O Level: M4-R5.1

Internet of Things and its Applications

(I) INTRODUCTION TO INTERNET OF THINGS - APPLICATIONS/DEVICES, PROTOCOLS AND COMMUNICATION MODEL (II) THINGS AND CONNECTIONS (III) SENSORS, ACTUATORS AND MICROCONTROLLERS (IV) BUILDING IOT APPLICATIONS (V) SECURITY AND FUTURE OF IOT ECOSYSTEM (VI) SOFT SKILLS-PERSONALITY DEVELOPMENT

Module Unit	Written Marks (Max.)
1. Introduction to IoT - Applications/Devices, Protocols and Communication Model	10
2. Things and Connections	10
3. Sensors, Actuators and Microcontrollers	15
4. Building IoT Applications	40
5. Security and Future of IoT Ecosystem	5
6. Soft skills-Personality Development	20

CHAPTER 1

Introduction to Internet of Things-Applications/ Devices, Protocols & Communication Model

इंटरनेट ऑफ थिंग्स कापरिचय एप्लीकेशन/ डिवाइसेज, प्रोटोकॉल्स और कम्यूनिकेशनमॉडल

Introduction - Overview of Internet of Things (IoT)

परिचय- इंटरनेट ऑफ थिंग्स का अवलोकन(IoT)

Internet of things is a platform where regular devices are connected to the internet, so they can interact, collaborate and exchange data with each other. The Internet of Things (IoT) is here to change the world we know. Smart cars, smart homes, smart cities, everything around us can be turned into a smart device with the help of the Internet of Things. IoT sure does bring the 'cool' factor to technology. While IoT is poised to bring the next big boom in the job market, lack of skill is facing the biggest barrier for companies looking to adopt this technology.

इंटरनेट ऑफ थिंग्स एक ऐसाप्लेट फॉर्म है जहाँ नियमित उपकरण इंटरनेट से जुड़े होते हैं, इसलिए वे एक-दूसरे के साथ डेटा काइंटरैक्ट, कोलाबोरेट और आदान-प्रदान कर सकते हैं।इंटरने ऑफ थिंग्स(IoT) यहां दुनिया में बदलाव को जानने के लिए है। स्मार्ट कार, स्मार्ट होम, स्मार्ट सिटीज, हमारे आस पास की प्रत्येक वस्तु को इंटरनेट ऑफ थिंग्स की मदद से स्मार्ट डिवाइस में बदला जा सकता है । IoT यकीनन प्रौद्योगिकी के लिए ' शांत ' का रक है । जब IoT नौकरी बाजार में बड़े बदलाव लाने की ओर अग्रसर है, कौशल की कमी इस

तकनीक को अपनाने की इच्छुक कंपनियों के लिए सब से बड़ी बाधा है।

History of IoT (IoT का इतिहास) :-

1. 1970: The actual idea of connected devices was proposed.

जुड़े उपकरणों का वास्तविक विचार प्रस्तावित किया गया था।

2. 1990: John Romkey created a toaster which could be turned on/off over the Internet.

जॉन रोमकी ने एक टोस्टर बनाया जिसे इंटरनेट पर चालू/बंद किया जा सकता था।

3. 1995: Siemens introduced the first cellular module built for M2M (Machine to Machine).

सीमेंस ने M2M (मशीन से मशीन) के लिए निर्मित पहला सेलुलर मॉड्यूल प्रस्तुत किया।

4. 1999: The term "Internet of Things" was used by Kevin Ashton during his work at P&G which became widely accepted.

"इंटरनेट ऑफ थिंग्स" शब्द का योग केविन एश्टन द्वारा P & G में अपने काम के दौरान किया गया था जिसे व्यापक रूप से स्वीकार किया गया।

5. 2004: The term was mentioned in famous publications like the Guardian, Boston Globe, and Scientific American.

गार्डियन, बोस्टन ग्लोब और साइंटिफिक अमेरिकन जैसे प्रसिद्ध प्रकाशनों में इस शब्द का उल्लेख किया गया था।

6. 2005: UN's International Telecommunications Union (ITU) published its first report on this topic.

संयुक्त राष्ट्र के अंतर्राष्ट्रीय दूर संचार संघ(ITU) ने इस विषय पर अपनी पहली रिपोर्ट प्रकाशित की।

7. 2008: The Internet of Things was born.

इंटरनेट ऑफ थिंग्स का जन्म हुआ।

8. 2011: Gartner, the market research company, includes "The Internet of Things" technology in their research.

मार्केट रिसर्च कंपनी गार्टनर ने अपने शोध में "इंटरनेट ऑफ थिंग्स" तकनीक को शामिल किया।

(Advantages of IOT) आईओटीकेलाभ:-

a) Communication - कम्यूनिकेशन

IoT encourages the communication between devices, also famously known as Machine-to-Machine (M2M) communication. Because of this, the physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and

greater quality.

IoT उपकरणों के बीच संचार को प्रोत्साहित करता है, जिसे मशीन-टू-मशीन(एम2 एम) संचार के रूप में भी जाना जाता है।इस वजह से, भौतिक उपकरण जुड़े हने में सक्षम हैं और इसलिए कुल पारदर्शिता कम अक्षमताओं और अधिक गुणवत्ता के साथ उपलब्ध है।

b) Access information – एक्सेस इनफार्मेशन

We can easily access any information sitting in any part of the globe in real time. This makes it very convenient for people to go about their work, even if they are not physically present.

हम वास्तविक समय में दुनिया के किसी भी हिस्से में बैठे किसी भी जानकारी को आसानी से एक्सेस कर सकते हैं।यह लोगों को अपने काम के बारे में ने के लिए बहुत सुविधा जनक बनाता है, भले ही वे ‘ारीरिक रूप से मौजूद न हों।

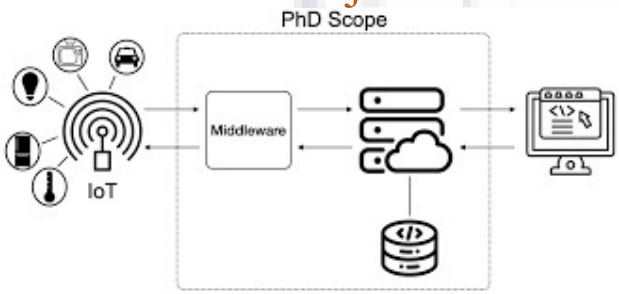
c) Automation and Control – ऑटोमेशन और कन्ट्रोल

With wireless infrastructure connecting and controlling digital objects digitally and centrally, there is a large amount of automation and control in its functioning. Without human intervention, machines are able to communicate with each other to deliver fast and timely output.

वायरलेस इंफ्रास्ट्रक्चर के साथ डिजिटल वस्तुओं को डिजिटल और सेन्ट्रली रूप से कनेक्ट और नियंत्रित करने के कारण, इसके काम काज में बड़ी मात्रा मेंस्वचालन(ऑटोमेशन) और नियंत्रण होता है। मानव हस्तक्षेप के बिना, मशीनें तेजी से और समय पर आउटपुट देने के लिए एक दूसरे के साथ संवाद करने में सक्षम हैं।

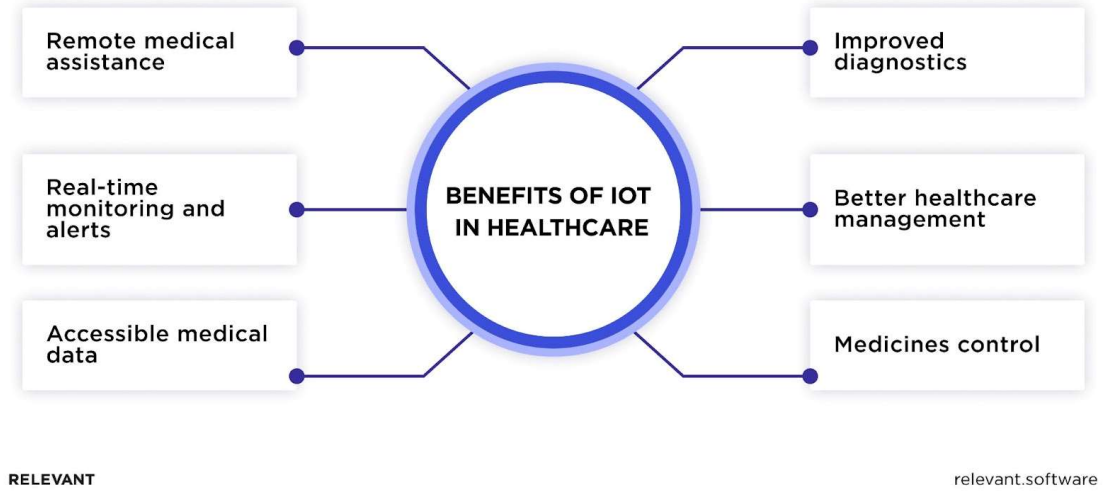
d) Efficient and Saves Time – कुशल और समय की बचत

The machine-to-machine interaction provides better efficiency, hence; accurate results can be obtained fast. This results in saving valuable time. Instead of repeating the same tasks every day, it enables people to do other creative jobs.



मशीन-टू-मशीन इंटरैक्शन बेहतर दक्षता प्रदान करता है, इसलिये सटीक परिणाम से सकिए जा सकते हैं।इससे बहुमूल्य समय की बचत होती है।हर दिन समान कार्यों को दोहराने के बजाय, यह लोगों को अन्य रचनात्मक कार्य करने में सक्षम बनाता है।

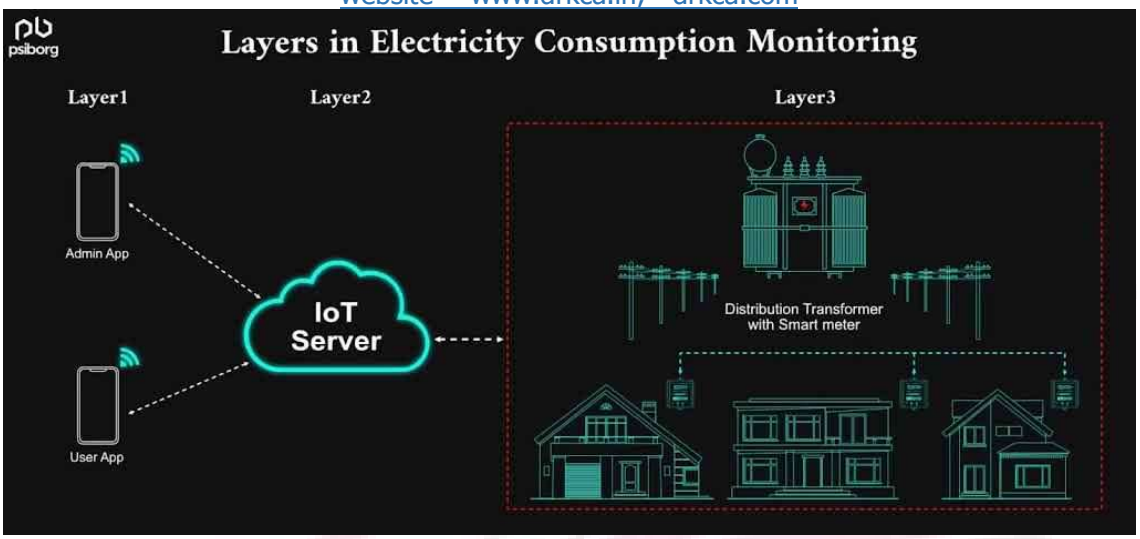
e) Better Health Care and Management – बेहतर स्वास्थ्य देखभाल औ प्रबंधन



The patient monitoring can be done on a real-time basis without a doctor's visit. It enables them to make decisions as well as evidence-based treatment.

रोगी की निगरानी बिना डॉक्टर के दौरे के वास्तविक समय के आधार पर की जा सकती है।यह उन्हें निर्णय लेने के साथ-साथ साक्ष्य-आधारित प्रस्ताव उपचार के लिए सक्षम बनाता है।

f) Efficient usage of electricity – बिजली का कुशल उपयोग

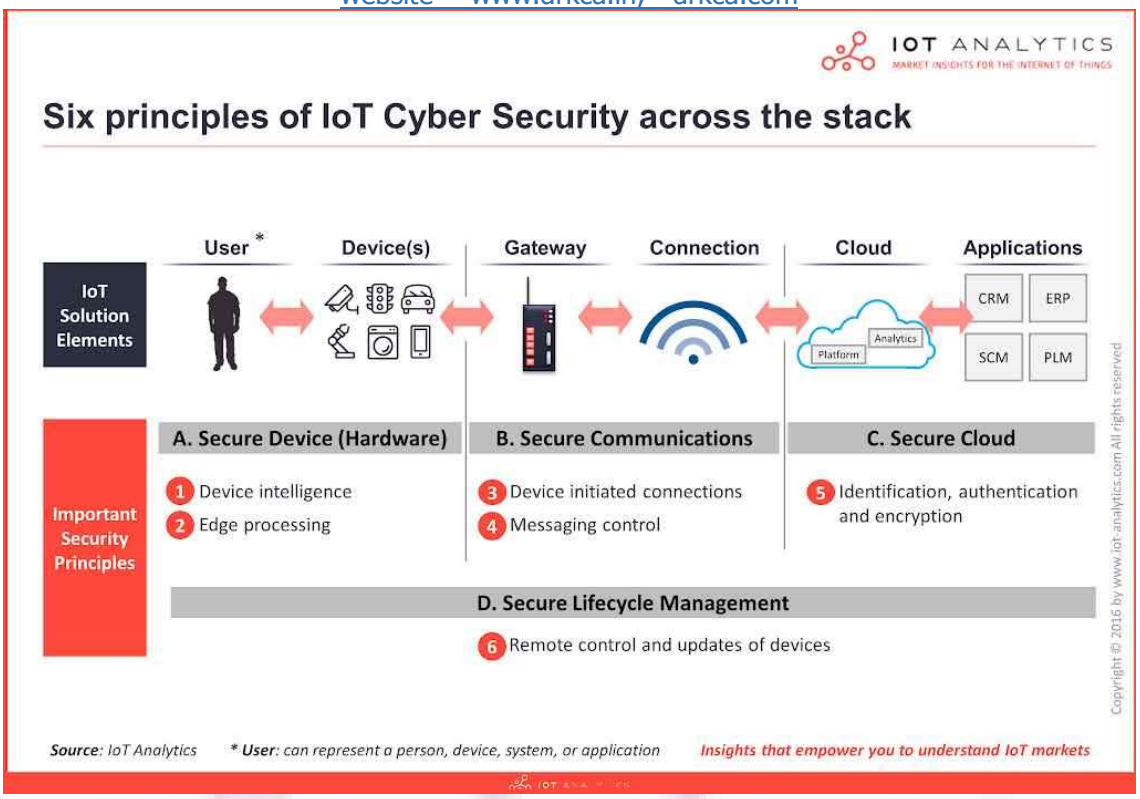


You can use electricity effectively because the devices are directly connected and communicate with controller devices such as mobile phones. So you don't have to worry about unnecessary use of electricity and appliances.

आप बिजली का प्रभावी ढंग से उपयोग कर सकते हैं क्योंकि डिवाइस सीधे जुडी हुई हैं और मोबाइल फोन जैसे नियंत्रक डिवाइस के साथ संवाद करते हैं।इसलिए आपको बिजली और उपकरणों के आवश्यक उपयोग के बारे में चिंता करने की जरूरत नहीं है।

g) Security - सुरक्षा

You can control your home through mobile phones. It provides personal protection.



आप मोबाइल फोन के माध्यम से अपने घर को नियंत्रित कर सकते हैं। यह व्यक्तिगत सुरक्षा प्रदान करता है।

Disadvantages of IOT आईओटीकेनुकसान:-

a) Compatibility - कम्पैटिबिलिटी- Currently, there is no international standard of compatibility for the tagging and monitoring equipment.

वर्तमान में, टैगिंग और निगरानी उपकरण के लिए अनुकूलता (कम्पैटिबल्टी) का कोई अंतरराष्ट्रीय स्टैंडर्ड नहीं है।

(b)The system offers little control which leads to various kinds of network attacks like hackers could break into the network and steal the information. So, data encryption is very much necessary for maintaining the privacy of the data shared over the internet.

चूंकि IoT सिस्टम आपस में जुड़े हुए हैं और नेटवर्क पर संचार करते हैं। सिस्टम थोड़ा नियंत्रण प्रदान करता है जो विभिन्न प्रकार के नेटवर्क हमलों का नेतृत्व करता है जैसे है कर्स नेटवर्क को तोड़ सकते हैं और जानकारी चुरा सकते हैं।इसलिए, इंटरनेट पर साझा किए गए डेटा की गोपनीयता बनाए रखने के लिए डेटा एन्क्रिप्शन बहुत आवश्यक है।

c) Privacy wişdent - Even without the active participation of the user, the IoT system provides substantial personal data in maximum detail.

उपयोग कर्ता पर सक्रिय भागीदारी के बिना भी, IoT सिस्टम अधिकतम विस्तार में पर्याप्त व्यक्तिगत डेटा प्रदान करता है।

d)Unemployment-बेरोजगारी: With every task being automated will lead to a lower need for manpower and eventually loss of jobs. This will have a direct impact on employment.

प्रत्येक कार्य के स्वतः होने सेजन शक्ति की कम आवश्यकता और ततः नौकरियों का नुकसान होगा। इसका सीधा अस रोजगार पर पड़ेगा।

सरल संस्करण (Simplified Version)

यह PDF O Level कोर्स M4-R5.1 के बारे में है, जो "Internet of Things (IoT) और उसके उपयोग" पर है। मैंने इसे आसान भाषा में समझाया है, लेकिन मूल अर्थ वही रखा है। मूल PDF में अंग्रेजी और हिंदी दोनों हैं, इसलिए मैं दोनों को सरल बनाकर दे रहा हूं।

मुख्य भाग (Syllabus)

यह कोर्स 6 हिस्सों में बंटा है:

1. IoT का परिचय - उपयोग, डिवाइस, प्रोटोकॉल और कम्युनिकेशन मॉडल (10 अंक)
2. चीजें और कनेक्शन (10 अंक)
3. सेंसर, एक्ट्यूएटर और माइक्रोकंट्रोलर (15 अंक)
4. IoT ऐप बनाना (40 अंक)
5. IoT की सुरक्षा और भविष्य (5 अंक)
6. सॉफ्ट स्किल्स - व्यक्तित्व विकास (20 अंक)

अध्याय 1: IoT का परिचय - उपयोग/डिवाइस, प्रोटोकॉल और कम्युनिकेशन मॉडल

IoT क्या है?

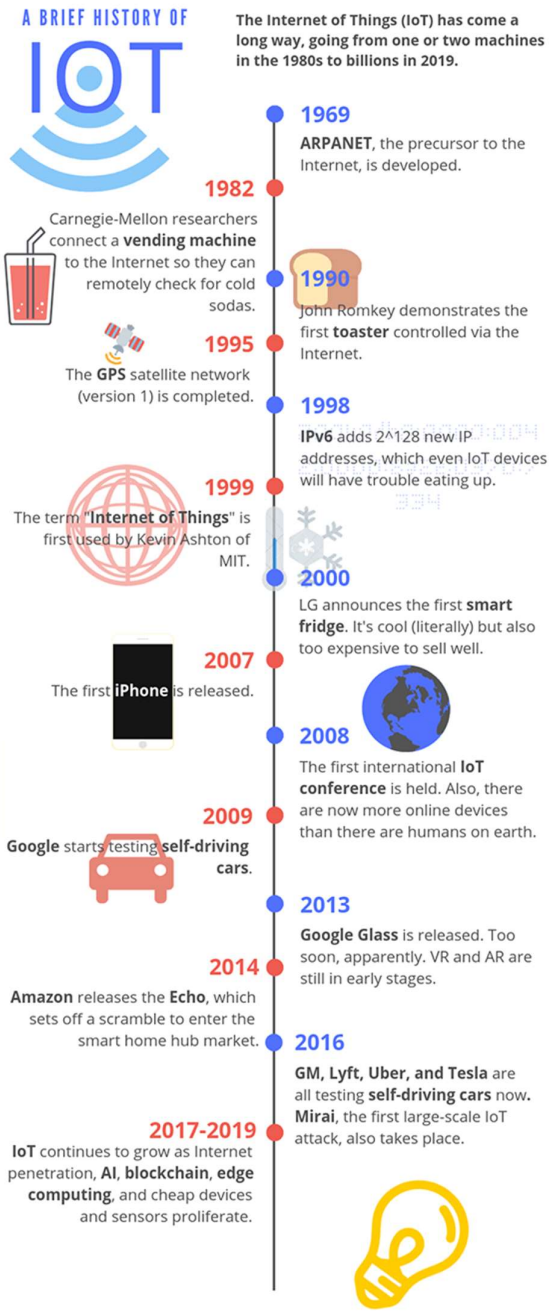
(Introduction)

IoT एक ऐसा सिस्टम है जहां सामान्य चीजें (जैसे फ्रिज, कार) इंटरनेट से जुड़ती हैं। वे एक-दूसरे से बात कर सकती हैं, डेटा शेयर कर सकती हैं। इससे स्मार्ट घर, स्मार्ट शहर बनते हैं। IoT तकनीक को 'कूल' बनाती है, लेकिन इसमें नौकरियां बढ़ेंगी पर स्किल्स की कमी है।

हिंदी में सरल: IoT एक प्लेटफॉर्म है जहां आम डिवाइस इंटरनेट से जुड़ती हैं। वे डेटा शेयर करती हैं। इससे दुनिया बदल रही है - स्मार्ट कार, घर, शहर। IoT तकनीक को मजेदार बनाती है, लेकिन कंपनियों को स्किल्ड लोग नहीं मिल रहे।

IoT का इतिहास (History)

1970: जुड़ी हुई डिवाइस का आइडिया आया।



- 1990: जॉन रोमकी ने इंटरनेट से ऑन/ऑफ होने वाला टोस्टर बनाया।
- 1995: सीमेंस ने M2M के लिए पहला सेलुलर मॉड्यूल बनाया।
- 1999: केविन एश्टन ने "Internet of Things" शब्द इस्तेमाल किया।
- 2004: बड़े अखबारों में इसकी चर्चा हुई।
- 2005: UN (ITU) ने पहली रिपोर्ट जारी की।
- 2008: IoT का जन्म हुआ।
- 2011: गार्टनर ने इसे अपनी रिसर्च में शामिल किया।

