Source Internet of Things Cloud Platforms, where anyone can freely implement their smart product concepts. KAA IoT has a lot of advanced features; among them, a well-thought-out functionality for the adjustment of mobile device compatibility, flexible management of an unlimited number of sensors for the collection of information through an SDK server, real-time sensor monitoring, cloud services, automation of software updates, automated user personal device settings distribution, etc. All these features, when combined, make KAA IoT one of the most advanced products for the development of this kind of software.
 10. **ZETTA IoT:** Zetta is nothing but a server-oriented platform developed based on the REST, NodeJS, and the Siren hypermedia-API-strip flow-based reactive programming philosophy. After being abstracted as REST APIs they are connected with cloud services. These internet services include tools for visualizing machine analytics and support such as Splunk. It builds a geo-distributed network through connectivity with systems like Heroku to endpoints like Arduino and Linux hackers.
 11. **Hewlett Packard Enterprise: IoT framework:** Hewlett Packard Enterprise's universal business platform offers scalability for its customers by offering solutions to most of their problems. The platform provides cloud-based assistance or local support. In smart cities and the automobile industry, HPE universal of things platform was used properly. The data monetization of several businesses has been carried out by HPE. Hewlett Packard Enterprise Collects analyzes information in order to grow the company.

Things to consider when choosing an off-the-shelf IOT platform

1. Check what data protocols their systems support -
 - MQTT
 - HTTP
 - WebSocket
 - DDS

All these protocols have their own rules on circulating data between hardware sensors and the cloud.

2. Another aspect worth checking out about ready-made IoT platforms is the network protocols they work with -

- Bluetooth
- WiFi
- LoRaWan
- Zigbee
- Z-Wave

The more protocols a platform supports, the better. You don't want to be locked into some



Fig 3.1.11: Wireless network for IoT

proprietary data transfer technology because this means you can't quickly or effectively move to a completely different provider or switch to a custom-built IoT environment.

3. **Examine their cloud architecture.** Does it allow for any flexibility? The ideal scenario is when an IoT system allows for various deployment scenarios -

- Cloud hosting setup
- On-premise option (when you host IoT running software on your own servers)
- a hybrid approach, combining the two said variants

4. If you also want to connect your IoT system with other existing solutions, e.g., a CRM or ERP, then also look for open APIs support from these platforms.

5. Check their scalability options.

- Will you be able to quickly double or triple the number of smart devices for your customers?
- How much data can your product process at any given time?

6. Hardware support and firmware updates provided by platform. Some platforms only work with designated hardware, while others function as platform-agnostic solutions. You will need to determine if the IoT provider supports OTA (over-the-air) firmware updates that drastically simplify device management.

7. Finally, double-check if an IoT platform of your choice includes prolific security mechanisms. In fact, the security aspect of internet of things app development is so critical.



Fig 3.1.12: Hardware support required

3.1.4 IoT Development Board

A development board is a printed circuit board with circuitry and hardware designed to assist experimentation with a certain microcontroller.

Typical components of a development board are:

- Power circuit
- Programming interface
- Basic input
- Basic output
- I/O pins

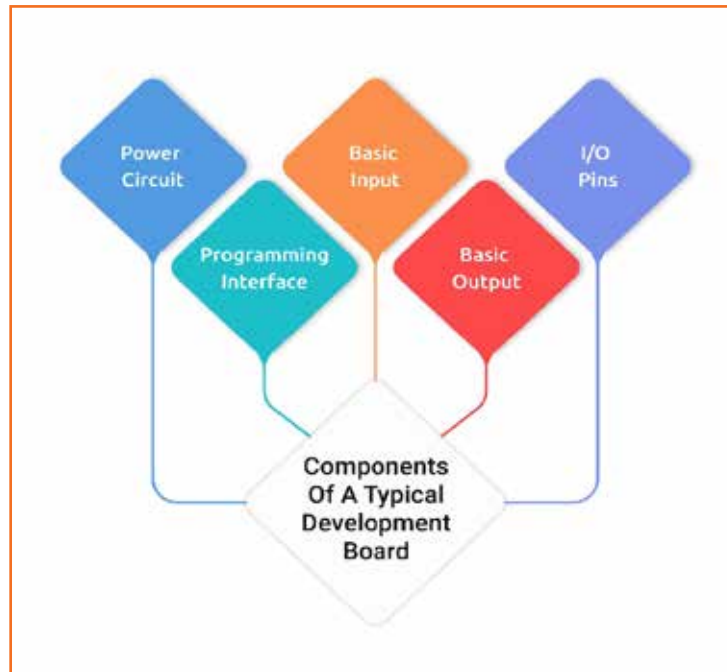


Fig 3.1.13: Components of a IoT board

Key features of development board

- Processing power
- Wireless capabilities
- Scalability
- Memory



Fig 3.1.14: Features of a IoT board

Categories of IoT Boards

1. **Microcontroller-based boards:** Even if you are new to programming and electronics, you must have come across the term “**microcontroller boards.**” They comprise a small computer developed on a metal oxide semiconductor circuit chip, and are mainly used in implantable medical devices, office machines, power tools, automobile engine control systems and so on.

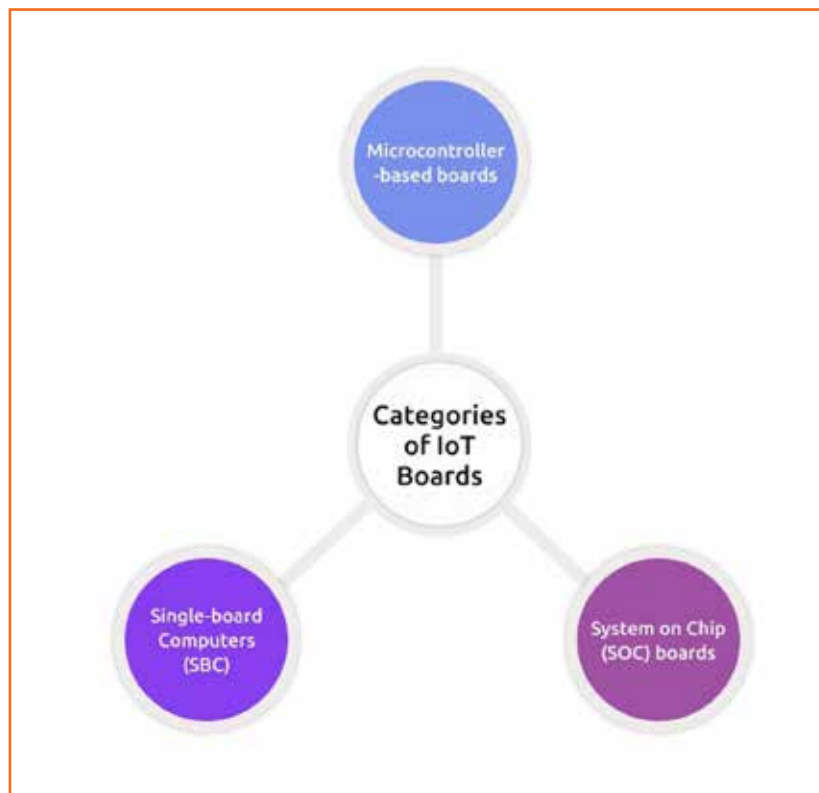


Fig 3.1.15: Categories of IoT boards

2. **System-on-Chip (SoC) boards:** A SoC board incorporates many system components into a single Si chip, primarily consisting of an audio receiver, memory, peripherals like USB, PCI and SATA, a microprocessor, and an application processor — basically all the required electronic components and circuits.

They find use in a number of industry niches, including medical care where SoC-based nano robots can perform as programmable antibodies to fight serious illnesses. SoC-based video and audio devices can be installed in the brains of blind and deaf people respectively.

3. **Single-board computers (SBC):** This is an easy one. As the name suggests, it is an entire computer developed on one circuit board. It comprises all the features of a functional computer such as memory, microprocessor(s), input/output (I/O) and so on.

SBCs are usually built as development or demonstration systems. A huge variety of portable computers and home computers integrate all their functions onto one circuit board easily. SBC designs are simple and they do not often rely on expansion slots for peripheral expansion.

List of most popular IoT Development Boards are:

1. **Raspberry Pi:** The raspberry pi Development Board is a small credit card size computer. That works on Linux based operating systems and is good for embedded projects. Raspberry boards can be easily plugged in to your monitor, computer or TV. It uses a standard keyboard and mouse. Even amateur users depend on it for configuring their digital media systems and surveillance cameras.

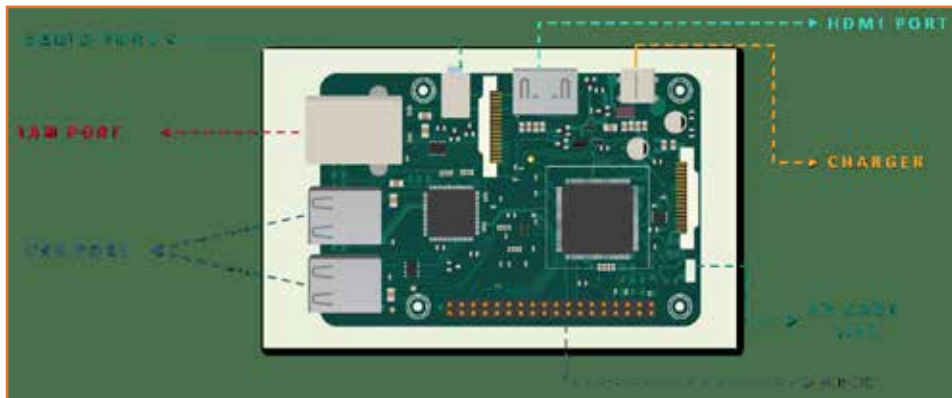


Fig 3.1.16: RaspberryPi board

Features:

- **Processor:** 1.2GHz, 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- videoCore IV 3D graphics core

2. **Arduino Nano 33 IoT:** The Arduino Nano 33 IoT is a dual-processor device that is perfect for experimentation. It offers a practical and low-cost solution for inventors seeking to add Wi-Fi connectivity to their projects with minimal previous experience in networking. The board is compatible with the Arduino IoT Cloud, where you can create IoT applications in a few simple steps.



Fig 3.1.17: Arduino Nano 33 board

Features :

- ARM Cortex-M0 32-bit SAMD21 processor
- 14 digital I/O pins and 8 analog input pins
- Support up to 12-bit ADC/PWM and 10-bit DAC resolutions.
- **Can operate as a few different USB devices:** (asynchronous serial, keyboard or mouse) also referred as HID, and USB MIDI.

- Can communicate via Synchronous serial communications.
- Inbuilt real-time clock module.

3. **Tessel 2:** It's a kind of System on Chipboards.

With WIFI capabilities, it allows you to build scripts in Node.js. The board also provides you with a connected hardware prototyping system that can be used in multiple different applications. Loaded with on-board features including two 10-pin module ports to add sensors and other external hardware, a 10/100 supported ethernet port, 2 USB ports for camera peripherals and flash storage, and a microUSB connector for power and tethered programming.



Fig 3.1.18: Tessel 2 board

Features:

- 2 USB ports (you can connect cameras or flash storage, for example)
- 10/100 ethernet port
- 802.11 b/g/n WiFi
- 580MHz Mediatek router-on-a-chip (you can turn your Tessel 2 into an access point!)
- 48MHz SAMD21 coprocessor (for making I/O faster)
- 64MB DDR2 RAM, 32MB of flash (lots of space for your programs and stuff)

4. **Omega 2:** Omega 2 is one of Onion's Linux-based WiFi development boards that allow makers of all skill levels to build connected hardware. This highly integrated board comes with a powerful processor and flexible GPIOs. The Platform lets you prototype hardware devices using familiar tools like Git, npm, pip, as well as high-level programming languages like Python, Javascript, and PHP.



Fig 3.1.19: Omega 2 board

Features :

- Linux Operating System, powerful processor, and flexible GPIOs.
- Compact size that easily fits into any project design.
- Modular design for a vast range of flexibility.
- Arduino compatible.
- Integrated Wi-Fi;
- Connectivity is expandable with 2G, 3G, Ethernet, Bluetooth®, Bluetooth Low Energy (BLE), GPS.
- U.FL Connector for external Wi-Fi antenna attachment.
- FCC and CE Certified.

5. **Particle Photon:** Particle Photon Board consists of an STM32 microcontroller, Wi-Fi, Switches, and LEDs. Simple to use, powerful, and connected to the cloud. Powered by a Cypress Wi-Fi chip alongside a powerful STM32 ARM Cortex M3 microcontroller, it is ideal for prototyping IoT projects.

Features :

- **Processor:** STM32F205 120Mhz ARM Cortex M3
- Real-time operating system (Free RTOS)
- **Memory:** 1MB flash, 128KB RAM
- Open source design
- On-board Wi-Fi module
- On-board RGB status LED.
- 18 Mixed-signal GPIO and advanced peripherals
- Soft AP setup
- B802.11b/g/n Wi-Fi
- roadcom BCM43362 Wi-Fi chip

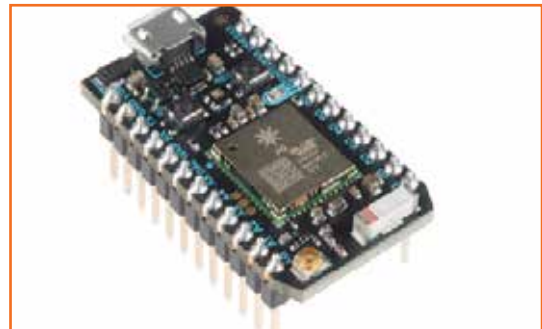


Fig 3.1.20: Particle photon board

6. **Beagle Bone:** The Beagle bone is a low power open-source single-board computer produced by Texas instruments. The board can boot Linux in under 10 seconds also you can start developing in less than 5 minutes with just a single USB cable.

It is a computer installed inside of a larger electronics project. The beagle board carries two rows of GPIO (general purpose Input/Output) pins

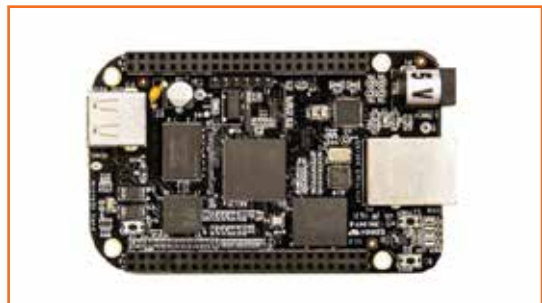


Fig 3.1.21: Beagle bone board

mounted along each side of the board. That allow it to communicate with a wide range of servos, sensors, outputs and other hardware, making it act as the brain of large & complex projects.

Its capabilities can be extended using plug-in boards referred to as “**capex**”. that are easily available for LCD, motor control, VGA, prototyping, battery power, and other functionalities.

Features :

- **DDR memory:** 512 MB
- Ability to run Ruby, Python, and INO Sketches directly in the Cloud9 IDE,
- **Ethernet:** On-chip 10/100 Ethernet
- **JTAG:** Optional
- **Memory:** 4GB eMMC memory
- **Power Options:** Via USB or 5V DC input

- **Price (USD) Per Unit:** \$55.00 (Suggested Retail Price)
- **Processor:** 1GHz AM3359 Sitara ARM Cortex-A8

7. **ESP 32:** ESP32 is a dual core low-footprint system development board powered by the latest ESP-WROOM-32 module that can be easily placed into a solderless breadboard. It has a pre-integrated antenna, power amplifier, low-noise amplifiers, filters, and power management module. Because of this, it's easy to build and test circuits as well as making projects related to IoT integrating with the cloud platform.



Fig 3.1.22: ESP 32 Board

Features :

- GHz dual-mode Wi-Fi.
 - Programmable with Arduino open-source IDE.
 - 8 independent LED.
 - Bluetooth chips by TSMC.
 - 40nm low power technology, power, and RF.
 - Easily embedded with other products.
 - Strong function with support LWIP protocol.
 - **Supports three modes:** AP, STA, and AP+STA.
 - Supporting the Lua program, easily to develop.
8. **Banana Pi:** Banana Pi is a line of low-cost credit card-sized single-board computers(SBC). IT is a router-based development board, which efficiently runs on various open-source operating systems including OpenWRT and Android, Lubuntu, Ubuntu, Debian, and Raspbian. Well, the hardware design of banana pi was influenced by the Raspberry Pi and it is compatible with Raspberry Pi boards.



Fig 3.1.23: Banana Pi board

Features :

- All winner A20 Dual-core 1.0 GHz CPU
- Mali-400 MP2 with Open GL ES 2.0/1.1.
- 1 GB DDR3 memory.
- 1x Gigabit LAN
- 1x SATA interface.

- 1X MIC
- 1x USB otg and 2x USB 2.0
- HDMI out
- Composite video out
- CSI camera interface
- DSI display interface
- 26 PIN GPIO

Selection the right IoT development board

1. **Confirm the development board type:** First confirm your project type. Is it visual-focused or sensing-based? Whatever it is should dictate your choice. There is a possibility you might have to use more than one type of boards. For instance, pairing a Raspberry Pi with Arduino Uno would fetch you better results.

Therefore, research thoroughly before making any decision because it is going to stick with you throughout your project, and you do not want it to hamper your progress.

2. **Observe the programming language and community:** What programming language you use is an important factor to consider in the IoT development board. For example, you could need support for multiple languages, OS and IDEs to create a rich and wholesome experience.

C and C++ are universal programming languages and many boards support them or a similar language like them. On the other hand, IDEs for some boards will allow you to experiment with more than one programming language.

Then there are single boards that usually run on Linux. For instance, the most popular OS for the Raspberry Pi is Raspbian. You also need to ensure you have a strong community of the programming language to fall back on.

3. **Check the specifications of the board:** The thing that the specifications of development board have been to be larger is wrong.

Also, the board with more specifications usually costs more,

3.1.5 Coding Languages for Application Development

IoT devices use software that includes instructions for them and is coded using a programming language. Coding is required for establishing a network. The frontend and backend of an IoT device are also built with coding.

IoT uses many programming languages to make a successful module. The devices are just the hardware that needs software to operate that has instructions in it. The following languages are used in IoT to instruct the module for a particular task. And every language has its worth and achievement in IoT.

- **Java and IoT:** Java is a versatile and flexible language, which is matched with the IoT. Therefore,

most IoT developers use Java as their first choice for the coding of their project's modules. If you are having a computing background, then you must be aware of the features of Java. Its built-in features are proving very helpful for IoT. The code written in Java can run independently on any machine. The developers only need to make few changes in the previous code and then they can transfer the code to the chip of a new IoT device. The codes of Java can run both on edge and cloud. But it needs more powerful resources because its codes are lengthy and heavy. Java coding is used in the following areas-

- i. Smart meters
 - ii. mHealth
 - iii. Industrial automation solutions
 - iv. Fleet management systems
 - v. Embedded applications
- **Python and IoT:** Python is used where IoT needs to build a data-intensive application. It will be a better choice when IoT developers need to work on a fully-fledged data processing center on a local level. They use it in straight forward data applications, analytics ability to the edge, and add a data science. It is mainly used for cloud applications. It is used by following in IoT.
 - i. **Mara:** This is a library skeleton that supports Python. This library can be used for all devices. So, you are not required to use different libraries for Edison and Pi. It provides a service for communication protocols.
 - ii. **Sockets:** It is a package that provides network services using python.
 - iii. **MySQL:** This is a small tool that uses python scripts to excuse shell commands for reading or writing a database.
 - iv. **MQTT:** It is a protocol that is specifically developed for IoT. In this protocol, you can directly request within python with extra settings.
 - **C/C++ and IoT:** Small devices that have low computing power and less memory are supported by C/C++. This can be the best choice for small devices of IoT. C/C++ writes a program that is superlative for firmware devices as it can run the optimized code only on RAM. The microcontrollers that are used in IoT hardware are also building using C/C++ coding. But you need to learn these languages very carefully because a simple error can stop your code to run. So, you can say that you must become an expert of C/C++ if you are working on devices that need less computing and can run on RAM directly. It is used for the following applications
 - i. Edge computing
 - ii. Gateways, hubs
 - iii. Game and game engine development
 - iv. Embedded systems
 - v. Microcontrollers
 - vi. Coherent solution development

- **GO:** GO is also a coding language that is becoming popular in IoT day by day. It is beneficial in IoT for the following

This language has not many features but it is still being used in small devices that used optimized code and less computing power. This code is a good choice for devices that have limited memory space.

GO is very suitable for the communication layer where multiple data streams require routing at the same time. This is a lightweight language that allows multiple data streams to process concurrently.

This language has multiple tools that are required for working in an IoT environment, like debugging, testing, and analyzing tools.

Things that need to be considered for the coding of an IoT device:

Now from the above information, you must have an idea that coding is required in IoT. It is required for all the modules of IoT like building a user interface, building network capabilities, and making cloud storage for storing massive data, generated by IoT devices. When you are choosing a coding language then you have to consider the following points

- **Hardware:** You need to choose a language that can work best with your hardware. Like if you have a small device that uses less power and memory then you should pick C/C++ as a coding tool. But if you are working on a big project like the devices that have increased computing power and memory then you should use Java and Python as they are the best choices here. [Click here to learn Data Science Training in Bangalore](#)
- **Speed:** If you are working on a project where you must build quick prototypes then you need to use Python as it is the best prototyping language and provides quick results. So where speedy work is required, Python will be best there.
- **Development cost:** The development cost worth a lot because Python requires more expensive hardware. Whereas C/C++ can work on inexpensive and small hardware devices. So, if you have a high budget and you can afford expensive hardware then you should use Java or Python otherwise you can use C/C++.
- **Tools:** You must keep an eye that which tools are available for you to use before choosing a programming language. Open-source languages like Swift and PHP can be best in this situation.

3.1.6 Programming in Python Language

Python is a high-level programming language used as a general-purpose language. Python design philosophy focuses on code readability. For this purpose, the language uses a significant number of spaces. Python provides constructs that allow you to program for small as well as large scales. This language has a very large standard library. It belongs to a dynamically typed language. It also has automatic memory management.

Python is the language of choice because it's easy to learn, has clean syntax and it is supported by a large community online. In IoT, Python is a great choice for the backend side of development as well as the software development of devices. In addition, Python is available to run on Linux devices and you can use MicroPython for microcontrollers.

Python programming powers intuitive interfaces of intelligent and effective Internet of Things (IoT) systems that are paramount in remote sensor networks, big data and data analysis, automation, and machine learning. IoT applications function efficiently with the help of Python libraries/packages which include -

1. **Numpy:** Numpy is a scientific computing package that helps to create datasets to test with the time series data in IoT. Numpy features are used in IoT to read sensor bulk data from the database inbuilt functions in the system
2. **Sockets and MySQLdb:** Sockets that facilitate networking in IoT devices include TCP/IP and UDP, which are compatible to work with Python packages. TCP/IP and UDP act as transport layer protocols for communication. The MySQLdb is a go-to relational format database that helps in the development of remote stores for the IoT system.
3. **MATPLOTLIB:** To get data insights, matplotlib visualizes the most paramount operations by giving a variety of graphs to represent the data.
4. **Requests, Tkinter and Tensorflow:** To make HTTP calls and parse responses in Python, the request package acts as a major protocol for data exchanges. Tkinter GUI puts the aspects of Python script in a controlled distribution, which enables functional testing and repeated executions in IoT Python devices. Therefore, the numerical computations of machine learning initiated into the IoT systems utilize the representation in data flow graphs dealing with huge non-linear datasets and deep learning aspects.

Some of the many advantages of working with Python for IoT devices are the speed at which you can develop code and a large number of libraries for all kinds of platforms.

Python is a great supporter to develop device prototypes. Even if you rewrite some of your code during production to C, C++, or Java to improve performance, in general, the system will function perfectly in Python.

IoT sensors simulators used in python programming include:

1. **MQ Telemetry Transport (MQTT) Sensor Simulator:**

The most popular connection method for IoT devices is MQTT, a protocol that is effectively implemented with Python.

The MQTT protocol is a machine-to-machine (M2M)/Internet of Things connectivity protocol that is designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

The Eclipse Paho MQTT Python client library implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol.

The Paho library code provides a client class that enables applications to connect to an MQTT

broker to publish messages, subscribe to topics, and receive published messages. It also provides some helper functions to make the publishing of one-off messages to MQTT servers straightforward.