IoT के फायदे (Advantages)

a) कम्युनिकेशन: डिवाइस एक-दूसरे से बात करती हैं (M2M), इससे पारदर्शिता बढ़ती है, गलतियां कम होती हैं।

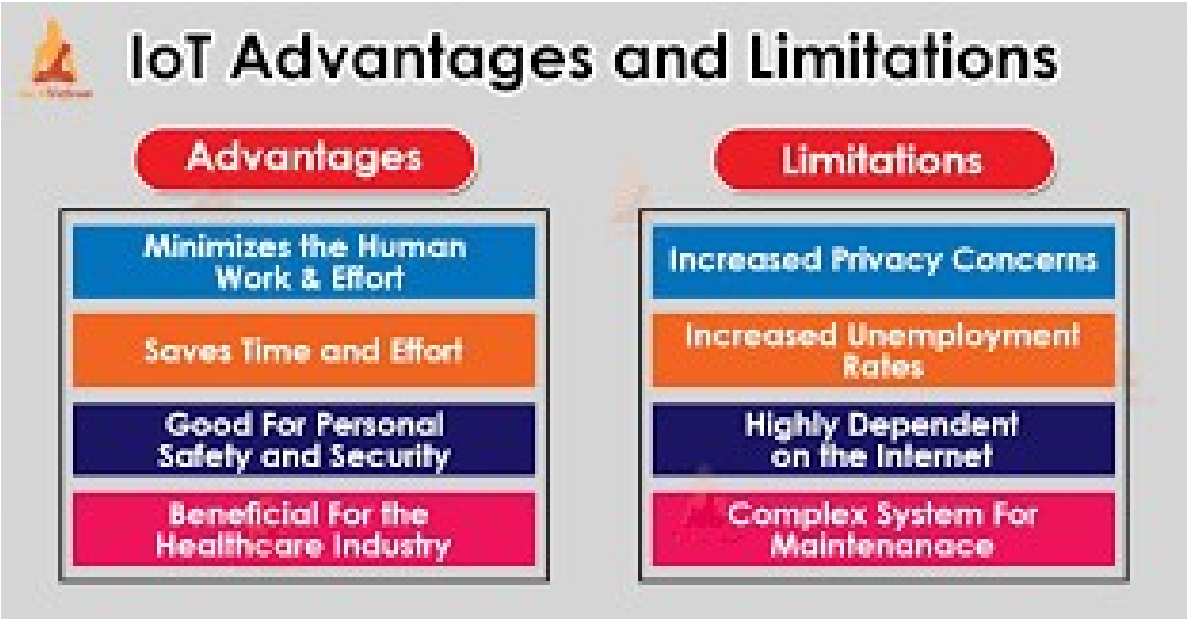
b) जानकारी एक्सेस: दुनिया के किसी कोने से रियल-टाइम डेटा मिलता है, भले आप वहां न हों।

c) ऑटोमेशन: वायरलेस से चीजें खुद चलती हैं, इंसान की जरूरत कम, तेज काम होता है।

d) समय बचत: मशीनें कुशल काम करती हैं, लोग क्रिएटिव काम कर सकते हैं।

e) स्वास्थ्य: मरीज की निगरानी बिना डॉक्टर के विजिट के, साक्ष्य-आधारित इलाज।

f) बिजली बचत: डिवाइस मोबाइल से जुड़ी, फालतू इस्तेमाल नहीं होता। g) सुरक्षा: घर को मोबाइल से कंट्रोल करो, पर्सनल प्रोटेक्शन मिलती है।



IoT के नुकसान (Disadvantages)

a) कम्पैटिबिलिटी: टैगिंग और मॉनिटरिंग के लिए कोई अंतरराष्ट्रीय स्टैंडर्ड नहीं।

b) सुरक्षा: नेटवर्क से जुड़े होने से हैकर्स चोरी कर सकते हैं, डेटा एन्क्रिप्शन जरूरी।

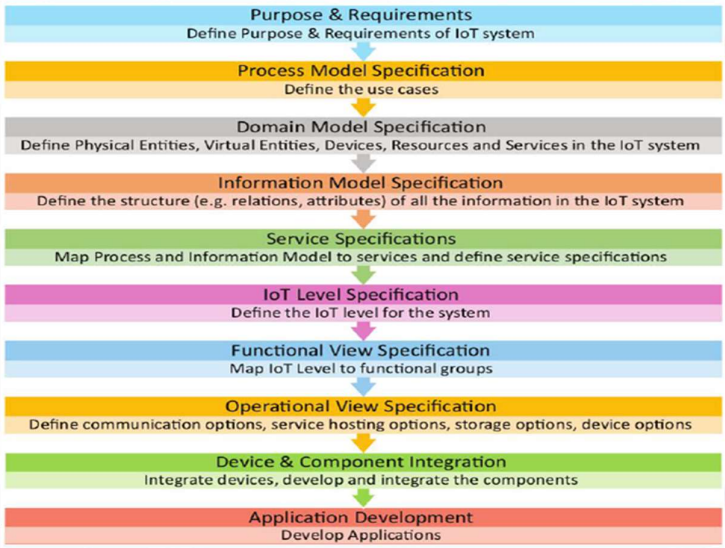
c) प्राइवेसी: यूजर की भागीदारी बिना भी पर्सनल डेटा ज्यादा शेयर होता है।

d) बेरोजगारी: सब ऑटोमेटेड होने से नौकरियां कम होंगी।

यह सरल संस्करण मूल सामग्री को आसान शब्दों में समझाता है, लेकिन कोई बदलाव नहीं किया। अगर और स्पष्ट करने की जरूरत हो, बताएं!

Chapter) - 1(Class - 2

IoT design methodology



IoT डिजाईनपद्धति

IoT Design Methodology that includes the following steps: IoT डिजाईन पद्धति जिसमें निम्नलिखित चरण समिल्लित है.....

Step 1 : Purpose and Requirements Specification First step is to define the purpose and requirements of the system. In this step, the system's purpose, behavior and requirements are captured.

पहला कदम सिस्टम के उद्देश्य और आवश्यकताओं को परिभाषित करना है।इस चरण में, सिस्टम के उद्देश्य, व्यवहार और आवश्यकताओं को मिल किया जाता है

- *Data collection requirements(डेटा संग्रह आवश्यकताएँ)*

- *System management requirements(सिस्टम प्रबंधन आवश्यकताएँ)*

- *Security requirements(सुरक्षा आवश्यकताएँ)*

- *User interface requirements(उपयोगकर्ता इंटरफ़ेस आवश्यकताएँ)*

Step 2 : Process Specification The use cases of the IoT system are formally described based on or derived from the purpose and requirements specifications IoT

प्रणाली के उपयोग के मामलों को उद्देश्य और आवश्यकताओं के विनिर्देशोंके आधार पर याउन से प्राप्त औपचारिक रूप से वर्णित किया गया है

Step 3 : Domain Model Specification The domain model describes the main concepts, entities and objects in the domain of the IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects. The domain model is independent of any specific technology or platform. Physical Entity, Virtual Entity, Device, Resource, Service

Step 4 : Information Model Specification The Information Model defines the structure of all the information in the IoT system.

सूचना मॉडल IoT प्रणाली में सभी सूचनाओं की रचना को परिभाषित करता है।

Step 5 : Service Specifications Service specifications define the services in the IoT system such as service types, service inputs/output, service endpoints, service schedules, service preconditions, and service effects

सेवाविनिर्देशIoT प्रणाली में सेवाओं को परिभाषित करते हैं जैसे सेवा प्रकार, सेवा इनपुट/आउटपुट, सेवा समापन बिंदु, सेवा शेड्यूल, सेवापूर्व शर्त और सेवाप्रभाव

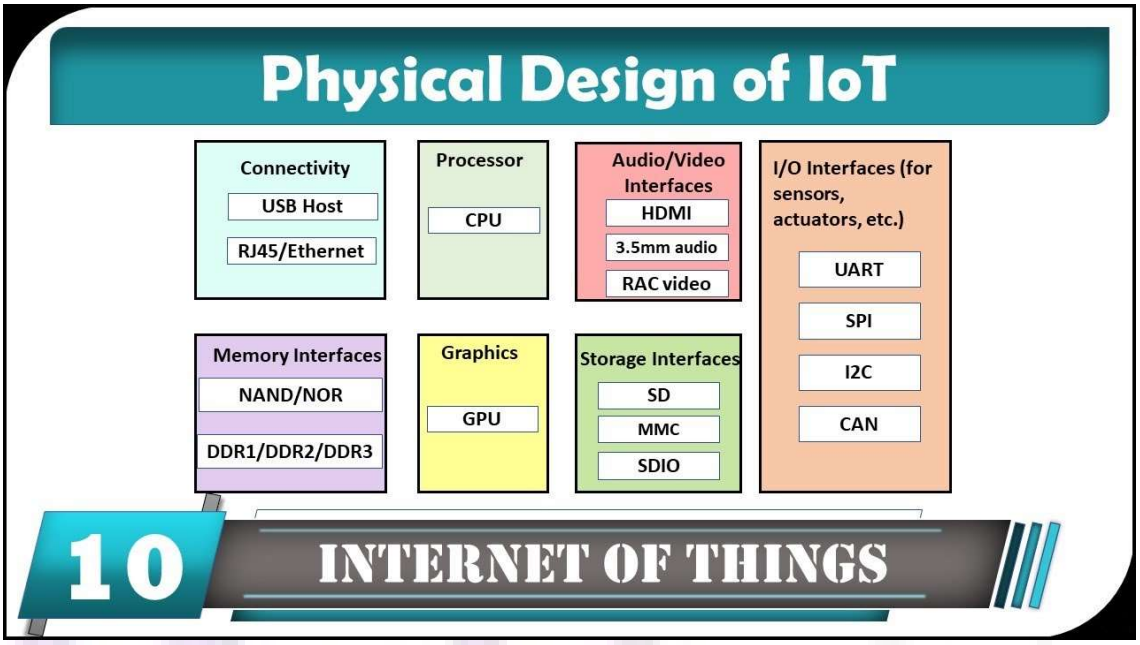
Step 6: IoT Level Specification In this step define the IoT level for the system. Five typess of IoT deployment levels are used according to different conditions.

इस चरण में सिस्टम केलिए IoT स्तर को परिभाषित करें । विभिन्न स्थितियों के अनुसार पाँच प्रकार के IoT परिनियोजन स्तरों का उपयोगकि या जाता है।

The Physical Design of IoT

Physical Design of IoT refers to Things in IoT and IoT Protocols. Things are Node devices which have unique identities and can perform remote sensing, actuating, and monitoring capabilities. IoT Protocols help Communication established between things and cloud-based servers over the Internet.

IoT काफि जिकल डिजाइन IoT में वस्तुओं और IoT प्रोटोकॉल को संदर्भित करता है।वस्तु यें नोड डिवाइस हैं जिनकी विशिष्ट पहचान हैं और रिमोट सेंसिंग, एक्टुवेटिंग और निगरानी की क्षमताओं का



प्रदर्शन कर सकती हैं। IoT प्रोटोकॉल इंटरनेट पर वस्तुओं और क्लाउड आधारित सर्वर के बीच संचार स्थापित करने में मदद करता है।

1. **Things in IoT** : Refers to IoT devices which have unique identities that can perform sensing, actuating and monitoring capabilities. IoT Devices can be various types, Sensing Devices, Smart Watches, Smart Electronics appliances, Wearable Sensors, Automobiles, and industrial machines.
2. **IoT मैंथिंग्स**: IoT उपकरणों का संदर्भ देता है जिनकी विशिष्ट पहचान होती है जोसेंसिंग , सक्रियण और निगरानी की क्षमताओं प्रदर्शन कर सकते हैं। IoT डिवाइसेस विभिन्न प्रकार के हो सकते

हैं, सेंसिंग डिवाइसेस, स्मार्टवॉचेस, स्मार्ट इलेक्ट्रॉनिक्स उपकरण, वेयरेबल सेंसर, ऑटोमोबाइल्स और इंडस्ट्रियल मशीनें।

3. **IoT Protocols** - IoT प्रोटोकॉल्स IoT protocols help to establish Communication between IoT Device (Node Device) and Cloud based Server over the Internet. It helps to send commands to IoT devices and receive data from an IoT device over the Internet.

IoT प्रोटोकॉलइंटरनेटपर IoT डिवाइस (नोडडिवाइस) और क्लाउड आधारित सर्वर के बीच संचार स्थापित करने में मदद करते हैं। यह IoT डिवाइस को कमांड भेजने में मदद करता है और इंटरनेट पर एक IoT डिवाइस से डेटा प्राप्त करता है।

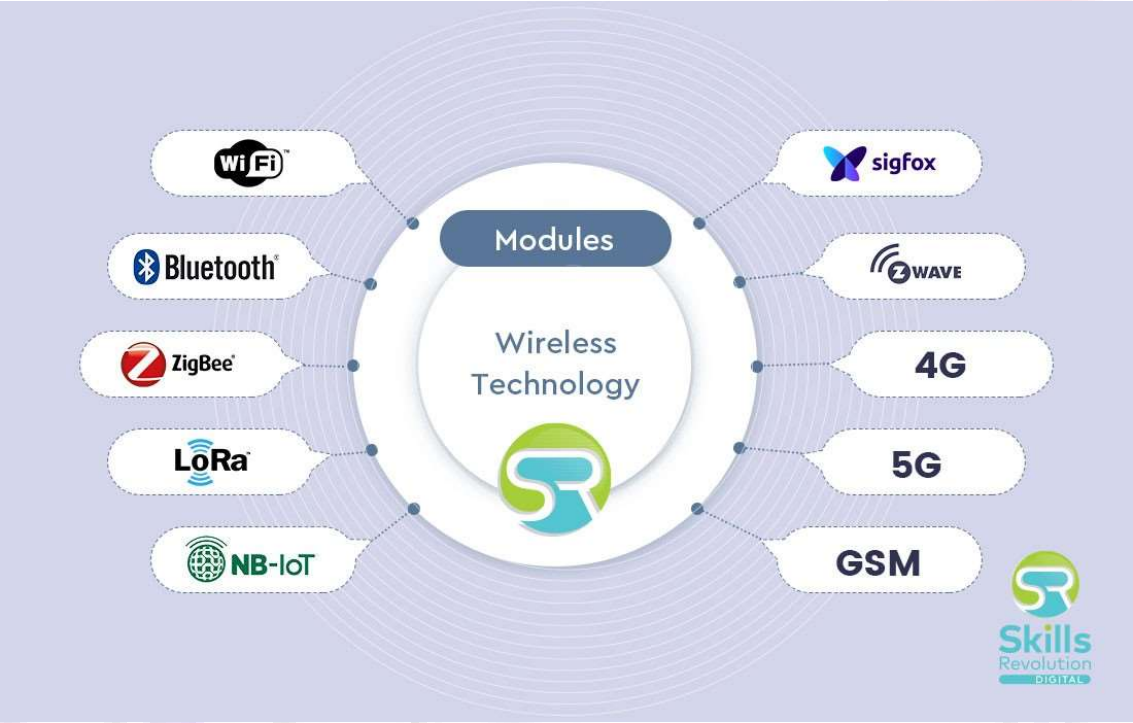
Link Layer – Link layer protocols determine how data is sent over the network's physical layer or medium (wired or wireless). Link Layer determines how the packets are coded and signalled by the hardware device over the medium to which the host is attached.

लिंक लेयर प्रोटोकॉल यह निर्धारित करते हैं कि नेटवर्क के फिजिकल लेयर या माध्यम (वायर्डयावायरलेस) पर डेटा कैसे भेजा जाता है। लिंक लेयर यह निर्धारित करता है कि जिस माध्यम से मेजबान होस्ट हुआ है उस माध्यम से हार्डवेयर डिवाइस द्वारा पैकेट को कैसे कोडित और संकेतित किया जाता है

Wireless connectivity :-

Wi–Fi: A widely used wireless networking technology that enables high–speed data *transfer over short to medium distances.*

एकव्यापक रूप से उपयोग की जाने वाली वायर लेसने टवर्किंगतकनी क जो छोटी से मध्यम दूरी पर उच्चगति डेटा



स्थानांतरण को सक्षम बनाती है।

Bluetooth: A short–range wireless technology used for connecting devices in proximity. It is commonly used for IoT devices that require low power consumption and intermittent data transfer, such as wearable devices and home automation systems.

एक छोटी दूरी वायर लेस तकनीक जिसका उपयोग निकटवर्ती उपकरणों को जोड़ने केलिए किया जाता है। इसका उपयोग आमतौर

पर IoT उपकरणों केलिए किया जाता है जिनके लिए कम बिजली की खपत और रुक-रुककर डेटा ट्रांसफर की आवश्यकता होती है, जैसे पहनने योग्य डिवाइस और होम ऑटोमेशन सिस्टम।

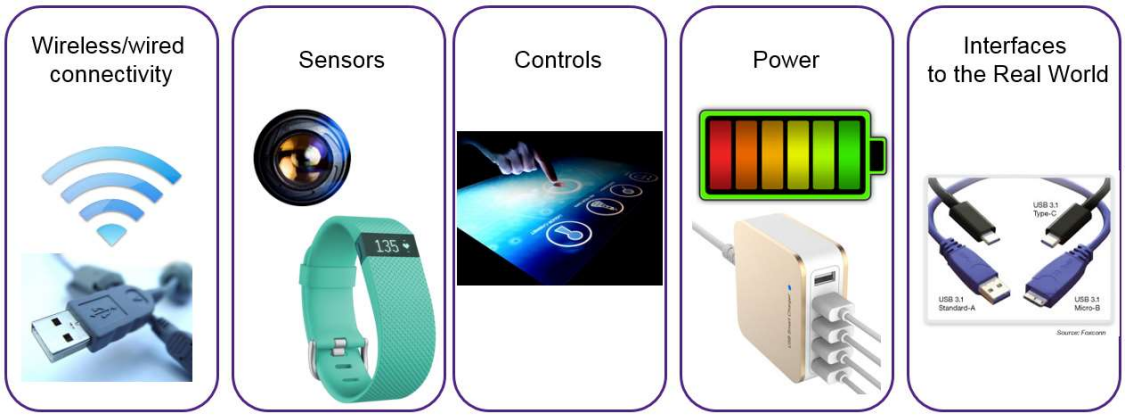
Zigbee: A low–power, low–data–rate wireless communication protocol designed for applications with low power consumption requirements and a large number of devices. It is commonly used in home automation, smart lighting, and industrial applications.

एक कम-शक्ति, कम-डेटा-दर वायरलेस संचार प्रोटोकॉल जोकम बिजली की खपत आवश्यकताओं और बड़ी संख्या में उपकरणों वाले अनुप्रयोगों के लिए डिज़ाइन किया गया है । इसका उपयोग आम तौर पर होम ऑटोमेशन, स्मार्ट लाइटिंग और औद्योगिक अनुप्रयोगों में किया जाता है।

LPWAN (Low–Power Wide Area Network): LPWAN technologies, such as LoRaWAN and NB–IoT, offer long–range connectivity with low power consumption, making them suitable for IoT applications that require wide area coverage, such as smart city deployments and agricultural monitoring.

एलपीडब्ल्यूएएन प्रौद्योगिकियां , जैसे लोरावन और एनबी -आईओटी, कम बिजली की खपत के साथ लंबीदूरी की कनेक्टिविटी प्रदान करती हैं, जो उन्हें आईओटी अनुप्रयोगों के लिए उपयुक्त बनाती हैं , जिनके लिए व्यापक क्षेत्र कवरेज की आवश्यकता होती है, जैसे स्मार्टसिटी तैनाती और कृषिनिगरानी।

Wired connectivity :- Ethernet: A standard wired networking technology that provides reliable and high-speed data transfer over a local area network (LAN). Ethernet is commonly used in industrial settings and for devices requiring high bandwidth and low latency.



एक मानकवायर्ड नेटवर्किंग तकनीक जो स्थानीय क्षेत्र नेटवर्क(LAN) परविश्वस नीय और उच्चगति डेटा स्थानांतरण प्रदान करती है । ईथरनेट का उपयोग आमतौर पर औद्योगिक से टिंग्समें और उच्चबैंड विड्थ और कमविलंबता की आवश्यकता वाले उपकरणों के लिए किया जाता है।

IEEE : Institute of Electrical and Electronics EngineersPowerline Communication: This technology allows data transmission over existing power lines, eliminating the need for additional wiring.

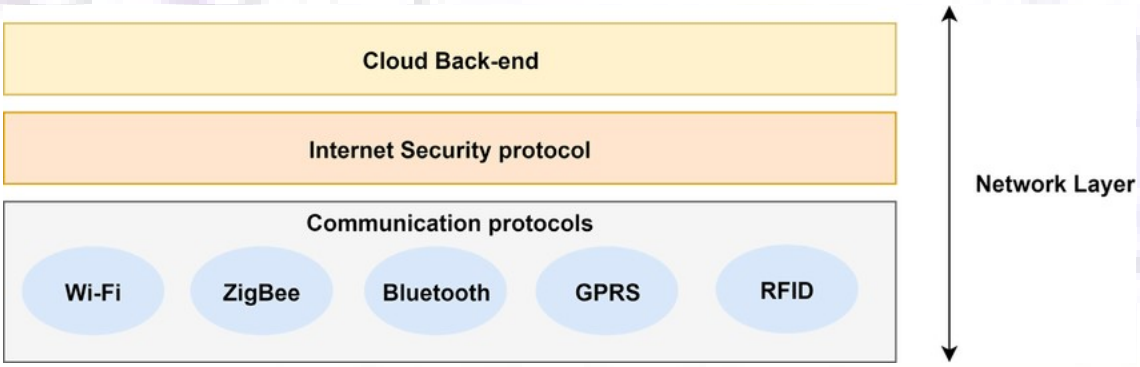
WHAT IS THE MEANING OF IEEE?

Institute of Electrical and Electronic Engineers (IEEE).