MQTT protocol for the IoT in Python enables high-speed data exchange with low payload communication between the devices. User-friendly requests of MQTT are made directly in Python. Data is collected in real-time and easily analyzed in mathematical computation libraries like matplotlib. The diagram below shows the steps used for the data flow-

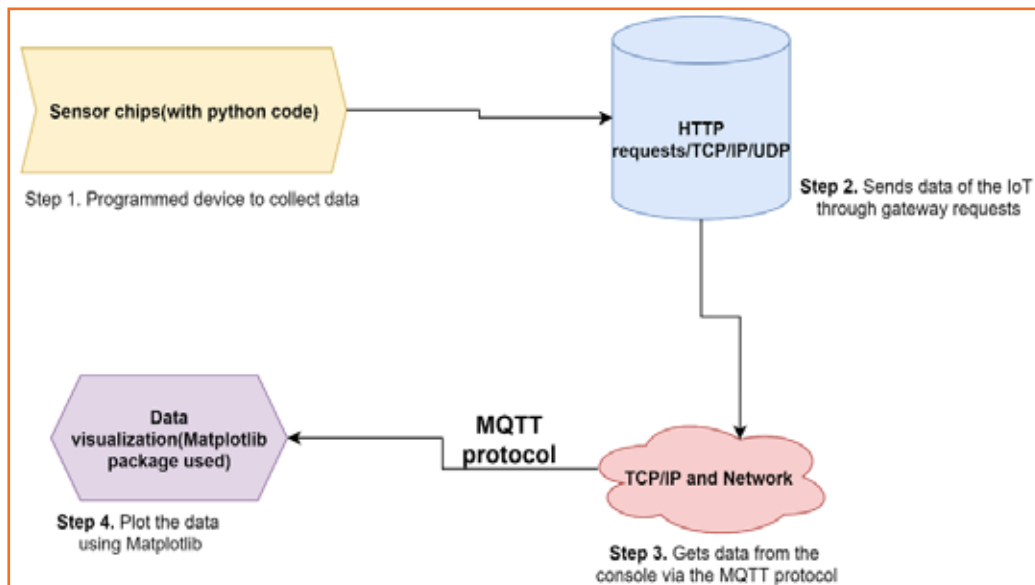


Fig 3.1.24: Data flow process in MQTT simulator

Data logging using MQTT (install using pip install paho-mqtt) Python is displayed below:

- i. `import paho.mqtt.client as mqtt`
- ii. `#Callback`
- iii. `for received data from server`
- iv. `def on_connect(data_iot, user, events)`
- v. `print("connected with code" + str(events))`
- vi. `data = mqtt.Client()`
- vii. `Data.on_connect = on_connect`
- viii. `Data.on_message = on_message`
- ix. `data.loop_forever()`
- x. For instance, the MQTT protocol is used to send data of a light bulb and install required dependencies and libraries

2. Azure IoT SDK in Python

Azure IoT hub offers a variety of features for IoT SDK usage which provides the ability to connect devices and services. The IoT SDK is supported by the MQTT protocol which facilitates the data exchange processes. The device requirements to be used along with Python include-

Python version 3.7+: helps in both asynchronous and synchronous API

The IoT hub SDK helps with the following aspects: access, processing, and analysis of data for machine learning applications.

The Azure IoT hub helps collect messages and feedback data collected by IoT devices.

3. IoT Backend on Flask In Python

For a quick, and hassle-free tool to write the backend for your IoT systems and easily set up server-side input/output information, the Flask microframework is here to rescue you and is packed with lots of functionalities.

To get started, decide on the requests you need to serve from your IoT devices, set up the Flask microframework, and write a couple of lines of code. The GET method will now return information upon request from the client's side.

In many cases, you can target the RESTful protocol when working with your IoT devices. This simplifies the exchange between the system components and allows you to expand the information exchange system in the future.

The disadvantage of using this approach is the potential lack of initiating the transfer of data from the server to the device. That is, the IoT must independently and periodically pull from the server. Rest easy, as there are solutions to address this risk. You can use web sockets or a Python library for Pushsafer. With PushSafer, you can easily and safely send and receive push notifications in real-time to your iOS, Android, and Windows devices (mobile and desktop), as well as browsers like Chrome, Firefox, Opera, etc.

4. IoT Raspberry Pi SDK

The first thing that comes to mind about running Python on an IoT device is grabbing Raspberry Pi from the table. Python is pre-installed in the operating system and the only thing left is for you to write your own script.

In this case, you can control the I/O ports on the Raspberry Pi expansion bar. Fortunately, the board supports wireless communication (Bluetooth and WiFi), Ethernet, and you can also connect a monitor to the HDMI outputs, a specialized 3.2" **320x240 TFT LCD, or a low energy consumption E-Ink 2.13"** 250x122 display for Raspberry Pi.

The controllers, available in a wide range of computing power and budgets, can be chosen for your IoT system - from the fast Raspberry Pi 4 Model B 8GB to the smallest Raspberry Pi Zero, all supporting Python. If necessary, you can install the previous version of Python 2.7 for past compatibility.

Sending data and visualizing data on a dashboard is simplified by involving the IoT Pi SDK, which relies on internet connectivity for efficient and effective data insights from the device. The code below is used to start the process of collecting data using IoT Pi SDK in Python. Install by running - pip install Raspberry_SDK

i. from Raspberry_SDK.Countly

- ii. `import Countly`
- iii. `#initiate the SDK`
- iv. `Countly = Countly("SERVER_URL", "APP_KEY", 0)`
- v. `#Send an event`
- vi. `countly.event("NAME", VALUE)`

IOT RASPBERRY PI SDK also helps to retrieve data events for both analog and digital circuits. Use case of IoT Raspberry Pi SDK is applicable in temperature room measuring and Bulb light. For instance, the server gets to pass the application key to collect the data and data is being manipulated by GroveAPI for raspberry IoT.

The advantages of using this approach are the availability of a wide range of development tools, libraries, and communications for the most complex devices based on Raspberry Pi, including video processing from cameras.

5. Python on PyBoard

The next great solution for Python in IoT devices is the PyBoard with an STM32F405RG microcontroller.

The PyBoard is a compact and powerful electronics development board that runs MicroPython. It connects to your PC via USB, giving you a USB flash drive to save your Python scripts and a serial Python prompt (a REPL) for instant programming. This works with Windows, Mac, and Linux.

MicroPython is a complete re-write of the Python (version 3.4) programming language so that it fits and runs on a microcontroller. It includes many optimizations for efficiency and it uses very little RAM.

MicroPython runs bare-metal on the PyBoard, essentially giving you a Python operating system. The built-in `pyb` module contains functions and classes to control the peripherals available on the board, such as UART, I2C, SPI, ADC, and DAC.

The dimensions of the board are impressive, taking up about two quarters, 33mm x 43mm, and weighing just 6 grams.

6. ESP8266, ESP32 with Micropython

If you need to create an IoT device with low power consumption, great capabilities, and integration with wireless WiFi networks, you can run Python on ESP8266, ESP32. More precisely, MicroPython.

After installing Python on computer, you can enter the `pip install esptool` from the command line. The MicroPython installation process is quite simple - download the firmware from the website and install it using `esptool`, not forgetting to format the board before installing it.

There are already quite a few IDEs for developing with Micropython. The entire development process is carried out on a working computer, then it is compiled and stored in the memory of an ESP8266 or ESP32 microcontroller. Code might look like

```

from machine import Pin
import time
led_pin = Pin(2,Pin.OUT)
while True:
    led_pin.on()
    time.sleep(1)
    led_pin.off()
    time.sleep(1)

```

MicroPython will impose a lot of restrictions compared to regular Python but, in general, you can easily write the necessary functionality on the client-side and run it efficiently on ESP microcontrollers. This solution is more cost-effective than purchasing PyBoard.

3.1.7 IoT Operating System (OS)

An IoT OS enables IoT devices to interact with cloud services over a global network within the tight parameters of limited memory bandwidth, data volumes, and processing power.

An embedded operating system is a specially designed operation system, which performs specific kinds of tasks in a small segment of memory unit like a Microcontroller/Microprocessor. We use the operating system depending on our purposes like completing tasks within the time-complexity, easy to manage multiple tasks, requiring fewer resources, etc. But when we talk about the Embedded OS for IoT, it actually means that an embedded operating system within the IoT devices connects to a network of devices.

An OS helps control traffic lights, digital televisions, smart meters, ATMs, airplane controls, and elevators, among many other use cases. The variety of one host to another means the IoT OS has a similar variability in requirements.

Examples of IoT operating systems:



Fig 3.1.25: IoT operating system

Internet of Things (IoT) capable microcontroller or sensor is a physical object or hardware, which is converted into an internet-ready computer to connect with the local network and running codebase applications. Devices like smartphones & tablets are IoT-enabled embedded operating system computers. As well as for the home the smart fridges, smart switches/fans, and entertainment devices all are enabled with IoT embedded OS. As a standard operating system like Windows, Linux, macOS helps us to manage or access software on a computer, but an IoT embedded-OS allows us to execute an operation with connected devices. Many of IoT-OS platforms are offered as open-source, so users are allowed to review the code and modify it as required for the IoT applications.

The IoT-OS is built for low resource utilization, with constraints related to size, memory, power, and processing capacity. By choosing the right IoT-OS, users will be able to set up IoT devices, which will run in any situation in any scenario. An IoT OS is considered successful when it can embed within an internet-connected application, run the software, and efficiently gather, process, and analyze data without any latency on the host IoT device.

Benefits of IoT operating system

Keeping up with the essence of connecting embedded systems all around us, IoT operating systems are critical for ensuring a series of tasks such as security, connectivity, interoperability, networking, storage, and remote device management, among other things.

IoT operating systems having real-time processing capabilities are called real-time operating systems or RTOS.

An IoT OS is essential for IoT apps to work as intended. It enables devices and apps to connect with other systems, such as cloud platforms and services. The IoT OS manages the processing power and other resources to transmit, collect and store data.

Characteristics of IoT operating systems

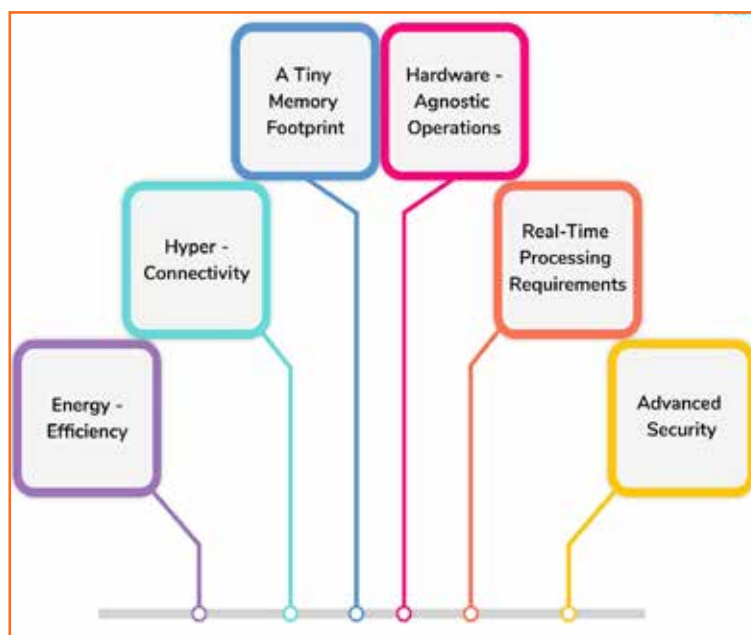



Fig 3.1.26: Characteristics of an IoT OS

1. **Energy-efficiency:** Imagine if an IoT app's battery keeps dying because its OS eats up all the power. That would defeat the purpose of IoT.
2. **Hyper-connectivity:** IoT is nothing without connectivity. If the devices do not connect, then what is the point?
3. **A tiny memory footprint:** IoT devices do store some amount of data, but their main focus is to ensure continuous transmission between two or more devices. An OS should be able to promise that.
4. **Hardware-agnostic operations:** An OS should be compatible with all sorts of hardware solutions. Because just like software, the hardware also evolves, and the OS must be able to work with it at all times.
5. **Real-time processing requirements:** IoT apps process data on the go. If the OS cannot support it, the purpose is lost, and the application would not function the way it is supposed to.
6. **Advanced security:** One of the biggest concerns, perhaps, in IoT is data security. It is the responsibility of the operating system to ensure the information being transmitted stays protected at all times and across all devices.



Moving ahead, several factors need to be kept in mind while selecting a suitable IoT OS.

Selecting the best IoT Operating System for your Application



After understanding the what and why of IoT-OS, we can easily select the right OS for completing our IoT operations. Because the IoT-OS is an open-source platform, so, in the market lots of IoT-OS are available. Now, we can check one by one.

IoT OS Names	Features	Logo
RIOT OS	<p>RIOT is a free open-source IoT-OS with built-in IoT stacks for IoT applications. RIOT released under the GNU Lesser General Public License (LGPL) and developed by a grassroots community gathering companies, academia, and hobbyists, distributed all around the world. This OS supports low-power IoT devices, and various microcontrollers like 8-bit, 16-bit, and 32-bit and developed for memory-constrained systems with a focus on low-power wireless IoT devices. Because of Microkernel (μ-kernel) architecture, the Kernel uses $\sim 1.5K$ RAM. We can also run a RIOT-OS code on a Linux/macOS computer. It also supports different network stacks like 6LoWPAN, IPV6, RPL, UDP, TCP, LoRaWAN, 802.15.4, MQTT, and much more. It also supports different PHY technologies (like Bluetooth, NFC, serial, CAN, etc). It also supports 3rd party packages (like lwIP Stack, uIP, Open-thread stack).</p>	

IoT OS Names	Features	Logo
Zephyr	<p>This is a real-time operating system (RTOS) that is built especially for IoT applications. Zephyr gets support from Linux Foundation and supports multiple hardware architectures (small scales of Cortex-M devices to multi-core 64-bit CPUs). It supports interconnectivity technology, like Bluetooth LE, Wi-Fi, NFC, LoraWAN. Minimum of 8kb RAM and 512kb ROM required to start operation with Zephyr. Zephyr comes with Apache 2.0 open-source license and it is free for commercial and non-commercial projects and also supported by Long-term support (LTS) with security updates.</p>	
Apache Mynewt	<p>This OS also comes with an open-source licence Apache License 2.0, which provides complete environment support for developing IoT applications. Mynewt uses 6kb kernel memory to deliver and process the IoT operations. It supports the connectivity of the Bluetooth Low Energy 4.2 stack. It supports multistage software watchdog, memory pool allocation, priority-based scheduling.</p>	
Android Things	<p>In 2018 Google launched an embedded operating system, which is called Android Things. Android Things uses only 32-64kb memory of RAM to complete the operations. Because Weave is connected with Android Things, the IoT device can detect each of them by the android smartphone. The SDK of Android Things can help developers to test, build and debug each IoT solution remotely. But unfortunately, Google shut down the Android Things dashboard.</p>	
Contiki-NG	<p>In 2002, Contiki was an open-source IoT-OS, invented for low-power microcontrollers. It runs effectively with IPv4 and IPv6 internet protocols. It also provides the support of wireless standard protocols suites like CoAP, 6LoWPAN, RPL. Only 10kb of RAM and 30kb of ROM are used to complete the processes. Contiki OS released under a BSD license and contributors of this open-source are also famous (like Texas-Instrument, Atmel, Cisco, ENEA, RedWire, Oxford University, SAP and many others)</p>	

IoT OS Names	Features	Logo
Amazon FreeRTOS	Amazon-FreeRTOS invented by Amazon, and it is an open-source released under the MIT open-source license OS with the support of IoT application development. It takes a 6-15kb memory footprint which makes it more reliable to be adopted by small power microcontrollers. Amazon-FreeRTOS comes with AWS IoT Core, which helps the developers to easily access the cloud service of the Amazon Web Service. Because of the data security of Amazon-FreeRTOS support with Transport Layer Security(TLS v1.2) and a variety of microcontroller chipsets over 40 architectures.	
TinyOS	TinyOS is also an open-source OS under a BSD license. In 2000, it initially released its first version for use. This OS is designed for low-power wireless devices, ubiquitous computing, personal area networks. It supports a complete stack of IPv4, IPv6, 6LoWPAN, RPL.	
Mbed OS	Mbed OS is an open-source OS release under Apache License 2.0, which supports 32-bit ARM Cortex-M IoT microcontrollers. Users can easily develop a code using Mbed online IDE, with the help of a free code editor and compiler. Mbed OS is based on Keil RTX5, which allows the developer to develop code with C/C++. Mbed OS comes with various protocol stacks like Ethernet, USB, CAN, SPI, I2C, RFID, NFC, WiFi, Bluetooth. It also provides multilayer security.	
EmMate	EmMate is FreeRTOS based open-source OS which comes under GPLv3 License. It supports multi-architecture and comes with internet protocol stacks to develop micro to small scale IoT applications. EmMate is a platform-independent OS written in C/C++, so developers can develop code with C/C++. Because of its open-source, users can easily modify, review the code and also contribute with his/her idea. EmMate also comes with migCloud, which helps the developers to easily access the cloud service for IoT applications.	

IoT OS Names	Features	Logo
<p>Raspbian Pi</p>	<p>It is now known as Raspberry Pi OS. It is an operating system used by the popular Raspberry Pi IoT devices. Used extensively by Raspberry Pi line CPUs, the operating system comes with preinstalled IoT software like Wolfram Mathematica, Chromium, and Minecraft Pi Edition.</p> <p>The user interface of Raspberry Pi OS is similar to Windows, macOS, and Ubuntu Linux, which makes it extremely useful for general, educational, and experimental purposes.</p> <p>There are certain substitutes or sub-branches of this operating system which are open-source, network-based educational solutions for educational institutions. You could use it to quickly set up and manage a classroom network with a Raspberry Pi.</p> <p>Such operating systems consist of a graphical user interface (GUI) and a scripting language called Python. Schools, for instance, can easily create and manage profiles of students, teachers, parents, and other staff. They can quickly access projects, workspaces, and other data easily.</p>	
<p>Ubuntu Core</p>	<p>There is no doubt that this is among the most secure, robust, and lightweight distributions of Linux, making it one of the most popular ones. It provides you with a low-level framework of the Linux kernel along with a host of preinstalled tools that are incredible to use.</p> <p>The OS runs comfortably on the Unity desktop environment, allowing it to perform efficiently on mobile devices or computers without any modifications. Being a foundation for IoT applications, it is equipped with tools to build, run, debug, test, and deploy applications.</p> <p>The built-in web server software allows you to handle large traffic spikes efficiently. The developers also get an impressive collection of embedded Linux systems for learning with and deploying IoT solutions.</p>	

IoT OS Names	Features	Logo
eLinux OS	<p>It is easy to install and runs seamlessly on embedded devices. Linus Torvalds developed the operating system to load balance for the ARMv7 CPU architecture. It offers a very intuitive user interface and provides a fully functional multiprocessor system.</p> <p>The main focus of the project is to provide a cutting-edge platform for developers. You can use it to build applications for embedded systems running the Linux operating system.</p> <p>These applications are apt for IoT devices as they run fast and are ready to be used with minimal configuration. It has been designed to be combined with other software such as PHPMyAdmin.</p>	
Windows 10 IoT	<p>It is pretty clear from the name that Windows 10 IoT is a member of the Microsoft community. It runs the Metamask Ethereum Wallet, which allows users to interact with their Windows 10 IoT devices using Ethereum, among other blockchains.</p> <p>This operating system is very similar to Ubuntu Linux, providing an easy-to-use base for creating and running IoT devices. When you install it on your device, it automatically downloads and installs all the necessary drivers and programs for your device.</p> <p>You can use it to create a network of connected devices that can work together to provide services when needed or as a standalone operating system for IoT applications. The latest Windows 10 IoT Core improves upon Windows 10.</p>	



IoT OS Names	Features	Logo
Micropython	<p>It is an open-source reimplementaion of the Python3 programming language. The Pyboard (a Python product) is very compact, with only 256k of code space and 16k RAM. It focuses on microcontrollers when you build compact electronic circuit boards for IoT projects.</p> <p>These bare-metal boards give you the capability to control all kinds of electronic projects and are compatible with normal Python as well. The operating system also gives you the option to transfer code from your desktop to the microcontroller quickly.</p> <p>The language is more beneficial for new developers developing robust applications for industrial use. You can leverage standard Python to develop and implement actual project code rapidly.</p>	
Embedded Linux	<p>This is basically an umbrella term to define the next generation of free and open-source operating systems. Embedded Linux provides an environment that helps developers build custom devices with higher functionality and lower cost.</p> <p>It expands the possibilities of building applications using programming languages such as C and Java. These applications run seamlessly within the confines of the embedded system without any need for explicit coordination with external servers or devices.</p> <p>Embedded Linux is simple to configure and offers accessible programming interfaces. Its purpose is to be an enhanced and expanded Reiser4 Linux distribution for IoT devices.</p> <p>An enhanced version of the Linux Kernel and an embedded graphics stack are being used by the operating system, resulting in agility.</p>	

Table 3.1.1: List of available IoT OS

Parameters for selecting the most suitable IoT OS

- **Scalability:** The operating system must be scalable for any type of device. That means both integrators and developers need to be familiar with the operating system when it comes to gateways and nodes.

- **Footprint:** Since the devices will always come with a bag of constraints, it is essential to choose an operating system with low power, processing, and memory requirements.

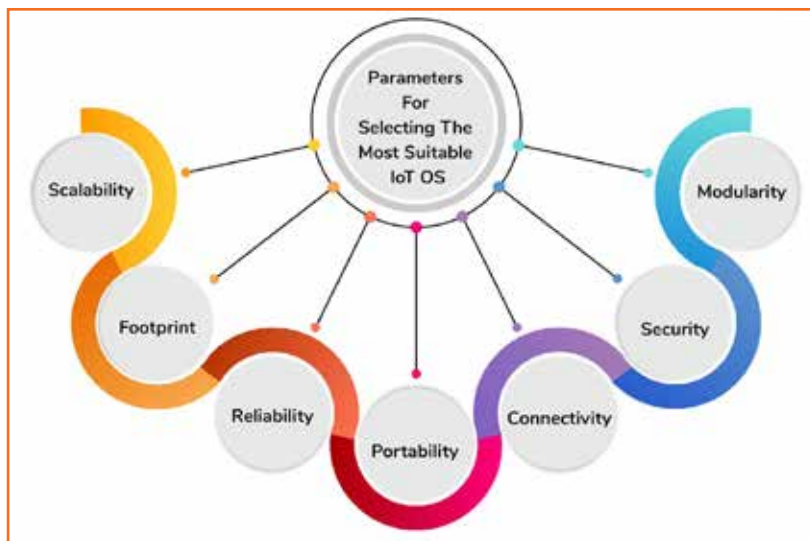


Fig 3.1.27: Parameters for selecting IoT OS

- **Reliability:** This is a critical factor to note for mission-critical systems. For instance, Industrial IoT devices are at remote locations and have to work for years without hampering business continuity. Your operating system should be able to fulfill specific certifications for IoT apps.
- **Portability:** Operating systems isolate apps from the specifics of the hardware, hence leading to greater portability. Usually, an OS is ported to different interface and hardware platforms to the board support package (BSP) in a standardized format, such as POSIX calls.
- **Connectivity:** This one is obvious, but your operating system should support connectivity protocols such as WiFi, IEEE, Ethernet, and so on. There is no point in IoT apps if they cannot connect without any hassle.
- **Security:** The operating system of your choice should be safe and secure to use, allowing you to add on some aspects in the form of SSL support, secure boot, components, and encryption drivers.
- **Modularity:** Every operating system must mandatorily have a kernel core. All other functionalities can be included as add-ons if so required by the IoT app you are building.

3.1.8 IoT App Security

Security is the fundamental design consideration when developing an IoT app. Smart homes, health, and security devices all transmit extremely sensitive personal data, and any data breaches could be harmful to users, not to mention disastrous for the device manufacturer's reputation.

- **Audit your chosen IoT platform:** Inspect the security credentials of your chosen IoT platform provider and check to see evidence of regular penetration tests and security updates.
- **Consider where the data is stored:** Most IoT environments store device data in a central server in the cloud. This presents an inherent vulnerability that needs to be managed with an appropriate

level of encryption. An alternate option is to store the data within the device using a P2P model, which removes the risk of storing data in a central server. This is discussed in more detail in the below section.

- **Decide on the most secure connection protocol:** This needs to be carefully balanced against broader design and operational considerations and constraints. For example, wired ethernet may provide greater security versus WiFi but is often impossible in remote settings.
- **Use two-factor authentication, obfuscation, and encryption:** This will help ensure only authorized users can access the IoT app.
- **Use PKI for authentication and encryption:** This ensures end-to-end privacy for your users. Nobody but the user will be able to access the collected data.