- IEEE – is an international technical professional society
- More than 375,000 members from 150 countries
- One of the pre-eminent standards bodies
- Advances the theory and application of electro-technologies and allied sciences
- Produces over 30% of world's published literatures in electrical engineering, computers, and controls.
- IEEE standards are recognized nationally and worldwide.

यह तकनीक मौजूदा बिजली लाइनों पर डेटा ट्रांसमिशन की अनुमति देती है, जिसस अतिरिक्त वायरिंग की आवश्यकता समाप्त हो जाती है।

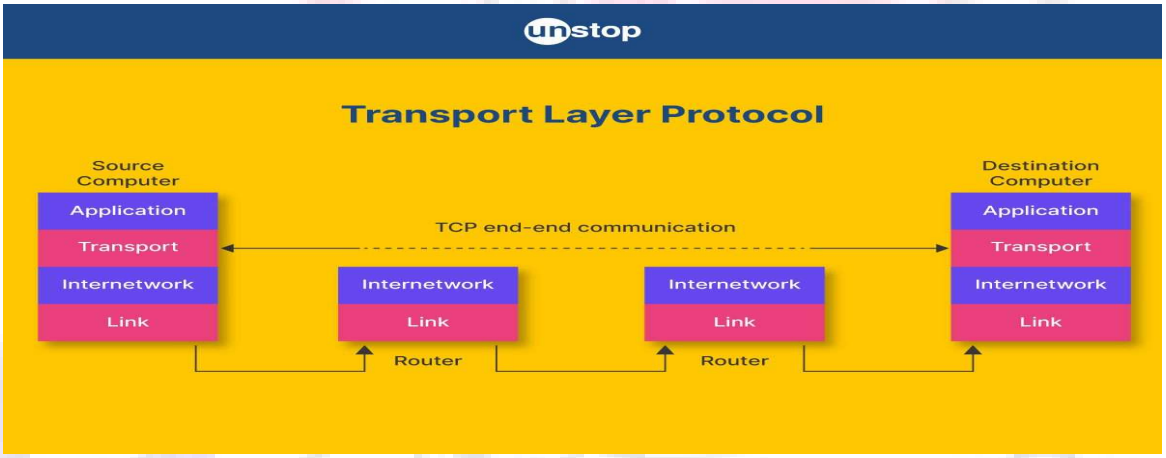
Network/Internet Layer: This layer is responsible for sending IP datagrams from source to destination network. Its host identification is done using hierarchical IP addressing schemes such as IPV4 or IPV6.



नेटवर्क/इंटरनेटलेयर:यह प्लेयर सोर्स से डेस्टिनेशन नेटवर्क तक आईपीडे टाग्राम भेजने के लिए रिस्पॉन्सिबल है।इसकी होस्ट पहचान

IPV4 याIPV6 जैसी पदानुक्रमित आईपीएड्रेसिंग योजनाओं का उपयोग करके की जाती है।

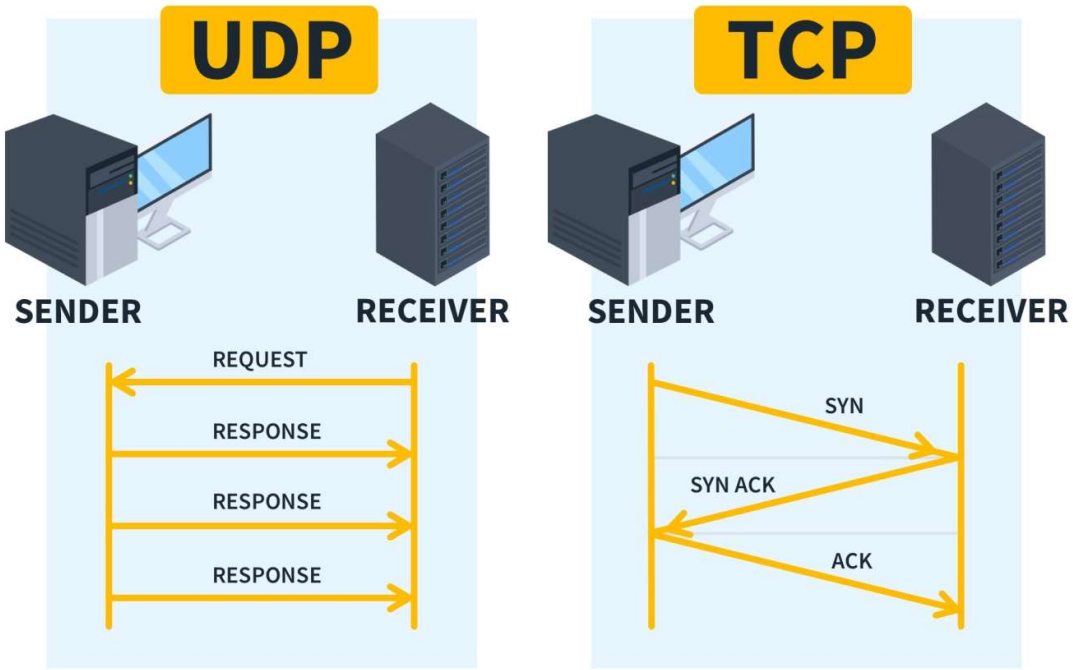
Transport Layer - ट्रांसपोर्टलेयर:- This layer provides end-to-end message transfer capability independent of the underlying network. It provides functions such as error control, segmentation, flow control and congestion control……



यह लेयर अंतर्निहित नेटवर्क से स्वतंत्रइंड –टू –इंड संदेश हस्तांतरण क्षमता प्रदान करती है ।यह त्रुटिनियंत्रण , विभाजन, प्रवाह नियंत्रण और भीड़ नियंत्रण जैसे फंक्शन्स प्रदान करता है।

Difference between TCP and UDP –

टीसीपी और यूडीपी के बीच अन्तर



1.TCP 2.UDP

1(a)It is a connection-oriented protocol.

यह एक कनेक्शन –ओरियन्टेड प्रोटोकॉल है।

2(a)It is the connectionless protocol.

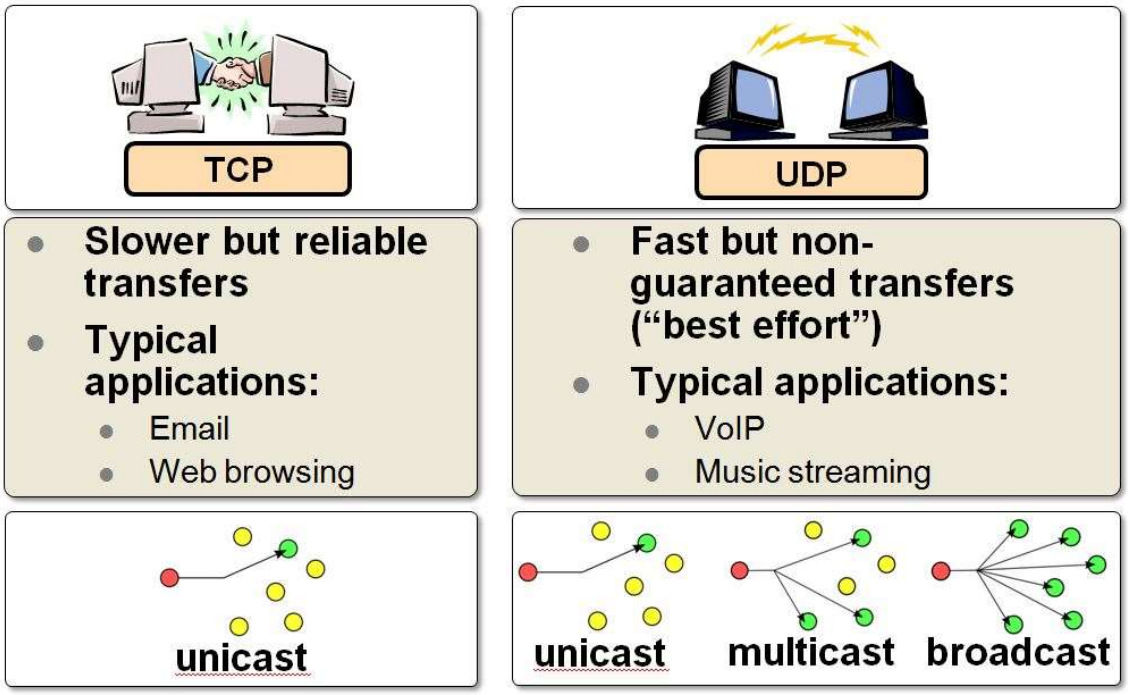
यह एक कनेक्शन लेस प्रोटोकॉल है।

1(b)TCP is reliable as it guarantees the delivery of the delivery of data to the destination router

टीसीपी विश्वसनीय है क्योंकि यह गंतव्य राउटर को डेटा कीडिलीवरी की गारंटी देता है

2(b)The delivery of data to the destination cannot be guaranteed in UDP.

UDP में गंतव्यतक डेटा की डिलीवरी की गारंटी नहीं दीजासकती है।



1(c)TCP is slower than UDP.

यूडीपी की तुलना में टीसीपी तुलनात्मक रूप से धीमा है।

2(c)UDP is faster, simpler, and more efficient than TCP.

UDP TCP की तुलना में अधिक तेज, सरल और अधिककुशल है।

1(d)It does error checking and also makes error It performs error checking, but it discards recovery.

यह एर रको चेक करता है और एररिकवरीभी करता है।

2(d)It checks for error and discards the pack with error.

यह एर रको चेककरता है और एरवाले पैकेट को डिस्कार्ड करता है।

1(e)It is heavy-weight.

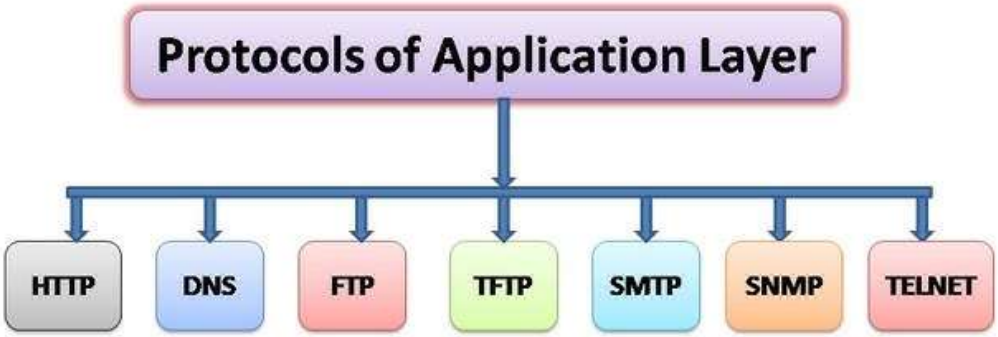
यह है विवेट है।

2(e)It is lightweight. But not that reliable.

यह लाइट वेट है ले किनरि लाइबल नही है।

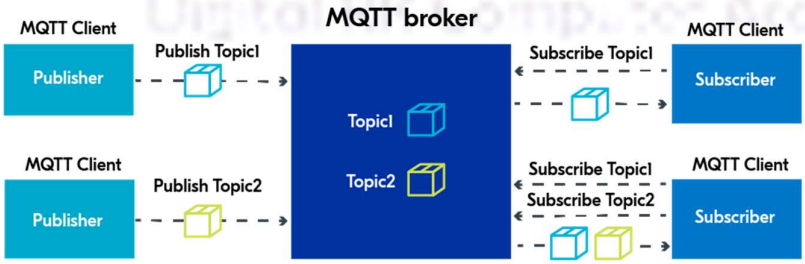
TCP **is used by HTTP, HTTPs, FTP, SMTP UDP is used by DNS, TFTP, SNMP, CoAP, and Telnet.**

Application Layer - Application layer protocols define how the applications interface with the lower layer protocols to send over the network….



एप्ली केशन लेयर प्रोटोकॉल यह परिभाषित करते हैं किनेटवर्क पर भेजने के लिए लोवर लेयर प्रोटोकॉल के साथ एप्ली केशनइंटर फेस कैसे होता है।

MQTT (Message Queuing Telemetry Transport): A lightweight protocol designed for efficient communication in constrained networks, suitable for low–power devices and unreliable connections.

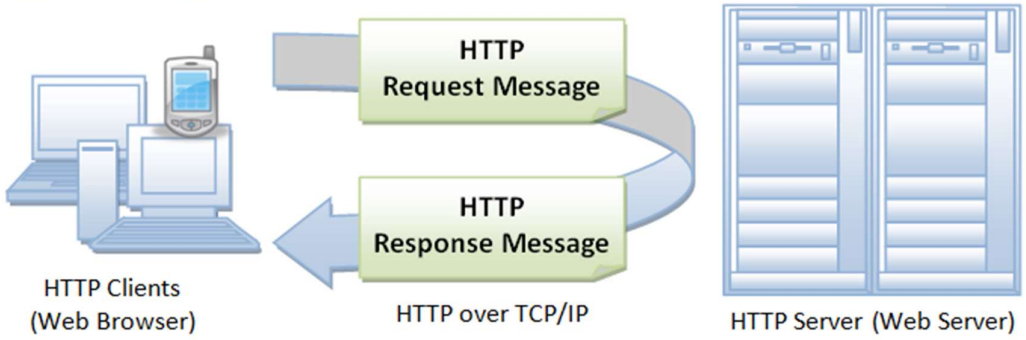


सीमित नेटवर्क में कुशल संचार के लिए डिज़ाइन किया गया एक हल्का प्रोटोकॉल, जोकम-शक्तिवाले उपकरणों और अविश्वसनीय कनेक्शन के लिए उपयुक्त है।

HTTP (Hypertext Transfer Protocol): A standard protocol used for communication between web browsers and servers, also employed in IoT for web–based interactions and data transfer.

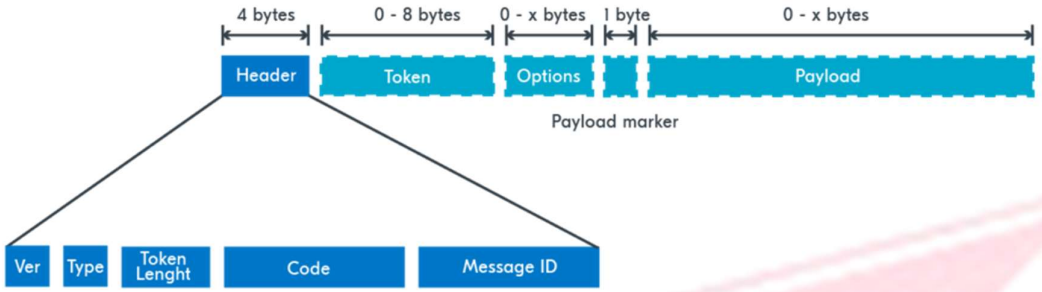
वेब ब्राउज़र और सर्वर के बीच संचार के लिए उपयोगकि याजाने वाला एक मानक प्रोटोकॉल , जिसे वेब-आधारित इंटरैक्शन और डेटाट्रांसफर के लिए IoT में भीनियोजित किया जाता है।

CoAP (Constrained Application Protocol): Designed for resource–c



onstrained devices, CoAP enables efficient communication and is often used in IoT applications

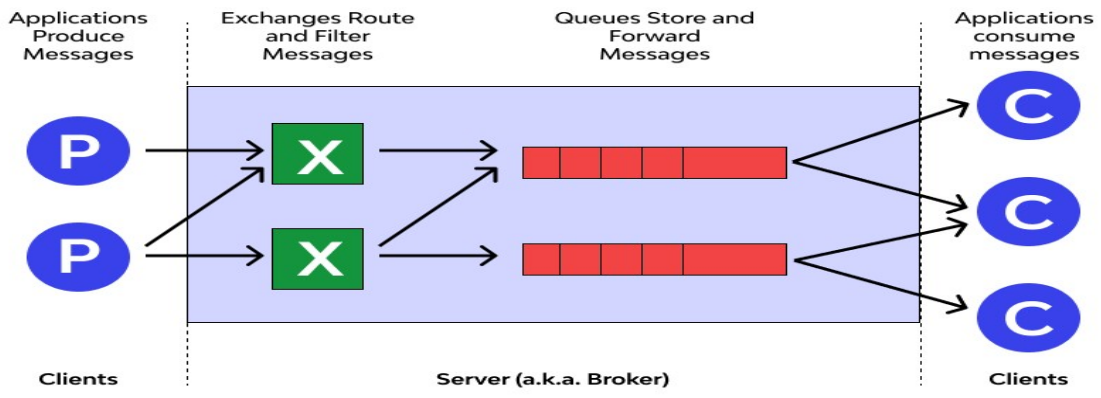
that require low power and low bandwidth.



संसाधन-बाधित उपकरणों के लिए डिज़ाइन किया गया, CoAP कुशल संचार सक्षम बनाता है और अक्सर IoT अनुप्रयोगों में उपयोग किया जाता है जिनके लिए कम बिजली और कम बैंडविड्थ की आवश्यकता होती है।

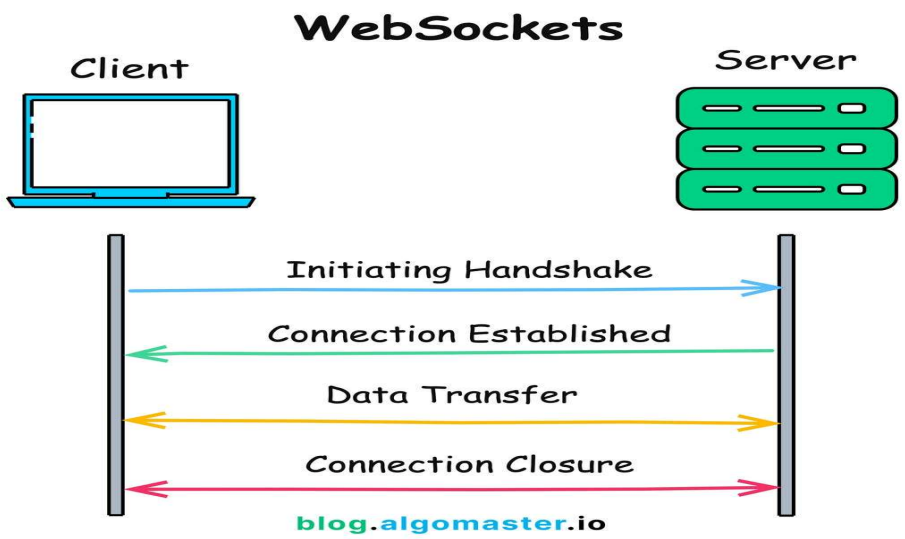
AMQP (Advanced Message Queuing Protocol):

A protocol for reliable messaging between devices and applications, capable of supporting complex messaging scenarios.



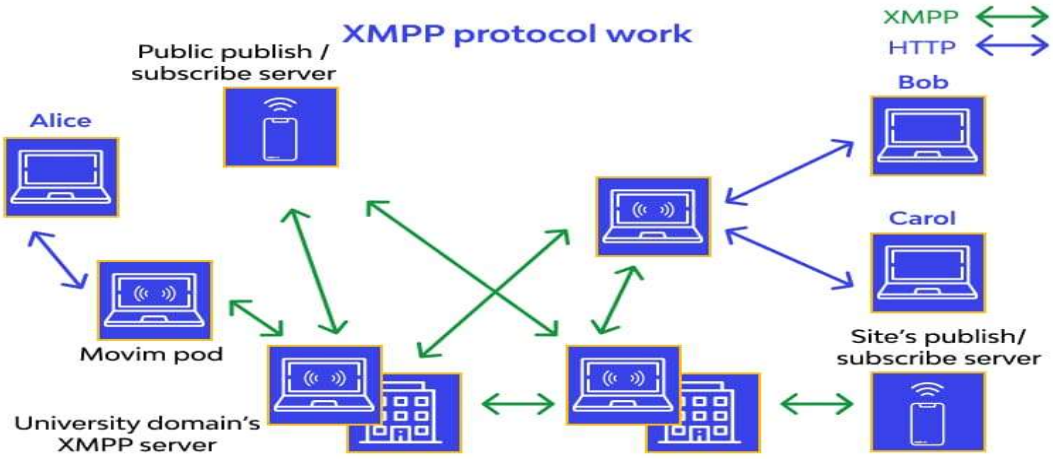
उपकरणों और अनुप्रयोगों के बीच विश्वसनीय संदेश भेजने के लिए एक प्रोटोकॉल, जो जटिल संदेश परिदृश्यों का समर्थन करने में सक्षम है।

WebSocket: A communication protocol that enables full–duplex communication over a single, long–lived connection, facilitating real–time data transfer between IoT devices and servers.



एक संचार प्रोटोकॉल जो एक, लंबे समय तक चलने वाले कनेक्शन पर पूर्ण-डुप्लेक्स संचार को सक्षम बनाता है, जिससे IoT उपकरणों और सर्वरों के बीच वास्तविक-समय डेटा स्थानांतरण की सुविधा मिलती है।

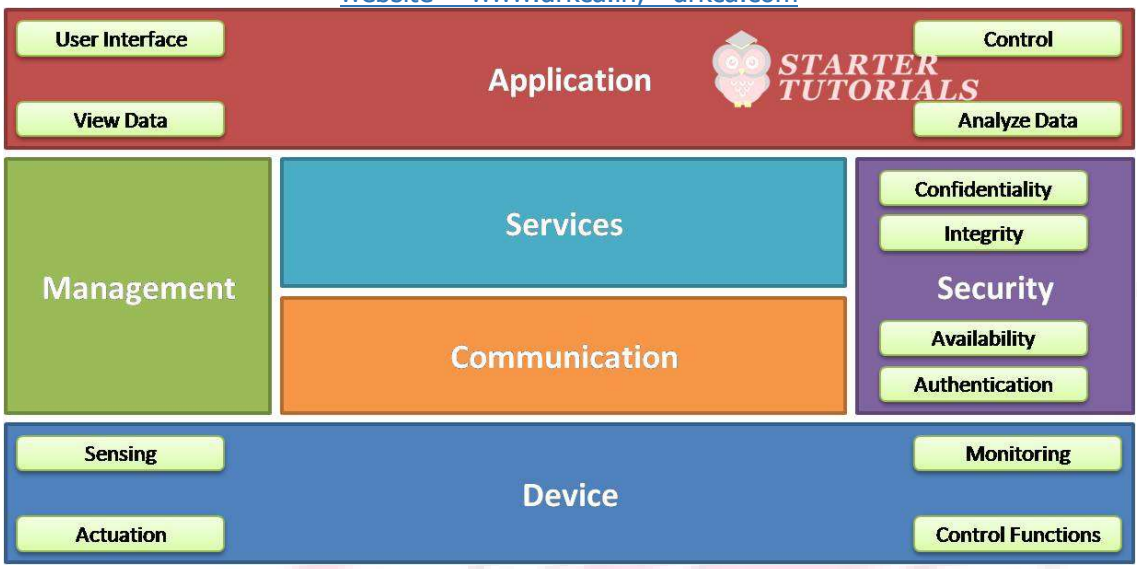
XMPP (Extensible Messaging and Presence Protocol): It is a communication protocal for message-oriented middleware based on XML (Extensible Markup Language). the protocol has been used also for publish-su systems, signaling for VoIP, video, file transfer, gaming, the Internet of Things (loT) applications such as the smart grid, and social networking services……..



यह एक्सएमएल(एक्सटेंसिबल मार्क अपलैंग्वेज) आधारित संदेश-उन्मुखमिड लवेयर के लिए एक संचार प्रोटोकॉल है ।प्रोटोकॉल का उपयोग पब्लिश-सब्सक्राइबसिस्टम, वीओआईपी, वीडियो, फाइलट्रागेमिंग, IoT एप्लिकेशन जैसे स्मार्ट ग्रिड और सोशलनेट वर्किंग सेवाओं के लिए भी किया जाता है।

The Logical Design of IoT

. Logical design of an management IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation. An IoT system comprises several functional blocks that provide the system the capabilities for identification, sensing, actuation, communication andIoT



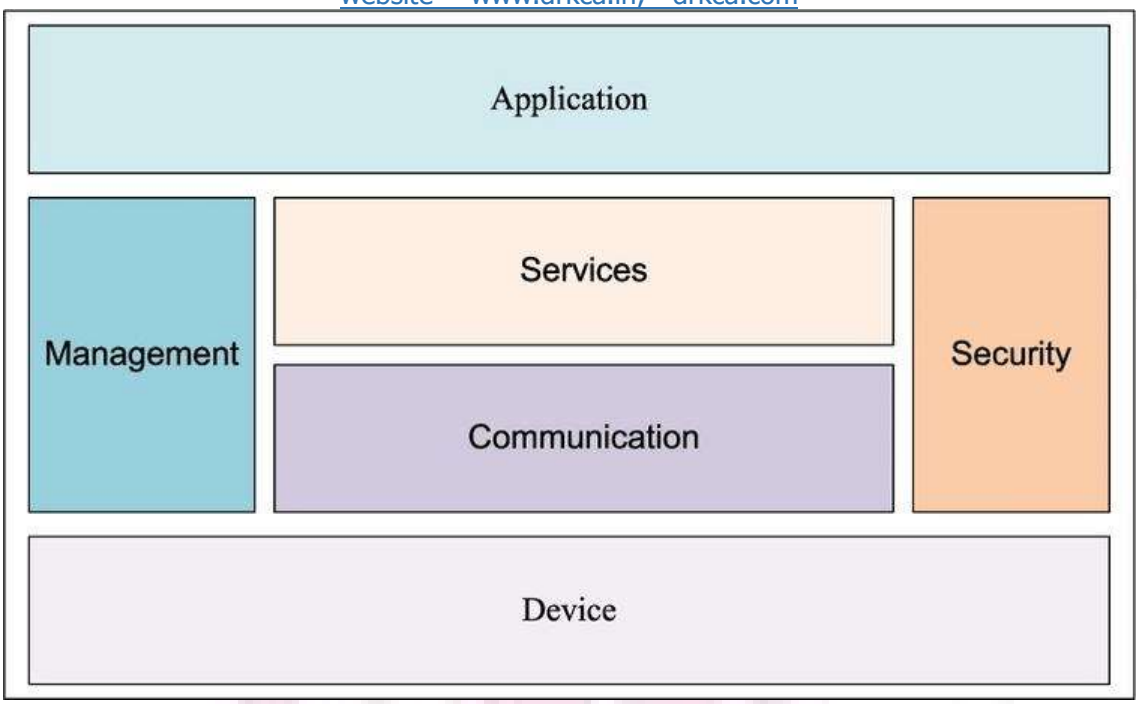
सिस्टम का लॉजिकल डिजाइन कार्यान्वयन के निम्न-स्तरीयबारी कियों में जाने के बिना संस्थाओं और प्रक्रियाओं के एक अमूर्तप्रतिनिधित्व को संदर्भित करता है।एक IoT सिस्टम में अनेक कार्यात्म कब्लॉक शामिल होते हैं जो सिस्टम की पहचान, सेन्सिंग, एक्चुवेशन, संचार और प्रबंधन के लिए क्षमताओं को प्रोवाइड करते हैं।

For understanding Logical Design of IoT, we consider the following terms: IoT केलॉजिकल डिजाइन को समझने केलिए, हम निम्नलिखित शब्दों परविचार करते हैं।

1. IoT Functional Blocks
2. IoT Communication Models
3. IoT Communication APIs

IoT Functional blocks

An IoT system comprises several functional blocks that provide the system capabilities of identification, sensing, actuation, communication, and management.



एक IoT सिस्टम में अनेक फंक्शनल ब्लॉक शामिल होते हैं जो पहचान, सेंसिंग, एक्चुवेशन, संचार और प्रबंधन की प्रणाली क्षमताओं को प्रदान करते हैं।

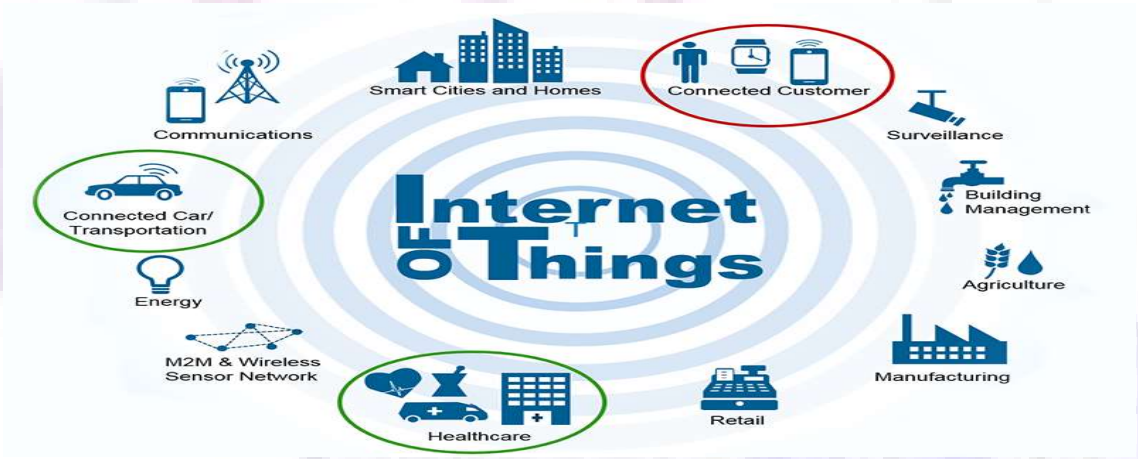
Device: An IoT system comprises devices that provide sensing, monitoring, actuation, and control functions.

एक IoT सिस्टम में ऐसे डिवाइस शामिल होते हैं जोसेंसिंग, मॉनिटरिंग और कंट्रोल फंक्शन प्रदान करते हैं।

Communication: Handles the communication for the IoT system.

IoT सिस्टम के लिए संचार को संभालता है।

Services: Services for device monitoring, device control service, data publishing services, and services for device discovery.

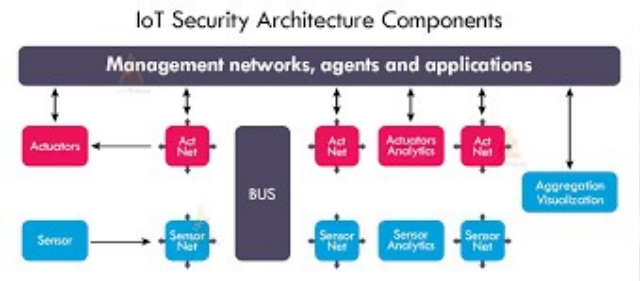


डिवाइस की निगरानी के लिए सेवाएं, डिवाइस नियंत्रण सेवा, डेटा पब्लिशिंग सेवाएं और डिवाइस खोज की सेवाएं।

Management: This block provides various functions to govern the IoT system.

यह ब्लॉक IoT सिस्टम को संचालित करने के लिए विभिन्न कार्य प्रदान करता है।

Security: This block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.



यह ब्लॉक IoT सिस्टम को सुरक्षित करता है और authentication, authorization, संदेश और कंटेंट की विश्वसनीयता, और डेटा सुरक्षा जैसे कार्य प्रदान करता है।

Application: This is an interface that the users can use to control and monitor various aspects of the loT system. Applications also allow users to view the system status and view or analyze the processed data.

यह एक इंटरफेस है जिसका उपयोग उपयोग कर्ता IoT सिस्टम के विभिन्न पहलुओं को नियंत्रित और मॉनिटर करने के लिए कर सकते हैं । एप्लिकेशन उपयोगकर्ताओं को सिस्टम की स्थिति देखने और प्रोसेस्ड डेटा को देखने याविश्लेषण करने की अनुमति भी देते हैं।

Communication Models – कम्यूनिकेशनमॉडल

Communication models are listed below कम्यूनिकेशन मॉडल नी चेसू चीबद्ध हैं:

(i) **Request-Response Communication** Model
Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.

रिक्वेस्ट-रिस्पॉन्स एक संचार मॉडल है जिसमें ग्राहक सर्वर को रिक्वेस्ट भेजता है और सर्वररिक्वे स्ट्सका जवाब देता है।जब सर्वर एक रिक्वेस्ट प्राप्त करता है, तो यह तय करता है कि कैसे प्रतिक्रिया दी जाए, डेटा प्राप्त किया जाए, रिसोर्सरि प्रेसेन्टेशन को पुनःप्राप्त किया जाए, प्रतिक्रिया तैयार की जाए और फिर ग्राहक को प्रति क्रिया भेजी जाए।

Publish-Subscribe Communication Model पब्लिश-सब्सक्राइबकम्यूनिकेशनमॉडल Publish-Subscribe is a communication model that involves publishers, brokers, and consumers.

पब्लिश-सब्सक्राइब एक संचार मॉडल है जिस में प्रकाशक, ब्रोकर और उपभोक्ता शामिल होते हैं।

Push -Pull Communication Model पुश-पुलकम्यूनिकेशनमॉडल Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers….

पुश-पुल एक कम्यूनिकेशन मॉडल है जिसमें डेटा निर्माता डेटा को क्यू में push करते हैं और उपभोक्ता क्यू से डेटा खींचते(pull) हैं।निर्माता को उपभोक्ताओं को जागरूक करने की आवश्यकता नहीं है।

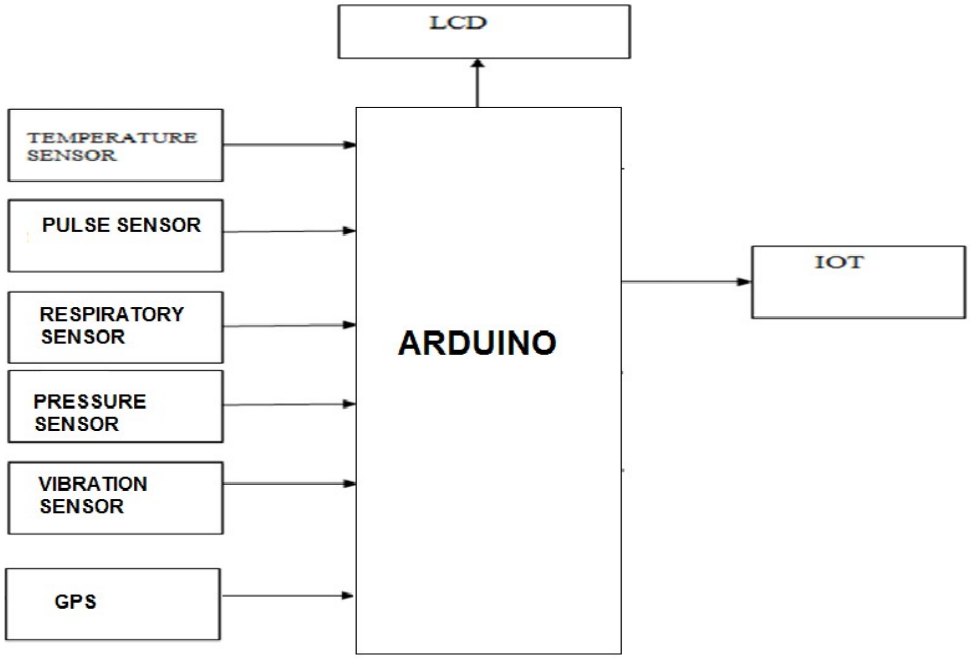
a. Exclusive Pair Communication Model एक्सक्ल्यूसिवपेयरकम्यूनिकेशनमॉडल Exclusive Pair is bidirectional, full duplex communication model that uses a persistent connection between the client and the server.

एक्सक्ल्यूसिव पेयर एक bidirectional, fully duplex संचार मॉडल है जोक्लाइंट और सर्वर के बीच एक स्थायी कनेक्शन का उपयोग करता है।

Development Tools used in IoT

IoT में उपयोग किए जाने वाले उपकरण

1. **Arduino** - आर्डयूनो Arduino is an Italian open-source hardware and software company, projects and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.



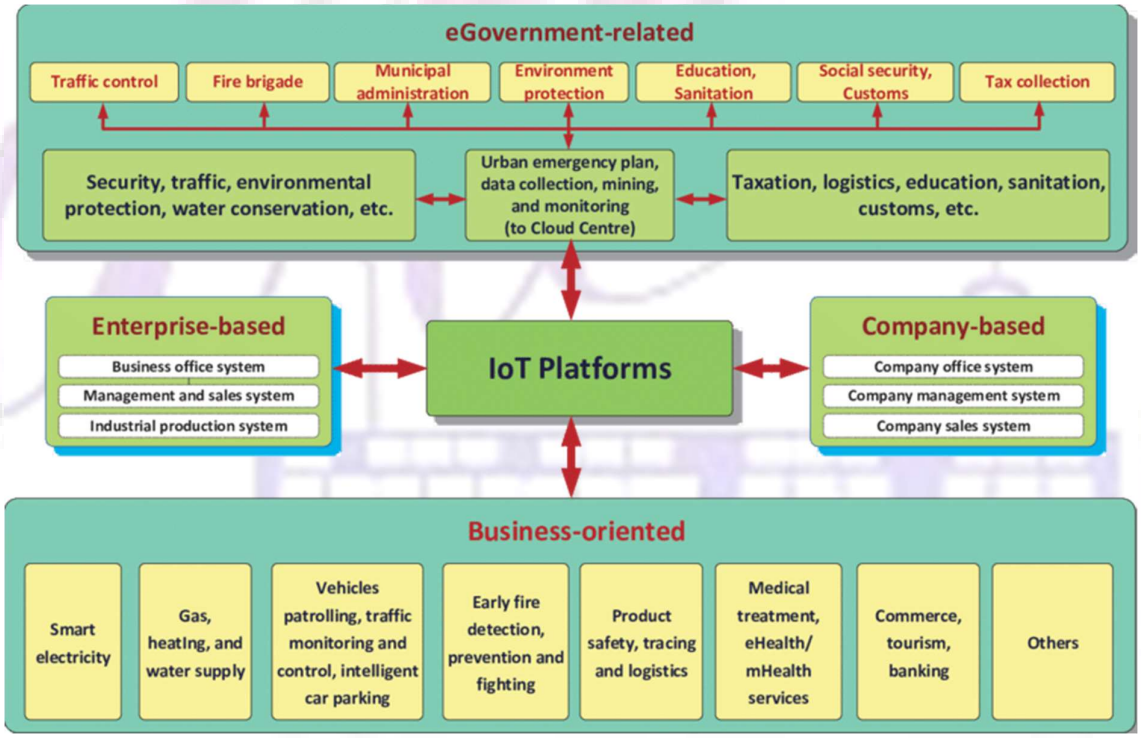
Arduino एक इतालवी ओपन-सोर्स हार्ड वेयर और सॉफ्टवेयर कंपनी, प्रोजेक्ट और उपयोगकर्ता समुदा य है जो डिजिटल उपकरणों के निर्माण के लिए सिंगल-बोर्ड माइक्रो कंट्रोलर और माइक्रो कंट्रोलर कि डिजाइन और निर्माण करता है।

2. **Eclipse IoT** - ईक्लिप्सआईओटी Eclipse IoT is an ecosystem of entities (industry and academia) working together to create a foundation for IoT based exclusively on open source technologies. Their focus

remains in the areas of producing open source implementations of IoT standard technology.

एक्लिप्स IoT संस्थाओं (उद्योगऔरशिक्षाजगत) का एक पारिस्थिति की तंत्र है जो विशेष रूप से ओपन सोर्स प्रौद्योगिकियों पर आधारित IoT के लिए एक आधार बनाने के लिए मिलकर काम कर रहा है।उनका ध्यान IoT मानक प्रौद्योगिकी के खुले स्रोत कार्यान्वयन के क्षेत्र में रहता है।

3. **PlatformIOT** It is a cross-platform IoT development environment. This platform comes with a build system, supported by a library manager and IDE…..



यह एक क्रॉस-प्लेटफॉर्म IoT विकास एनवायरनमेंट है। यह प्लेटफॉर्म एक बिल्डसिस्टम के साथ आता है, जो एकलाइब्रेरी मैनेजर और आईडीई द्वारा समर्थित है।

4. **IBM Watson** IBM Watson, an API that allows you to attach a host of cognitive computing features to your applications. This is an innovative tool that can also be used to predict the future.

आईबीएम वाटसन, एक एपीआई जो आपको अपने loT ऐप्लिकेशनो में संज्ञानात्मक कंप्यूटिंग सुविधाओं के एक होस्ट को संलग्न करने की अनुमति देता है।यह एक इनोवेटिव उपकरण है जिसका उपयोग भविष्य की भविष्यवाणी करने के लिए भी किया जा सकता है।

5. **Raspbian** Raspberry Pi OS is a Unix-like operating system based on the Debian GNU/Linux distribution for the Raspberry Pi family of compact single-board computers. First developed independently in 2012, it has been produced as the primary operating system for these boards since 2013, distributed by the Raspberry Pi Foundation

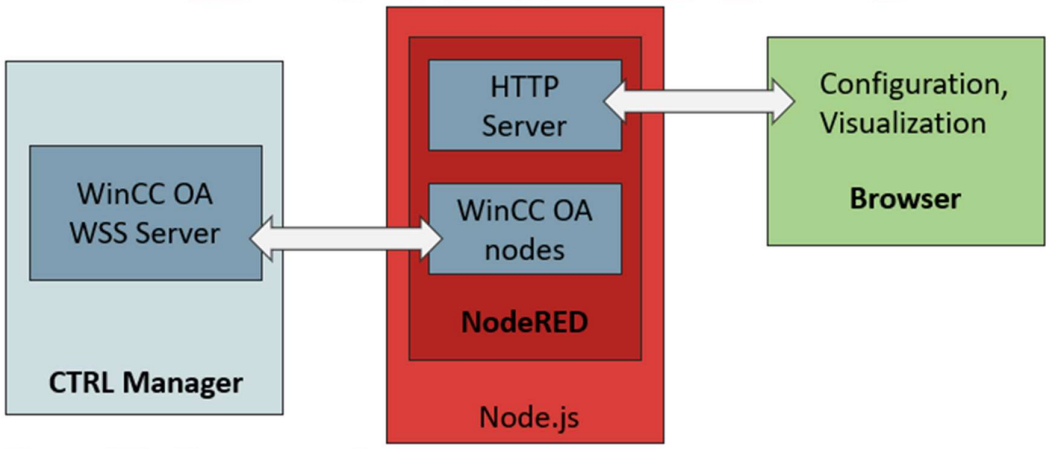
रास्प बेरी पाईओ एस एक यूनिक्स जैसा ऑपरेटिंग सिस्टम है जो कॉम्पैक्ट सिंगल-बोर्ड कंप्यूटरों के रास्पबेरी पाईप रिवार के लिए डेबियन जी एनयू/लिनक्सवितरण पर आधारित है।पहलीबार 2012 में स्वतंत्र रूप से विकसित किया गया था, इसे 2013 से इन बोर्डों के लिए प्राथमिक ऑपरेटिंग सिस्टम के रूप में उत्पादित किया गया है, जिसे रास्पबेरी पाई फाउंडेशन द्वारा वितरित किया गया है।

6. OpenSCADA (Supervisory Control And Data Acquisition) This tool is a part of the SCADA project by Eclipse IoT industry groups. It is Independent to any platform and is known for its security and flexibility along with a modern design. OpenSCADA also supports editing and debugging and comes with front-end applications, back-end applications, libraries, configuration tools and interface applications. Its different tools can be combined with the development of advanced loT applications. Unlike other IDEs, OpenSCADA supports various programming languages and consists of sub-projects like Atlantis, Utgard, Orilla and 0thers….

यह टूल एक्लिप्स IoT उद्योग समूहों द्वारा SCADA प्रोजेक्ट का एक हिस्सा है।यह किसी भी मंच के लिए स्वतंत्र है और आधुनिक डिजाइन के साथ अपनी सुरक्षा और लचीलेपन के लिए जाना जाता है। Open SCADA संपादन और डिबगिंग का भी समर्थन करता है और फ्रंट-एंड एप्लिकेशन, बैक-एंड एप्लिकेशन, लाइब्रेरी, कॉन्फ़िगरेशन टूल और इंटरफ़ेस एप्लिकेशन के साथ आता है।इसके विभिन्न उपकरणों को उन्नत एल ओटी अनुप्रयोगों के विकास के साथ जोड़ा जा सकता है।अन्य IDE के विपरीत , Open SCADA विभिन्न प्रोग्रामिंग भाषाओं का समर्थन

करता है और इसमें अटलांटिस , Ut gard, Orilla और 0thers जैसी उप-परियोजनाएँ शामिल हैं।

7. **Node-RED** Node-RED is a simple visual tool that is built on Node.js, a server-side JavaScript platform that is widely used in IoT projects. It is an open-source tool mainly used to connect devices, services, and APIs together with an integrated browser-based flow editor. With over 60,000modules, it was developed by IBM with the aim of providing a user-friendly interface for Node-RED developers allowing them to connect devices very quickly and easily.