Here are a few tips on ensuring security in your IoT app:

- **Choose hardware vendors carefully:** Some vendors that offer software for their devices can miss potential software vulnerabilities. For your IoT app, make sure to choose reliable hardware or run it by security specialists.
- **Use proven IoT platforms:** Much depends on the platform you use for your IoT application. Be sure to use reliable platforms that test their tools and update them regularly.
- **Consider not only network attacks but also physical attacks:** Any data stored on a device should be not only encrypted but also physically protected - make sure it isn't easy to remove the storage.
- **Use protected networks:** All data that circulates between your app and devices through a server or the cloud should be encrypted.
- **Apply best practices for app security:** Use encryption, obfuscation, two-factor authentication, and other techniques to ensure that your IoT app and devices are used securely by authorized users.

Unit 3.2: Develop GUI/web UI for IoT System

Unit Objectives

At the end of this unit, participants will be able to:

1. Describe process of GUI/UI development
2. List various features of GUI/UI
3. Describe various toolkits available for GUI/UI in Python
4. Demonstrate GUI programming

3.2.1 Graphic User Interface (GUI)

In the previous units we learn about how data is collected by sensor devices, sent to a cloud service via a network solution, and transformed into useful information. The last thing we need to do is to deliver the information to the end user. This is done via user interface (UI).

Users need a way to view and understand the data captured by IoT. That's where the user interface comes in. The user interface consists of the features by which a user interacts with a computer system. This includes screens, pages, buttons, icons, forms, etc. The most obvious examples of user interfaces are software and applications on computers and smartphones.

A user interface doesn't necessarily require a screen, however. For example, a TV remote has a user interface that consists of various buttons, and devices such as Amazon Echo can be controlled with voice commands.

User interface (UI) software is often large, complex, and difficult to implement, debug, and modify. As interfaces become easier to use, they become harder to create. Today, direct-manipulation interfaces (also called GUIs for graphical user interfaces) are almost universal. The Graphical User Interface (GUI) is the way a user can interact with a computing device without entering text commands into a console. It is a friendly visual environment that allows the user to perform any action without having to have programming knowledge. An example of the GUI are the Windows, MacOs or Android environments etc.

A GUI includes GUI objects, like icons, cursors, and buttons. These graphical elements are sometimes enhanced with sounds, or visual effects like transparency and drop shadows. Using these objects, a user can use the computer without having to know commands.

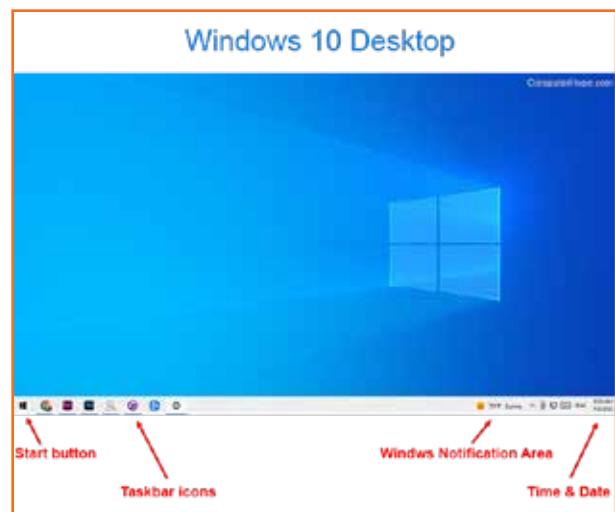


Fig 3.2.1: WINDOWS GUI

Elements of a GUI

To make a GUI as user-friendly as possible, there are different elements and objects that the user use to interact with the software. Below is a list of each of these with a brief description.

- **Button:** A graphical representation of a button that performs an action in a program when pressed
- **Dialog box:** A type of window that displays additional information, and asks a user for input.
- **Icon:** Small graphical representation of a program, feature, or file.
- **Menu:** List of commands or choices offered to the user through the menu bar.
- **Menu bar:** Thin, horizontal bar containing the labels of menus.
- **Ribbon:** Replacement for the file menu and toolbar that groups programs activities together.
- **Tab:** Clickable area at the top of a window that shows another page or area.
- **Toolbar:** Row of buttons, often near the top of an application window, that controls software functions.
- **Window:** Rectangular section of the computer's display that shows the program currently being used.

How does a GUI work?

A GUI uses windows, icons, and menus to carry out commands, such as opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a mouse, a keyboard can also be used via keyboard shortcuts or the arrow keys.

For example, if you want to open a program on a GUI system, you would move the mouse pointer to the program's icon and double-click it. With a command line interface, you need to know the commands to navigate to the directory containing the program, list the files, and then run the file.

Benefits of GUI

A GUI is considered to be more user-friendly than a text-based command-line interface, such as MS-DOS, or the shell of Unix-like operating systems.

Unlike a command-line operating system or CUI, like Unix or MS-DOS, GUI operating systems are easier to learn and use because commands do not need to be memorized. Additionally, users do not need to know any programming languages. Because of their ease of use and more modern appearance, GUI operating systems have come to dominate today's market.

Why IoT design is different from regular UX/UI design

The creation of UX/UI design for IoT apps is often much more challenging in comparison to conventional software products.

- Unlike traditional web or mobile apps, IoT digital solutions involve additional layers, e.g. a number of devices and interfaces with different functionality, Artificial Intelligence (AI), input-output data streams, user rights distribution, specialized platforms, and much more. Designers have to entirely understand how to work with each aspect of an IoT network while making the system as smooth and spontaneous for end-users as possible.

- IoT devices exchange large volumes of data with each other and provide only processed results to end-users. Therefore, designers have to consider the compatibility of diverse interfaces, the way they gather data, connect with the cloud and other platforms, as well as interact with humans.
- They also have to take into account the characteristics of a particular network, whether it's automated car diagnostics, climate control, supply-chain tracking, or any other activities. This increased complexity and specificity of IoT networks result in non-standard approaches in interface design.
- Another considerable challenge is the necessity to develop UX/UIs that can support more devices and data points. Therefore, IoT products should have flexible and adaptive interfaces which can be simply amended without much interference into the core functionalities of the products.

IoT App UI Design and Its Challenges

When developing UX/UIs for IoT applications, software developers have to take into account general design principles that are invariably important for building a successful interface in any app type. Here are the key ones to consider for IoT.

- **Simplicity:** Simplicity and user-friendliness are important in any software app. Though, things get to an entirely new level with IoT networks. It's essential that designers find ways for apps to deliver the most needed and well-processed information from numerous devices in the simplest way. For this, they have to carefully consider which data to provide for different user groups and when to provide it.

For example, when a smart-watch user utilizes their gadget as a fitness tracker, the device should display relevant data such as the number of steps made and calories burnt instead of a weather forecast or the latest messages.

- **Performance:** No lags, fast data processing, and fluid motion are key to successful UX/UI. To ensure robust IoT application performance, software developers have to use well-established and efficient UX/UI development tools.
- **Connectivity:** Most of the IoT apps are internet or cloud-dependent. It means that they send their data to the cloud within a certain period of time. If there are any difficulties with the internet connection or a device can't upload files in the cloud, the UI should inform its users about that. This way, when designers develop UIs they have to consider this aspect so that the UI could reflect the information on the latest data updates, e.g. **"The last message was received 1 hour ago"**, etc.
- **Physical UI:** What should a user do if their IoT app is having an unplanned downtime or just lagging? Designers should be prepared for this scenario and offer end-users an alternative - a physical user interface. For example, if there isn't any WiFi connection, users should still be able to use smart lights or use a smart kettle in their homes.

Though physical UI isn't limited to any unexpected disruptions, it can also be used to quickly inform users about the state of a device, e.g. a small green LED light on a sensor to notify that everything works as it needs to.

3.2.2 GUI Programming

A GUI program is very different from a program that uses a command line interface which receives user input from typed characters on a keyboard. Typically programs that use a command line interface perform a series of tasks in a predetermined order and then terminate. However, a GUI program creates the icons and widgets that are displayed to a user and then it simply waits for the user to interact with them. The order that tasks are performed by the program is under the user's control - not the program's control! This means a GUI program must keep track of the **"state"** of its processing and respond correctly to user commands that are given in any order the user chooses. This style of programming is called **"event driven programming."** In fact, by definition, all GUI programs are event-driven programs.

An GUI program has the following structure:

- Create the icons and widgets that are displayed to a user and organize them inside a screen window.
- Define functions that will process user and application events.
- Associate specific user events with specific functions.
- Start an infinite event-loop that processes user events. When a user event happens, the event-loop calls the function associated with that event.

A GUI program's interface is composed of widgets displayed in a window. Your computer's operating system controls the creation and manipulation of windows on your computer's display screen. The operating system also controls the pointing devices on your computer, such as a mouse or a touch screen. Therefore, your computer's operating system is what recognizes events that happen in a window. Your operating system sends events to your program in the order they are generated by a user. Your program's event-loop responds to these events. All GUI programs have the same event-loop, so there is an event-loop provided for you and it looks something like this

```
while True:
    # Get the next event from the operating system
    event = get_next_event()

    # Get the function that is assigned to handle this event
    a_function_to_handle_the_event = event-handlers[event]

    # If a function has been assigned to handle this event, call the function
    if a_function_to_handle_the_event:
        a_function_to_handle_the_event() # Call the event-handler function

    # Stop processing events if the user gives a command to stop the applica-
    # tion
    if window_needs_to_close:
        break # out of the event-loop
```

Again, you do not implement an event-loop in a GUI program. The event loop has already been written for you. You make this event-loop work by associating a function (which is called an event-handler or a callback function) to a specific event. We will show you how to do this in a few lessons. First, let's learn

how to create a GUI interface which is the widgets a user sees when a GUI program runs.

GUI Programming by using Python Toolkits

Python does not implement GUI, event-driven-programming in its core functionality. GUI programming is implemented using imported modules which are often referred to as “**toolkits.**” Once you understand how GUI programming works, you should be able to learn how to use any of the other available toolkits without much difficulty.

Python UI libraries can use for the GUI. Writing a Python program that uses a graphical user interface (GUI) to interact with the user involves using the provided libraries that come with the Python language. Using these special libraries, Python will interact with the operating system to present windows, buttons, and event handlers to detect what a user is doing with the keyboard and mouse. By learning a few important methods, a programmer won't find Python GUI development difficult as well.

Python supports many programming paradigms, including:

- Object-oriented;
- Imperative;
- Functional;
- Procedural.

3.2.3 GUI Toolkits for Python

A toolkit is a library of widgets that can be called by application programs. A widget (also called a control) is a way of using a physical input device to input a certain type of value. Typically, widgets in toolkits include menus, buttons, scroll bars, text type-in fields, etc.

Using a toolkit has the advantage that the final UI will look and act similarly to other UIs created using the same toolkit, and each application does not have to rewrite the standard functions, such as menus.

GUI library: The GUI library contains widgets. Widgets are collections of graphical controls. When creating a GUI program, a cascade method is usually used. Graphical controls are added on top of each other.

Python GUI: Python GUI is a GUI that is written in the Python programming language. When writing an application using Python, you need to use the GUI for that. There are many options for the Python GUI. There are over 30 cross-platform frameworks for GUI programming in Python. Few of them are listed and briefly described below.

1. **Tkinter:** Tkinter is a set of tools that can render GUIs using Python. It allows you to run Python scripts in a GUI format. Tkinter is considered as the de-facto GUI toolkit for many Python developers, with many beginners picking it up as their first GUI programming framework.



Fig 3.2.2: Tkinter toolkit

Tkinter is a combination of the Tcl and Python standard GUI frameworks giving you all the widgets you need to create a rich UI for whatever application you're working on, but it's especially suitable for developing desktop apps.

Advantages:

- If you're using a recent version of Python, Tkinter is most likely already installed
- Tkinter offers a vast collection of well known widgets, including all the most common ones like buttons, labels, checkboxes etc.
- Adding code to each widget is straightforward

Limitations:

- The basic Tkinter widgets are good, but you'll want to ensure your version of Tkinter also supports the extended set of Ttk widgets

2. **PyQt5:** PyQt5 is a very well-known GUI framework used by both Python coders and UI designers. One of its components, the PyQt package, is built around the Qt framework, which is a leading cross-platform GUI design tool for just about any kind of application. The PyQt5 package has a detailed set of bindings for Python based on the latest version (v5) of the Qt application framework. It also provides SQL support for connecting to databases.

Creating GUIs is wonderfully easy with the QtGUI and QtDesigner module, providing plenty of visual elements that can be implemented simply by dragging and dropping widgets.

Advantages:

- Drag and drop functionality for implementing visual elements
- Add code to visual elements enabling the creation of small- and large-scale apps
- Cross-platform support for Windows, Mac, Android and Raspberry Pi
- Easy install with pip

Limitations:

- Not free! PyQt5 requires you to purchase a commercial licence.
- Not pre-installed with Python
- Python 3 support only

3. **PySide:** Wrapper for Qt. PySide2 and Qt5 are not exactly the same framework, but they were developed by the same company under the Qt for Python project, which essentially increases compatibility between the two frameworks to having nearly 99.9% identical APIs.

PySide2 is Qt for Python, offering the official Python bindings for Qt. This enables the use of Qt APIs in Python apps, and also a binding generator tool (Shiboken2) to expose C++ projects in Python.



Fig 3.2.3: QT toolkit

Advantages:

- Easy install via pip
- Supports both Python 3 and Python 2.7
- Cross platform
- Extensive community support and documentation
- Used by well known companies like TomTom and Mercedes

Limitations:

- Not pre-installed with Python
- Licensed under LGPL, which makes it questionable for use in commercial applications since you will need to make your source code available
- Python 2.7 builds are only available for 64 bit versions of MacOS and Linux. Windows 32 bit support is provided on Python 2 only.

4. **PyGUI:** PyGUI is designed for Unix, Macintosh, and Windows platforms. The trick of this MVC framework is to fit into the Python ecosystem as easily as possible. It is a simple API for developers to create user interfaces using native elements for Python applications. As a lightweight API, not a lot of code is needed between the app and the target platform, making it far more efficient than many of the other frameworks on this list.

Advantages:

- Documentation all written in Python
- Available in Python 2 and 3
- Support for other Python extensions like OpenGL and GTK
- Cross-platform
- Open source
- Easy install via pip

Limitations:

- Not pre-installed with Python

5. **wxPython:** wxPython is a cross-platform GUI toolkit you can use to create robust, functional GUIs in a simple and easy manner. The implementation is a set of Python extension modules that wrap the GUI components of the wxWidgets cross-platform library, which is written in C++.

wxPython creates native user interfaces that add zero additional overhead to the application, giving you the capabilities of a functional library without the burden.

Advantages:

- Open source
- Cross platform
- Supports both Python 2.7 and 3

- Simple to use
- Smooth installation with pip

Limitations:

- Not a part of the pre-installed Python on MacOS

6. **Wax:** Wax is the wrapper for wxPython (see #3 in this list). It offers the same functionality as wxPython, but is far more user-friendly. I've also included an example of how to use Wax at the bottom of this post.

Advantages:

- Open source
- Cross-platform
- Easy to use

Limitations:

- Not pre-installed with Python

7. **Pyforms:** Pyforms-GUI is a software layer that forms part of the Pyforms main library, which also includes PyForms-Web, and PyForms-Terminal. Pyforms is the Python implementation of Windows Forms, which lets you develop interactive interfaces for Windows GUI mode, Web mode, and Terminal mode.

Advantages:

- Open source
- Cross platform
- It has a minimal API, so interfaces can be easily defined with just a few lines of Python code
- You can also code advanced functionalities with minimal effort
- The code is organized in modules, ready to be reused by other applications
- Simplifies applications maintenance
- Fast and easy prototyping
- Low learning curve
- Easy install with pip

Limitations:

- Only available for Python3, and it's not pre-installed
- Not suitable for handling a large number of data fields in a form
- Displays record attribute values, but does not support calling method functions of objects being browsed
- It can't display dynamically computed attributes

8. **Kivy:** Kivy is a cross-platform, open source Python library (based on Python and Cython) designed for rapid development of apps with complex UIs, like multi-touch apps. Kivy runs on Linux,

Windows, OS X, Android, iOS, and Raspberry Pi, and you can run the same code on all these platforms. Kivy is based on OpenGL ES 2. It is an event-driven framework that is great for game development.

Advantages:

- Open source
- MIT license
- Built-in widgets
- Cross platform

Limitations:

- Not pre-installed with Python

9. **Libavg:** Libavg is a great Python GUI framework specifically for building touch-based interfaces. It's built in C++, enabling fast execution times, which are necessary when handling touch-based interfaces. It has the following features -

- displaying items in the form of Python variables;
- event management system;
- timers;
- support for logs.

Advantages:

- Open source
- Cross platform
- Supports common drivers for touch screens
- Broad range of features, including camera and animation support, as well as GPU effects and text aligning
- Offers a screen layout engine for different visualization manipulation techniques
- Supports both Python 2.7 and 3

Limitations:

- Not pre-installed with Python

10. **Flexx:** Many Python GUI libraries are based on libraries written in other languages such as C++. For example wxWidgets and libavg. Flexx is built in Python. The GUI uses web technology to display.

11. **CEF Python:** This framework targets Windows, MAC OS, and Linux. Based on Google Chromium. It focuses on making the built-in browser easier to use in third-party applications.

12. **Dabo:** The target of this framework is WxPython. It is a three-tier framework for cross-platform application development.

13. **Pyforms:** Pyforms is a Python 2.7 / 3.x environment framework used to develop a GUI application. It encourages code reuse.

14. **PyGObject:** With PyGObject, you can write Python applications for the GNOME project. It is also possible to write Python applications using GTK +.
15. **PyGTK | PyGObject: “GTK +”,** widely used on Linux, is a “GTK +” wrapper from “PyGTK”. Compared to Kivy and PyQt, PyGTK is very lightweight for Unix, Macintosh, Windows platforms. This MVC framework was developed by Greg Ewing of the University of Canterbury. Its main focus is to adapt to the Python ecosystem as simply as possible.

Selection of Python GUI Toolkit

It depends on your needs, for example -

- **For Absolute Beginners:** I would recommend starting with PySimpleGUI. This framework is easy to follow and has lots of documentation to help you get your head around GUI design and development with Python.
- **If you find PySimpleGUI too limiting:** Try something a little more complete like wxPython. It offers a better set of controls in an easy-to-use framework, although its wrapper (Wax) is more user-friendly. Either one would make a good step up over PySimpleGUI - the choice is yours.
- **Design-focused Coders:** Keen on GUI design? Try PyQt5, Tkinter or PySide 2. All of them will give you wonderfully flexible controls so you can build the UI of your dreams. However, keep in mind that PyQt5 and PySide 2 will require you to purchase a license.
- **Touch Screen UIs:** Need a UI for a touch screen application? Kivy and Libavg are your best bets. If fast execution times are important for you, I would recommend Libavg. On the other hand, if visual design is of paramount importance, stick with Kivy.
- **Form-driven UI:** If your application is form-driven, Pyforms is a good choice. Otherwise, Pyforms is hard to recommend.

3.2.4 Creating GUI Using Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter provides a set of standard GUI dialog boxes that can be used with minimal programming.

For creating a GUI application using Tkinter, perform the following steps:

1. Import the Tkinter module.
2. Create the GUI application main window.
3. Add one or more widgets to the GUI application.
4. Create instances of the widgets you want in your interface.
5. Define the layout of the widgets (i.e., the location and size of each widget).
6. Create functions that will perform your desired actions on user generated events.
7. Connect your functions to specific user events.
8. Start a GUI event-loop.

9. Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window:



Fig 3.2.4: Tk dialogue box

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

S. No.	Operator & Description
1	<p>Button: The Button widget is used to display buttons in your application.</p> <p>Syntax to create this widget –</p> <p>w = Button (master, option=value, ...)</p>
2	<p>Canvas: The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.</p> <p>Syntax to create this widget –</p> <p>w = Canvas (master, option=value, ...)</p>

S. No.	Operator & Description
3	<p>Check button: The Check button widget is used to display a number of options as checkboxes. The user can select multiple options at a time.</p> <p>Syntax to create this widget –</p> <p>w = Checkbutton (master, option, ...)</p>
4	<p>Entry: The Entry widget is used to display a single-line text field for accepting values from a user.</p> <p>Syntax to create this widget –</p> <p>w = Entry (master, option, ...)</p>
5	<p>Frame: The Frame widget is used as a container widget to organize other widgets.</p> <p>Syntax to create this widget –</p> <p>w = Frame (master, option, ...)</p>
6	<p>Label: The Label widget is used to provide a single-line caption for other widgets. It can also contain images.</p> <p>Syntax to create this widget –</p> <p>w = Label (master, option, ...)</p>
7	<p>List box: The List box widget is used to provide a list of options to a user.</p> <p>Syntax to create this widget –</p> <p>w = Listbox (master, option, ...)</p>
8	<p>Menu button: The Menu button widget is used to display menus in your application.</p> <p>Syntax to create this widget –</p> <p>w = Menubutton (master, option, ...)</p>
9	<p>Menu: The Menu widget is used to provide various commands to a user. These commands are contained inside Menu button.</p> <p>Syntax to create this widget –</p> <p>w = Menu (master, option, ...)</p>
10	<p>Message: The Message widget is used to display multiline text fields for accepting values from a user.</p> <p>Syntax to create this widget –</p> <p>w = Message (master, option, ...)</p>

S. No.	Operator & Description
11	<p>Radio button: The Radio button widget is used to display a number of options as radio buttons. The user can select only one option at a time.</p> <p>Syntax to create this widget –</p> <p>w = Radiobutton (master, option, ...)</p>
12	<p>Scale: The Scale widget is used to provide a slider widget.</p> <p>Syntax to create this widget –</p> <p>w =Scale (master, option, ...)</p>
13	<p>Scrollbar: The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.</p> <p>Syntax to create this widget –</p> <p>w = Scrollbar (master, option, ...)</p>
14	<p>Text: The Text widget is used to display text in multiple lines.</p> <p>Syntax to create this widget –</p> <p>w = Text (master, option, ...)</p>
15	<p>Toplevel: The Toplevel widget is used to provide a separate window container.</p> <p>Syntax to create this widget –</p> <p>w = Toplevel (master, option, ...)</p>
16	<p>Spinbox: The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.</p> <p>Syntax to create this widget –</p> <p>w = Spinbox (master, option, ...)</p>
17	<p>Paned Window: A Paned Window is a container widget that may contain any number of panes, arranged horizontally or vertically.</p> <p>Syntax to create this widget –</p> <p>w = PanedWindow (master, option, ...)</p>
18	<p>Label Frame: A label frame is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.</p> <p>Syntax to create this widget –</p> <p>w = LabelFrame (master, option, ...)</p>

S. No.	Operator & Description
19	<p>tkMessageBox: This module is used to display message boxes in your applications.</p> <p>Syntax to create this widget –</p> <p>w = tkMessageBox (master, option, ...)</p>

Table 3.2.1: Tkinter widgets

Standard attributes

Let us take a look at how some of their common attributes such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes - pack, grid, and place.

- **The pack() Method:** This geometry manager organizes widgets in blocks before placing them in the parent widget.

Syntax

widget.pack(pack_options)

List of possible options

- Expand:** When set to true, widget expands to fill any space not otherwise used in widget's parent.
- Fill:** Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions - NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
- Side:** Determines which side of the parent widget packs against - TOP (default), BOTTOM, LEFT, or RIGHT.
- **The grid() Method:** This geometry manager organizes widgets in a table-like structure in the parent widget.

Syntax

widget.grid(grid_options)

List of possible options

- ❑ **Column:** The column to put widget in; default 0 (leftmost column).
- ❑ **columnspan:** How many columns widget occupies; default 1.
- ❑ **ipadx, ipady:** How many pixels to pad widget, horizontally and vertically, inside widget's borders.
- ❑ **padx, pady:** How many pixels to pad widget, horizontally and vertically, outside widget's borders.
- ❑ **Row:** The row to put widget in; default the first row that is still empty.
- ❑ **rowspan:** How many rows widget occupies; default 1.
- ❑ **Sticky:** What to do if the cell is larger than widget. By default, with sticky="", widget is centered in its cell. Sticky may be the string concatenation of zero or more of N, E, S, W, NE, NW, SE, and SW, compass directions indicating the sides and corners of the cell to which widget sticks.
- **The place() Method:** This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Syntax

widget.place(place_options)

List of possible options:

- ❑ **Anchor: The exact spot of widget other options refer to:** may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
- ❑ **bordermode:** INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.
- ❑ **Height, width:** Height and width in pixels.
- ❑ **relheight, relwidth:** Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- ❑ **relx, rely:** Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- ❑ **x, y:** Horizontal and vertical offset in pixels.

Scan the QR code or click on the link to watch related videos



<https://www.youtube.com/watch?v=9SMmbc-TFrQ>
Graphical User Interface (GUI)

4. Process of Testing and Troubleshooting the Firmware



Unit 4.1 - Testing of Firmware

Unit 4.2 - Debugging of Firmware



Key Learning Outcomes



At the end of this module, participants will be able to:

1. Describe types of testing in IoT
2. List tools available for IoT testing
3. Describe Interoperability testing
4. Describe Root Cause Analysis (RCA)
5. Describe process of debugging a firmware
6. Describe various features of GDB
7. Demonstrate how to debug a firmware by GDB

Unit 4.1: Testing of Firmware

Unit Objectives

At the end of this unit, participants will be able to:

1. Describe types of testing in IoT
2. List tools available for IoT testing
3. Describe Interoperability testing

4.1.1 IoT Testing

IoT testing is a sub-category of testing to check IoT devices. We now need to provide better and faster services. There is a huge global demand to access, create, use and transfer data. The aim is to provide insight and control, of various interconnected devices. That is why IoT testing framework is so important.

IoT Testing Approaches

To ensure the high quality of IoT products and services, you should develop a thoroughly planned strategy and choose the most efficient IoT testing toolas.

General infrastructure of IoT systems comprises four layers:

- Physical layer — sensors, controllers, other connected devices that collect data
- Network layer — gateways, communication units that ensure connectivity and data transmission
- Data management layer — local or cloud centers (or backend) that provide data storage, aggregation and analysis
- Application layer — software for user interaction (or frontend) that provides reporting and control capabilities

That's why, due to the complexity of such solutions, there are several approaches to IoT testing.

- Check all layers separately.
- Check the interoperability of several layers.
- Check the operation of the entire system.

Types of Testing in IOT

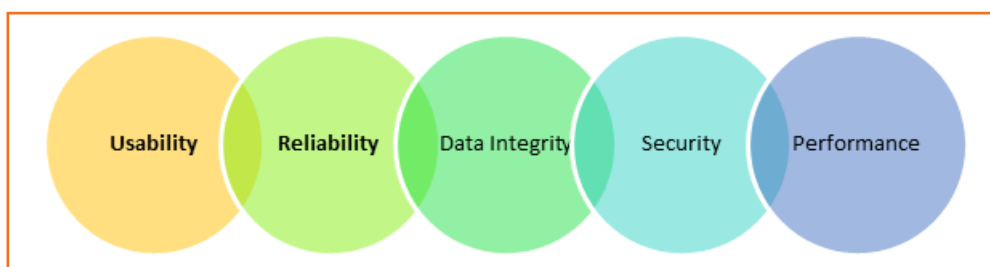


Fig 4.1.1: Types of Testing in IOT

- **Usability Testing:** Users use many devices of varying shape and form factors. Also, the perception varies from user to user. This is why investigating the usability of the system is very important in IoT testing. The usability of each device used in IoT must be determined. In healthcare, the tracking devices used must be portable so that they can be moved to different divisions. The equipment used should be smart enough to push notifications, error messages, warnings, etc. The system must log all the events occurring to provide clarity to the end users.
- **Compatibility Testing:** There are lots of devices which can be connected through IOT system. These devices have varied software and hardware configuration. Each one of them has a high degree of variability in terms of the firmware and hardware models and versions; network type, speed, protocols and versions; operating system type and versions; browser type and versions; screen sizes and display resolutions to name a few. It is important to test the application in all possible combinations of these versions to reduce failures in the field.

Compatibility testing is also important due to the complex architecture of the IoT system. Testing items like OS versions, browser types, devices' generation, communication modes is vital for compatibility testing.

- **Connectivity Testing:** This testing involves checking the device and application behavior by subjecting the network through a load, intermittent failures, and total loss of connectivity. By inducing these real-life scenarios, the robustness of the device, edge, platform and application are checked.
- **Reliability and Scalability Testing:** Reliability and Scalability is important for building an IOT test environment which involves simulation of sensors by utilizing virtualization tools and technologies.
- **Data Integrity Testing:** It's important to check the Data integrity in IOT testing as it involves large amount of data and its applications.
- **Security testing:** In the IOT environment, a large number of users are accessing a massive amount of data. Thus, it is important to validate user via authentication, have data privacy controls as part of security testing.
- **Performance Testing:** Performance testing is important to create strategic approach for developing and implementing an IOT testing plan. On the device, these tests check their responsiveness to user actions and on a platform layer, they check the ability to handle spikes in traffic gracefully. They are based on metrics for assessing the responsiveness of the device/application and underlying system performance. Load generators and performance measuring tools on the cloud rate system performance under normal and full load.

The chart below is the applications of the different types of testing for various IoT components.

IoT Testing Types	Sensor	Application	Network	Backend
Functional testing	True	True	False	False
Usability testing	True	True	False	False
Security testing	True	True	True	True

IoT Testing Types	Sensor	Application	Network	Backend
Performance testing	False	True	True	True
Compatibility testing	True	True	False	False
Services testing	False	True	True	True
Operational testing	True	True	True	True

Table 4.1.1: Applications of the different types of testing

Challenges faced in IoT testing

- Both, the network and internal communication, needs to be checked.
- One of the biggest concerns in IoT testing is security and privacy because the tasks are done via Internet.
- The software complexity as well as the system itself may conceal the bugs or defects found in the IoT technology.
- There are limitations on memory, processing power, bandwidth, battery life, etc.
- Suggestions to make IoT testing effective
- Gray box testing and IoT testing should be performed simultaneously as it enables the designing of effective test cases. This helps us understand the operating system, architecture, third-party hardware, new connectivity, and hardware restrictions.
- Real-time OS is vital to provide scalability, modularity, connectivity, and security, all of which are essential to IoT.
- To make it effective, IoT testing can be automated.
- Tools for IoT testing
- **Shodan:** This tool can be used to determine which device/s is/are connected to the Internet. It helps track all the computers which can be directly accessed from the Internet. Shodan is also used in connectivity testing. It helps in the verification of the devices connected to the IoT hub. It provides the connected devices, their locations, user information, etc. It tracks and records all the computers connected to the network.
- **Thingful:** This is a Search Engine for IoT. It helps keep interoperability between millions of objects through the Internet, secured. Thingful is used to control how the data is used. It also helps take more decisive and valuable decisions.
- **Wireshark:** This open-source tool is used to monitor the traffic in interfaces, source/destination host addresses, and so on.
- **Tcpdump:** This tool is quite similar to Wireshark, but for the absence of GUI (Graphical User Interface). This tool is based on command line. It helps users display packets such as TCP/IP that are transmitted over a network.
- **JTAG Dongle:** This tool is quite like a debugger in desktop applications. It is used in debugging the

target platform or device code, and display variables step by step.

- **Digital Storage Oscilloscope:** This tool is used to investigate the different events with time stamps, glitches in power supply, and signal integration.
- **Software Defined Radio:** This tool is used to mimic receiver and transmitter for a wide range of wireless gateways.
- **MQTT Spy:** If the device supports MQTT protocol, then this tool is the most useful. MQTT Spy is an efficient open-source tool for IoT testing. It is particularly useful for day-to-day usage.
- Prerequisites of IoT Testing
- **Setting up IoT device:** The IoT device must be turned on, and can be accessed and used in real life. E.g., while testing a smart watch, make sure to wear it on wrist. Placing it on a table would not be regarded as a real user case.
- **Setting up of IoT Hub:** IoT hub is a server that can connect with IoT devices and gather information from them. An IoT hub may be an application in a mobile device or a web server on a cloud. The IoT hub must be set up properly.
- **Setting up network:** We need a strong wireless connection to connect IoT hub and the IoT device together. This can be possible with a Wi-Fi, Bluetooth, satellite signals, NFC (near field communication), etc. While connecting wearable device with a mobile app, ensure the following –
 - The Bluetooth of both the devices is turned on.
 - Both the devices are paired together.
 - Both the devices are in range of each other.

4.1.2 Interoperability Testing

Interoperability Testing is a type of testing where it is checked if a component can interact with other software components. It checks the functionality between two software systems as per the requirement of customers. It validates that end to end functionality between two systems is as required. It ensures that there is the end to end communication and reduces the compatibility issue between two systems when data is being transferred. It provides uniform data type and data format which is present between two software systems. This testing is an integral part as many different kinds of technologies and architecture are being used where seamless operations are to be performed. Other technically qualified software applications or specific software components from other relevant applications which, in effect, are linked to form an effective operating product as specified by the customer as a requirement.

Need of Interoperability Testing

- The need for Interoperability testing arises because it is important to make sure that end to end service is being provided across two or more software that is involved in the system. These systems can be of different vendors and may have different architectures or may be using different technologies.
- The communication and data exchange between the systems should be smooth and the

software that is included should be smooth. There should not be any compatibility issues for any communication which will take place.

- As there can be different architectures, technologies, products, and vendors involved in a complete system all these components must be in working condition with one another. There should not be any issues arising between the different components involved.
- It should be validated that the data which is being exchanged does not get modified and is in its original state. Data exchange can happen without any prior notice. It should also be noted that all applications in the network perform their expected behavior on their own.

Interoperability Testing Process

The testing process involves the below steps in Interoperability testing:

1. **Test Environment Setup:** This is the first step in testing. The environment needs to be set up to test interoperability. Without having a proper environment being setup it will not be possible to test. A formal statement of work needs to be set up for the infrastructure.
2. **Create Test Case:** Different test cases are created to check different scenarios and connection behaviors. To cover different scenarios different test cases should be created. This is done to perform testing more efficiently. Before this, all setup must be done like setting up automation tools to reduce test cases and reuse them. All database configurations should be made and metrics should be measured.
3. **Test Case Execution:** Once the test cases are made they need to be executed on the environment which is set up. The execution lets us know the actual behavior of software and lets us know how the software will behave when it goes live and how it communicates with the other components.
4. **Test Result Analysis:** Once the execution is complete all the test results should be analyzed and verified. The defects that are found should be noted and resolved. The test team should get the root cause of the failure that is found. These should be made sure that they are resolved.
5. **Retest:** The defects that are noted should be made sure are resolved. Once the development team resolves the defect then it should be made sure that the testing is performed again, and the entire process is repeated. The issues should now be resolved.

Once these activities are done it should be made sure all results are documented and a record is maintained of all test logs and test results.

Types of Interoperability Testing

There are five Types of Interoperability Testing

- **Data type Interoperability:** It mainly focuses on checking that data types are being transferred from one type to another. There should not be any inconsistency of data when data is transferred among the systems.
- **Semantic Interoperability:** This type focuses on the algorithm which is used to transfer the data. It checks for the semantics which is involved and verifies if the algorithm is reliable or not.
- **Physical Interoperability:** This checks if the connections between the two or more systems are proper or not. The ports and cables which are used should not affect the speed or rate of transfer.

- **Protocol Interoperability:** The protocol which is used for data transfer is checked for the security of data. Checksum should be enabled to transfer data without any error.
- **Data Format Interoperability:** The format in which data is sent and received should be the same in both systems.

Advantages and Disadvantages of Interoperability Testing

Advantages:

- Interoperability testing helps in having an establishment of the connection between two systems. This connection helps in getting a better picture of how the system will work in synchronization with other products. It helps in fostering better communication between two disparate systems.
- It boosts efficiency. When data is presented consistently then decision making can be easier.
- It ensures that uniform data type is being transferred and there are no mismatches with the data type. The data type should be uniform and compatible over the system so that there are no issues.
- The data formatting ensures that there is uniform formatting followed in the entire system. All software is in sync and there is no incompatibility due to data formatting
- It makes sure that all interacting systems have the same semantics or algorithm.

Disadvantages:

- It requires accurate measurements so that all systems can work well in an end to end environment
- The network complexity is more in this testing as all components are to be tested
- The requirements are inadequate in this testing.

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=9QiNMUdAgg
Types of IoT testing



www.youtube.com/watch?v=Am9SW1T_Qvs
Interoperability testing

Unit 4.2: Debugging of Firmware

Unit Objectives



At the end of this unit, participants will be able to:

1. Describe Root Cause Analysis (RCA)
2. Describe process of debugging a firmware
3. Describe various features of GDB
4. Demonstrate how to debug a firmware by GDB

4.2.1 Root Cause Analysis (TCA) of Issue

Root Cause Analysis (RCA) is a structured and effective process to find the root cause of issues in a software. If performed systematically, it can improve the performance and quality of the deliverables and the processes, not only at the team level but also across the organization.

RCA is a mechanism of analyzing the Defects, to identify its cause. We brainstorm, read and dig the defect to identify whether the defect was due to **“testing miss”**, **“development miss”** or was a **“requirement or designs miss”**.

When RCA is done accurately, it helps to prevent defects in the later releases or phases. If we find, that a defect was due to design miss, we can review the design documents and can take appropriate measures. Similarly, if we find that a defect was due to testing miss, we can review our test cases or metrics, and update it accordingly.

RCA Process

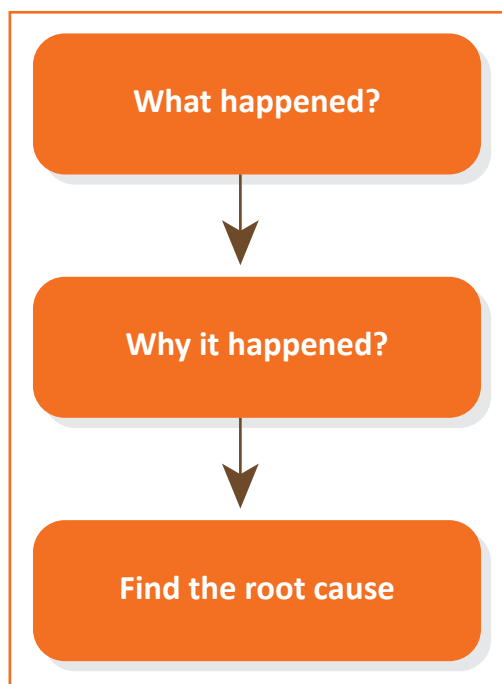


Fig 4.2.1: RCA Process

RCA is aimed at finding the root cause and not treating the symptom, by following a specific set of steps and associated tools. It is different from defect analysis, troubleshooting, and other problem-solving methods as these methods try to find the solution for the specific issue, but RCA tries to find the underlying cause.

Advantages of RCA

- Prevent the reoccurrence of the same problem in the future.
- Eventually, reduce the number of defects reported over time.
- Reduces developmental costs and saves time.
- Improve the software development process and hence aiding quick delivery to market.
- Improves customer satisfaction.
- Boost productivity.
- Find hidden problems in the system.
- Aids in continuous improvement.

Steps to do RCA

1. Form RCA Team

- If the problem has been around for a while, it is probably not easy to find and fix. A team approach pools the experience and expertise of all members to focus on and solve the problem.
- A Problem Statement communicates the scope of the problem, establishes a common understanding of the problem among all team members and focuses the team on the task at hand.

2. Define the problem: Collect the details of the problem like, incident reports, problem evidence (screenshot, logs, reports, etc.), then study/analyze the problem by asking the below questions -

- What is the problem?
- What is the sequence of events that led to the problem?
- What systems were involved?
- How long the problem existed?
- What is the impact of the problem?
- Who was involved and determine who should be interviewed?

3. Identify root cause: Conduct the BRAINSTORMING session within the RCA team formed to identify the causes.

Think of root cause analysis as “**problem-finding**”; once the real problem (the root cause) has been found, the solution to the problem is often obvious.

Follow the trail of evidence; the evidence (data) will point the way from the symptoms to the root cause of the problem.

4. Implement Root Cause Corrective Action (RCCA): Correction action involves giving fix to the solution by identifying the real root cause.

Once the root cause or causes have been found, typically one or more potential solutions will become apparent as well. Now you have to select the best solution or combination of solutions that will lead to a robust yet cost-effective solution.

Screen the solutions by testing the practicality, feasibility, and cost-effectiveness of the solution. Once the solution is approved for implementation, it is time to create an action plan.

RCCA should be implemented in such a way that this root cause will not occur again in the future. Fix given by the support team will be temporary for the customer site where the issue is reported. When this fix is merged into an ongoing version, do proper impact analysis to ensure no existing feature is broken.

5. **Implement Root Cause Preventive Action (RCPA):** The team needs to come up with a plan for how such a similar issue can be prevented in the future. For example, Update Instruction Manual, improve skillset, update the team assessment checklist, etc. Follow proper documents of preventive actions and monitor whether the team is adhering to the preventive actions taken.

4.2.2 Debugging

Debugging is a technique to find and remove the number of errors or bugs or defects in a program is called Debugging. It is a multistep process in software development. It involves identifying the bug, finding the source of the bug and correcting the problem to make the program error-free.

Differences between testing and debugging

Parameters	Testing	Debugging
Basics	It is the process using which we find errors and bugs.	It is the process using which we correct the bugs that we found during the testing process.
Code Failure	We can identify the failure of any implemented code using this process.	We use this process to provide the code failure with an absolution.
Errors	Errors get displayed in this process.	Errors get deducted and dissolved in this process.
Performer	A tester performs testing on any given software or application.	Either a developer or a programmer performs this step of debugging in an application or software.
Design Knowledge	One does not need any design knowledge to perform testing.	Design knowledge is a prerequisite to debugging.

Parameters	Testing	Debugging
Insiders and Outsiders	Both- insiders and outsiders can perform testing.	Only an insider can perform debugging. No outsider can perform it on the intended software.
Mode of Operation	The process of testing can be both- automated as well as manual.	The process of debugging must always be manual since we are fixing the available errors and bugs. We cannot debug a system/ software/ application on auto-pilot.
Basis of Operation	The process of testing is based on various levels of testing- system testing, integration testing, unit testing, etc.	The process of debugging is based on various types of bugs present in a system.
SDLC	It is a stage of SDLC (Software Development Life Cycle).	It's not at all an aspect of the SDLC. It rather occurs as a consequence of the process of testing.
Moment of Initiation	This process can begin after we have completed writing the code.	This process can begin after the execution of the test case and the identification of errors.
Steps	This process consists of both- validation as well as verification of the errors.	This process seeks to match the available symptoms with the cause. Only then it leads to the correction of the errors.

Table 4.2.2 Difference between testing and debugging

To resolve the defect in the program or application, you need to identify the piece of source code that is causing the problem. You can trace the program execution step by step by evaluating the values of the variables and the return values of the functions.

The types of errors and the debugging techniques vary depending on the stage of development and the environment.

Types of Possible Errors

To understand different types of debugging, we need to understand the different types of errors. Errors are usually of three types -

- Build and compile-time errors happen at the development stage when the code is being built.

These errors are made by the compiler or the interpreter while building the source code. Build or compile-time errors prevent the application from even starting. These errors often result from syntax errors, like missing semicolons at the end of a statement or class not found. These errors are easy to spot and rectify because most IDE or compilers find them for you.

- Runtime errors occur and can be identified only while running the application. They occur only when the source code doesn't have any compiler or syntax error, and the compiler or the interpreter cannot identify the runtime error during the build stage. Mostly, runtime errors depend on the user input or the environment. These kinds of errors can be identified by using try-catch blocks in your program and logging the error message properly.
- Logic errors occur after the program is successfully compiled and running and it gives you an output. A logic error is when the result of the program is incorrect. These errors cannot be caught using the try-catch blocks. Logic errors are also called semantic errors, and they occur due to some incorrect logic used by the developer to solve a problem while building the application.

Types of debugging

Not all errors can be treated or debugged in the same way. You have to set up the right strategy to fix different errors. There are two types of debugging techniques -

1. **Reactive Debugging:** Reactive debugging refers to any debugging protocol that is employed after the bug manifests itself. Reactive debugging is deployed to reduce runtime and logic errors. Examples of reactive debugging are print debugging and using a debugger.
 - **Print debugging:** Print debugging is the print statements developers write in their source code (printf in C or System.Out.print in Java or print in Python). Developers write them to print out the values of the variables to the console or to the file to get a better sense of what is happening. These printed statements help find the root cause of the problem, especially when a variable has unexpected values.
 - **Debugger:** Print debugging is an easy technique, and sometimes developers use it to find the root cause of an error. However, it is a time-consuming and tedious process if the developer is dealing with several modules, functions, and variables. If a developer finds themselves writing more than 10 print statements to identify the issue, then it's time to switch to a debugger.

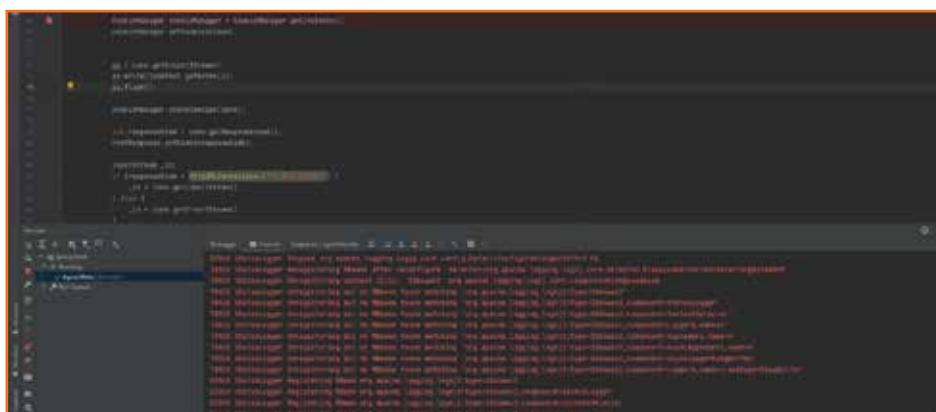


Fig 4.2.2: Debugger screen

The debugger allows the developer to trace the execution of the program and identify the state

of the variable functions as the program runs. In a sense, it is not too different from a print debugger. What makes it better for larger applications is that it prints every single variable and line of source code as the program executes. Integrated development environments like Eclipse, JetBrains IntelliJ, Pycharm, and Visual Studio include built-in GUI debuggers. There are many command-line debuggers like GDB for C and pdb for Python available in the market.

Additionally, sometimes one can face a situation where the application in question runs on a different host. Collecting data from that system thus becomes really challenging. In such cases, remote debugging allows developers to trace the issue on a different host.

Most debugging is reactive — a defect is reported in the application or an error occurs, and the developer tries to find the root cause of the error to fix it. Solving build errors is easier, and generally, the compiler or build system clearly tells you the errors.

Solving a runtime error can be done using a debugger, which can provide additional information about the error and the stack trace of the error. It will tell you exactly the line where the fault is happening or exception is raised.

Solving logic errors is trickier. We don't get clues as we do in other errors from the compiler or the stack trace. We need to use other strategies to identify the code causing the error.

2. **Preemptive Debugging:** Preemptive debugging involves writing code that doesn't impact the functionality of the program but helps developers, either catch bugs sooner or debug the source code easily when the bug occurs.

4.2.3 Debugging of Firmware

Debugging is an integral part of embedded systems as much as designing. Embedded systems have become very complex these days and the boundaries of software and hardware are merging. So when an issue occurs at system level it becomes difficult to find out the root cause. As an analyst working on embedded systems it is important that you quickly understand the issue and find the root cause.

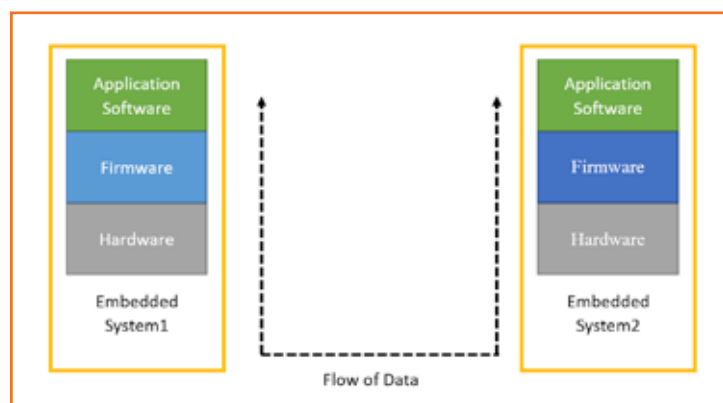


Fig 4.2.3: Generic diagram of an embedded system

An embedded system consists of hardware, firmware and application software. Sometimes when an issue is reported it is not clear where the issue is present in the system. It can be due to hardware,

firmware code or application software. Below is a generic diagram showing the basics of an embedded system along with the direction of data flow from one system to another.