नोड-रेड एक सरल दृश्य उपकरण है जिसे Node .js पर बनाया गया है, जो एक सर्वर-साइड जावास्क्रिप्ट प्लेटफॉर्म है जो व्यापक रूप से IoT के प्रोजेक्टो में उपयोग किया जाता है।यह एक ओपन-सोर्सटूल है जिसका उपयोग मुख्य रूप से उपकरणों, सेवाओं और एपीआई को एक एकीकृत ब्राउजर-आधारित फ्लो एडिटर के साथ जोड़ने के लिए किया जाता है।60,000 से अधिक मॉड्यूल के साथ, यह आईबीएम द्वारा डेवल पर्स के लिए उपयोगकर्ता के अनुकूलइंटरफेस प्रदान करने के उद्देश्य से

विकसित किया गया था, जिससे उन्हें बहुत जल्दी और आसानी से डिवाइस कनेक्ट करने की अनुमति मिलती है।

8. **Device Hive** Device Hive is a free open source machine to machine (M2M) communication framework which was launched in 2012. It is a Data Art's Alljoyn based device and is considered one of the most preferred IoT app development platforms.

डिवाइस हाइव एक मुफ्त ओपन सोर्स मशीन टूमशीन(M2M) संचार फ्रेम वर्क है जिसे 2012 में लॉन्च किया गया था।यह डेटा आर्ट का Alljoyn आधारित डिवाइस है और इसे सबसे पसंदीदा IoT ऐप डेवलपमेंट प्लेटफॉर्म में से एक माना जाता है।

(नोट: PDF में कुछ स्थानों पर "……." या ट्रंकेटेड हिस्से हैं, जो मूल में हैं। इमेज/डायग्राम जहां हैं, वहां मैंने नोट किया है क्योंकि यह टेक्स्ट ट्रांसक्रिप्शन है। PDF में कई इमेज हैं जैसे

डायग्राम, चार्ट आदि, लेकिन टेक्स्ट आधारित उत्तर में उन्हें शामिल नहीं किया।)

Simplified Version

यह PDF O Level M4 कोर्स के अध्याय 1 (क्लास 2) का है, जो IoT डिजाइन मेथडोलॉजी, फिजिकल डिजाइन, प्रोटोकॉल्स, लॉजिकल डिजाइन, कम्युनिकेशन मॉडल्स और डेवलपमेंट टूल्स पर है। मैंने मुश्किल शब्दों को आसान बना दिया है (जैसे "actuating" को "क्रिया करना", "authentication" को "पहचान सत्यापन" आदि), लेकिन मूल अर्थ वही रखा है। मूल में अंग्रेजी और हिंदी दोनों हैं, इसलिए दोनों को सरल बनाया।

IoT Design Methodology

IoT डिजाइन मेथडोलॉजी में ये स्टेप्स शामिल हैं:

STEP 1: उद्देश्य और आवश्यकताएं निर्धारित करना (Purpose and Requirements Specification) पहला कदम सिस्टम के उद्देश्य और जरूरतों को परिभाषित करना है। इसमें सिस्टम का उद्देश्य, व्यवहार और जरूरतें कैद की जाती हैं।

- डेटा इकट्ठा करने की जरूरतें (Data collection requirements)
- सिस्टम प्रबंधन की जरूरतें (System management requirements)
- सुरक्षा की जरूरतें (Security requirements)
- यूजर इंटरफेस की जरूरतें (User interface requirements)

STEP 2: प्रक्रिया निर्धारण (Process Specification) IoT सिस्टम के उपयोग मामलों को उद्देश्य और जरूरतों के आधार पर औपचारिक रूप से वर्णित किया जाता है।

STEP 3: डोमेन मॉडल निर्धारण (Domain Model Specification) डोमेन मॉडल IoT सिस्टम के मुख्य अवधारणाओं, संस्थाओं और वस्तुओं का वर्णन करता है। यह वस्तुओं की विशेषताएं और उनके बीच संबंध परिभाषित करता है। यह किसी खास तकनीक या प्लेटफॉर्म से स्वतंत्र है। (जैसे: भौतिक संस्था, आभासी संस्था, डिवाइस, संसाधन, सेवा)

STEP 4: सूचना मॉडल निर्धारण (Information Model Specification) सूचना मॉडल IoT सिस्टम में सभी सूचनाओं की संरचना परिभाषित करता है।

STEP 5: सेवा निर्धारण (Service Specifications) सेवा निर्धारण IoT सिस्टम में सेवाओं को परिभाषित करता है जैसे सेवा प्रकार, इनपुट/आउटपुट, सेवा अंतबिंदु, सेवा समयसारणी, सेवा पूर्वशर्तें और सेवा प्रभाव।

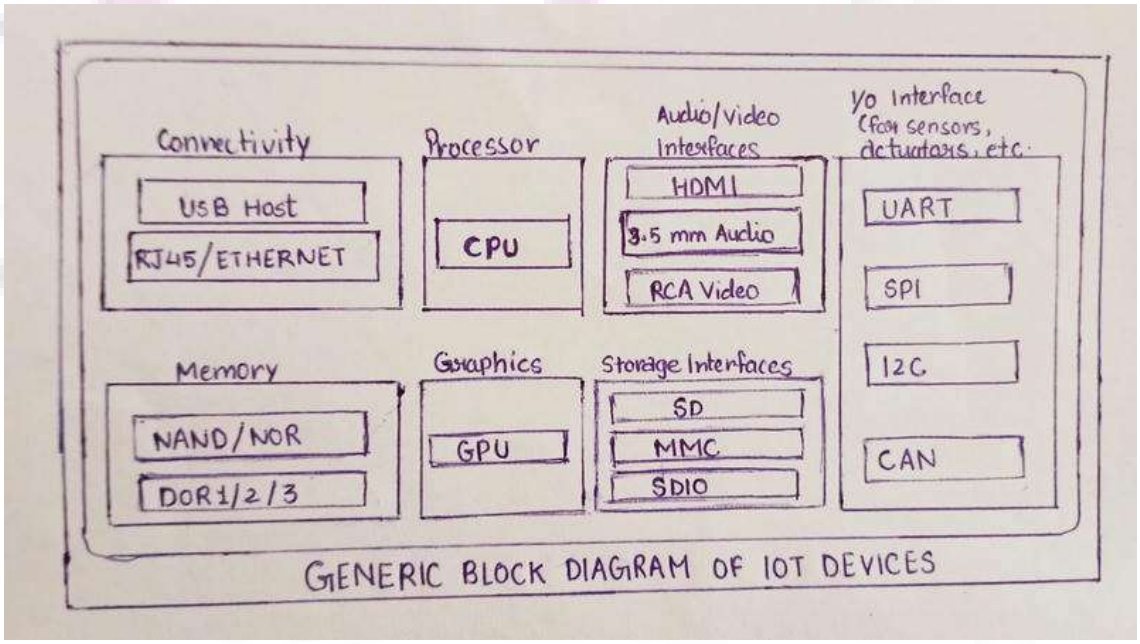
STEP 6: IoT स्तर निर्धारण (IoT Level Specification) इसमें सिस्टम के लिए IoT स्तर परिभाषित करें। विभिन्न स्थितियों के अनुसार 5 प्रकार के IoT परिनियोजन स्तर उपयोग किए जाते हैं।

The Physical Design of IoT

IoT का फिजिकल डिजाइन IoT में वस्तुओं और प्रोटोकॉल्स को संदर्भित करता है। वस्तुएं नोड डिवाइस हैं जिनकी विशिष्ट पहचान होती है और दूर से सेंसिंग (महसूस करना), क्रियाकरण और निगरानी

कर सकती हैं। IoT प्रोटोकॉल वस्तुओं और क्लाउड सर्वर के बीच इंटरनेट पर संचार स्थापित करने में मदद करते हैं।

1. IoT में वस्तुएं (Things in IoT): IoT डिवाइस जिनकी विशिष्ट पहचान होती है जो सेंसिंग, क्रियाकरण और निगरानी कर सकते हैं। ये विभिन्न प्रकार के हो सकते हैं जैसे सेंसिंग डिवाइस, स्मार्ट वॉच, स्मार्ट इलेक्ट्रॉनिक्स उपकरण, पहनने योग्य सेंसर, कारें और औद्योगिक मशीनें।
2. IoT प्रोटोकॉल्स (IoT Protocols): ये IoT डिवाइस और क्लाउड सर्वर के बीच इंटरनेट पर संचार स्थापित करने में मदद करते हैं। ये IoT डिवाइस को कमांड भेजने और डेटा प्राप्त करने में मदद करते हैं



लिंक लेयर (Link Layer): लिंक लेयर प्रोटोकॉल तय करते हैं कि डेटा नेटवर्क के भौतिक लेयर (वायर्ड या वायरलेस) पर कैसे भेजा जाता है।

यह तय करता है कि पैकेट हार्डवेयर डिवाइस द्वारा कैसे कोडित और संकेतित किए जाते हैं।

वायरलेस कनेक्टिविटी (Wireless Connectivity):

- Wi-Fi: व्यापक उपयोग वाली वायरलेस नेटवर्किंग तकनीक जो छोटी से मध्यम दूरी पर तेज डेटा ट्रांसफर सक्षम बनाती है।
- ब्लूटूथ (Bluetooth): छोटी दूरी की वायरलेस तकनीक जो निकट डिवाइस जोड़ने के लिए उपयोग होती है। IoT डिवाइस के लिए जहां कम बिजली और रुक-रुक कर डेटा ट्रांसफर चाहिए, जैसे पहनने योग्य डिवाइस और घरेलू ऑटोमेशन।
- जिगबी (Zigbee): कम शक्ति, कम डेटा दर वाला वायरलेस संचार प्रोटोकॉल जो कम बिजली वाली जरूरतों और ज्यादा डिवाइस वाले अनुप्रयोगों के लिए डिजाइन किया गया। घरेलू ऑटोमेशन, स्मार्ट लाइटिंग और औद्योगिक में उपयोग।
- LPWAN (Low-Power Wide Area Network): जैसे LoRaWAN और NB-IoT, कम शक्ति के साथ लंबी दूरी कनेक्टिविटी देते हैं। स्मार्ट सिटी और कृषि निगरानी के लिए उपयुक्त।

वायर्ड कनेक्टिविटी (Wired Connectivity):

- ईथरनेट (Ethernet): मानक वायर्ड नेटवर्किंग तकनीक जो स्थानीय नेटवर्क (LAN) पर विश्वसनीय और तेज डेटा ट्रांसफर देती है। औद्योगिक सेटिंग्स और उच्च बैंडविड्थ वाले डिवाइस के लिए।

- पावरलाइन कम्युनिकेशन (Powerline Communication): मौजूदा बिजली लाइनों पर डेटा ट्रांसमिशन, अतिरिक्त वायरिंग की जरूरत नहीं।

नेटवर्क/इंटरनेट लेयर (Network/Internet Layer): यह सोर्स से डेस्टिनेशन नेटवर्क तक IP डेटाग्राम भेजने के लिए जिम्मेदार। होस्ट पहचान IPV4 या IPV6 जैसी पदानुक्रमित IP एड्रेसिंग से होती है।

ट्रांसपोर्ट लेयर (Transport Layer): यह अंतर्निहित नेटवर्क से स्वतंत्र एंड-टू-एंड संदेश ट्रांसफर देता है। इसमें त्रुटि नियंत्रण, विभाजन, प्रवाह नियंत्रण और भीड़ नियंत्रण जैसे कार्य।

TCP और UDP में अंतर (Difference between TCP and UDP):

- TCP: कनेक्शन-ओरिएंटेड, विश्वसनीय, डेटा डिलीवरी गारंटी, धीमा, त्रुटि चेक और रिकवरी, हैवीवेट। HTTP, HTTPS, FTP, SMTP, Telnet में उपयोग।
- UDP: कनेक्शनलेस, डिलीवरी गारंटी नहीं, तेज, सरल, कुशल, त्रुटि चेक लेकिन पैकेट डिस्कार्ड, लाइटवेट लेकिन कम विश्वसनीय। DNS, TFTP, SNMP, CoAP में उपयोग।

एप्लीकेशन लेयर (Application Layer): एप्लीकेशन लेयर प्रोटोकॉल तय करते हैं कि एप्लीकेशन लोवर लेयर प्रोटोकॉल के साथ कैसे इंटरफेस करते हैं।

- MQTT: सीमित नेटवर्क में कुशल संचार के लिए हल्का प्रोटोकॉल, कम शक्ति वाले डिवाइस और अविश्वसनीय कनेक्शन के लिए।

- HTTP: वेब ब्राउजर और सर्वर के बीच संचार के लिए मानक, IoT में वेब-आधारित इंटरैक्शन और डेटा ट्रांसफर के लिए।
- CoAP: संसाधन-बाधित डिवाइस के लिए, कुशल संचार, कम शक्ति और बैंडविड्थ वाले IoT अनुप्रयोगों में।
- AMQP: डिवाइस और अनुप्रयोगों के बीच विश्वसनीय संदेश, जटिल संदेश परिदृश्यों के लिए।
- WebSocket: एक लंबे कनेक्शन पर पूर्ण-डुप्लेक्स संचार, IoT डिवाइस और सर्वर के बीच रियल-टाइम डेटा ट्रांसफर।
- XMPP: XML आधारित संदेश मिडलवेयर के लिए संचार प्रोटोकॉल, पब्लिश-सब्सक्राइब, VoIP, वीडियो, फाइल ट्रांसफर, गेमिंग, स्मार्ट ग्रिड, सोशल नेटवर्किंग में उपयोग।

The Logical Design of IoT

IoT सिस्टम की लॉजिकल डिजाइन संस्थाओं और प्रक्रियाओं का अमूर्त प्रतिनिधित्व है बिना कार्यान्वयन की बारीकियों के। यह पहचान, सेंसिंग, क्रियाकरण, संचार और प्रबंधन की क्षमताएं देता है।

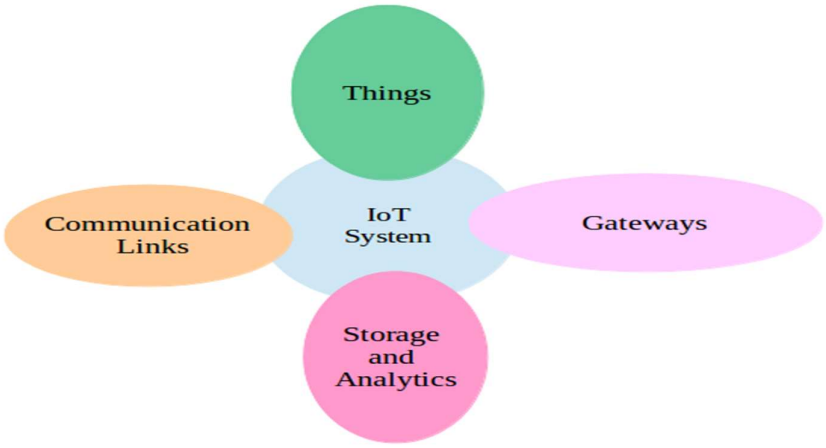
IoT को समझने के लिए:

1. IoT फंक्शनल ब्लॉक्स (Functional Blocks)
2. IoT कम्युनिकेशन मॉडल्स (Communication Models)
3. IoT कम्युनिकेशन APIs

IoT फंक्शनल ब्लॉक्स:

- डिवाइस: सेंसिंग, मॉनिटरिंग, क्रियाकरण और नियंत्रण प्रदान करते हैं।

- कम्युनिकेशन: IoT सिस्टम के संचार को संभालता है।



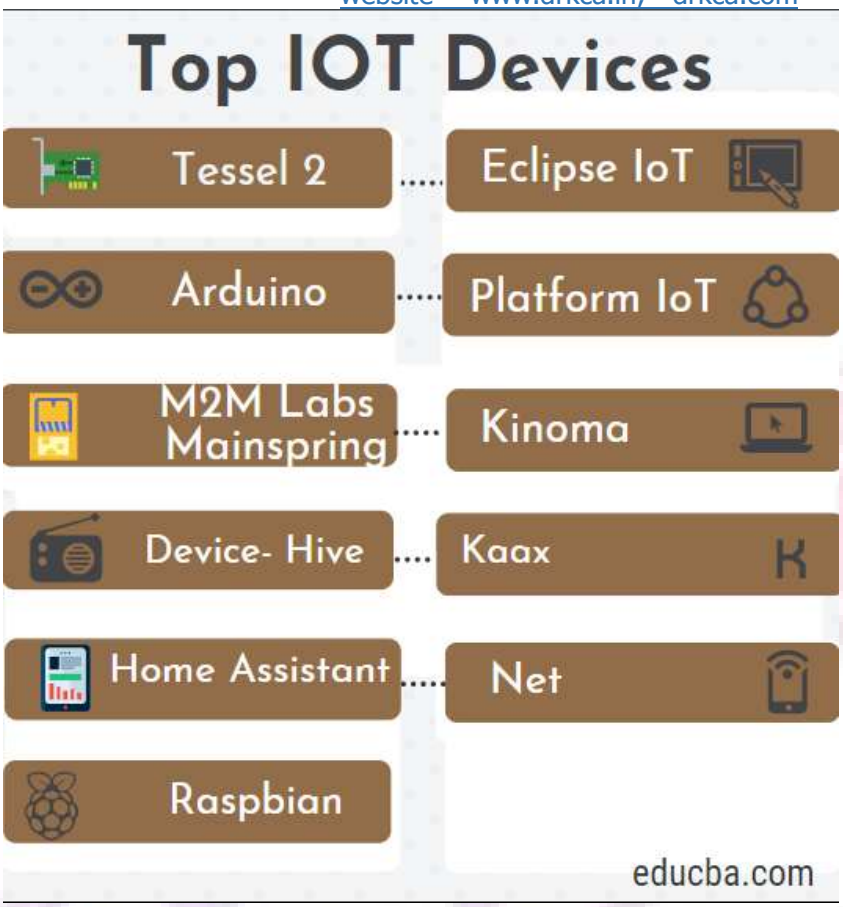
- सेवाएं: डिवाइस मॉनिटरिंग, नियंत्रण, डेटा पब्लिशिंग और डिवाइस खोज की सेवाएं।
- प्रबंधन: IoT सिस्टम को संचालित करने के कार्य।
- सुरक्षा: प्रमाणीकरण (authentication), प्राधिकरण (authorization), संदेश अखंडता और डेटा सुरक्षा।
- एप्लीकेशन: यूजर को सिस्टम नियंत्रित और मॉनिटर करने का इंटरफेस, स्थिति देखने और डेटा विश्लेषण।

कम्युनिकेशन मॉडल्स:

- रिक्वेस्ट-रिस्पॉन्स: क्लाइंट सर्वर को रिक्वेस्ट भेजता है, सर्वर जवाब देता है।
- पब्लिश-सब्सक्राइब: प्रकाशक, ब्रोकर और उपभोक्ता शामिल।
- पुश-पुल: डेटा उत्पादक क्यू में पुश करते हैं, उपभोक्ता पुल करते हैं।
- एक्सक्ल्यूसिव पेयर: द्विदिश, पूर्ण डुप्लेक्स, क्लाइंट-सर्वर के बीच स्थायी कनेक्शन।

Development Tools used in IoT

1. आर्डयूनो (Arduino): इटालियन ओपन-सोर्स कंपनी जो सिंगल-बोर्ड माइक्रोकंट्रोलर बनाती है डिजिटल डिवाइस बनाने के लिए।
2. ईक्लिप्स IoT (Eclipse IoT): संस्थाओं का पारिस्थितिकीतंत्र जो ओपन-सोर्स पर आधारित IoT फाउंडेशन बनाता है। मानक तकनीक के ओपन-सोर्स कार्यान्वयन पर फोकस।
3. प्लेटफॉर्मIO (PlatformIO): क्रॉस-प्लेटफॉर्म IoT विकास वातावरण, बिल्ड सिस्टम, लाइब्रेरी मैनेजर और IDE के साथ।
4. IBM वाटसन (IBM Watson): API जो संज्ञानात्मक कंप्यूटिंग फीचर्स को एप्लीकेशन में जोड़ता है, भविष्य भविष्यवाणी के लिए।
5. रास्पबियन (Raspbian): रास्पबेरी पाई के लिए डेबियन आधारित यूनिक्स जैसा OS, 2012 में विकसित।
6. ओपनSCADA (OpenSCADA): एक्लिप्स IoT का SCADA प्रोजेक्ट भाग, प्लेटफॉर्म स्वतंत्र, सुरक्षित, लचीला। संपादन, डिबगिंग, फ्रंट-एंड, बैक-एंड, लाइब्रेरी आदि सपोर्ट। विभिन्न भाषाएं और सब-प्रोजेक्ट्स जैसे Atlantis, Utgard।
7. नोड-RED (Node-RED): Node.js पर आधारित सरल दृश्य टूल, डिवाइस, सेवाएं और APIs जोड़ने के लिए। IBM द्वारा विकसित, यूजर फ्रेंडली इंटरफेस।



8. डिवाइस हाइव (Device Hive): फ्री ओपन-सोर्स M2M फ्रेमवर्क, 2012 में लॉन्च, Data Art का Alljoyn आधारित, पसंदीदा IoT ऐप डेवलपमेंट प्लेटफॉर्म।

यह सरल संस्करण मूल सामग्री को आसान शब्दों में समझाता है (मुश्किल शब्द जैसे "actuating" को "क्रिया करना", "duplex" को "द्विदिश" आदि), लेकिन कोई बदलाव नहीं किया। अगर और स्पष्ट करने की जरूरत हो, बताएं!