Debugging Process

The process of finding bugs or errors and fixing them in any application or software is called debugging. To make the software programs or products bug-free, this process should be done before releasing them into the market. The steps involved in this process are -

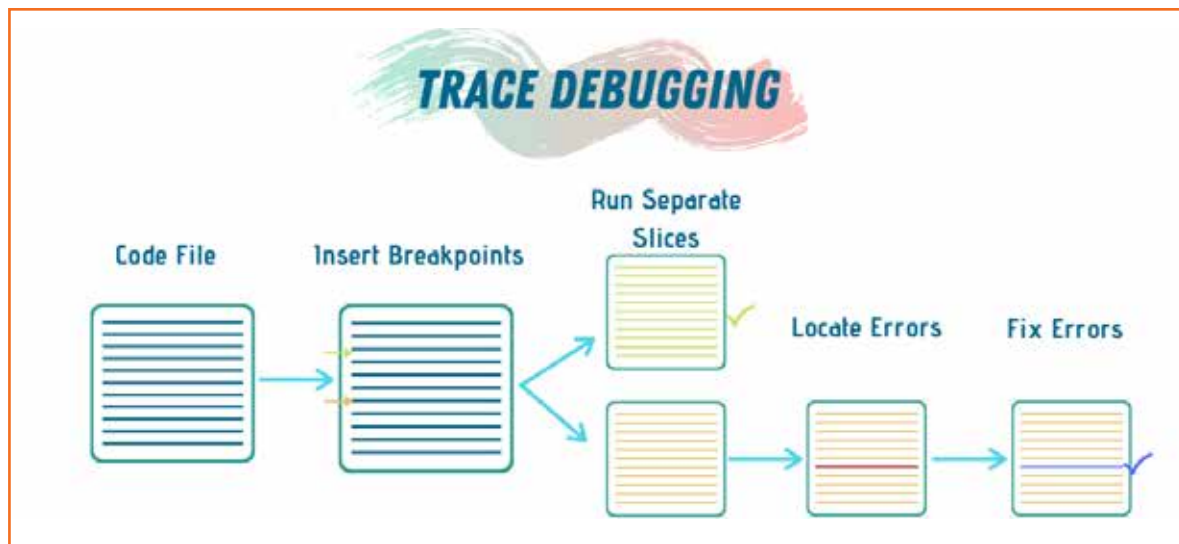


Fig 4.2.4: Debugging process

1. **Identifying the error:** It saves time and avoids the errors at the user site. Identifying errors at an earlier stage helps to minimize the number of errors and wastage of time.
2. **Identifying the error location:** The exact location of the error should be found to fix the bug faster and execute the code.
3. **Analyzing the error:** To understand the type of bug or error and reduce the number of errors we need to analyze the error. Solving one bug may lead to another bug that stops the application process.
4. **Prove the analysis:** Once the error has been analyzed, we need to prove the analysis. It uses a test automation process to write the test cases through the test framework.
5. **Cover the lateral damage:** The bugs can be resolved by making the appropriate changes and move onto the next stages of the code or programs to fix the other errors.
6. **Fix and Validate:** This is the final stage to check all the new errors, changes in the software or program and executes the application.

Debugging Software

Debugging software finds the bugs, analyze the bugs and enhance the quality and performance of the software. The process of resolving the bugs using manual debugging is very tough and time-consuming. We need to understand the program, it's working, and the causes of errors by creating breakpoints.

As soon as the code is written, the code is combined with other stages of programming to form a new software product. Several strategies like unit tests, code reviews, and pair programming are used to debug

the large program (contains thousands of lines of code). The standard debugger tool or the debug mode of the Integral Development Environment (IDE) helps determine the code's logging and error messages.

The steps involved in debugging software are:

- The bug is identified in a system and defect report is created. This report helps the developer to analyze the error and find the solutions.
- The debugging tool is used to know the cause of the bug and analyze it by step-by-step execution process.
- After identifying the bug, we need to make the appropriate changes to fix the issues.
- The software is retested to ensure that no error is left and checks all the new errors in the software during the debugging software process.
- A sequence-based method used in this software process made it easier and more convenient for the developer to find the bugs and fix them using the code sequences.

Debugging techniques in embedded systems

These techniques reduce the error count and increase the quality and functionality of the code. Debugging of the embedded systems depends on physical memory addresses and virtual memory.

There are 6 debugging techniques in an embedded system.

- Simplify the complex data
- Divide and conquer
- Slow down the process
- Change only one variable at a time
- Creating off-line models
- Start from a known-good state

Different debugging techniques are used in different cases. A combination of one or more approaches may cause errors. This process includes -

- Reproduce the bug or problem
- Explain the bug using input from the user
- Try to get all the variable values and state of the program when the bug appears
- Analyze the bug and find the cause of the bug
- Fix the bug and check all the causes of new bugs

Listed discuss the step by step approach on how to go forward in debugging an issue. To illustrate the approach, an example of a video display issue seen on an embedded system is considered.

7. **Understand the setup & reproduce the issue correctly:** The first and foremost thing you need to do is to correctly reproduce the issue. Sometimes the issues are seen locally and the you are able to reproduce the issue easily. Whereas sometimes the issues are seen at a remote location or customer site and then you have to rely purely on logs available to understand the setup and reproduce the issue. In the second case, you need to understand the setup correctly as it will help

in successfully reproducing the issue. If you fails to do so, then it can happen that the issue can re-occur at the remote location or customer site in the future. This happens primarily because you may not have correctly understood the issue and hence has not developed the right solution. Thus it would lead to multiple repetitions of fixing the issue.

For our example, let's consider a video display where noise was seen only on some monitors. Since the devices were installed in a remote location so only the video logs were available for debugging. From the logs it was difficult to make out the root cause of noise. Initially we tried playing with different clips but were not able to reproduce the issue at our end. We were not sure of why the issue was not reproducible. We could not tell clearly if it was due to the video file used or due to the setup. Finally we were able to reproduce the issue by getting the same exact clip that the customer used and one of the monitors.

- 8. Breaking the issue into smaller issues:** Once the issue is correctly reproduced the next step would be to break the whole issue into smaller issues. The first step would be to break the flow of data at the interface of application layer with firmware layer and then firmware layer with hardware layer. This way each layer can be reviewed and tested independently for any issue.

For our example, we divided the whole path of video frame data from the application to the hardware layer. We understood the whole path of how encoded video data is received and then decoded, and after that how it is passed to the firmware layer. Here we understood how the pointers are assigned for each video frame and how they are given by the firmware to send each frame over the hardware layer. At the hardware layer, we understood the protocol of how a video frame is sent out on the physical lines. Once we understood the whole path, we divided it into logical blocks. One block was for application layer, where the encoded video data is decoded to raw video and stored in video buffers. Then another block is for firmware layer, where we check how the video buffers are given out to the hardware. The final block is for hardware where we check how the video data is given out on the actual physical lines.

- 9. Fix each of the smaller issues:** Once we have broken the whole data path into logical blocks for each layer, we needed to individually test each block and validate it in their own respective ways. This would help in finding where the root cause of the issue lies. Sometimes a system issue can be fixed by changing only one layer whereas sometimes it would need a change in more than one layer. By logically breaking the whole path, we can properly find out where all changes are needed and then fix them accordingly.

For our example, at application layer we used reading and writing into a file for validating the data path flow. The video data buffers generated after decoding were compared with the expected values. At the firmware level, fixed pattern was used as data input instead of the data coming from application layer. Here we observed that the video data was given to the firmware layer in terms of fields and not frames but the field information (top or bottom) was not given correctly from application layer to firmware layer. So we had to modify the code accordingly so that the buffer contains the correct field information.

At the hardware level, these fixed patterns and the corresponding control signal for the interface was validated as per the specifications using logic analysers. For our example, the protocol we

were using was BT.1120 and we saw that the protocol timings were not as per the specification. So we realized that this was the reason why some monitors worked properly and others did not. Once we made the protocol as per the specification we saw that all the monitors worked properly. We also realized that the whole issue of noise was actually a combination of the wrong field information and wrong protocol timings. This was the reason why some monitors were able to work and others did not.

10. **Negative Testing:** Testing is of course a very important aspect of problem solving and it is important to test that the issue is fixed properly and should not come back. So it is important that we do negative testing along with the usual testing done after fixing an issue. Negative testing basically means to ensure that the issue is forced into the system and then the system's response is validated as per the designed solution. This basically means that if we are getting the wrong output from a system by giving correct input and we come up with a solution, so then we should be able to generate a wrong input and feed to the system so that it will generate the correct output. If this happens that means the root cause is correctly identified and the fix is validated.

For our example, we tested the following way. For protocol timings, we saw that after making them as per the specification, all the monitors showed the noise issue. Even one slight modification of the specification led to change in behaviour of monitors. This confirmed that the wrong protocol timings at the hardware layer was the root cause for different behaviour of monitors. Next, for a monitor where we were seeing the issue, we knowingly swapped the top and bottom fields at the application layer. Then we saw that the issue did not occur. This confirmed that the incorrect top and bottom field pointers was the root cause for the noise issue. This way we tested that the solution actually addressed the root cause and it included a fix in both application and hardware layer.

Debugging Tools

A software tool or program used to test and debug the other programs is called a debugger or a debugging tool. It helps to identify the errors of the code at the various stages of the software development process. These tools analyze the test run and find the lines of codes that are not executed. Simulators in other debugging tools allow the user to know about the display and behavior of the operating system or any other computing device. Most of the open-source tools and scripting languages don't run an IDE and they require the manual process.

Mostly used Debugging Tools are:

- **GDB Tool:** This type of tool is used in Unix programming. GDB is pre-installed in all Linux systems if not, it is necessary to download the GCC compiler package.
- **DDD Tool:** DDD means Data Display Debugger, which is used to run a Graphic User Interface (GUI) in Unix systems.
- **Eclipse:** An IDE tool is the integration of an editor, build tool, debugger and other development tools. IDE is the most popular Eclipse tool. It works more efficiently when compared to the DDD, GDB and other tools.

Here, we will discuss about GDB in detail in next section.

4.2.4 Gnu DeBugger (GDB)

A debugger enables the programmer to step through code in execution, place breakpoints in the code, view memory, change variables, and track variables, among other capabilities. One of the most common serial debuggers available is the GNU debugger (GDB). The GDB debugger is a command-line debugger where the user can give a series of commands to set a break point, continue execution, examine a variable, set a watch point, etc.

GDB is a powerful source-level debugging package that lets you see what is going on inside your program. You can step through the code, set breakpoints, examine and change variables, and so on. There are several graphical front ends for GDB that translate GUI commands into GDB text commands.

GDB can be set up to run on a host workstation while debugging code on a separate target machine. The two machines can be connected via serial or network ports, or you can use an in-circuit emulator (ICE).

There are also specialized debug ports built into some processor chips. These go by the names Joint Test Action Group (JTAG) and Background Debug Mode (BDM). These ports provide access to debug features such as hardware breakpoint registers. GDB back ends exist for these ports as well.

How GDB Debugs?

GDB allows you to run the program up to a certain point, then stop and print out the values of certain variables at that point, or step through the program one line at a time and print out the values of each variable after executing each line.

GDB uses a simple command line interface.

GDB - Installation

You can install GDB on your system by following the simple steps discussed below.

1. Make sure you have the prerequisites for installing GDB –
 - An ANSI-compliant C compiler (gcc is recommended – note that gdb can debug codes generated by other compilers)
 - 115 MB of free disk space is required on the partition on which you're going to build gdb.
 - 20 MB of free disk space is required on the partition on which you're going to install gdb.

2. Use the following command to install GDB on linux machine.

```
$ sudo apt-get install libc6-dbg gdb valgrind
```

3. Now use the following command to find the help information.

```
$gdb -help
```

You now have GDB installed on your system and it is ready to use.

If GDB is already installed on your system, you can check it by issuing the following command –

```
$gdb -help
```

If GDB is installed, then it will display all the available options within your GDB.

GDB - Debugging Symbols

A Debugging Symbol Table maps instructions in the compiled binary program to their corresponding variable, function, or line in the source code. This mapping could be something like

Program instruction \Rightarrow item name, item type, original file, line number defined.

Symbol tables may be embedded into the program or stored as a separate file.

GDB - Commands

List of commands in GDB are -

- b main - Puts a breakpoint at the beginning of the program
- b - Puts a breakpoint at the current line
- b N - Puts a breakpoint at line N
- b +N - Puts a breakpoint N lines down from the current line
- b fn - Puts a breakpoint at the beginning of function “fn”
- d N - Deletes breakpoint number N
- info break - list breakpoints
- r - Runs the program until a breakpoint or error
- c - Continues running the program until the next breakpoint or error
- f - Runs until the current function is finished
- s - Runs the next line of the program
- s N - Runs the next N lines of the program
- n - Like s, but it does not step into functions
- u N - Runs until you get N lines in front of the current line
- p var - Prints the current value of the variable “var”
- bt - Prints a stack trace
- u - Goes up a level in the stack
- d - Goes down a level in the stack
- q - Quits gdb

Interfaces / Protocols

Two common protocols exist to communicate and introspect microcontrollers -

JTAG (Joint Test Action Group):

JTAG has a long history, and was originally intended to daisy-chain multiple micro-processors, FPGAs, memory chips, simply anything which supports JTAG, and access them over the same JTAG bus. To enable this, it requires more signals than SWD -

- TDI (Test Data In)
- TDO (Test Data Out)

- TCK (Test Clock)
- TMS (Test Mode Select)
- TRST (Test Reset) which is optional

(a variant of JTAG called cJTAG, c for compact exists that uses 2 signals)

The following image illustrates a common communication example with a JTAG bus with multiple devices:

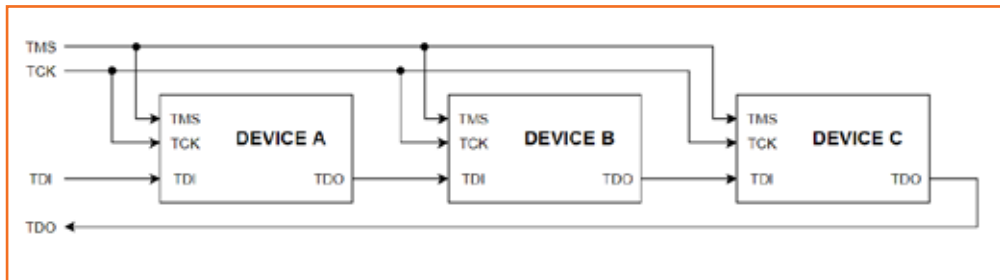


Fig 4.2.5: JTAG bus with multiple devices

SWD (Serial Wire Debug)

SWD is quite similar to JTAG, as it uses the same protocol but it is limited to two electrical signals and cannot address multiple targets on a single connection. SWD is also a proprietary ARM technology. While JTAG is vastly more versatile, SWD's strength is in its simplicity as many systems only have one microcontroller to program. In these cases the reduction of signals while still having full debugability is a major win. In SWD the following signals are available

- SWDIO (Serial Wire Debug Input / Output)
- SWCLK (Serial Wire Clock)

The following image illustrates a common communication example with SWD communicating with a single microcontroller:

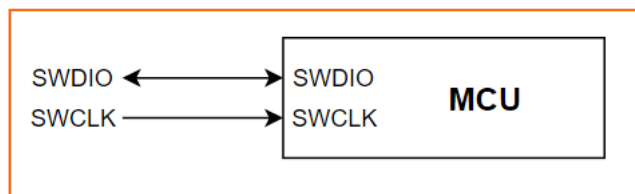


Fig 4.2.6: SWD communication with single microcontroller

Which one to use?

When designing a system, we should select the interface which is most convenient for the system, and there are simple criteria to consider -

- Does the target support SWD?
- Does the system have more than a single component which needs to be programmed or debugged?
- How valuable is board space and is the larger area required for JTAG acceptable?
- What does the component manufacturer recommend?

4.2.5 Using Gnu Debugger (GDB)

Starting and Stopping GDB

- **gcc -g myprogram.c**

Compiles myprogram.c with the debugging option (-g). You still get an a.out, but it contains debugging information that lets you use variables and function names inside GDB, rather than raw memory locations (not fun).

- **gdb a.out**

Opens GDB with file a.out, but does not run the program. You'll see a prompt (gdb) - all examples are from this prompt.

- **r**

r arg1 arg2

r < file1

Three ways to run "a.out", loaded previously. You can run it directly (r), pass arguments (r arg1 arg2), or feed in a file. You will usually set breakpoints before running.

- **Help - h breakpoints**

Lists help topics (help) or gets help on a specific topic (h breakpoints). GDB is well-documented.

- **q - Quit GDB**

Stepping through Code

Stepping lets you trace the path of your program, and zero in on the code that is crashing or returning invalid input.

- **l**

- **l 50**

- **l myfunction**

Lists 10 lines of source code for current line (l), a specific line (l 50), or for a function (l myfunction).

- **next**

Runs the program until next line, then pauses. If the current line is a function, it executes the entire function, then pauses. next is good for walking through your code quickly.

- **step**

Runs the next instruction, not line. If the current instruction is setting a variable, it is the same as next. If it's a function, it will jump into the function, execute the first statement, then pause. step is good for diving into the details of your code.

- **finish**

Finishes executing the current function, then pause (also called step out). Useful if you accidentally stepped into a function.

Breakpoints or Watchpoints

Breakpoints play an important role in debugging. They pause (break) a program when it reaches a certain point. You can examine and change variables and resume execution. This is helpful when some input failure occurs, or inputs are to be tested.

- **break 45**

- **break myfunction**

Sets a breakpoint at line 45, or at myfunction. The program will pause when it reaches the breakpoint.

- **watch x == 3**

Sets a watchpoint, which pauses the program when a condition changes (when `x == 3` changes). Watchpoints are great for certain inputs (`myPtr != NULL`) without having to break on every function call.

- **continue**

Resumes execution after being paused by a breakpoint/watchpoint. The program will continue until it hits the next breakpoint/watchpoint.

- **delete N**

Deletes breakpoint N (breakpoints are numbered when created).

Setting Variables

Viewing and changing variables at runtime is a critical part of debugging. Try providing invalid inputs to functions or running other test cases to find the root cause of problems. Typically, you will view/set variables when the program is paused.

- **print x**

Prints current value of variable x. Being able to use the original variable names is why the (-g) flag is needed; programs compiled regularly have this information removed.

- **set x = 3**

- **set x = y**

Sets x to a set value (3) or to another variable (y)

- **call myfunction()**

- **call myotherfunction(x)**

- **call strlen(mystring)**

Calls user-defined or system functions. This is extremely useful, but beware of calling buggy functions.

- **display x**

Constantly displays the value of variable x, which is shown after every step or pause. Useful if you

are constantly checking for a certain value.

- **undisplay x**

Removes the constant display of a variable displayed by display command.

Backtrace and Changing Frames

A stack is a list of the current function calls - it shows you where you are in the program. A frame stores the details of a single function call, such as the arguments.

- **bt**

Backtraces or prints the current function stack to show where you are in the current program. If main calls function a(), which calls b(), which calls c(), the backtrace is

```
c <= current location
b
a
main
```

- **up**

- **down**

Move to the next frame up or down in the function stack. If you are in c, you can move to b or a to examine local variables.

- **return**

Returns from current function.

Handling Signals

Signals are messages thrown after certain events, such as a timer or error. GDB may pause when it encounters a signal; you may wish to ignore them instead.

- handle [signalname] [action]
- handle SIGUSR1 nostop
- handle SIGUSR1 noprint
- handle SIGUSR1 ignore

Instruct GDB to ignore a certain signal (SIGUSR1) when it occurs. There are varying levels of ignoring.

4.2.6 Debugging C Program Using GDB

First create a C program that calculates and prints the factorial of a number. However this C program contains some errors in it for our debugging purpose.

```
$ vim factorial.c
# include <stdio.h>
int main()
{
    int i, num, j;
    printf ("Enter the number: ");
    scanf ("%d", &num );

    for (i=1; i<num; i++)
        j=j*i;
    printf("The factorial of %d is %d\n",num,j);
}
```

```
$ cc factorial.c

$ ./a.out
Enter the number: 3
The factorial of 3 is 12548672
```

Let us debug it while reviewing the most useful commands in GDB

Step 1. Compile the C program with debugging option -g

Compile your C program with -g option. This allows the compiler to collect the debugging information.

```
$ cc -g factorial.c
```

Step 2. Launch GDB

Launch the C debugger (GDB) as shown below.

```
$ gdb a.out
```

Step 3. Set up a break point inside C program

Syntax:

```
break line_number
```

Other formats:

- **break [file_name]:line_number**
- **break [file_name]:func_name**

Places break point in the C program, where you suspect errors. While executing the program, the debugger will stop at the break point, and gives you the prompt to debug.

So before starting up the program, let us place the following break point in our program.

```
break 10
Breakpoint 1 at 0x804846f: file factorial.c, line 10.
```

Step 4. Execute the C program in GDB debugger

```
run [args]
```

You can start running the program using the run command in the gdb debugger. You can also give command line arguments to the program via run args. The example program we used here does not requires any command line arguments so let us give run, and start the program execution.

```
run
Starting program: /home/sathiyamoorthy/Debugging/c/a.out
```

Once you executed the C program, it would execute until the first break point, and give you the prompt for debugging.

```
Breakpoint 1, main () at factorial.c:10
10          j=j*i;
```

You can use various GDB commands to debug the C program as explained in the sections below.

Step 5. Printing the variable values inside GDB debugger

```
Syntax: print {variable}
```

Examples:

```
print i
print j
print num
```

```
(gdb) p i
$1 = 1
(gdb) p j
$2 = 3042592
(gdb) p num
$3 = 3
(gdb)
```

As you see above, in the factorial.c, we have not initialized the variable j. So, it gets garbage value resulting in a big numbers as factorial values.

Fix this issue by initializing variable `j` with 1, compile the C program and execute it again.

Even after this fix there seems to be some problem in the `factorial.c` program, as it still gives wrong factorial value.

So, place the break point in 10th line, and continue as explained in the next section.

Step 6. Continue, stepping over and in - GDB commands

There are three kind of GDB operations you can choose when the program stops at a break point. They are continuing until the next break point, stepping in, or stepping over the next program lines.

- **c or continue:** Debugger will continue executing until the next break point.
- **n or next:** Debugger will execute the next line as single instruction.
- **s or step:** Same as next, but does not treats function as a single instruction, instead goes into the function and executes it line by line.

By continuing or stepping through you could have found that the issue is because we have not used the `<=` in the 'for loop' condition checking. So changing that from `<` to `<=` will solve the issue.

4.2.7 Debugging an Arduino Board Using GDB

External Debugger Connections

If you don't have a debugger built-in then you can use an External Debugger which supports SWD to attach to your board, as seen in the examples below.

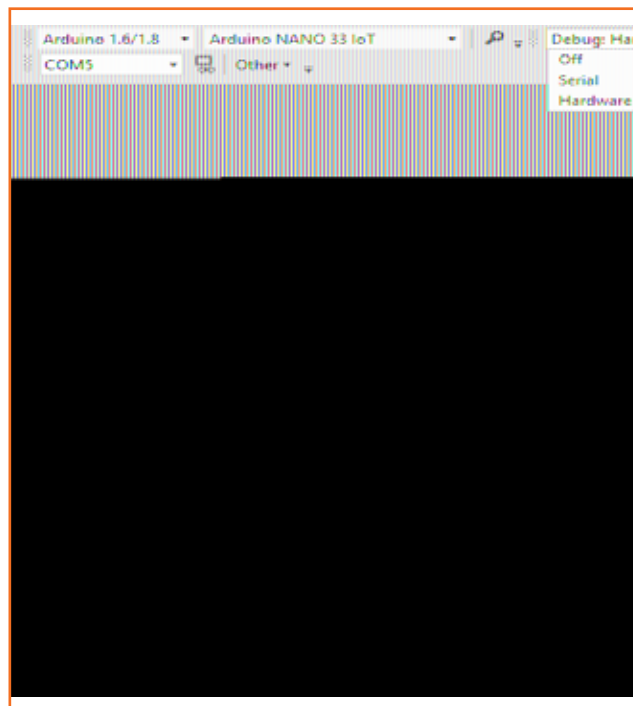


Fig 4.2.7: Connecting external debugger with IoT board

Arduino Nano 33 IOT SWD Pads will need to be connected for the debugger as shown below, and in this case the debugger needs a modification to fully support SWD reliably.