CHAPTER 2

THINGS AND CONNECTIONS

Control System- A control system is a system that manages, regulates, and controls the behavior of other devices or systems. It uses input from sensors, processes the information, and sends commands to actuators to achieve a desired output.

कंट्रोल सिस्टम वह प्रणाली है जो अन्य उपकरणों या सिस्टम के व्यवहार को नियंत्रित और प्रबंधित करती है। यह सेंसर से इनपुट लेता है, जानकारी प्रोसेस करता है, और वांछित आउटपुट प्राप्त करने के लिए एक्ट्यूएटर को निर्देश भेजता है।

---

2. Components of a Control System

1. Sensor (सेंसर)

Measures physical quantities like temperature, moisture, pressure, etc.

→ तापमान, नमी, दबाव जैसे मानों को मापता है।

---

2. Controller (कंट्रोलर)

Processes the sensor data and makes decisions.
Examples: Microcontroller, ESP32, Arduino, Raspberry Pi.

→ सेंसर के डेटा को प्रोसेस कर निर्णय लेता है।

---

3. Actuator (एक्ट्यूएटर)

Carries out the action such as turning ON/OFF motor, opening valve, etc.

→ कार्रवाई करता है, जैसे मोटर चालू/बंद करना।

4. Feedback (फीडबैक)

Output is measured again and sent back to controller for correction.

→ आउटपुट को दोबारा मापा जाता है और सुधार के लिए कंट्रोलर को भेजा जाता है।

5. Set Point (सेट प्वाइंट)

Desired output value (example: 25°C in AC).

→ वांछित आउटपुट मान (जैसे AC को 25°C पर सेट करना)।

3. Types of Control Systems

A. Open-Loop Control System

An open-loop system has no feedback. It performs action based on predefined settings. Output is not compared with input.

ओपन-लूप सिस्टम में फीडबैक नहीं होता। यह पहले से तय सेटिंग के अनुसार काम करता है।
आउटपुट की तुलना इनपुट से नहीं की जाती।

Examples / उदाहरण:

- Washing machine (time-based)
- Microwave oven
- Electric iron (older models)
- Street lights using timers

Advantages (फायदे):

- Simple design
- Low cost
- Easy maintenance

Disadvantages (नुकसान):

- No error correction
- Less accuracy
- Output changes with disturbance

B. Closed-Loop (Feedback) Control System

A closed-loop system uses feedback.
It continuously compares output with set point and corrects errors automatically.

क्लोज्ड-लूप सिस्टम फीडबैक का उपयोग करता है।
यह आउटपुट की तुलना सेट प्वाइंट से करता है और त्रुटियों को स्वयं सुधारता है।

Examples / उदाहरण:

- Air conditioner with thermostat
- Smart irrigation system
- Automatic room heater
- Self-driving cars

Advantages (फायदे):

- High accuracy
- Automatic error correction
- Stable system

Disadvantages (नुकसान):

- Complex design

- High cost
- Requires more power

---

4. Control Actions / Control Algorithms

---

1. ON/OFF Control

Turns device ON or OFF based on threshold.
Used in simple systems.

थ्रेशहोल्ड के आधार पर डिवाइस को ON या OFF करता है।
सरल सिस्टम में उपयोग।

Example:

- Thermostat (AC, heater)

---

2. Proportional Control (P Control)

Output is proportional to error.
If error increases → control action increases.

आउटपुट त्रुटि के अनुपात में होता है।
त्रुटि बढ़े → नियंत्रण क्रिया बढ़े।

---

3. PID Control (Proportional + Integral + Derivative)

☆☆ Most important in IoT and industry

PID provides stable and smooth control.
Used in robots, drones, automotive systems.

PID नियंत्रण स्थिर और स्मूद कंट्रोल देता है।
रोबोट, ड्रोन और ऑटोमोबाइल सिस्टम में उपयोग होता है।

---

5. Block Diagram of Control System

Input → Controller → Actuator → Output → Feedback → Controller

इनपुट → कंट्रोलर → एक्ट्यूएटर → आउटपुट → फीडबैक → कंट्रोलर

---

6. Applications of Control Systems in IoT

- Smart home automation
- Smart agriculture (soil moisture control)
- Industrial automation
- Smart healthcare
- Smart traffic systems
- Energy management systems

- स्मार्ट होम ऑटोमेशन
- स्मार्ट कृषि (मिट्टी नमी नियंत्रण)
- औद्योगिक ऑटोमेशन
- स्मार्ट हेल्थकेयर
- स्मार्ट ट्रैफिक सिस्टम
- ऊर्जा प्रबंधन सिस्टम

---

7. Advantages of IoT-based Control Systems

- Automation reduces human effort
- Improves accuracy
- Real-time control
- Remote monitoring
- Energy efficient

- ऑटोमेशन से मानव श्रम कम
- सटीकता में सुधार
- रियल-टाइम कंट्रोल
- दूर से निगरानी
- ऊर्जा की बचत

---

8. Challenges of Control Systems

- Network failure affects control
- Security vulnerabilities
- Sensor noise affects accuracy
- High cost in advanced systems

- नेटवर्क फेल होने पर कंट्रोल प्रभावित
- सुरक्षा समस्याएँ
- सेंसर की त्रुटियाँ
- उन्नत सिस्टम में अधिक लागत

SUMMARY

A control system is a system that regulates the behavior of devices using sensors, processors, and actuators. It can be open-loop or closed-loop depending on whether feedback is used or not. Open-loop systems are simple but less accurate, while closed-loop systems use feedback to correct errors and provide accurate control. IoT-based control systems are widely used in smart homes, agriculture, industries, and healthcare. They improve automation, accuracy, and efficiency but face challenges like network issues and security risks.

कंट्रोल सिस्टम वह प्रणाली है जो सेंसर, प्रोसेसर और एक्ट्यूएटर का उपयोग करके उपकरणों के व्यवहार को नियंत्रित करती है। इसे फीडबैक के आधार पर ओपन-लूप और क्लोज्ड-लूप में विभाजित किया जाता है। ओपन-लूप सिस्टम सरल होते हैं लेकिन कम सटीक होते हैं, जबकि क्लोज्ड-लूप सिस्टम फीडबैक का उपयोग करके त्रुटियों को सुधारते हैं और अधिक सटीक नियंत्रण देते हैं। IoT आधारित कंट्रोल सिस्टम स्मार्ट होम, कृषि, उद्योग और हेल्थकेयर में उपयोग होते हैं। ये ऑटोमेशन, सटीकता और दक्षता को बढ़ाते हैं, लेकिन नेटवर्क और सुरक्षा जैसी चुनौतियों का सामना करते हैं।

M4 - 3.1

M4 - R5 (IoT)

Chapter _ 3

Sensors, Actuators & Microcontrollers

सेंसर, एक्चुएटर्स और माइक्रोकंट्रोलर्स

- **Sensor** – Measuring physical quantities in digital world e.g. light sensor, moisture sensor, temperature sensor, etc.

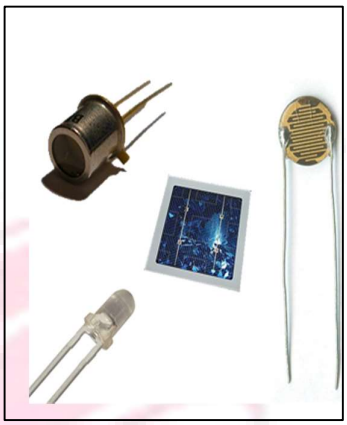
- **Actuator** – Moving or controlling systems e.g. DC motor, different types of actuators.

- **Controller** – Role of microcontroller as gateway to interfacing sensors and actuators.

- **Microcontroller vs Microprocessor**

- **Different types of microcontrollers in embedded ecosystem**

Sensor - सेंसर :



Sensors detect the presence of energy, changes in or the transfer of energy.
Sensors detect by receiving a signal from a device such as a transducer, then responding to that signal by converting it into an output can easily be read and understood.

सेंसर ऊर्जा की उपस्थिति, परिवर्तन या ऊर्जा के हस्तांतरण का पता लगाते हैं।

सेंसर एक ट्रांसड्यूसर जैसे उपकरण से सिग्नल प्राप्त करके पता लगाते हैं, फिर उस सिग्नल को एक आउटपुट में परिवर्तित करके इसका जवाब देते हैं, जिसे आसानी से पढ़ा और समझा जा सकता है।

Examples of Sensors:

- **Temperature Sensor**
- **Proximity Sensor**

- Accelerometer
- IR Sensor (Infrared Sensor)
- Moisture Sensor
- Light Sensor

Two Basic Groups of Sensors _ सेंसर के दो बेसिक प्रकार

Active Sensors:

The effect of the measured physical quantity on the sensor changes the electric parameters (resistance, inductance, capacitance, frequency); therefore, the sensor needs a power supply to function properly.
Example: Radar, GPS, X-Ray, Sonar (Sound Navigation and Ranging), etc.

सक्रिय सेंसर:

सेंसर पर मापा गया भौतिक मात्रा का प्रभाव विद्युत मापदंडों (रेज़िस्टेंस, इंडक्टेंस, कैपेसिटेंस, फ्रीक्वेंसी) को बदलता है, इसलिए सेंसर को सही से काम करने के लिए बिजली की आपूर्ति की आवश्यकता होती है।

1. Passive Sensors:

The effect of the measured physical quantity on the sensor does not change the sensor's electrical properties. The sensor does not need a power supply.
Example: Photographic, thermal, electric field sensing, chemical, infrared, etc.

निष्क्रिय सेंसर:

सेंसर पर मापा गया किसी भौतिक मात्रा के प्रभाव से सेंसर के विद्युत गुण में परिवर्तन नहीं होता है। सेंसर को बिजली की आपूर्ति की आवश्यकता नहीं होती है।

NOTE:

All living organs contain biological sensors with functions similar to the mechanical devices described.

For example _ **light: eyes, smell: nose, sound: ears, and so on.**

उदाहरण _ प्रकाश: आँखें, गंध: नाक, ध्वनि: कान, आदि।

Some Important Sensors

Temperature Sensor _ तापमान सेंसर

A temperature sensor is a device to measure the temperature through an electrical signal. It requires a thermocouple or RTD (Resistance Temperature Detectors).

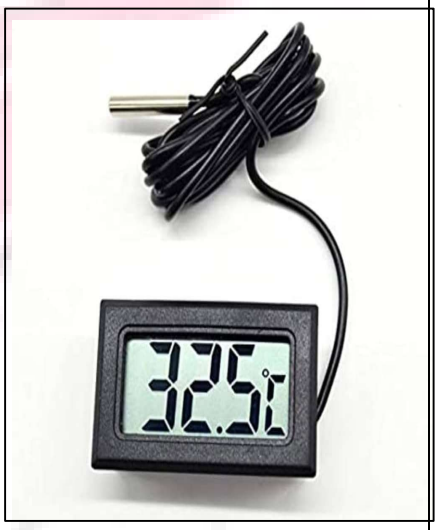
DHT, LM35

ताप

मान सेंसर एक उपकरण है; एक विद्युत संकेत के माध्यम से तापमान को मापने के लिए इसे थर्मोकपल

या RTD (रेज़िस्टेंस टेम्परेचर डिटेक्टर) की आवश्यकता होती है।

Some Types of Temperature Sensors _ कुछ तापमान सेंसर:



1. Thermocouples:

These are voltage devices that indicate temperature by a change in voltage.
As temperature increases, the output voltage of the thermocouple rises.

ये वोल्टेज डिवाइस हैं जो वोल्टेज में बदलाव के sसाथ तापमान मापने का संकेत देते हैं। जैसे ही तापमान बढ़ता है, थर्मोकपल का आउटपुट वोल्टेज बढ़ जाता है।

2. Resistor Temperature Detectors (RTD):

The resistance of the device is directly proportional to the temperature increasing in a positive direction when the temperature rises.

डिवाइस का रेजिस्टेंस तापमान के सीधे अनुपात में होता है —जैसे-जैसे तापमान बढ़ता है, रेजिस्टेंस भी बढ़ता है।

3. Thermostats:

It is a temperature-sensitive resistor that changes its physical resistance with the change in temperature.

यह एक टेम्परेचर-सेंसिटिव रेसिस्टर है जो तापमान में परिवर्तन के साथ अपने रेसिस्टेंस को बदलता है।

4. Semiconductor Sensors:

They are linear devices where the conductivity of the semiconductor increases linearly with temperature.

They provide a direct temperature reading in digital form, especially at low temperatures.

ये रैखिक उपकरण हैं जहां सेमीकंडक्टर की चालकता तापमान के साथ रैखिक रूप से बढ़ती है।

यह विशेष रूप से कम तापमान पर डिजिटल रूप में सटीक तापमान रीडिंग प्रदान करते हैं।

5. Infrared Sensors:

They detect temperature by intercepting a portion of emitted infrared energy of the object or substance, and sensing its intensity.

Used to measure temperature of solids and liquids only (not gases due to transparency).

ये वस्तु या पदार्थ द्वारा उत्सर्जित अवरक्त ऊर्जा के एक हिस्से को रोककर तापमान का पता लगाते हैं।

इनका उपयोग केवल ठोस और तरल पदार्थों के तापमान को मापने के लिए किया जाता है —गैसों के लिए नहीं।

Proximity Sensor _ निकटता सेंसर

A device that detects the presence or absence of a nearby object, or properties of that object, and converts it into a signal which can be easily read by a user.

एक उपकरण जो किसी नज़दीकी वस्तु या उस वस्तु के गुणों की उपस्थिति या अनुपस्थिति का पता लगाता है, और इसे एक सिग्नल में परिवर्तित करता है जिसे उपयोगकर्ता आसानी से पढ़ सकता है।

Some Types of Proximity Sensors _ कुछ प्रॉक्सिमिटी सेंसर

1. Inductive Sensors:

Used for non-contact detection of metallic objects using an electromagnetic field or a beam of electromagnetic radiation.
They can operate at higher speeds than mechanical switches and are more reliable because of robustness.

इंडक्टिव प्रॉक्सिमिटी सेंसर का उपयोग बिना संपर्क किए धातु की वस्तुओं की उपस्थिति का पता लगाने के लिए किया जाता है।

ये यांत्रिक स्विचों की तुलना में अधिक गति पर काम कर सकते हैं और मज़बूती के कारण अधिक विश्वसनीय होते हैं।

2. Capacitive Sensors:

Can detect both metallic and non-metallic targets.
Used to sense very small objects and for complex or difficult applications.

कैपेसिटिव प्रॉक्सिमिटी सेंसर धातु और गैर-धातु दोनों प्रकार की वस्तुओं को पहचान सकते हैं।

इनका उपयोग छोटे ऑब्जेक्ट्स का पता लगाने या जटिल परिस्थितियों में किया जाता है।

3. Photoelectric Sensors:

Made of light-sensitive parts; use a beam of light to detect presence or absence of an object.

Ideal for long-distance sensing and non-metal objects.

फोटोइलेक्ट्रिक सेंसर प्रकाश-संवेदनशील भागों से बने होते हैं और किसी वस्तु की उपस्थिति या अनुपस्थिति का पता लगाने के लिए प्रकाश की किरण का उपयोग करते हैं।

ये लंबी दूरी के सेंसिंग या गैर-धातु वस्तुओं के लिए आदर्श होते हैं

4. Ultrasonic Sensors:

An ultrasonic sensor measures the distance to an object using ultrasonic sound waves.

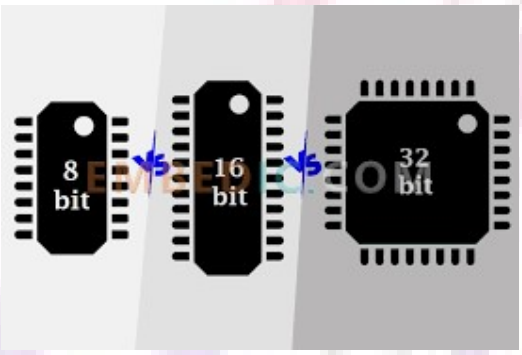
Used for distance and presence detection like

radar or sonar.

अल्ट्रासोनिक सेंसर एक उपकरण है जो ध्वनि तरंगों का उपयोग करके किसी वस्तु की दूरी मापता है।

इनका उपयोग वस्तु की उपस्थिति या दूरी मापने के लिए किया जाता है, जैसे रडार या सोनार।

Pressure Sensor _ प्रेशर सेंसर



A pressure sensor senses pressure and converts it into an electrical signal.
The signal amount depends on the level of pressure applied.

प्रेशर सेंसर दबाव को महसूस करता है और उसे एक विद्युत सिग्नल में बदल देता है।

सिग्नल की मात्रा लगाए गए दबाव के स्तर पर निर्भर करती है।

Such sensors are used in many devices that rely on liquid or pressure systems, especially in IoT applications.

ऐसे सेंसर का उपयोग कई उपकरणों में किया जाता है जो तरल या दबाव पर आधारित होते हैं, खासकर IoT सिस्टम में।

Water Quality Sensor _ वॉटर क्वालिटी सेंसर

Used to detect water quality and ion monitoring, primarily in water distribution systems.
Water is practically used everywhere –hence these sensors are used in many industries.

जल गुणवत्ता सेंसर का उपयोग मुख्य रूप से जल वितरण प्रणालियों में पानी की गुणवत्ता और आयन मॉनिटरिंग का पता लगाने के लिए किया जाता है। पानी लगभग हर जगह उपयोग होता है, इसलिए इन सेंसर का उपयोग कई उद्योगों में होता है।

Some of the Water Quality Sensors

_कुछ वॉटर क्वालिटी सेंसर

1. Chlorine Residual Sensor:

Measures chlorine residual in water –widely used as a disinfectant because of its efficiency.

यह पानी में क्लोरीन के अवशेष को मापता है और दक्षता के कारण सबसे अधिक उपयोग किया जाने वाला कीटाणुनाशक सेंसर है।

2. Total Organic Carbon (TOC) Sensor:

Measures organic elements in water.

यह सेंसर पानी में कार्बन आधारित (ऑर्गेनिक) तत्वों को मापता है।

3. Turbidity Sensor:

Measures suspended solids in water;

typically used in rivers, streams, and wastewater treatment.

टर्बिडिटी सेंसर पानी में ठोस कणों की मात्रा को मापते हैं।

इनका उपयोग नदियों, धाराओं, अपशिष्ट जल और औद्योगिक अपशिष्ट निगरानी में किया जाता है।

4. pH Sensor:

Used to measure the pH level in water — determines whether it is acidic or alkaline.

इसका उपयोग पानी के pH स्तर (अम्लीय या क्षारीय होने की स्थिति) को मापने के लिए किया जाता है।

PH = Potential of Hydrogen

5. Oxygen-Reduction Potential (ORP) Sensor:

Measures oxidation/reduction reactions occurring in the solution.

ओआरपी सेंसर किसी घोल में होने वाली ऑक्सीकरण/अपचयन प्रतिक्रियाओं के स्तर को मापता है।

Chemical Sensor _ केमिकल सेंसर

Chemical sensors are used in various industries.
Their goal is to indicate changes in liquids or to detect air chemical changes.
They are mainly used for industrial environmental monitoring and process control.

कई अलग-अलग उद्योगों में रासायनिक सेंसर लगाए जाते हैं।

इनका उद्देश्य तरल में परिवर्तन का संकेत देना या वायु में रासायनिक परिवर्तन का पता लगाना होता है।

रासायनिक सेंसर का मुख्य उपयोग औद्योगिक पर्यावरण निगरानी और प्रक्रिया नियंत्रण में किया जाता है।

Gas Sensor _ गैस सेंसर

Gas sensors are similar to chemical sensors but are specifically used to monitor air quality and detect the presence of various gases.

Like chemical sensors, they are used in industries such as manufacturing, agriculture, and healthcare.
They help in air quality monitoring, detection of toxic or combustible gases, and hazardous gas monitoring in coal mines, oil & gas industries, chemical labs, etc.

गैस सेंसर रासायनिक सेंसर के समान होते हैं, लेकिन इनका उपयोग विशेष रूप से हवा की गुणवत्ता की निगरानी और विभिन्न गैसों की उपस्थिति का पता लगाने के लिए किया जाता है।
इनका उपयोग निर्माण, कृषि और स्वास्थ्य जैसे कई उद्योगों में किया जाता है।
ये सेंसर वायु गुणवत्ता की निगरानी, विषैली या ज्वलनशील गैसों का पता लगाने, कोयला खदानों में खतरनाक गैसों की निगरानी और तेल-गैस उद्योगों में भी उपयोग किए जाते हैं।

Examples of Gas Sensors

_ गैस सेंसर के उदाहरण:

- Carbon Dioxide Sensor
- Carbon Monoxide Detector
- Hydrogen Sensor
- Air Pollution Sensor
- Nitrogen Oxide Sensor
- Oxygen Sensor
- Ozone Monitor
- MQ-135

Smoke Sensor _ स्मोक सेंसर

A smoke sensor detects smoke (airborne particles and gases) and its level. These sensors are widely used in manufacturing industries, HVAC systems, buildings, and residential infrastructure to detect fire and gas incidents.

They help protect people working in dangerous environments.

स्मोक सेंसर एक ऐसा उपकरण है जो धुएं (हवा में मौजूद कणों और गैसों) का पता लगाता है।

इनका उपयोग निर्माण उद्योग, HVAC सिस्टम, इमारतों और आवासीय क्षेत्रों में आग और गैस से संबंधित घटनाओं का पता लगाने के लिए किया जाता है।यह खतरनाक वातावरण में काम करने वाले लोगों की सुरक्षा में मदद करता है।

Level Sensor _ लेवल सेंसर

A sensor that determines the level or amount of fluids or liquids in an open or closed system is called a Level Sensor.

They are primarily known for measuring fuel levels but are also used in many liquid-handling businesses.

लेवल सेंसर एक ऐसा उपकरण है जो किसी खुले या बंद सिस्टम में तरल पदार्थों के स्तर या मात्रा को मापने के लिए उपयोग किया जाता है।

इन्हें आमतौर पर ईंधन स्तर मापने के लिए जाना जाता है, लेकिन इनका उपयोग अन्य तरल पदार्थों वाले उद्योगों में भी किया जाता है।

Image Sensor _ इमेज सेंसर

Image sensors convert optical images into electrical signals for display or digital storage.
They are mainly used in digital cameras, medical imaging, night vision devices, thermal imaging, radar, sonar, media, and biometric/IRIS devices.