Software Setup

Ensure you have Visual Studio and the vMicro Extension Installed

Open your Sketch and select the Debug > Hardware, and the relevant Debugger you have available, whether it's built in, or External -

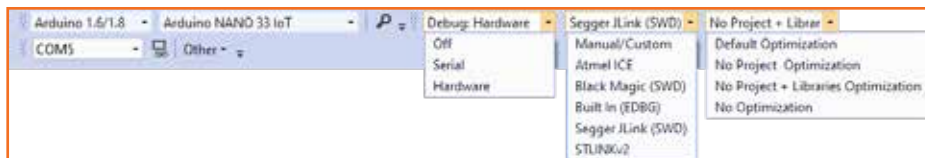


Fig 4.2.8: GDB software setup

Start Debugger

1. Ensure you have the Debug Configuration selected from the Configuration Manager Window
2. Add the first breakpoint in your code,
3. **To start the debugging process, you can either:**
 - **“Debug > Attach to Process”** button if your code has already been uploaded to the SAMD board
 - **“Debug > Start Debugging”** if your code has not been uploaded

Upload Note - Set the COM port to the Nano port to Upload, as programming is not available at present for this debugger.

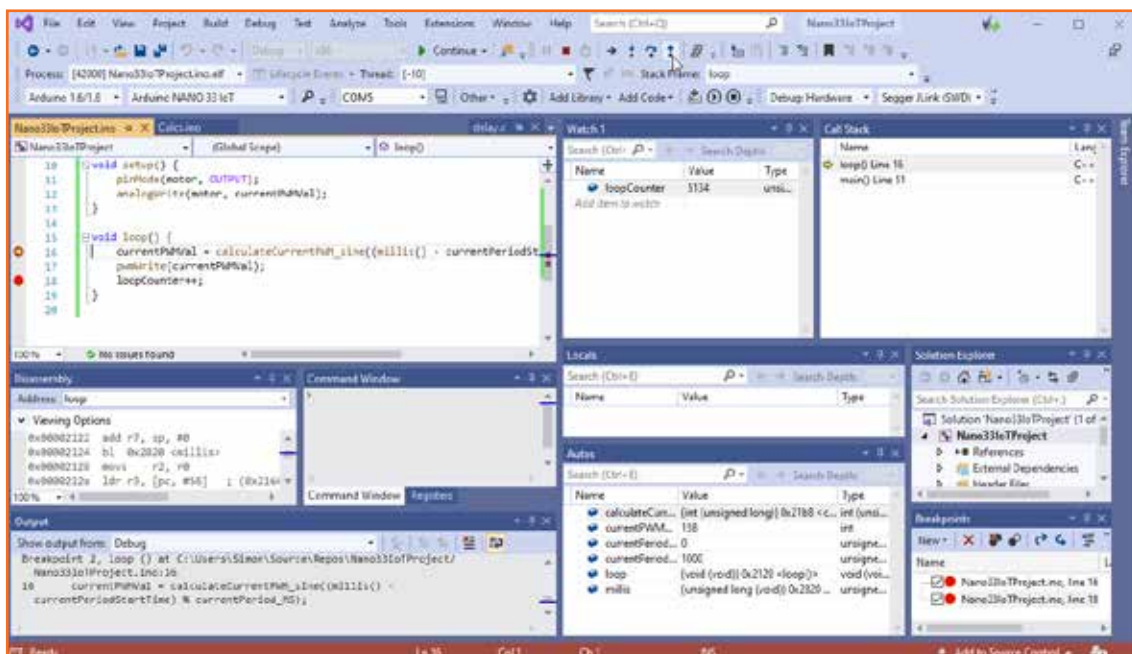


Fig 4.2.9: Starting debugger

Now the debugger is running, and further windows can be opened from the **“Debug > Windows”** menu once you have started debugging.

Activity

Use GDB to debug the following codes

```
#include <stdio.h>
/* This code contains some logical errors. Try using GDB to find out */
/* For two given 2 by 1 vectors 1 and 2, compute the inner product */
double inner(double *a, double *b, int n);
main()
{
    int i, j;
    double a[2][2], r;
    for(i=0; i<2; i++) {
        printf("Vector %d:\n", (i+1));
        printf("1st element:");
        scanf("%f", &a[i][1]);
        printf("2nd element:");
        scanf("%f", &a[i][2]);
    }
    r = inner(a[0], a[1], 2);
    printf("The inner product is %f\n", r);
}

double inner(double *a, double *b, int n) {
    int i, res;
    for(i=0; i<n; i++) {
        res = res + a[i]*b[i];
    }
    return res;
}
```

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=Dq8l1_-QgAc
Debugging using GDB



www.youtube.com/watch?v=6H_6lgVOuiA
Debugging Arduino Nano 33 IoT with GDB



5. Work effectively at the workplace

Unit 5.1 - Effective Communication and Coordination at Work

Unit 5.2 - Working Effectively and Maintaining Discipline at Work

Unit 5.3 - Maintaining Social Diversity at Work



Key Learning Outcomes



At the end of this unit, the participant will be able to:

1. Discuss about work ethics and workplace etiquettes
2. Describe ways to communicate work effectively at workplace
3. Discuss the importance of following organizational guidelines for dress code, time schedules, language usage and other behavioural aspect
4. Explain the importance of working as per the workflow of the organization to receive instructions and report problems
5. Explain the importance of conveying information/instructions as per defined protocols to the authorised persons/team members
6. Explain the common workplace guidelines and legal requirements on non-disclosure and confidentiality of business-sensitive information
7. Describe the process of reporting grievances and unethical conduct such as data breaches, sexual harassment at the workplace, etc.
8. Discuss ways of dealing with heightened emotions of self and others.
9. Explain the concept and importance of gender sensitivity and equality
10. Discuss ways to create sensitivity for different genders and Persons with Disabilities (PwD)

Unit 5.1: Effective Communication and Coordination at Work

Unit Objectives

At the end of this unit, participants will be able to:

1. Discuss about work ethics and workplace etiquettes
2. Describe ways to communicate work effectively at workplace

5.1.1 Importance of Work Ethics and Workplace Etiquette

Workplace ethics are a set of moral and legal guidelines that organizations follow. These guidelines influence the way customers and employees interact with an organization. Workplace ethics essentially guide how an organization serves its clients and treats its employees.

For example, if a company seeks to fulfil the promises it makes, it may develop processes and set up a robust support system to address this policy and build customer/client loyalty. To achieve this goal, the company may implement specific incentive programs for employees to encourage them to produce high-quality work and ensure the organization fulfils the promises it makes to its clients/ customers.

Many organizations, often the large ones, set detailed ethical codes to guide their operations and control how the organizational processes impact the stakeholders. These ethics usually help organizations maintain certain standards of responsibility, accountability, professionalism and among others, as they navigate through different challenges and day-to-day circumstances. By following these guidelines, organizations often experience several benefits that improve the lives of stakeholders, such as customers, employees, leaders, etc.

Examples of Common Workplace Ethics

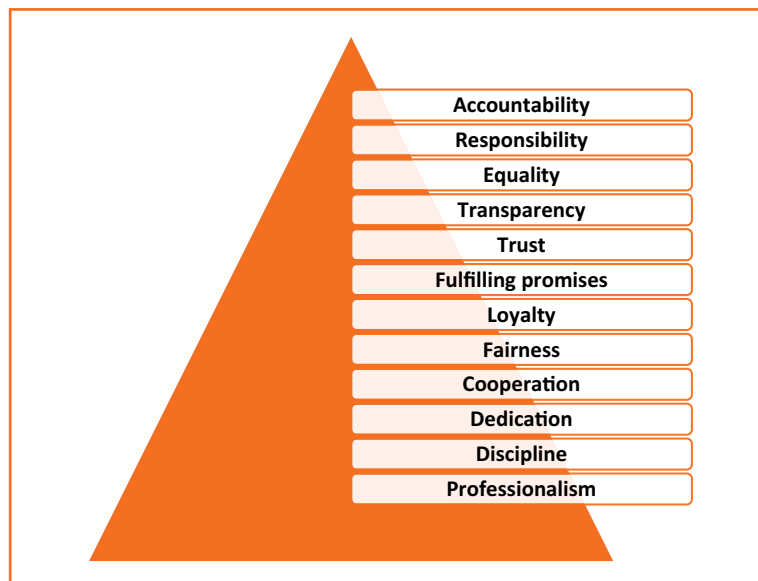


Fig 5.1.1: Examples of Common Workplace Ethics

Workplace ethics are essential for a successful organization with a satisfied and loyal team. High ethical standards help in ensuring all stakeholders, such as customers, investors, employees, and other individuals involved in the workplace operations, feel the organization is safeguarding their interests. By creating and implementing ethical guidelines, organizations can keep the best interests of their employees in mind while maintaining a positive influence on those they impact through their processes.

As a result, employees maintain the organization's best interests by being ethical in their daily work duties. For example, fairly-treated employees of an organization who understand the organization's commitments to environmental sustainability are usually less likely to behave in a manner that causes harm to the environment. Thus, they help maintain a positive public image of the organization. It means that workplace ethics help in maintaining reciprocal relationships that benefit organizations at large and the individuals associated with and influenced by the organizational policies.

Benefits of Workplace Ethics

There are various benefits of implementing workplace ethics. When organizations hold themselves to high ethical standards, leaders, stakeholders, and the general public can experience significant improvements. Following are some of the key benefits of employing ethics in the workplace -



Fig 5.1.2: Benefits of Workplace Ethics

5.1.2 Interpersonal Communication

Interpersonal communication is a process that involves sharing ideas and emotions with another person, both - verbally and non-verbally. It is essential to interact effectively with others in both personal and professional lives. In professional life or the workplace, strong interpersonal skills play a crucial role in achieving effective collaboration with colleagues.

Interpersonal Skills

Interpersonal skills, in other terms, are known as people skills, which are used to communicate and interact with others effectively. These are soft skills one uses to communicate with others and understand them. One uses these skills in daily life while interacting with people

Examples of Interpersonal Skills

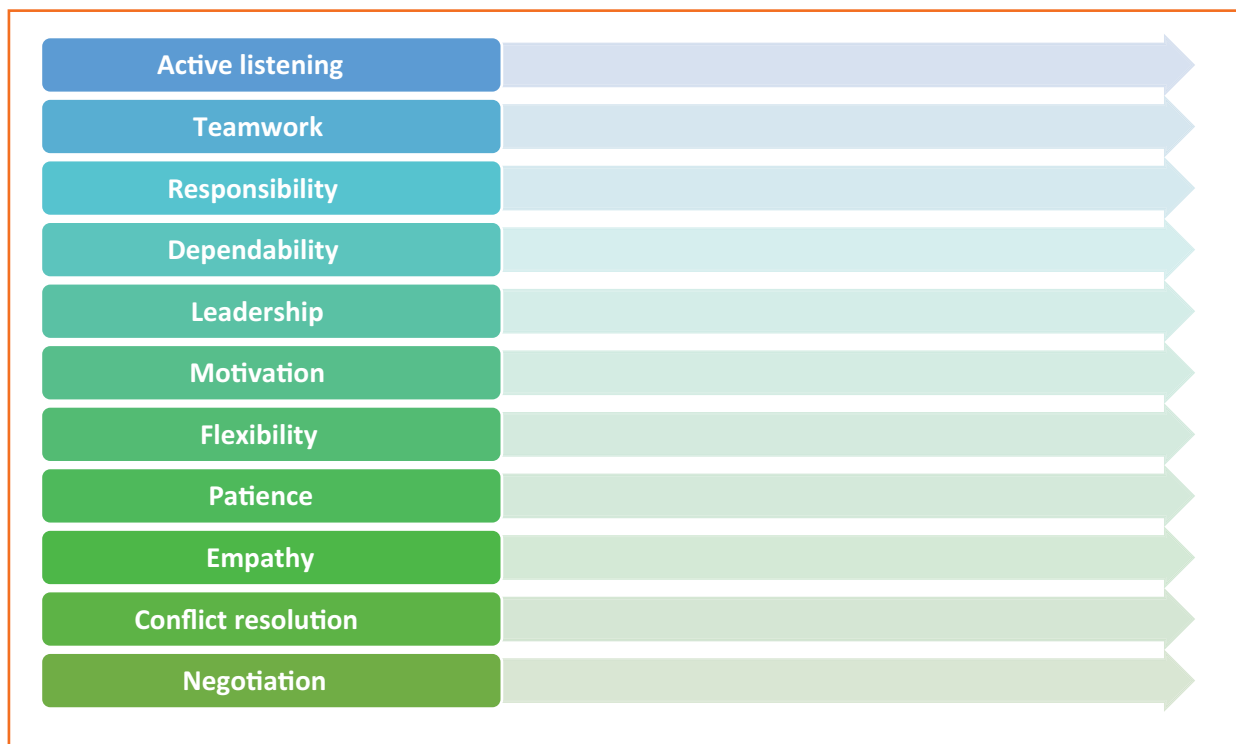


Fig 5.1.3: Examples of Interpersonal Skills

Numerous interpersonal skills involve communication. Communication can be verbal, such as persuasion or tone of voice - or non-verbal, such as listening and body language.

Importance of Interpersonal Skills

Interpersonal skills are essential for communicating and collaborating with groups and individuals in both personal and professional life. People with strong interpersonal skills often are able to build good relationships and also tend to work well with others. Most people often enjoy working with co-workers who have good interpersonal skills.

Among other benefits of good interpersonal skills is the ability to solve problems and make the best decisions. One can use the ability to understand others and good interpersonal communication skills to find the best solution or make the best decisions in the interest of everyone involved. Strong interpersonal skills help individuals work well in teams and collaborate effectively. Usually, people who possess good

interpersonal skills also tend to be good leaders, owing to their ability to communicate well with others and motivate the people around them.

Interpersonal communication is the key to working in a team environment and working collectively to achieve shared goals. Following are the interpersonal communication skills that vital for success at work -

Verbal Communication

The ability to speak clearly, appropriately and confidently can help one communicate effectively with others. It is vital to select the appropriate vocabulary and tone for the target audience.

For example: one should speak formally and professionally in the work environment, while informal language is acceptable in an intimate environment with close friends and family. Also, one should avoid using complex or technical language while communicating with an audience that may not be familiar with it. Using simple language in a courteous tone helps achieve better communication, irrespective of the audience.

Active Listening

Active listening is defined as the ability to pay complete or undivided attention to someone when they speak and understand what they are saying. It is important for effective communication because without understanding what the speaker is saying, it becomes difficult to carry forward a conversation. One should ensure to use appropriate verbal and non-verbal responses, e.g. eye contact, nodding, or smiling, to show interest in what the speaker says. Active listening is also about paying attention to the speaker's body language and visual cues. Asking and answering questions is one of the best ways to demonstrate an interest in conversing with the other person.

Active listening is critical for communicating effectively without ambiguity. It helps one understand the information or instructions being shared. It may also encourage co-workers to share their ideas, which ultimately helps achieve collaboration.

Body Language

One's expression, posture, and gestures are as important as verbal communication. One should practice open body language to encourage positivity and trust while communicating. Open body language includes - maintaining eye contact, nodding, smiling and being comfortable. On the other hand, one should avoid closed body language, e.g. crossed arms, shifting eyes and restless behaviour.

Empathy

Empathy is the ability to understand the emotions, ideas and needs of others from their point of view. Empathy is also known as emotional intelligence. Empathetic people are good at being aware of others' emotions and compassionate when communicating with them. Being empathetic in the workplace can be good to boost the morale of employees and improve productivity. By showing empathy, one can gain the trust and respect of others.

Conflict Resolution

One can use interpersonal communication skills to help resolve disagreements and conflicts in the workplace. This involves the application of negotiation and persuasion skills to resolve arguments

between conflicting parties. It is also important to evaluate and understand both sides of the argument by listening closely to everyone involved and finding an amicable solution acceptable to all.

Good conflict resolution skills can help one contribute to creating a collaborative and positive work environment. With the ability to resolve conflicts, one can earn the trust and respect of co-workers.

Teamwork

Employees who communicate and work well in a team often have better chances of achieving success and common goals. Being a team player can help one avoid conflicts and improve productivity. One can do this by offering to help co-workers when required and asking for their feedback and ideas. When team members give their opinions or advice, one should positively receive and react to the opinions/advice. One should be optimistic and encouraging when working in groups.

Improving Interpersonal Skills

One can develop interpersonal skills by practising good communication and setting goals for improvement. One should consider the following tips to improve their interpersonal skills -

- One should ask for feedback from co-workers, managers, family or friends to figure out what needs improvement concerning their interpersonal skills.
- One can identify the areas of interpersonal communication to strengthen by watching others.
- One can learn and improve interpersonal skills by observing co-workers, company leaders and professionals who possess good interpersonal skills. This includes watching and listening to them to note how they communicate and the body language used by them. It is vital to note their speed of speaking, tone of voice, and the way they engage with others. One should practice and apply such traits in their own interactions and relationships.
- One should learn to control their emotions. If stressed or upset, one should wait until being calm to have a conversation. One is more likely to communicate effectively and confidently when not under stress.
- One can reflect on their personal and professional conversations to identify the scope of improvement and learn how to handle conversations better or communicate more clearly. It helps to consider whether one could have reacted differently in a particular situation or used specific words or positive body language more effectively. It is also vital to note the successful and positive interactions to understand why they are successful.
- One should practice interpersonal skills by putting oneself in positions where one can build relationships and use interpersonal skills. For example, one can join groups that have organized meetings or social events. These could be industry-specific groups or groups with members who share an interest or hobby.
- Paying attention to family, friends and co-workers and making efforts to interact with them helps a lot. One should complement their family, friends and co-workers on their good ideas, hard work and achievements. Trying to understand someone's interests and showing interest in knowing them can help one build strong interpersonal skills. Offering to help someone, especially in difficult situations, helps build stronger and positive workplace relationships.

- One should avoid distractions, such as a mobile phone, while interacting with someone. Giving someone full attention while avoiding distractions helps achieve a clear exchange of ideas. By listening with focus, one can understand and respond effectively.
- One can attend appropriate courses on interpersonal skills or sign up for workshops at work to improve interpersonal skills. One can find many resources online also, such as online videos.
- For personal mentoring, one can approach a trusted family member, friend, co-worker, or current/former employer. A person one looks up to with respect and admires is often a good choice to be selected as a mentor. One can even hire a professional career or communication coach.
- Interpersonal communication skills often help one boost their morale, be more productive in the workplace, complete team projects smoothly and build positive and strong relationships with co-workers.

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=u16EPwFmdis
Effective communication skills

Unit 5.2: Working Effectively and Maintaining Discipline at Work

Unit Objectives

At the end of this unit, participants will be able to:

1. Discuss the importance of following organizational guidelines for dress code, time schedules, language usage and other behavioural aspect
2. Explain the importance of working as per the workflow of the organization to receive instructions and report problems
3. Explain the importance of conveying information/instructions as per defined protocols to the authorised persons/team members
4. Explain the common workplace guidelines and legal requirements on non-disclosure and confidentiality of business-sensitive information
5. Describe the process of reporting grievances and unethical conduct such as data breaches, sexual harassment at the workplace, etc.
6. Discuss ways of dealing with heightened emotions of self and others.

5.2.1 Discipline at Work

Discipline is essential for organizational success. It helps improve productivity, reduce conflict and prevent misconduct in the workplace. It is important to have rules concerning workplace discipline and ensure that all employees comply with them. In the absence of discipline, a workplace may experience conflicts, bullying, unethical behaviour and poor employee performance. An efficient workplace disciplinary process helps create transparency in the organization. Benefits of disciplinary standards:

All employees follow the same rules which helps establish uniformity and equality in the workplace

Managers and supervisors have defined guidelines on what action to take while initiating disciplinary action

With well defined and enforced disciplinary rules, an organization can avoid various safety, security, reputational risks

Fig 5.2.1: Benefits of Disciplinary Standards

Maintaining an organized and cohesive workforce requires maintaining discipline in both personal and professional behaviors. It is important to follow the appropriate measures to keep employees in line without affecting their morale.

Defining Discipline

The first and crucial step in maintaining workplace discipline is to define what is meant by discipline. It helps to evaluate common discipline problems and devise guidelines for handling them effectively. Among a number of areas, discipline usually covers -

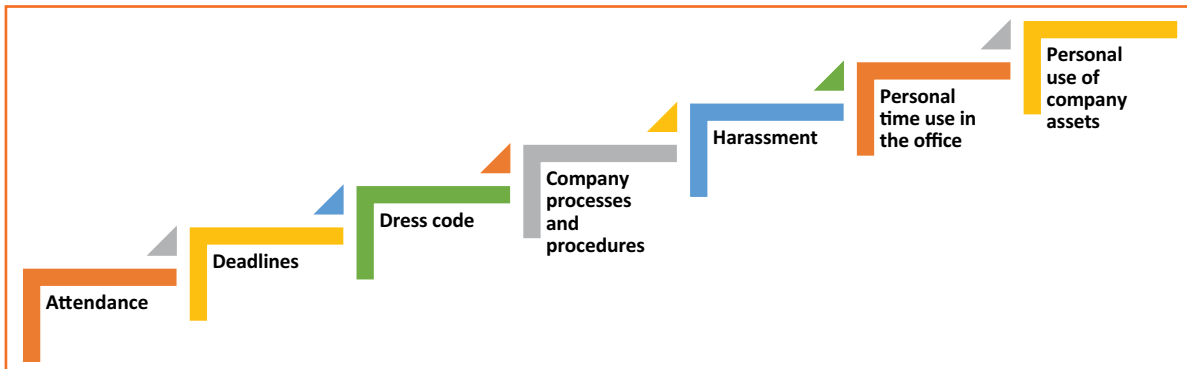


Fig 5.2.2: Examples of Workplace Discipline

According to demography and local issues, it may also include substance use and related issues. It is vital for a workplace to have an employee handbook or company policy guide, to serve as a rulebook for employees to follow. The employee handbook/ company policy guide should be reviewed and updated periodically according to any issues or areas, or concerns identified concerning workplace discipline. Such manuals should also cover all the laws and regulations governing workplace behaviour.

Defining and documenting workplace rules aids in their implementation, ensuring little or no ambiguity. All employees in a workplace should also have easy access to the workplace guidelines so that they can refer to them to get clarity whenever required. To maintain discipline at work, it is also critical to ensure uniform application of workplace guidelines to all employees without exception.

5.2.2 Employee Code of Conduct

The employee code of conduct manual serves as a guide for employees to inform them regarding the behavior expected from them at work. It helps create a good work environment with consistent behavior from employees. The manual should list examples of acceptable and not acceptable behaviors at work. The code of conduct should be discussed with employees so that they have the clarifications required.

For example, an organization may create guidelines concerning the conduct with clients to ensure no contact is made with them except for business purposes, also prescribing the use of appropriate means of communication.

Employees should have a clear understanding concerning their job responsibilities and the behaviors expected from them with all stakeholders, e.g. company personnel, clients and associated third parties. It is critical to have documented guidelines for employees to follow concerning all aspects of work. It should also document the disciplinary action to be followed in case of non-compliance, e.g. verbal and then written warning, temporary suspension or eventual termination of service in case of repeated non-compliance with the employee code of conduct. Employees should know what the company rules are

and what will happen if they break the rules. However, disciplinary action should be initiated only when reasonably required to avoid its misuse for employee harassment.

There should also be an effective mechanism for employees to raise their concerns/ grievances and have them addressed while maintaining privacy, as required, e.g. raising concerns regarding the behavior of a co-worker.

The employee code of conduct manual must be duly reviewed and approved by the concerned stakeholders, such as the Human Resources (HR) department and company executives.

5.2.3 Interpersonal Conflicts

Interpersonal conflict is any type of conflict between two or more people. These are found in both - personal and professional relationships - among friends, family, and co-workers. In the workplace, interpersonal conflict is often observed when a person or group of people interfere with another person's attempts at completing assignments and achieving goals. It is critical to resolve conflicts in the workplace to boost the morale of employees, repair working relationships among them, and improve customer satisfaction.

Reasons for Workplace Conflicts

Workplace conflicts are often observed when two or more people have different points of view. This can happen between managers, co-workers, or clients and customers. In general, interpersonal conflicts are caused by a lack of communication or unclear communication.

Some of the leading reasons for workplace conflicts are:

- Difference in values
- Personality clashes
- Poor communication

Example of poor communication: if a manager reassigns a task to another employee without communicating with the employee to whom it was originally assigned, interpersonal conflict can arise among them. This may potentially make the first employee, i.e. who was originally assigned the task, feel slighted and mistrusted by the manager. It may even cause animosity in the first employee toward the employee who has now been assigned the task.

Types of Interpersonal Conflict

Following are the four types of interpersonal conflicts -

1. Policy-related interpersonal conflict

When a conflict relates to a decision or situation that involves both parties, it can be called a policy-related interpersonal conflict. Example – two people or groups working on the same project, trying to adopt different approaches. To resolve policy-related interpersonal conflicts, the parties involved should try to look for a win-win situation or make a compromise. This is especially critical

to resolve trivial issues so that work is not affected and common goals are achieved.

2. Pseudo-conflicts

Pseudo-conflict arises when two people or groups want different things and cannot reach an agreement. Pseudo-conflicts usually involve trivial disagreements that tend to hide the root of the issue.

3. Ego-related interpersonal conflicts

In ego conflicts, losing the argument may hurt or damage a person's pride. Sometimes ego conflicts arise when a number of small conflicts pile up on being left unresolved. To resolve ego-related conflicts, it's best to find the root of the issue and work towards a resolution.

4. Value-related interpersonal conflicts

Sometimes conflicts may occur between people when they have different value systems. Such conflicts can be difficult to identify initially, making the people involved think the other party is being disagreeable or stubborn, wherein they just have different values. Some co-workers may highly value their personal/ family time after office that they may be unreachable to clients during non-office hours, while others may place a high value on client satisfaction and may still be available for clients during non-office hours. Conflict may arise among such people when they may be required to coordinate to help a client during after-office hours. Value-related interpersonal conflicts are often difficult to settle since neither party likes to compromise.

Resolving Interpersonal Conflicts

Conflicts are usually likely in the workplace; they can, however, be prevented. Often resolving interpersonal conflicts through open communication helps build a stronger relationship, paving the way for effective coordination and success. Some ways to resolve interpersonal conflict:

- **Communication:** A great way to resolve interpersonal conflicts is for the opposing parties to listen to one another's opinions and understand their viewpoints. Meeting in person and keeping the conversation goal-oriented is important. One can have effective communication by following some measures, e.g. staying on the topic, listening actively, being mindful of the body language, maintaining eye contact, etc.
- **Active Listening:** One should patiently listen to what the other person is saying without interrupting or talking over them. It helps one display empathy and get to the root of the issue. Asking questions to seek clarification when required helps in clear communication and conveys to the other person that one is listening to them. Practising active listening is a great way to improve one's communication skills.
- **Displaying Empathy:** Listening attentively and identifying the anxieties/ issues of co-workers is a great way to show empathy and concern. It is essential to understand their feelings and actions to encourage honesty and avoid future conflict.
- **Not Holding Grudges:** With different types of people and personalities in a workplace, it is common for co-workers to have conflicts. It is best to accept the difference in opinions and move on. Being forgiving and letting go of grudges allows one to focus on the positive side of things and

perform better at work.

Work-related interpersonal conflicts can be complicated because different people have different leadership styles, personality characteristics, job responsibilities and ways in which they interact. One should learn to look above interpersonal conflicts, resolving them to ensure work goals and environment are not affected.

5.2.4 Importance of Following Organizational Guidelines

Policies and procedures or organizational guidelines are essential for any organization. These provide a road map for the operations of the organization. These are also critical in ensuring compliance with the applicable laws and regulations by guiding the decision-making process and business operations.

Organizational guidelines help bring uniformity to the operations of an organization, which helps reduce the risk of unwanted and unexpected events. These determine how employees are supposed to behave at work, which ultimately helps the business achieve its objectives efficiently.

However, organizational guidelines are ineffective and fail to serve their purpose if they are not followed. Many people don't like the idea of following and abiding by specific guidelines. Such people should be made to understand the benefits of following the organizational guidelines. Some of the key benefits are given below:

With well-defined organizational guidelines in place, no individual can act arbitrarily, irrespective of their position in the organization. All individuals will know the pros and cons of taking certain actions and what to expect in case of unacceptable behaviour. Benefits of following organizational guidelines:

- **Consistent processes and structures:** Organization guidelines help maintain consistency in operations, avoiding any disorder. When all employees follow the organizational guidelines, an organization can run smoothly. These ensure that people in different job roles operate as they are supposed to, knowing what they are responsible for, what is expected of them, and what they can expect from their supervisors and co-workers. With clarity in mind, they can do their jobs with confidence and excellence. With every person working the way intended, it's easy to minimise errors.
- With all the staff following organizational guidelines, the organization has a better scope of using time and resources more effectively and efficiently. This allows the organization to grow and achieve its objectives.
- **Better quality service:** By following organizational guidelines, employees perform their duties correctly as per the defined job responsibilities. It helps enhance the quality of the organization's products and services, helping improve the organization's reputation. Working with a reputable organization, employees can take pride in their work and know they are contributing to the reputation.
- **A safer workplace:** When all employees follow organizational guidelines, it becomes easy to minimise workplace incidents and accidents. It reduces the liabilities associated with risks for

the organization and limits the interruptions in operations. Employees also feel comfortable and safe in the workplace, knowing their co-workers are ensuring safety at work by following the applicable guidelines.

Different organizations may have different guidelines on dress code, time schedules, language usage, etc. For example – certain organizations in a client-dealing business requiring employees to meet clients personally follow a strict dress code asking their employees to wear formal business attire. Similarly, organizations operating in specific regions may require their employees to use the dominant regional language of the particular region to build rapport with customers and serve them better. Certain organizations, such as banks, often give preference to candidates with knowledge of the regional language during hiring.

Working hours may also differ from one organization to another, with some requiring employees to work extra compared to others. One should follow the organizational guidelines concerning all the aspects of the employment to ensure a cohesive work environment.

5.2.5 Workflow

Workflow is the order of steps from the beginning to the end of a task or work process. In other words, it is the way a particular type of work is organised or the order of stages in a particular work process.

Workflows can help simplify and automate repeatable business tasks, helping improve efficiency and minimise the room for errors. With workflows in place, managers can make quick and smart decisions while employees can collaborate more productively.

Other than the order that workflows create in a business, these have several other benefits, such as -

- **Identifying Redundancies:** Mapping out work processes in a workflow allows one to get a clear, top-level view of a business. It allows one to identify and remove redundant or unproductive processes.

Workflow gives greater insights into business processes. Utilizing such useful insights, one can improve work processes and the bottom line of the business. In many businesses, there are many unnecessary and redundant tasks that take place daily. Once an organization has insight into its processes while preparing workflow, it can determine which activities are really necessary.

Identifying and eliminating redundant tasks creates value for a business. With redundant tasks and processes eliminated, an organization can focus on what's important to the business.

- **Increase in Accountability and Reduction in Micromanagement:** Micromanagement often causes problems in a business setting as most employees don't like being micromanaged, and even many managers don't like the practice. Micromanagement is often identified as one of the reasons why people quit their job.

However, the need for micromanagement can be minimized by clearly mapping out the workflow. This way, every individual in a team knows what tasks need to be completed and by when and

who is responsible for completing them. This makes employees more accountable also.

With clearly defined workflow processes, managers don't have to spend much time micromanaging their employees, who don't have to approach the manager to know what the further steps are. Following a workflow, employees know what is going on and what needs to be done. This, in turn, may help increase the job satisfaction of everyone involved while improving the relationships between management and employees.

- **Improved Communication:** Communication at work is critical because it affects all aspects of an organization. There are instances when the main conflict in an organization originates from miscommunication, e.g. the management and employees disagreeing on an aspect, despite pursuing the same objectives. Poor communication is a common workplace issue that is often not dealt with.

This highlights why workflow is important. Workplace communication dramatically can increase with the visibility of processes and accountability. It helps make the daily operations smoother overall.

- **Better Customer Service:** Customers or clients are central to a business. Therefore, it is imperative to find and improve ways to improve customer experience. Relying on outdated manual systems may cause customer requests or complaints to be overlooked, with dissatisfied customers taking their business elsewhere. However, following a well-researched and defined workflow can help improve the quality of customer service.

By automating workflows and processes, an organization can also reduce the likelihood of human error. This also helps improve the quality of products or services over time, resulting in a better customer experience.

5.2.6 Following Instructions and Reporting Problems

All organizations follow a hierarchy, with most employees reporting to a manager or supervisor. For organizational success, it is vital for employees to follow the instructions of their manager or supervisor. They should ensure they perform their duties as per the given instructions to help achieve the common objectives of the organization and deliver quality service or products. This consequently helps maintain the reputation of the organization.

It is also important to be vigilant and identify problems at work or with the organizational work processes. One should deal with the identified within their limits of authority and report out of authority problems to the manager/ supervisor or the concerned person for a prompt resolution to minimise the impact on customers/clients and business.

5.2.7 Information or Data Sharing

Information or data is critical to all organizations. Depending on the nature of its business, an organization may hold different types of data, e.g. personal data of customers or client data concerning their business operations and contacts. It is vital to effective measures for the appropriate handling of different types of data, ensuring its protection from unauthorized access and consequent misuse.

One should access certain data only if authorised to do so. The same is applicable when sharing data which must be shared only with the people authorised to receive it to use it for a specific purpose as per their job role and organizational guidelines. For example – one should be extra cautious while sharing business data with any third parties to ensure they get access only to the limited data they need as per any agreements with them. It is also critical to monitor how the recipient of the data uses it, which should strictly be as per the organizational guidelines. It is a best practice to share appropriate instructions with the recipient of data to ensure they are aware of the purpose with which data is being shared with them and how they are supposed to use and handle it. Any misuse of data must be identified and reported promptly to the appropriate person to minimise any damage arising out of data misuse.

These days most organizations require their employees and business partners or associated third parties to sign and accept the relevant agreement on the non-disclosure of business-sensitive information. In simple terms, business-sensitive information is confidential information. It is proprietary business information collected or created during the course of conducting business, including information about the business, e.g. proposed investments, intellectual property, trade secrets, or plans for a merger and information related to its clients. Business-sensitive information may sometimes also include information regarding a business's competitors in an industry.

The release of business-sensitive information to competitors or the general public poses a risk to a business. For example, information regarding plans for a merger could be harmful to a business if a competitor gets access to it.

5.2.8 Reporting Issues at Work

Most organizations have defined guidelines on appropriate reporting processes to be followed for reporting different types of issues. For example – one can report any grievances or dissatisfaction concerning co-workers to their manager/supervisor, e.g. data breaches or unethical conduct. If the concern is not addressed, then the employee should follow the organizational guidelines and hierarchy for the escalation of such issues that are not addressed appropriately.

For example: any concern related to sexual harassment at the workplace should be escalated to the concerned spokesperson, such as Human Resources (HR) representative, and if not satisfied with the

5.2.8 Reporting Issues at Work

Most organizations have defined guidelines on appropriate reporting processes to be followed for reporting different types of issues. For example – one can report any grievances or dissatisfaction concerning co-workers to their manager/supervisor, e.g. data breaches or unethical conduct. If the concern is not addressed, then the employee should follow the organizational guidelines and hierarchy for the escalation of such issues that are not addressed appropriately.

For example: any concern related to sexual harassment at the workplace should be escalated to the concerned spokesperson, such as Human Resources (HR) representative, and if not satisfied with the the senior management for their consideration and prompt action.

5.2.9 Dealing with Heightened Emotions

Humans are emotional beings. There may be occasions when one is overwhelmed by emotions and is unable to suppress them. However, there may be situations when one must manage emotions well, particularly at work.

Stress in one's personal and professional life may often cause emotional outbursts at work. Managing one's emotions well, particularly the negative ones, is often seen as a measure of one's professionalism. Anger, dislike, frustration, worry, and unhappiness are the most common negative emotions experienced at work.

Ways to manage negative emotions at work:

- **Compartmentalisation:** It's about not confining emotions to different aspects of one's life. For example, not letting negative emotions from personal life affect work-life and vice versa. One should try to leave personal matters and issues at home. One should train their mind to let go of personal matters before reaching work. Similarly, one can compartmentalise work-related stresses so that negative emotions from work don't affect one's personal life.
- **Deep breathing and relaxation:** Deep breathing helps with anxiety, worry, frustration and anger. One should take deep breaths, slowly count to ten - inhaling and exhaling until one calms down. One can also take a walk to calm down or listen to relaxing music. Talking to someone and sharing concerns also helps one calm down.
- **The 10-second rule:** This is particularly helpful in controlling anger and frustration. When one feels their temper rising, they should count to 10 to calm down and recompose. If possible, one should move away to allow temper to come down.
- **Clarify:** It is always good to clarify before reacting, as it may be a simple case of misunderstanding or miscommunication.
- **Physical activity:** Instead of losing temper, one should plan to exercise, such as running or going to the gym, to let the anger out. Exercise is also a great way to enhance mood and release any physical tension in the body.
- **Practising restraint:** One should avoid replying or making a decision when angry, not allowing anger or unhappiness to cloud one's judgement. It may be best to pause any communication while one is angry, e.g. not communicating over email when angry or upset.

- **Knowing one's triggers:** It helps when one is able to recognise what upsets or angers them. This way, one can prepare to remain calm and plan their reaction should a situation occur. One may even be able to anticipate the other party's reaction.
- **Be respectful:** One should treat their colleagues the same way one would like to be treated. If the other person is rude, one need not reciprocate. It is possible to stay gracious, firm and assertive without being aggressive. Sometimes, rude people back away when they don't get a reaction from the person they are arguing with.
- **Apologise for any emotional outburst:** Sometimes, one can get overwhelmed by emotions, reacting with an emotional outburst. In such a case, one should accept responsibility and apologise immediately to the affected persons without being defensive.
- **Doing away with negative emotions:** It is recommended to let go of anger, frustration and unhappiness at the end of every workday. Harboring negative emotions affects one emotionally, affecting their job performance also. Engaging in enjoyable activities after work is a good stress reliever.

Notes



Unit 5.3 : Maintaining Social Diversity at Work

Unit Objectives



At the end of this unit, participants will be able to:

1. Explain the concept and importance of gender sensitivity and equality
2. Discuss ways to create sensitivity for different genders and Persons with Disabilities (PwD)

5.3.1 Gender Sensitivity

Gender sensitivity is the act of being sensitive towards people and their thoughts regarding gender. It ensures that people know the accurate meaning of gender equality, and one's gender should not be given priority over their capabilities.



Fig 5.3.1. Gender Equality

Women are an important source of labour in many sectors, yet they have limited access to resources and benefits. Women should receive the same benefits and access to resources as men. A business can improve its productivity and quality of work by providing better support and opportunities to women.

Important Terms

- Gender Sensitivity-Gender sensitivity is the act of being sensitive to the ways people think about gender.
- **Gender Equality:** It means persons of any gender enjoy equal opportunities, responsibilities, and rights in all areas of life.
- **Gender Discrimination:** It means treating an individual unequally or disadvantageously based on their gender, e.g. paying different wages to men and women for similar or equal job positions.

Strategies for Enhancing Gender Equity

To enhance gender equity, one should -

- Follow gender-neutral practices at all levels at work.

- Participate together in decision-making.
- Help in promoting women's participation in different forums.
- Assist women in getting exposure to relevant skills and practices.
- Assist women in capacity building by mentoring, coaching or motivating them, as appropriate.
- Assist in the formation and operation of women support groups.
- Assist in the implementation of women-centric programmes.
- Combine technical training with reproductive health and nutrition for coffee farming households.
- Assist in making a work environment that is healthy, safe, and free from discrimination.

Bridging Gender Differences

Men and women react and communicate very differently. Thus, there are some work differences as both genders have their style and method of handling a situation.

Although, understanding and maturity vary from person to person, even between these genders, based on their knowledge, education, experience, culture, age, and upbringing, as well as how one's brain functions over a thought or problem.

In order to bridge the gap, one should:

- Not categorize all men and women in one way.
- Be aware of the verbal and non-verbal styles of communication of every gender to avoid any miscommunication and work better.
- Be aware of partial behaviour and avoid it.
- Encourage co-workers of different genders to make room by providing space to others.

Ways to reduce Gender Discrimination

- Effective steps against sexual harassment by the concerned authorities and general public.
- Gender stereotypes are how society expects people to act based on their gender. This can only be reduced by adopting appropriate behaviour and the right attitude.
- Objectification of females must be abolished.
- Ways to Promote Gender Sensitivity in the Workplace
- Practices that promote gender diversity should be adopted and promoted.
- All genders should receive equal responsibilities, rights, and privileges.
- All genders should have equal pay for similar or the same job roles/ positions.
- Strict and effective workplace harassment policies should be developed and implemented.
- An open-minded and stress-free work environment should be available to all the employees, irrespective of their gender.
- Women should be encouraged to go ahead in every field of work and assume leadership roles.
- Follow appropriate measures for women's empowerment.
- Men should be taught to be sensitive to women and mindful of their rights.

5.3.2 PwD Sensitivity

Some individuals are born with a disability, while others may become disabled due to an accident, illness or as they get old. People with Disabilities (PwD) may have one or more areas in which their functioning is affected. A disability can affect hearing, sight, communication, breathing, understanding, mobility, balance, and concentration or may include the loss of a limb. A disability may contribute to how a person feels and affect their mental health.

Important Terms

- **Persons with Disabilities (PwD):** Persons with Disabilities means a person suffering from not less than 40% of any disability as certified by a medical authority.
- **Types of Disability:**
 - i. **Blindness:** Visually impaired
 - ii. Low Vision
 - iii. Leprosy Cured
 - iv. Hearing impairment
 - v. Locomotor disability
 - vi. Mental retardation
 - vii. Mental illness

PwD Sensitivity

PwD sensitivity promotes empathy, etiquette and equal participation of individuals and organizations while working with individuals with a disability, e.g. sensory, physical or intellectual.

Ways to be PwD Sensitive

To be sensitive to PwD, one should -

- Be respectful to all Persons with Disabilities (PwD) and communicate in a way that reflects PwD sensitivity.
- Always be supportive and kind towards a PwD with their daily chores.
- Be ready to assist a PwD to help them avail of any benefit/ livelihood opportunity/ training or any kind that helps them grow.
- Encourage and try to make things easier and accessible to PwD so that they can work without or with minimum help.
- Protest where feasible and report any wrong act/behaviour against any PwD to the appropriate authority.
- Learn and follow the laws, acts, and policies relevant to PwD.

Appropriate Verbal Communication

As part of appropriate verbal communication with all genders and PwD, one should -

- Talk to all genders and PwD respectfully, maintaining a normal tone of voice with appropriate politeness. It is important to ensure one's tone of voice does not have hints of sarcasm, anger, or unwelcome affection.

- Avoid being too self-conscious concerning the words to use while also ensuring not to use words that imply one's superiority over the other.
- Make no difference between a PwD and their caretaker. Treat PwD like adults and talk to them directly.
- Ask a PwD if they need any assistance instead of assuming they need it and offering assistance spontaneously.

Appropriate Non-verbal Communication

Non-verbal communication is essentially the way someone communicates through their body language. These include -

- **Facial expressions:** The human face is quite expressive, capable of conveying many emotions without using words. Facial expressions must usually be maintained neutral and should change according to the situation, e.g. smile as a gesture of greeting.
- **Body posture and movement:** One should be mindful of how to sit, stand, walk, or hold their head. For example - one should sit and walk straight in a composed manner. The way one moves and carries self, communicates a lot to others. This type of non-verbal communication includes one's posture, bearing, stance, and subtle movements.
- **Gestures:** One should be very careful with their gestures, e.g. waving, pointing, beckoning, or using one's hands while speaking. One should use appropriate and positive gestures to maintain respect for the other person while being aware that a gesture may have different meanings in different cultures.
- **Eye contact:** Eye contact is particularly significant in non-verbal communication. The way someone looks at someone else may communicate many things, such as interest, hostility, affection or attraction. Eye contact is vital for maintaining the flow of conversation and for understanding the other person's interest and response. One should maintain appropriate eye contact, ensuring not to stare or look over the shoulders. To maintain respect, one should sit or stand at the other person's eye level to make eye contact.
- **Touch:** Touch is a very sensitive type of non-verbal communication. Examples are - handshakes, hugs, pat on the back or head, gripping the arm, etc. A firm handshake indicates interest, while a weak handshake indicates the opposite. One should be extra cautious not to touch others inappropriately and avoid touching them inadvertently by maintaining a safe distance.

Rights of PwD

PwD have the right to respect and human dignity. Irrespective of the nature and seriousness of their disabilities, PwD have the same fundamental rights as others, such as -

- Disabled persons have the same civil and political rights as other people
- Disabled persons are entitled to the measures designed to enable them to become as self-dependent as possible
- Disabled persons have the right to economic and social security
- Disabled persons have the right to live with their families or foster parents and participate in all

social and creative activities.

- Disabled persons are protected against all exploitation and treatment of discriminatory and abusive nature.

Making Workplace PwD Friendly

- One should not make PwD feel uncomfortable by giving too little or too much attention
- One should use a normal tone while communicating with a PwD and treat them as all others keeping in mind their limitations and type of disability
- Any help should be provided only when asked for by a PwD
- One should help in ensuring the health and well-being of PwD.

Expected Employer Behaviour

Some of the common behavioural traits that employees expect from their employers are -

- **Cooperation:** No work is successful without cooperation from the employer's side. Cooperation helps to understand the job role better and complete it within the given timeline.
- **Polite language:** Polite language is always welcomed at work. This is a basic aspect that everybody expects.
- **Positive Attitude:** Employers with a positive attitude can supervise the work of the employees and act as a helping hand to accomplish the given task. A person with a positive attitude looks at the best qualities in others and helps them gain success.
- **Unbiased behaviour:** Employers should always remain fair towards all their employees. One should not adopt practices to favour one employee while neglecting or ignoring the other. This might create animosity among co-workers.
- **Decent behaviour:** The employer should never improperly present oneself before the employee. One should always respect each other's presence and behave accordingly. The employer should not speak or act in a manner that may make the employee feel uneasy, insulted, and insecure.

Notes



A large rectangular area enclosed by an orange border, containing 25 horizontal lines for writing notes.



6. Basic Health and Safety Practices

Unit 6.1 - Workplace Hazards

Unit 6.2 – Fire Safety

Unit 6.3 – First Aid

Unit 6.4 – Waste Management



Key Learning Outcomes



At the end of this unit, the participant will be able to:

1. Discuss job-site hazards, risks and accidents
2. Explain the organizational safety procedures for maintaining electrical safety, handling tools and hazardous materials
3. Describe how to interpret warning signs while accessing sensitive work areas
4. Explain the importance of good housekeeping
5. Describe the importance of maintaining appropriate postures while lifting heavy objects
6. List the types of fire and fire extinguishers
7. Describe the concept of waste management and methods of disposing of hazardous waste
8. List the common sources of pollution and ways to minimize them
9. Elaborate on electronic waste disposal procedures
10. Explain how to administer appropriate first aid to victims in case of bleeding, burns, choking, electric shock, poisoning and also administer first aid to victims in case of a heart attack or cardiac arrest due to electric shock

Unit 6.1: Workplace Hazards

Unit Objectives



At the end of this unit, participants will be able to:

1. Discuss job-site hazards, risks and accidents
2. Explain the organizational safety procedures for maintaining electrical safety, handling tools and hazardous materials
3. Describe how to interpret warning signs while accessing sensitive work areas
4. Explain the importance of good housekeeping
5. Describe the importance of maintaining appropriate postures while lifting heavy objects
6. Explain safe handling of tools and Personal Protective Equipment to be used.

6.1.1. Workplace Safety

Workplace safety is important to be established for creating a safe and secure working for the workers. The workplace has to be administered as per the rules of the Occupational Safety and Health Administration (OSHA). It refers to monitoring the working environment and all hazardous factors that impact employees' safety, health, and well-being. It is important to provide a safe working environment to the employees to increase their productivity, wellness, skills, etc.

The benefits of workplace safety are:

- Employee retention increases if they are provided with a safe working environment.
- Failure to follow OSHA's laws and guidelines can result in significant legal and financial consequences.
- A safe environment enables employees to stay invested in their work and increases productivity.
- Employer branding and company reputation can both benefit from a safe working environment.

6.1.2. Workplace Hazards

A workplace is a situation that has the potential to cause harm or injury to the workers and damage the tools or property of the workplace. Hazards exist in every workplace and can come from a variety of sources. Finding and removing them is an important component of making a safe workplace.

Common Workplace Hazards

The common workplace hazards are:

- **Biological:** The threats caused by biological agents like viruses, bacteria, animals, plants, insects and also humans, are known as biological hazards.

- **Chemical:** Chemical hazard is the hazard of inhaling various chemicals, liquids and solvents. Skin irritation, respiratory system irritation, blindness, corrosion, and explosions are all possible health and physical consequences of these dangers.
- **Mechanical:** Mechanical Hazards comprise the injuries that can be caused by the moving parts of machinery, plant or equipment.
- **Psychological:** Psychological hazards are occupational hazards caused by stress, harassment, and violence.
- **Physical:** The threats that can cause physical damage to people is called physical hazard. These include unsafe conditions that can cause injury, illness and death.
- **Ergonomic:** Ergonomic Hazards are the hazards of the workplace caused due to awkward posture, forceful motion, stationary position, direct pressure, vibration, extreme temperature, noise, work stress, etc.