इमेज सेंसर ऐसे उपकरण हैं जो ऑप्टिकल छवियों को इलेक्ट्रॉनिक संकेतों में बदलकर उन्हें डिजिटल रूप में

प्रदर्शित या संग्रहित करने में मदद करते हैं।

इनका उपयोग मुख्य रूप से डिजिटल कैमरा, मेडिकल इमेजिंग, नाइट विज़न, थर्मल इमेजिंग, रडार, सोनार, मीडिया हाउस, बायोमेट्रिक और आईरिस उपकरणों में किया जाता है।

Motion Detection Sensor _ मोशन डिटेक्शन सेंसर

A motion detector detects physical movement in a given area and converts it into an electrical signal.
It can detect motion of objects or human beings.

मोशन डिटेक्टर एक इलेक्ट्रॉनिक उपकरण है जो किसी क्षेत्र में भौतिक गति का पता लगाता है और उसे विद्युत संकेत में परिवर्तित करता है।

यह वस्तुओं या मनुष्यों की गति का पता लगाने के लिए प्रयोग किया जाता है।

Gyroscope Sensor _ जाइरोस्कोप सेंसर

A device used to measure angular rate or angular velocity is called a Gyroscope Sensor. It measures the speed of rotation around an axis and is mainly used in navigation and orientation monitoring.

जाइरोस्कोप सेंसर कोणीय वेग या घूर्णन दर को मापने के लिए प्रयोग किया जाता है।

यह किसी वस्तु की दिशा और गति की निगरानी करता है, और नेविगेशन प्रणालियों में प्रमुख रूप से उपयोग होता है।

Accelerometer Sensor _ एक्सेलेरोमीटर सेंसर

An accelerometer measures vibration or acceleration of motion of a structure. These sensors are found in many devices like smartphones.

They are used for vibration detection, tilt sensing, and acceleration monitoring.

एक्सेलेरोमीटर एक उपकरण है जो किसी संरचना के कंपन या गति के त्वरण को मापता है।

ये सेंसर लाखों उपकरणों (जैसे स्मार्टफ़ोन) में पाए जाते हैं।

इनका उपयोग कंपन, झुकाव और गति का पता लगाने के लिए किया जाता है।

Light Sensor _ लाइट सेंसर

A light sensor consists of a photodiode with an LED at the center.
When light hits the photodiode, it produces a voltage that is amplified by a microprocessor. By analyzing the voltage, the sensor detects light intensity.

लाइट सेंसर एक साधारण फोटो डायोड होता है जिसके केंद्र में एक एलईडी लगी होती है।

जब प्रकाश इस पर पड़ता है, तो यह वोल्टेज उत्पन्न करता है जिसे माइक्रो प्रोसेसर द्वारा बढ़ाया जाता है। फोटो डायोड के वोल्टेज स्तर को देखकर प्रकाश का पता लगाया जा सकता है।

Example: **LDR (Light Dependent Resistor) / Photoresistor**

Moisture Sensor _ नमी सेंसर

The Moisture Sensor is used to measure the water content (moisture) of soil.
When the soil has a shortage of water, the module output is at a high level; otherwise, it is at a low level.

नमी सेंसर का उपयोग मिट्टी की नमी को मापने के लिए किया जाता है।

जब मिट्टी में पानी की कमी होती है, तो मॉड्यूल का

आउटपुट उच्च स्तर पर होता है, अन्यथा आउटपुट निम्न स्तर पर होता है।

Example: DHT11, DHT22

Humidity Sensor _ आर्द्रता सेंसर

The humidity sensor is a device that senses, measures, and reports the relative humidity (RH) of air or determines the amount of water vapor present in a gas mixture (air) or pure gas.

आर्द्रता सेंसर एक ऐसा उपकरण है जो हवा की सापेक्ष आर्द्रता (RH) को मापता है और रिपोर्ट करता है या गैस मिश्रण (वायु) या शुद्ध गैस में मौजूद जलवाष्प की मात्रा निर्धारित करता है।

Optical Sensor _ ऑप्टिकल सेंसर

A sensor that measures the physical quantity of light rays and converts it into electrical signals that can be easily read by a user or an electronic device is called an Optical Sensor.

Due to this, they are widely used in healthcare, environmental monitoring, energy, aerospace, and many other industries.

ऑप्टिकल सेंसर वह उपकरण है जो प्रकाश किरणों की भौतिक मात्रा को मापता है और उसे विद्युत संकेतों में परिवर्तित करता है जिन्हें उपयोगकर्ता या कोई इलेक्ट्रॉनिक यंत्र आसानी से पढ़ सकता है।
इसी कारण इनका उपयोग स्वास्थ्य सेवाओं, पर्यावरण निगरानी, ऊर्जा, एयरोस्पेस और कई अन्य उद्योगों में किया जाता है।

Actuator - गत देने वाला

An actuator is something that actuates or moves something.
An actuator converts energy into motion or mechanical energy.
An actuator requires a control signal and a source of energy.
Actuators are typically used in manufacturing or industrial applications and might be used

in devices such as motors, pumps, switches, and valves.

एक एक्चुएटर एक ऐसी वस्तु है जो किसी वस्तु को सक्रिय करता है या गति प्रदान करता है।

एक एक्चुएटर ऊर्जा को गति या यांत्रिक ऊर्जा में परिवर्तित करता है।

एक्चुएटर को एक नियंत्रण सिग्नल और ऊर्जा स्रोत की आवश्यकता होती है।

एक्चुएटर आमतौर पर विनिर्माण या औद्योगिक अनुप्रयोगों में उपयोग किए जाते हैं और इनका उपयोग मोटर, पंप, स्विच और वाल्व जैसे उपकरणों में किया जा सकता है।

(Moving or controlling system e.g. DC motor)

सिस्टम को स्थानांतरित करना

या नियंत्रित करना जैसे -

डीसी मोटर

DC motor speed control is one of the most useful features of the motor.
By controlling the speed of the motor, you can vary the speed according to requirements and get the desired operation.

डीसी मोटर गति नियंत्रण मोटर की सबसे उपयोगी विशेषताओं में से एक है।

मोटर की गति को नियंत्रित करके, आप आवश्यकता अनुसार गति को बदल सकते हैं और आवश्यक ऑपरेशन प्राप्त कर सकते हैं।

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy.
In a practical DC motor, when the conductor (armature) is supplied with current, it produces its own magnetic flux.

The magnetic flux either adds up to or cancels out the magnetic flux due to the field windings.

The accumulation of magnetic flux in one direction exerts a force on the conductor, and therefore, it starts rotating.

एक डीसी मोटर रोटरी विद्युत मोटर का एक वर्ग है जो डीसी विद्युत ऊर्जा को यांत्रिक ऊर्जा में परिवर्तित करता है।

डीसी मोटर में, जब कंडक्टर (आर्मेचर) को करंट दिया जाता है, तो यह अपना स्वयं का चुंबकीय फ्लक्स उत्पन्न करता है।

यह फ्लक्स फील्ड वाइंडिंग द्वारा उत्पन्न फ्लक्स के साथ या तो जुड़ता है या उसे निरस्त करता है।

एक दिशा में फ्लक्स का संचय कंडक्टर पर बल लगाता है, जिससे यह घूमना शुरू कर देता है।

Types of Actuators _

विभिन्न प्रकार के एक्चुएटर

1. Electric Actuators _ इलेक्ट्रिकल एक्चुएटर

An electric actuator may provide actuation force/torque in several ways.
Electromechanical actuators power a motor that converts electrical energy into mechanical torque.
This type of actuator is generally powered by a motor (DC, AC, or Stepper) that converts electrical energy into mechanical motion.

एक इलेक्ट्रिक एक्चुएटर कई तरीकों से बल या टॉर्क प्रदान कर सकता है।

इलेक्ट्रोमैकेनिकल एक्चुएटर एक मोटर को शक्ति देते हैं जो विद्युत ऊर्जा को यांत्रिक टॉर्क में बदलता है।

इस प्रकार के एक्चुएटर आमतौर पर मोटर (डीसी मोटर, एसी मोटर, या स्टेपर मोटर) द्वारा संचालित होते हैं जो विद्युत ऊर्जा को यांत्रिक गति में परिवर्तित करते हैं।

2. Hydraulic Actuators _ हाइड्रॉलिक एक्चुएटर

A hydraulic actuator consists of a cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation.
The motion output can be linear, rotary, or oscillatory.
Because liquids are almost incompressible, a hydraulic actuator can exert a considerable amount of force.
These are used to operate tools like balers, cranes, excavators, loaders, and presses.

हाइड्रॉलिक एक्चुएटर में एक सिलिंडर या फ्लुइड मोटर होता है जो यांत्रिक संचालन को सुगम बनाने के लिए हाइड्रॉलिक पावर का उपयोग करता है।

इसकी यांत्रिक गति रैखिक, घूर्णनशील यादोलनशील हो सकती है।

3. Pneumatic Actuators _ न्यूमैटिक एक्चुएटर

A pneumatic actuator uses energy from compressed air or vacuum at high pressure to produce either linear or rotary motion.

Example: Used in robotics –sensors that function like human fingers using compressed air.

एक वायवीय (न्यूमैटिक) एक्चुएटर उच्च दबाव वाली संपीड़ित हवा या वैक्यूम से उत्पन्न ऊर्जा का उपयोग रैखिक या घूर्णी गति उत्पन्न करने के लिए करता है।

उदाहरण: रोबोटिक्स में ऐसे सेंसरों में उपयोग होता है जो संपीड़ित हवा के माध्यम से मानव उंगलियों की तरह कार्य करते हैं।

4. Mechanical Actuators _ मैकेनिकल एक्चुएटर

A mechanical actuator converts one type of motion (such as rotary) into another (such as linear).

Example: Rack and pinion mechanism.

Operation is based on structural components like gears and rails or pulleys and chains.

एक यांत्रिक (मैकेनिकल) एक्चुएटर एक प्रकार की गति (जैसे घूर्णी) को दूसरे प्रकार की गति (जैसे रैखिक) में परिवर्तित करता है।

उदाहरण: रैक और पिनियन तंत्र।

इनका संचालन संरचनात्मक घटकों जैसे गियर, रेल, पुली और चेन पर आधारित होता है।

क्योंकि द्रव को संपीड़ित करना लगभग असंभव होता है, हाइड्रॉलिक एक्चुएटर बहुत अधिक बल लगा सकता है। इनका उपयोग बेलर, क्रेन, खुदाई मशीन, लोडर और प्रेस जैसे उपकरणों को संचालित करने के लिए किया जाता है।

5. Thermal / Magnetic Actuators _ थर्मल / मैग्नेटिक एक्चुएटर

These actuators can be activated by applying thermal or magnetic energy.
They are compact, lightweight, economical, and have high power density.
They use shape memory materials (SMMs) such as Shape Memory Alloys (SMAs) or Magnetic Shape Memory Alloys (MSMAs).

इन प्रकार के एक्चुएटर को थर्मल या चुंबकीय ऊर्जा लागू करके सक्रिय किया जा सकता है।

ये आकार में छोटे, हल्के, किफायती और उच्च शक्ति घनत्व वाले होते हैं।

इनमें "शेप मेमोरी मटेरियल" (SMM) जैसे "शेप मेमोरी एलॉय" (SMA) या "मैग्नेटिक शेप मेमोरी एलॉय" (MSMA) का उपयोग किया जाता है।

M4 - 3.3

Controller or Microcontroller _ कंट्रोलर और माइक्रोकंट्रोलर

A controller or microcontroller is also known as an embedded controller.
A microcontroller is a semiconductor IC (Integrated Circuit) chip consisting of multiple functional blocks such as CPU, ROM, RAM, EPROM, Timer, Counter & Interrupts etc.
It is essentially a small computer with all peripherals inside a single package that can be used as an embedded system.

एक कंट्रोलर या माइक्रोकंट्रोलर को **एम्बेडेड कंट्रोलर** के रूप में भी जाना जाता है।

एक माइक्रोकंट्रोलर एक **सेमीकंडक्टर IC (इंटीग्रेटेड सर्किट)** चिप होता है जिसमें कई कार्यात्मक ब्लॉक होते हैं, जैसे CPU, ROM, RAM, EPROM, Timer, Counter और Interrupts आदि।

यह मूलतः एक छोटा कंप्यूटर होता है जिसमें सभी पेरिफेरल्स एक ही

पैकेज में होते हैं और इसका उपयोग एक एम्बेडेड सिस्टम के रूप में किया जा सकता है।

Sometimes referred to as an Embedded Controller or Microcontroller Unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines, and home appliances.

कभी-कभी इसे **एम्बेडेड कंट्रोलर** या **माइक्रोकंट्रोलर यूनिट (MCU)** कहा जाता है।

माइक्रोकंट्रोलर वाहनों, रोबोट्स, कार्यालय मशीनों, चिकित्सा उपकरणों, मोबाइल रेडियो ट्रांसीवर, वेंडिंग मशीनों और घरेलू उपकरणों में पाए जाते हैं।

Role of Microcontroller as Gateway to Interfacing Sensors and Actuators

सेंसर और एक्चुएटर्स के इंटरफेसिंग में माइक्रोकंट्रोलर की भूमिका

Main Role of the Microcontroller _ माइक्रोकंट्रोलर की मुख्य भूमिका

It satisfies market needs; microcontrollers have so many features and are designed to meet embedded system requirements. That means it performs functions only for its embedded system purpose.

यह बाजार की आवश्यकताओं को पूरा करता है; माइक्रोकंट्रोलर में कई विशेषताएँ होती हैं और यह एम्बेडेड सिस्टम की ज़रूरतों को पूरा करने के लिए डिज़ाइन किया गया है।

इसका मतलब है कि यह केवल अपने एम्बेडेड सिस्टम उद्देश्य के लिए कार्य करता है।

It simplifies the complexity of market needs.

यह बाजार की ज़रूरतों की जटिलता को सरल करता है।

It acts as a motivational factor for the embedded system market.

यह एम्बेडेड सिस्टम बाजार के लिए प्रेरणादायक तत्व का कार्य करता है।

Microcontrollers help to make innovations (standard applications).

माइक्रोकंट्रोलर नवाचारों (मानक अनुप्रयोगों) को विकसित करने में मदद करते हैं।

Power Consumption Factor _ बिजली की खपत का कारक

Microcontrollers require less energy –this is a key factor that satisfies user needs.

हम जानते हैं कि माइक्रोकंट्रोलर को बहुत कम ऊर्जा की

आवश्यकता होती है, यही कारण है कि यह उपयोगकर्ताओं की जरूरतों को प्रभावी ढंग से पूरा करता है।

Heart of the Embedded System _ एम्बेडेड सिस्टम का हृदय

Microcontrollers are the heart of embedded systems; they give life to embedded applications.
The invention of microcontrollers brought revolutionary changes in embedded technology.

माइक्रोकंट्रोलर एम्बेडेड सिस्टम का "दिल" हैं —ये एम्बेडेड अनुप्रयोगों को जीवन देते हैं।
माइक्रोकंट्रोलर के आविष्कार ने एम्बेडेड टेक्नोलॉजी में अप्रत्याशित बदलाव किए।

Microcontroller vs Microprocessor _ माइक्रोकंट्रोलर बनाम माइक्रोप्रोसेसर

Both microprocessors and microcontrollers are IC-based electronic devices used in modern equipment such as computers, laptops, washing machines, air conditioners, etc.

Their primary function is to automate processes.

माइक्रोप्रोसेसर और माइक्रोकंट्रोलर दोनों ही इंटीग्रेटेड सर्किट (IC) आधारित इलेक्ट्रॉनिक डिवाइस हैं जिनका उपयोग कंप्यूटर, लैपटॉप, वॉशिंग मशीन, एयर कंडीशनर और अन्य उपकरणों में किया जाता है। दोनों का मुख्य उद्देश्य प्रक्रियाओं का स्वचालन (Automation) करना है।

What is a Microprocessor?

It is a processing device that converts data into information based on a set of instructions.

It is a compact electronic chip and the most crucial component of a computer or computing device.

It processes data and produces output based on the given instructions.

यह एक प्रोसेसिंग डिवाइस है जो निर्देशों के आधार पर डेटा को सूचना में परिवर्तित करता है।

यह एक बहुत कॉम्पैक्ट इलेक्ट्रॉनिक चिप है और कंप्यूटर का सबसे महत्वपूर्ण घटक है।

यह निर्देशों के अनुसार डेटा को प्रोसेस करके सूचना उत्पन्न करता है।

Example: Intel 8085, Intel 8086

What is a Microcontroller?

A microcontroller is a compact electronic system that contains a processing element, small memory (RAM, ROM, EPROM), and I/O ports on a single chip.
Hence, it is like a miniature version of a microcomputer.
It is small, low-cost, and serves as the main functional device in appliances like microwaves, washing machines, and smart refrigerators.

माइक्रोकंट्रोलर एक छोटा इलेक्ट्रॉनिक सिस्टम है जिसमें प्रोसेसिंग यूनिट, छोटी मेमोरी (RAM, ROM, EPROM) और I/O पोर्ट एक ही चिप में होते हैं। यह मूल रूप से माइक्रो कंप्यूटर का छोटा रूप है। यह कम लागत वाला और छोटा डिवाइस है, जिसका उपयोग घरेलू और औद्योगिक उपकरणों में मुख्य नियंत्रक के रूप में किया जाता है।

Microcontroller (माइक्रोकंट्रोलर)	Microprocessor (माइक्रोप्रोसेसर)
Acts as the heart of an embedded system. एम्बेडेड सिस्टम का हृदय।	Acts as the heart of a computer system. कंप्यूटर सिस्टम का हृदय।
Memory & I/O components are internal, making the circuit less complex.	Memory & I/O are external, making the circuit more complex.
Used mainly in washing	Used mainly in

Microcontroller (माइक्रोकंट्रोलर)	Microprocessor (माइक्रोप्रोसेसर)
machines, air conditioners, etc.	personal computers.
More efficient.	Less efficient.
Has more registers.	Has fewer registers.
Can be used with compact systems.	Cannot be used in compact systems.

Embedded System _ एम्बेडेड सिस्टम

An Embedded System is a combination of computer software and hardware, either fixed or programmable.
Example: Fire Alarm –it can sense only smoke.

एम्बेडेड सिस्टम कंप्यूटर हार्डवेयर और सॉफ़्टवेयर का संयोजन होता है जो या तो स्थिर होता है या प्रोग्राम योग्य।

उदाहरण: फायर अलार्म —जो केवल धुआं पहचान सकता है।

Types of Microcontrollers in Embedded Ecosystem

Microcontrollers are categorized according to Memory, Architecture, Bus Width, and Instruction Set.

माइक्रोकंट्रोलर को उनकी **मेमोरी, आर्किटेक्चर, बस चौड़ाई, और निर्देश सेट** के आधार पर वर्गीकृत किया गया है।

According to Bus Width _ बस चौड़ाई के अनुसार

The □□□□□ refers to parallel connections between microcontroller components that transmit instructions and data.
Based on bus width, microcontrollers are divided into:

8-bit, 16-bit, and 32-bit microcontrollers.

“बस” माइक्रोकंट्रोलर के घटकों के बीच समानांतर लाइनों का समूह है जो निर्देश और डेटा का संचार करती हैं।

बस चौड़ाई के आधार पर माइक्रोकंट्रोलर तीन प्रकार के होते हैं:

8-बिट, 16-बिट, और 32-बिट माइक्रोकंट्रोलर।

According to Bus Width _ बस चौड़ाई के अनुसार

The □□□□□ refers to parallel connections between microcontroller components that transmit instructions and data.
Based on bus width, microcontrollers are divided into:

8-bit, 16-bit, and 32-bit microcontrollers.

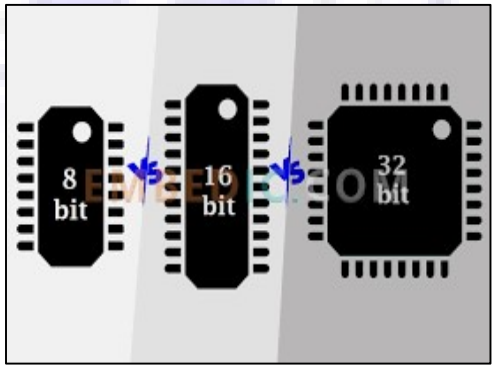
“बस” माइक्रोकंट्रोलर के घटकों के बीच समानांतर लाइनों का समूह है जो निर्देश और डेटा का संचार करती हैं।

बस चौड़ाई के आधार पर माइक्रोकंट्रोलर तीन प्रकार के होते हैं:

8-बिट, 16-बिट, और 32-बिट

माइक्रोकंट्रोलर।

According to Memory Devices _ मेमोरी डिवाइस के अनुसार



Microcontrollers need memory (RAM, ROM, EPROM, EEPROM, Flash Memory, etc.) to store programs and data.
Based on the memory configuration, microcontrollers are divided into two types:

माइक्रोकंट्रोलर को प्रोग्राम और डेटा स्टोर करने के लिए मेमोरी (RAM, ROM, EPROM, EEPROM, Flash Memory आदि) की आवश्यकता होती है।
मेमोरी कॉन्फ़िगरेशन के आधार पर इन्हें दो प्रकारों में बाँटा गया है:

1. Embedded Memory Microcontroller _ एम्बेडेड मेमोरी माइक्रोकंट्रोलर

In this type, both data and program memory are embedded into the IC itself.
For example _ 8051 microcontroller contains

program & data memory, I/O ports, serial communication, counters, timers, and interrupts.

इस प्रकार के माइक्रोकंट्रोलर में डेटा और प्रोग्राम मेमोरी दोनों IC के अंदर एम्बेडेड होती हैं।

उदाहरण: 8051 माइक्रोकंट्रोलर, जिसमें प्रोग्राम और डेटा मेमोरी, I/O पोर्ट, सीरियल कम्युनिकेशन, काउंटर, टाइमर और इंटरप्ट्स —सब एक ही चिप में होते हैं।

2. External Memory Microcontroller _ बाहरी मेमोरी माइक्रोकंट्रोलर

These microcontrollers do not have program memory embedded on the chip and require an external chip for the same.
For example _ 8031 microcontroller has no internal program memory and requires an external memory chip.