Workplace Hazards Analysis

A workplace hazard analysis is a method of identifying risks before they occur by focusing on occupational tasks. It focuses on the worker's relationship with the task, the tools, and the work environment. After identifying the hazards of the workplace, organisations shall try to eliminate or minimize them to an acceptable level of risk.

Control Measures of Workplace Hazards

Control measures are actions that can be taken to reduce the risk of being exposed to the hazard. Elimination, Substitution, Engineering Controls, Administrative Controls, and Personal Protective Equipment are the five general categories of control measures.

- **Elimination:** The most successful control technique is to eliminate a specific hazard or hazardous work procedure or prevent it from entering the workplace.
- **Substitution:** Substitution is the process of replacing something harmful with something less hazardous. While substituting the hazard may not eliminate all of the risks associated with the process or activity, it will reduce the overall harm or health impacts.
- **Engineering Controls:** Engineered controls protect workers by eliminating hazardous situations or creating a barrier between the worker and the hazard, or removing the hazard from the person.
- **Administrative Controls:** To reduce exposure to hazards, administrative controls limit the length of time spent working on a hazardous task that might be used in combination with other measures of control.
- **Personal Protective Equipment:** Personal protective equipment protects users from health and safety hazards at work. It includes items like safety helmets, gloves, eye protection, etc.

6.1.3. Risk for a Drone Technician

A drone technician may require to repair the propeller, motor and its mount, battery, mainboards, processor, booms, avionics, camera, sensors, chassis, wiring and landing gear. A technician may face some risks while repairing the drones' equipment.

- The technician is susceptible to being physically harmed by propellers.
- Direct contact with exposed electrical circuits can injure the person.
- If the skin gets in touch with the heat generated from electric arcs, it burns the internal tissues.
- Major electrical injuries can occur due to poorly installed electrical equipment, faulty wiring, overloaded or overheated outlets, use of extension cables, incorrect use of replacement fuses, use of equipment with wet hands, etc.

6.1.4. Workplace Warning Signs

A Hazard sign is defined as 'information or instruction about health and safety at work on a signboard, an illuminated sign or sound signal, a verbal communication or hand signal.'

There are four different types of safety signs:

- Prohibition / Danger Alarm Signs
 - Mandatory Signs
 - Warning Signs
 - And Emergency
1. **Prohibition Signs:** A "prohibition sign" is a safety sign that prohibits behaviour that is likely to endanger one's health or safety. The colour red is necessary for these health and safety signs. Only what or who is forbidden should be displayed on a restriction sign.



Fig 6.1.1: Prohibition Warning Signs

2. **Mandatory Signs:** Mandatory signs give clear directions that must be followed. The icons are white circles that have been reversed out of a blue circle. On a white background, the text is black.



Fig 6.1.2: Mandatory Signs

3. **Warning Signs:** Warning signs are the safety information communication signs. They are shown as a 'yellow colour triangle'.



Fig 6.1.3: Mandatory Signs

4. **Emergency Signs:** The location or routes to emergency facilities are indicated by emergency signs. These signs have a green backdrop with a white emblem or writing. These signs convey basic information and frequently refer to housekeeping, company procedures, or logistics.



Fig 6.1.4: Emergency Signs

6.1.5. Cleanliness in the Workplace

Workplace cleanliness maintenance creates a healthy, efficient and productive environment for the employees. Cleanliness at the workplace is hindered by some elements like cluttered desks, leftover food, waste paper, etc. A tidy workplace is said to improve employee professionalism and enthusiasm while also encouraging a healthy working environment.

Benefits of cleanliness in the workplace:

1. **Productivity:** Cleanliness in the workplace can bring a sense of belonging to the employees, also motivating and boosting the morale of the employees. This results in increasing their productivity.
2. **Employee Well-being:** Employee well-being can be improved by providing a clean work environment. Employees use fewer sick days in a workplace where litter and waste are properly disposed of, and surfaces are cleaned regularly, resulting in increased overall productivity.
3. **Positive Impression:** Cleanliness and orderliness in the workplace provide a positive impression on both employees and visitors.
4. **Cost saving:** By maintaining acceptable levels of cleanliness in the workplace, businesses can save money on cleaning bills and renovations, which may become necessary if the premises are not properly kept.

Reasons for cleaning the workplace

- Cleaning of dry floors, mostly to prevent workplace slips and falls.

- Disinfectants stop bacteria in their tracks, preventing the spread of infections and illness.
- Proper air filtration decreases hazardous substance exposures such as dust and fumes.
- Light fixture cleaning improves lighting efficiency.
- Using environmentally friendly cleaning chemicals that are safer for both personnel and the environment.
- Work environments are kept clean by properly disposing of garbage and recyclable items.

6.1.6. Lifting and Handling of Heavy Loads

Musculoskeletal Injuries (MSIs), such as sprains and strains, can occur while lifting, handling, or carrying objects at work. When bending, twisting, uncomfortable postures and lifting heavy objects are involved, the risk of injury increases. Ergonomic controls can help to lower the risk of injury and potentially prevent it.

Types of injuries caused while lifting heavy objects:

- Cuts and abrasions are caused by rough surfaces.
- Crushing of feet or hands.
- Strain to muscles and joints



Fig 6.1.5: Lifting loads technique

Preparing to lift

A load that appears light enough to bear at first will grow increasingly heavier as one carries it further. The person carrying the weight should be able to see over or around it at all times.

The amount of weight a person can lift, depends on their age, physique, and health.



It also depends on whether or not the person is used to lifting and moving hefty objects.

Common causes of back injuries

The most common causes of back injuries are -

1. **Inadequate training:** The individual raising the load receives no sufficient training or guidance.
2. **Lack of awareness of technique:** The most common cause of back pain is incorrect twisting and posture, which causes back strain.
3. **Load size:** The load size to consider before lifting. If the burden is too much for one's capacity or handling, their back may be strained and damaged.
4. **Physical strength:** Depending on their muscle power, various persons have varied physical strengths. One must be aware of their limitations.
5. **Teamwork:** The operation of a workplace is all about working together. When opposed to a single person lifting a load, two people can lift it more easily and without difficulty. If one of two people isn't lifting it properly, the other or both of them will suffer back injuries as a result of the extra strain.

Techniques for lifting heavy objects

Technique	Demonstration
<ol style="list-style-type: none"> 1. Ensure one has a wide base of support before lifting the heavy object. Ensure one's feet are shoulder-width apart, and one foot is slightly ahead of the other at all times. This will help one maintain a good balance during the lifting of heavy objects. This is known as the Karate Stance. 	
<ol style="list-style-type: none"> 2. Squat down as near to the object as possible when one is ready to lift it, bending at the hips and knees with the buttocks out. If the object is really heavy, one may wish to place one leg on the floor and the other bent at a straight angle in front of them. 	

3. Maintain proper posture as one begin to lift upward. To do so, one should keep their back straight, chest out, and shoulders back while gazing straight ahead.



4. By straightening one's hips and knees, slowly elevate the thing (not the back). As one rises, they should extend their legs and exhale. Lift the heavy object without twisting the body or bending forward.



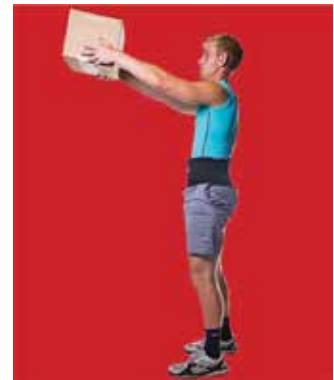
5. Do not lift bending forward.



6. Hold the load close to the body.



7. Never lift heavy objects above the shoulder



8. Use the feet (not the body) to change direction, taking slow, small steps.



9. Set down the heavy object carefully, squatting with the knees and hips only.



Table 6.1.1. Techniques for lifting heavy objects

6.1.7. Safe Handling of Tools


Workers should be trained on how to use tools safely. When tools are misplaced or handled incorrectly by workers, they can be dangerous. The following are some suggestions from the National Safety Council for safe tool handling when they are not in use -

- Never carry tools up or down a ladder in a way that makes it difficult to grip them. Instead of being carried by the worker, tools should be lifted up and down using a bucket or strong bag.
- Tools should never be tossed but should be properly passed from one employee to the next. Pointed tools should be passed with the handles facing the receiver or in their carrier.
- When turning and moving around the workplace, workers carrying large tools or equipment on their shoulders should pay particular attention to clearances.
- Pointed tools such as chisels and screwdrivers should never be kept in a worker's pocket. They can be carried in a toolbox, pointing down in a tool belt or pocket tool bag, or in hand with the tip always held away from the body.
- Tools should always be stored while not in use. People below are put in danger when tools are left sitting around on an elevated structure, such as a scaffold. In situations when there is a lot of vibration, this risk increases.

6.1.8. Personal Protective Equipment

Personal protective equipment, or “PPE,” is equipment worn to reduce exposure to risks that might result in significant occupational injuries or illnesses. Chemical, radiological, physical, electrical, mechanical, and other job dangers may cause these injuries and diseases.

PPE used for protection from the following injuries are:

Injury Protection	Protection	PPE
Head Injury Protection	Falling or flying objects, stationary objects, or contact with electrical wires can cause impact, penetration, and electrical injuries. Hard hats can protect one's head from these injuries. A common electrician's hard hat is shown in the figure below. This hard hat is made of non conductive plastic and comes with a set of safety goggles.	

Injury Protection	Protection	PPE
Foot and Leg Injury Protection	In addition to foot protection and safety shoes, leggings (e.g., leather) can guard against risks such as falling or rolling objects, sharp objects, wet and slippery surfaces, molten metals, hot surfaces, and electrical hazards.	
Eye and Face Injury Protection	Spectacles, goggles, special helmets or shields, and spectacles with side shields and face shields can protect against the hazards of flying fragments, large chips, hot sparks, radiation, and splashes from molten metals. They also offer protection from particles, sand, dirt, mists, dust, and glare.	
Protection against Hearing Loss	Hearing protection can be obtained by wearing earplugs or earmuffs. High noise levels can result in permanent hearing loss or damage, as well as physical and mental stress. Self-forming earplugs composed of foam, waxed cotton, or fibreglass wool usually fit well. Workers should be fitted for moulded or prefabricated earplugs by a specialist.	



Injury Protection	Protection	PPE
Hand Injury Protection	Hand protection will aid workers who are exposed to dangerous substances by skin absorption, serious wounds, or thermal burns. Gloves are a frequent protective clothing item. When working on electrified circuits, electricians frequently use leather gloves with rubber inserts. When stripping cable with a sharp blade, Kevlar gloves are used to prevent cuts.	
Whole Body Protection	Workers must protect their entire bodies from risks such as heat and radiation. Rubber, leather, synthetics, and plastic are among the materials used in whole-body PPE, in addition to fire-retardant wool and cotton. Maintenance staff who operate with high-power sources such as transformer installations and motor-control centres are frequently obliged to wear fire-resistant clothes.	

Table 6.1.2. Personal protective equipment

Unit 6.2: Fire Safety

Unit Objectives



At the end of this unit, participants will be able to:

1. List the types of fire and fire extinguishers

6.2.1. Fire Safety

Fire safety is a set of actions aimed at reducing the amount of damage caused by fire. Fire safety procedures include both those that are used to prevent an uncontrolled fire from starting and those that are used to minimise the spread and impact of a fire after it has started. Developing and implementing fire safety measures in the workplace is not only mandated by law but is also essential for the protection of everyone who may be present in the building during a fire emergency.

The basic Fire Safety Responsibilities are:

- To identify risks on the premises, a fire risk assessment must be carried out.
- Ascertain that fire safety measures are properly installed.
- Prepare for unexpected events.
- Fire safety instructions and training should be provided to the employees.

6.2.2. Respond to a Workplace Fire

- Workplace fire drills should be conducted on a regular basis.
- If one has a manual alarm, they should raise it.
- Close the doors and leave the fire-stricken area as soon as possible. Ensure that the evacuation is quick and painless.
- Turn off dangerous machines and don't stop to get personal items.
- Assemble at a central location. Ascertain that the assembly point is easily accessible to the employees.
- If one's clothing catches fire, one shouldn't rush about it. They should stop and descend on the ground and roll to smother the flames if their clothes catch fire.

6.2.3. Fire Extinguisher

Fire extinguishers are portable devices used to put out small flames or minimise their damage until fire-fighters arrive. These are maintained on hand in locations such as fire stations, buildings, workplaces, public transit, and so on. The types and quantity of extinguishers that are legally necessary for a given region are determined by the applicable safety standards.

Types of fire extinguishers are:

There are five main types of fire extinguishers -

1. **Water:** Water fire extinguishers are one of the most common commercial and residential fire extinguishers on the market. They're meant to be used on class-A flames.



2. **Powder:** The L2 powder fire extinguisher is the most commonly recommended fire extinguisher in the Class D Specialist Powder category, and is designed to put out burning lithium metal fires.



3. **Foam:** Foam extinguishers are identified by a cream rectangle with the word "foam" printed on it. They're mostly water-based, but they also contain a foaming component that provides a quick knock-down and blanketing effect on flames. It suffocates the flames and seals the vapours, preventing re-ignition.



4. **Carbon Dioxide (CO2):** Class B and electrical fires are extinguished with carbon dioxide extinguishers, which suffocate the flames by removing oxygen from the air. They are particularly beneficial for workplaces and workshops where electrical fires may occur since, unlike conventional extinguishers, they do not leave any toxins behind and hence minimise equipment damage.



5. **Wet Chemical:** Wet chemical extinguishers are designed to put out fires that are classified as class F. They are successful because they can put out extremely high-temperature fires, such as those caused by cooking oils and fats.



Table 6.2.1 Fire Extinguishers

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=DaYwcH1GMEg
Workplace emergency procedures

Unit 6.3: First Aid

Unit Objectives

At the end of this unit, participants will be able to:

1. Explain how to administer appropriate first aid to victims in case of bleeding, burns, choking, electric shock, poisoning
2. Explain how to administer first aid to victims in case of a heart attack or cardiac arrest due to electric shock.

6.3.1. First Aid

First aid is the treatment or care given to someone who has sustained an injury or disease until more advanced care can be obtained or the person recovers.

The aim of first aid is to:

- Preserve life
- Prevent the worsening of a sickness or injury
- If at all possible, relieve pain
- Encourage recovery
- Keep the unconscious safe.

First aid can help to lessen the severity of an injury or disease, and in some situations, it can even save a person's life.

6.3.2. Need for First Aid at the Workplace

In the workplace, first aid refers to providing immediate care and life support to persons who have been injured or become unwell at work.

Many times, first aid can help to lessen the severity of an accident or disease.

It can also help an injured or sick person relax. In life-or-death situations, prompt and appropriate first aid can make all the difference.

6.3.3. Treating Minor Cuts and Scrapes

Steps to keep cuts clean and prevent infections and scars:

- **Wash Hands:** Wash hands first with soap and water to avoid introducing bacteria into the cut and causing an infection. One should use the hand sanitiser if one is on the go.
- **Stop the bleeding:** Using a gauze pad or a clean towel, apply pressure to the wound. For a few minutes, keep the pressure on.
- **Clean Wounds:** Once the bleeding has stopped, clean the wound by rinsing it under cool running water or using a saline wound wash. Use soap and a moist washcloth to clean the area around the wound. Soap should not be used on the cut since it may irritate the skin. Also, avoid using hydrogen peroxide or iodine, as these may aggravate the wound.
- **Remove Dirt:** Remove any dirt or debris from the area. Pick out any dirt, gravel, glass, or other material in the cut with a pair of tweezers cleaned with alcohol.

6.3.4. Heart Attack

When the blood flow carrying oxygen to the heart is blocked, a heart attack occurs. The heart muscle runs out of oxygen and starts to die.

Symptoms of a heart attack can vary from person to person. They may be mild or severe. Women, older adults, and people with diabetes are more likely to have subtle or unusual symptoms.

Symptoms in adults may include:

- Changes in mental status, especially in older adults.
- Chest pain that feels like pressure, squeezing, or fullness. The pain is most often in the centre of the chest. It may also be felt in the jaw, shoulder, arms, back, and stomach. It can last for more than a few minutes or come and go.
- Cold sweat.
- Light-headedness.
- Nausea (more common in women).
- Indigestion.
- Vomiting.
- Numbness, aching or tingling in the arm (usually the left arm, but the right arm may be affected alone, or along with the left).
- Shortness of breath.
- Weakness or fatigue, especially in older adults and in women.

First Aid for Heart Attack If one thinks someone is experiencing a heart attack, they should:

- Have the person sit down, rest, and try to keep calm.
- Loosen any tight clothing.

- Ask if the person takes any chest pain medicine, such as nitro-glycerine for a known heart condition, and help them take it.
- If the pain does not go away promptly with rest or within 3 minutes of taking nitro-glycerine, call for emergency medical help.
- If the person is unconscious and unresponsive, call 911 or the local emergency number, then begin CPR.
- If an infant or child is unconscious and unresponsive, perform 1 minute of CPR, then call 911 or the local emergency number.

Scan the QR code or click on the link to watch related videos



www.youtube.com/watch?v=BumbKHqXJo0

First-aid practices

Unit 6.4: Waste Management

Unit Objectives

At the end of this unit, participants will be able to:

1. Describe the concept of waste management and methods of disposing of hazardous waste.
2. List the common sources of pollution and ways to minimize them.
3. Elaborate on electronic waste disposal procedures.

6.4.1. Waste Management and Methods of Waste Disposal.

The collection, disposal, monitoring, and processing of waste materials is known as waste management. These wastes affect living beings' health and the environment. For reducing their effects, they have to be managed properly. The waste is usually in solid, liquid or gaseous form.

The importance of waste management is:

Waste management is important because it decreases waste's impact on the environment, health, and other factors. It can also assist in the reuse or recycling of resources like paper, cans, and glass. The disposal of solid, liquid, gaseous, or dangerous substances is the example of waste management.

When it comes to trash management, there are numerous factors to consider, including waste disposal, recycling, waste avoidance and reduction, and garbage transportation. Treatment of solid and liquid wastes is part of the waste management process. It also provides a number of recycling options for goods that aren't classified as garbage during the process.

6.4.2. Methods of Waste Management

Non-biodegradable and toxic wastes, such as radioactive remains, can cause irreversible damage to the environment and human health if they are not properly disposed of. Waste disposal has long been a source of worry, with population increase and industrialisation being the primary causes. Here are a few garbage disposal options.

1. **Landfills:** The most common way of trash disposal today is to throw daily waste/garbage into landfills. This garbage disposal method relies on burying the material in the ground.
2. **Recycling:** Recycling is the process of transforming waste items into new products in order to reduce energy consumption and the use of fresh raw materials. Recycling reduces energy consumption, landfill volume, air and water pollution, greenhouse gas emissions, and the preservation of natural resources for future use.
3. **Composting:** Composting is a simple and natural bio-degradation process that converts organic

wastes, such as plant remnants, garden garbage, and kitchen waste, into nutrient-rich food for plants.

4. **Incineration:** Incineration is the process of combusting garbage. The waste material is cooked to extremely high temperatures and turned into materials such as heat, gas, steam, and ash using this technology.

6.4.3. Recyclable, Non-Recyclable and Hazardous Waste

1. **Recyclable Waste:** The waste which can be reused or recycled further is known as recyclable waste.
2. **Non-recyclable Waste:** The waste which cannot be reused or recycled is known as non-recyclable waste. Polythene bags are a great example of non-recyclable waste.
3. **Hazardous Waste:** The waste which can create serious harm to the people and the environment is known as hazardous waste.

6.4.4. Sources of Pollution

Pollution is defined as the harm caused by the presence of a material or substances in places where they would not normally be found or at levels greater than normal. Polluting substances might be in the form of a solid, a liquid, or a gas.

- **Point source of pollution:** Pollution from a point source enters a water body at a precise location and can usually be identified. Effluent discharges from sewage treatment plants and industrial sites, power plants, landfill sites, fish farms, and oil leakage via a pipeline from industrial sites are all potential point sources of contamination. Point source pollution is often easy to prevent since it is feasible to identify where it originates, and once identified, individuals responsible for the pollution can take rapid corrective action or invest in longer-term treatment and control facilities.
- **Diffuse source of pollution:** As a result of land-use activities such as urban development, amenity, farming, and forestry, diffuse pollution occurs when pollutants are widely used and diffused over a large region. These activities could have occurred recently or in the past. It might be difficult to pinpoint specific sources of pollution and, as a result, take rapid action to prevent it because prevention often necessitates significant changes in land use and management methods.

Pollution Prevention

Pollution prevention entails acting at the source of pollutants to prevent or minimise their production. It saves natural resources, like water, by using materials and energy more efficiently.

Pollution prevention includes any practice that:

- Reduces the amount of any hazardous substance, pollutant, or contaminant entering any waste stream or otherwise released into the environment (including fugitive emissions) prior to

recycling, treatment, or disposal;

- Reduces the hazards to public health and the environment associated with the release of such substances, pollutants, or contaminants (these practices are known as “**source reduction**”);
- Improved efficiency in the use of raw materials, energy, water, or other resources, or Conservation is a method of safeguarding natural resources.
- Improvements in housekeeping, maintenance, training, or inventory management; equipment or technology adjustments; process or method modifications; product reformulation or redesign; raw material substitution; or improvements in housekeeping, maintenance, training, or inventory control.

6.4.5. Electronic Waste







Lead, cadmium, beryllium, mercury, and brominated flame retardants are found in every piece of electronic waste. When gadgets and devices are disposed of illegally, these hazardous compounds are more likely to contaminate the earth, pollute the air, and leak into water bodies.







When e-waste is dumped in a landfill, it tends to leach trace metals as water runs through it. The contaminated landfill water then reaches natural groundwater with elevated toxic levels, which can be dangerous if it reaches any drinking water bodies. Despite having an environmentally benign approach, recycling generally results in international shipment and dumping of the gadgets in pits.

Some eco-friendly ways of disposing of e-waste are:

- Giving back the e-waste to the electronic companies and drop-off points
- Following guidelines issued by the government
- Selling or donating the outdated technology-based equipment
- Giving e-waste to a certified e-waste recycler

ANNEXURE - QR Codes

S.No.	Chapter No.	Unit No.	Topic Name	Page No.	QR code(s)	URL
1	Chapter 2: Process of Developing and Testing Design for IoT Based System	Unit 2.1: IOT System	Introduction of IoT	24		https://www.youtube.com/watch?v=HI6XA-HeX9y0
2		Unit 2.1: IOT System	Components of IoT	24		https://www.youtube.com/watch?v=Z-n4ozz3CkhY
3		Unit 2.2 – IOT System Architecture and Requirements	IoT architecture	31		https://www.youtube.com/watch?v=IxTBxZd-6jao
4		Unit 2.2 – IOT System Architecture and Requirements	IoT design methodology	31		https://www.youtube.com/watch?v=nw_O23o6Dr0
5	Chapter 3: Process of Building GUI and Applications in a Framework	Unit 3.2 – Develop GUI/web UI for IoT System	Graphical User Interface (GUI)	86		https://www.youtube.com/watch?v=9SMmbc-TFrQ
6	Chapter 4: Process of Testing and Troubleshooting the Firmware	Unit 4.1 – Testing of Firmware	Types of IoT testing	94		https://www.youtube.com/watch?v=9QiN-MU UdAgg

S.No.	Chapter No.	Unit No.	Topic Name	Page No.	QR code(s)	URL
7	Chapter 4: Process of Testing and Troubleshooting the Firmware	Unit 4.1 – Testing of Firmware	Interoperability testing	94		https://www.youtube.com/watch?v=Am9SW1T_Qvs
8		Unit 4.2 – Debugging of Firmware	Debugging using GDB	115		https://www.youtube.com/watch?v=D-q8l1_-QgAc
9		Unit 4.2 – Debugging of Firmware	Debugging Arduino Nano 33 IoT with GDB	115		https://www.youtube.com/watch?v=6H_6lg-VOuiA
10	Chapter 5: Work effectively at the workplace	Unit 5.1: Effective Communication and Coordination at Work	Communication skills	124		https://www.youtube.com/watch?v=u16EP-wFmdis
11	Chapter 6: Basic Health and Safety Practices	Unit 6.2 – Fire Safety	Workplace emergency procedures	157		https://www.youtube.com/watch?v=DaYw-cH1GMEg
12		Unit 6.3 – First Aid	First-aid practices	160		https://www.youtube.com/watch?v=Bumb-KHqXJo0





Skill India
कौशल भारत - कुशल भारत



Scan this QR Code to access eBook



Address: Electronics Sector Skill Council of India

155, 2nd Floor, ESC House Okhla Industrial Area-Phase 3, New Delhi- 110020

Email: info@essc-india.org

Web: www.essc-india.org

Phone: +91-84477-38-501

Price: ₹