इन माइक्रोकंट्रोलर में प्रोग्राम मेमोरी चिप के अंदर नहीं होती और इसके लिए बाहरी मेमोरी चिप की आवश्यकता होती है।

उदाहरण: 8031 माइक्रोकंट्रोलर, जिसमें कोई आंतरिक प्रोग्राम मेमोरी नहीं होती।

According to Instruction Set _ इंस्ट्रक्शन सेट के अनुसार

The instruction set (ISA _ Instruction Set Architecture) defines how the CPU executes commands.
Based on this, microcontrollers are classified into two types:

इंस्ट्रक्शन सेट (ISA _ Instruction Set Architecture) यह निर्धारित करता है कि CPU निर्देशों को कैसे निष्पादित करता है।

इसके आधार पर माइक्रोकंट्रोलर दो प्रकार के होते हैं:

1. CISC (Complex Instruction Set Computer) _ जटिल निर्देश सेट कंप्यूटर

This type of microcontroller's CPU is designed to execute a single complex command that performs multiple operations.
It can execute several steps using a single instruction.

CISC माइक्रोकंट्रोलर का CPU इस प्रकार बनाया जाता है कि वह एक ही जटिल निर्देश से कई कार्य कर सके।

यह एक ही इंस्ट्रक्शन से कई ऑपरेशन कर सकता है।

2. RISC (Reduced Instruction Set Computer) _ सीमित निर्देश सेट कंप्यूटर

This type of microcontroller's CPU is designed to execute smaller and simpler instructions.
RISC processors are faster and consume less power.

RISC माइक्रोकंट्रोलर का CPU छोटे और सरल निर्देशों को तेज़ी से निष्पादित करने के लिए बनाया गया है। ये प्रोसेसर तेज़ होते हैं और कम ऊर्जा की खपत करते हैं।

According to Architecture _ आर्किटेक्चर के अनुसार

Microcontrollers are classified into two types according to architecture:

माइक्रोकंट्रोलर को उनकी संरचना के आधार पर दो प्रकारों में वर्गीकृत किया गया है:

. Harvard Architecture Microcontroller _ हार्वर्ड आर्किटेक्चर माइक्रोकंट्रोलर

It has physically separate memory for program (instructions) and data.
Hence, both can be accessed simultaneously, which increases processing speed.

Example: **Most modern embedded controllers.**

इस प्रकार के माइक्रोकंट्रोलर में प्रोग्राम (निर्देश) और डेटा के लिए अलग-अलग मेमोरी होती है।

इससे दोनों को एक साथ एक्सेस किया जा सकता है, जिससे निष्पादन की गति बढ़ जाती है।

उदाहरण: अधिकांश आधुनिक एम्बेडेड कंट्रोलर।

2. Von Neumann (Princeton) Architecture Microcontroller _ वॉन न्यूमैन (प्रिंसटन) आर्किटेक्चर माइक्रोकंट्रोलर

It uses a single memory for both program and data storage.
This concept was proposed by John Von Neumann (1945) and is widely used in computers and laptops.

इस आर्किटेक्चर में प्रोग्राम और डेटा दोनों के लिए एक ही मेमोरी का उपयोग किया जाता है।

यह अवधारणा **जॉन वॉन न्यूमैन** द्वारा 1945 में प्रस्तुत की गई थी और आज भी कंप्यूटरों और लैपटॉप में सबसे अधिक उपयोग की जाती है।

Various Types of Commonly Used Microcontrollers _ सामान्यतः उपयोग किए जाने वाले माइक्रोकंट्रोलर के प्रकार

1. 8051 Microcontroller _ 8051 माइक्रोकंट्रोलर

Developed by Intel in 1981, it is an 8-bit microcontroller having a 40-pin DIP package, 4KB ROM, 128 bytes RAM, and two 16-bit timers.

It has four parallel 8-bit ports, all programmable and addressable.

इंटेल द्वारा 1981 में विकसित, यह एक 8-बिट माइक्रोकंट्रोलर है जिसमें 40 पिन DIP पैकेज, 4KB ROM, 128 बाइट RAM, और दो 16-बिट टाइमर होते हैं।

इसमें चार समानांतर 8-बिट पोर्ट होते हैं, जो सभी प्रोग्रामेबल और एड्रेसेबल होते हैं।

It is widely used in automobile systems, robotics, medical equipment, and industrial machines.

इसका उपयोग व्यापक रूप से **ऑटोमोबाइल सिस्टम, रोबोटिक्स, मेडिकल उपकरण** और **औद्योगिक मशीनों** में किया जाता है।

2. ARM Microcontrollers _ ए.आर.एम. माइक्रोकंट्रोलर

ARM stands for Advanced RISC Machine.
It is based on RISC architecture and provides high-speed instruction execution.

Used in mobile phones, tablets, multimedia devices, and wearable electronics.

ARM का अर्थ है **Advanced RISC Machine**।

यह **RISC आर्किटेक्चर** पर आधारित होता है और **तेज़ निर्देश निष्पादन** की सुविधा प्रदान करता है।

इसका उपयोग **मोबाइल फोन, टैबलेट, मल्टीमीडिया डिवाइसेज़ और वियरेबल इलेक्ट्रॉनिक्स** में किया जाता है।

It is a 32-bit RISC processor, compact, power-efficient, and ideal for embedded applications.

यह एक **32-बिट RISC प्रोसेसर** है, जो आकार में छोटा, ऊर्जा-कुशल और एम्बेडेड एप्लीकेशन के लिए उपयुक्त होता है।

3. PIC Microcontrollers _ पी.आई.सी. माइक्रोकंट्रोलर

PIC stands for Peripheral Interface Controller.
It is based on Harvard Architecture and made by Microchip Technology

.

It is programmable using Assembly, C, or Basic-C language.

PIC का अर्थ है Peripheral Interface Controller।

यह **हार्वर्ड आर्किटेक्चर** पर आधारित है और इसे **Microchip Technology** द्वारा निर्मित किया जाता है।

इसे **Assembly, C** या **Basic-C** भाषा में प्रोग्राम किया जा सकता है।

Used in vehicles, robotics, vending machines, office automation, and medical devices.

इसका उपयोग वाहनों, रोबोटिक्स, वेंडिंग मशीनों, कार्यालय स्वचालन, और चिकित्सा उपकरणों में किया जाता है।

Used in vehicles, robotics, vending machines, office automation, and medical devices.

इसका उपयोग वाहनों, रोबोटिक्स, वेंडिंग मशीनों, कार्यालय स्वचालन, और चिकित्सा उपकरणों में

4. AVR Microcontrollers _ ए.वी.आर. माइक्रोकंट्रोलर

AVR stands for Alf and Vegard's RISC Processor.
It was developed by Atmel Corporation and is based on a modified Harvard architecture.
AVR controllers are faster than PIC and 8051 microcontrollers.

AVR का अर्थ है Alf and Vegard's RISC Processor।

यह **Atmel Corporation** द्वारा विकसित किया गया है और **संशोधित हार्वर्ड आर्किटेक्चर** पर आधारित है। AVR माइक्रोकंट्रोलर **PIC और 8051** माइक्रोकंट्रोलरों से तेज़ होते हैं।

Examples: ATmega16, ATmega32, ATmega64

उदाहरण: ATmega16, ATmega32, ATmega64

Applications of Microcontrollers _ माइक्रोकंट्रोलर के अनुप्रयोग

Microcontrollers are used in:

- **Robotics**
- **Home appliances (TV, Washing Machine, Microwave Oven)**
- **Automobiles**

- **Industrial control systems**
- **Communication devices**

माइक्रोकंट्रोलर का उपयोग निम्न क्षेत्रों में किया जाता है:

- रोबोटिक्स
- घरेलू उपकरण (टीवी, वॉशिंग मशीन, माइक्रोवेव ओवन)
- ऑटोमोबाइल
- औद्योगिक नियंत्रण प्रणाली

Parts of Embedded System _ एम्बेडेड सिस्टम के भाग

An embedded system consists of both hardware and software.
Hardware includes CPU, Memory, Input/Output Ports, Timers, Counters, and Communication Interfaces.

एम्बेडेड सिस्टम हार्डवेयर और सॉफ़्टवेयर दोनों से मिलकर बना होता है।

हार्डवेयर भागों में CPU, मेमोरी, इनपुट/आउटपुट पोर्ट, टाइमर, काउंटर और कम्युनिकेशन इंटरफेस शामिल होते हैं।

1. CPU (Central Processing Unit) _ सेंट्रल प्रोसेसिंग यूनिट

The CPU is the brain of the embedded system.
It performs all arithmetic and logical operations, processes data, and controls other units of the system.
It executes program instructions stored in memory.

CPU एम्बेडेड सिस्टम का "मस्तिष्क" होता है।

यह सभी गणनात्मक और तार्किक क्रियाएँ करता है, डेटा प्रोसेस करता है और सिस्टम की अन्य इकाइयों को नियंत्रित करता है।

यह मेमोरी में संग्रहीत प्रोग्राम निर्देशों को निष्पादित करता है।

2. Memory _ मेमोरी

Memory is used to store programs and data. It is mainly of two types:

मेमोरी का उपयोग प्रोग्राम और डेटा को संग्रहित करने के लिए किया जाता है।

इसके दो प्रमुख प्रकार होते हैं:

a. ROM (Read Only Memory) _

Used for permanent program storage.

b. RAM (Random Access Memory)

_ Used for temporary data storage during execution.

a. ROM (रीड ओनली मेमोरी) _ स्थायी रूप

से प्रोग्राम को स्टोर करने के लिए उपयोग की जाती है।

b. RAM (रैंडम एक्सेस मेमोरी) _ प्रोग्राम के

निष्पादन के दौरान अस्थायी डेटा स्टोरेज के लिए उपयोग होती है।

3. I/O Ports (Input/Output Ports) _ इनपुट / आउटपुट पोर्ट

Input ports are used to receive signals from sensors or user input devices.
Output ports are used to send control signals to actuators or display devices.

4. Timers _ टाइमर

Timers are used to generate time delays, schedule events, or measure time intervals. They help in controlling tasks at fixed intervals and synchronize system operations.

टाइमर का उपयोग समयांतराल उत्पन्न करने, घटनाओं को निर्धारित करने और समय मापने के लिए किया जाता है।

यह निश्चित अंतराल पर कार्यों को नियंत्रित करने और

सिस्टम संचालन को सिंक्रनाइज़ करने में मदद करते हैं।

इनपुट पोर्ट सेंसर या उपयोगकर्ता उपकरणों से सिग्नल प्राप्त करने के लिए उपयोग किए जाते हैं।
आउटपुट पोर्ट एक्ट्यूएटर्स या डिस्प्ले डिवाइस को नियंत्रण सिग्नल भेजने के लिए उपयोग किए जाते हैं।

5. Counters _ काउंटर

Counters are used to count external events or internal pulses within the system.
They are useful in frequency measurement and event counting applications.

काउंटर का उपयोग सिस्टम के अंदर बाहरी घटनाओं या आंतरिक पल्स की गणना करने के लिए किया जाता है। इनका उपयोग फ्रीक्वेंसी मापन और इवेंट काउंटिंग जैसे अनुप्रयोगों में किया जाता है।

6. ADC (Analog to Digital Converter) _ एनालॉग टू डिजिटल कनवर्टर

ADC converts analog signals from sensors into digital form which the microcontroller can process.
It is essential in IoT and embedded systems where sensors output analog data.

ADC सेंसर से आने वाले एनालॉग सिग्नल को डिजिटल रूप में बदलता है ताकि माइक्रोकंट्रोलर उन्हें प्रोसेस कर सके।

यह IoT और एम्बेडेड सिस्टम में अत्यंत आवश्यक होता है, क्योंकि सेंसर प्रायः एनालॉग आउटपुट देते हैं।

7. Communication Interface _ कम्युनिकेशन इंटरफेस

Used for data exchange between microcontrollers, sensors, and other external devices.

Common communication protocols include UART, SPI, and I²C.

कम्युनिकेशन इंटरफेस माइक्रोकंट्रोलर, सेंसर और अन्य बाहरी उपकरणों के बीच डेटा के आदान-प्रदान के लिए उपयोग किया जाता है।

सामान्य संचार प्रोटोकॉल हैं — **UART, SPI, और I²C।**

Chapter 5 : Security and Future of IoT Ecosystem

(IoT पारितंत्र की सुरक्षा और भविष्य)

1. Need of Security in IoT _ Why Security?

(IoT में सुरक्षा की आवश्यकता क्यों है?)

Every connected device creates opportunities for attackers.

→ प्रत्येक connected डिवाइस हमलावरों के लिए अवसर पैदा करता है

Every connected device creates opportunities for attackers.

कमजोरियाँ छोटे उपकरणों में भी होती हैं —जोखिमों में शामिल हैं

- **Data transfer issues** (डेटा स्थानांतरण समस्याएँ)

- **Unauthorized device access** (अनधिकृत डिवाइस एक्सेस)
- **Malfunctioning devices** (डिवाइस का खराब होना)
- **Always-on / always-connected devices** (हमेशा चालू / कनेक्टेड उपकरण)
- **Main challenges in IoT security:**

IoT सुरक्षा की मुख्य चुनौतियाँ:

- **Low-cost device production limits security features.**
  (कम लागत वाले उपकरणों में सीमित सुरक्षा सुविधाएँ होती हैं)

Increasing number of devices creates more attack opportunities.
(डिवाइसों की बढ़ती संख्या से हमलों के अवसर बढ़ते हैं)

2. Privacy for IoT-Enabled Devices

(IoT-सक्षम उपकरणों में गोपनीयता)

- **(a) Secure the IoT Network (IoT नेटवर्क को सुरक्षित करें):**
- **Protect IoT devices connected to backend systems using traditional endpoint security features:**
  Antivirus, Anti-malware, Firewalls, Intrusion Prevention & Detection Systems.
- एंटीवायरस, एंटी-मालवेयर, फायरवॉल और घुसपैठ रोकथाम/पहचान प्रणाली का उपयोग करें।
- **(b) Authenticate the IoT Devices (IoT उपकरणों को प्रमाणित करें):**
- **Allow user authentication via:**
  Multiple user management, Two-factor authentication, Digital certificates, Biometrics.

- उपयोगकर्ताओं को मल्टीयूज़र प्रबंधन-, दोकारक - प्रमाणीकरण, डिजिटल प्रमाणपत्र और बायोमेट्रिक्स के माध्यम से प्रमाणीकरण की अनुमति दें।
- **Allow user authentication via: Multiple user management, Two-factor authentication, Digital certificates, Biometrics.**

- उपयोगकर्ताओं को मल्टीयूज़र प्रबंधन-, दोकारक - प्रमाणीकरण, डिजिटल प्रमाणपत्र और बायोमेट्रिक्स के माध्यम से प्रमाणीकरण की अनुमति दें।

- **(d) Use IoT PKI Security Methods (IoT PKI सुरक्षा विधियों का उपयोग करें):**
  - **Implement Public Key Infrastructure (PKI): X.509 Certificates, Public/Private key generation, Management, Revocation.**

- पब्लिक की इंफ्रास्ट्रक्चर (PKI) विधियों जैसे X.509 सर्टिफिकेट और की प्रबंधन अपनाएँ।

(e) Use IoT Security Analytics (IoT सुरक्षा विश्लेषण का उपयोग करें):

- **Apply analytics tools capable of detecting IoT-specific attacks not seen by traditional firewalls.**
- ऐसे सुरक्षा विश्लेषण उपकरणों का उपयोग करें जो IoT-विशिष्ट हमलों को पहचान सकें।

(f) Use IoT API Security Methods (IoT API सुरक्षा विधियाँ):

- **Secure API communication among devices, backend systems, and apps.**
- **Ensure only authorized devices and users access APIs.**
- केवल अधिकृत डिवाइस और डेवलपर्स को API उपयोग की अनुमति दें।

3. IoT Security for Consumer Devices

(उपभोक्ता उपकरणों के लिए IoT सुरक्षा)

- **IoT security includes four essential activities:**

  (IoT सुरक्षा में चार मुख्य क्रियाएँ होती हैं)

- **Identification (पहचान)**

- **Devices must have secure, traceable identifiers (e.g., serial numbers).**
- प्रत्येक डिवाइस में सुरक्षित पहचान संख्या होनी चाहिए।

- **Authentication (प्रमाणीकरण)**

- **A secure mechanism to establish device_ network connection.**
- नेटवर्क और डिवाइस के बीच सुरक्षित कनेक्शन के लिए प्रमाणीकरण आवश्यक है।

- **Authorization (अधिकार प्रदान करना)**
- **Controls device behavior based on identity and permissions.**
- डिवाइस के उपयोग को उसकी पहचान और अनुमति के आधार पर नियंत्रित किया जाता है।
- **Monitoring (निगरानी करना)**
- **Detect unusual data patterns and potential breaches.**
- असामान्य डेटा पैटर्न और संभावित सुरक्षा उल्लंघन की पहचान करें।
- **4. Security Levels (सुरक्षा के स्तर)**

IoT devices face both active and passive attacks.

(आईओटी उपकरण सक्रिय और निष्क्रिय दोनों प्रकार के हमलों का सामना करते हैं।)

1. Low-level attack (निम्न-स्तरीय हमला):

- **Attack attempt fails; no harm done.**
- जब हमला असफल रहता है।

2. Medium-level attack (मध्यम-स्तरीय हमला):

- **Attacker listens but doesn't modify data**
- हमलावर केवल सुनता है, डेटा में परिवर्तन नहीं करता।

3. High-level attack (उच्च-स्तरीय हमला):

- **Attacker modifies or alters the integrity of data.**
- हमलावर डेटा में बदलाव करता है।

4. Extremely High-level attack (अत्यंत उच्च-स्तरीय हमला):

- **Unauthorized access, illegal operations, jamming, or flooding network.**
- अनधिकृत पहुँच, अवैध कार्यवाही, या नेटवर्क को बाधित करना।

5. Protecting IoT Devices (IoT उपकरणों की सुरक्षा)

(a) Keep Changing Your Password (पासवर्ड बदलते रहें):

- **Regularly update passwords on PCs, accounts, and devices.**
- समय-समय पर अपने पासवर्ड बदलें।

(b) Update Your Device (उपकरण को अपडेट रखें):

- **Enable automatic updates and install security patches.**
- ऑटो-अपडेट चालू करें और सुरक्षा पैच स्थापित करें।

(c) Reduce Use of Cloud (क्लाउड का उपयोग कम करें):

- **Cloud connectivity may expose data to hackers.**
- क्लाउड का कम उपयोग करें ताकि डेटा सुरक्षित रहे।

(d) Use Secondary Network (द्वितीय नेटवर्क का उपयोग करें):

- **Create a separate network for IoT devices.**
- IoT उपकरणों के लिए अलग Wi-Fi नेटवर्क बनाएँ।

6. Botnet (बॉटनेट

- **A Botnet is a network of internet-connected devices like PCs, servers, mobiles, and IoT devices.**
- **"Botnet" word = "Robot" + "Network".**

- बॉटनेट इंटरनेट से जुड़े उपकरणों का समूह होता है।

Common uses:

- **Sending spam emails** (स्पैम ईमेल भेजना)
- **Fraudulent activities** (धोखाधड़ी वाले अभियान)
- **Distributed Denial of Service (DDoS) attacks to overload networks**.
- **DDoS** हमलों से नेटवर्क को ठप करना।

M4-R5

Chapter 4 – Building IoT Application

(IoT एप्लीकेशन का निर्माण)

❖ Introduction to Arduino IDE

(Arduino IDE का परिचय)

Arduino was born at the *Ivrea Interaction Design Institute* as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. Arduino is an open-source computer hardware and software company.

Arduino boards can read inputs –a light on a sensor, a finger on a button, or even a Twitter message –and turn it into an output like activating a motor, turning on an LED, or publishing something online.
You tell your board what to do by sending a set of instructions to the microcontroller using the Arduino programming language (based on Wiring) and the Arduino IDE (based on Processing).

Arduino का जन्म *Ivrea Interaction Design Institute* में हुआ था, इसे तेज़ी से प्रोटोटाइप बनाने के लिए एक आसान उपकरण के रूप में विकसित किया गया था, जिसका उद्देश्य उन छात्रों की मदद करना था जिनकी पृष्ठभूमि इलेक्ट्रॉनिक्स या प्रोग्रामिंग में नहीं थी।Arduino एक ओपन-सोर्स

कंप्यूटर हार्डवेयर और सॉफ्टवेयर कंपनी है।

Arduino बोर्ड इनपुट पढ़ सकता है — जैसे किसी सेंसर पर रोशनी, बटन पर उंगली, या ट्विटर संदेश — और उसे आउटपुट में बदल सकता है, जैसे मोटर चालू करना, LED जलाना या कुछ ऑनलाइन प्रकाशित करना।
आप बोर्ड को बताते हैं कि क्या करना है, इसके लिए आप Arduino प्रोग्रामिंग भाषा (जो *Wiring* पर आधारित है) और Arduino सॉफ़्टवेयर IDE (जो *Processing* पर आधारित है) का उपयोग करते हैं।

❖ What is Arduino IDE?

(Arduino IDE क्या है?)

Arduino IDE (Integrated Development Environment) is open-source software used for writing, compiling, and uploading code to Arduino boards.
It acts like a text editor (similar to Notepad) with additional features.
It is cross-platform software available for Windows, Linux, and macOS, and supports both C and C++ languages.

Arduino IDE (इंटीग्रेटेड डेवलपमेंट एनवायरनमेंट) एक ओपन-सोर्स सॉफ्टवेयर है जिसका उपयोग Arduino बोर्ड पर कोड

लिखने, कंपाइल करने और अपलोड करने के लिए किया जाता है।

यह नोटपैड की तरह एक टेक्स्ट एडिटर है, लेकिन इसमें अतिरिक्त सुविधाएँ होती हैं।

यह क्रॉस-प्लेटफ़ॉर्म सॉफ्टवेयर है जो Windows, Linux, और macOS सभी पर चलता है और C व C++ भाषाओं को सपोर्ट करता है।

➢ Parts of the Arduino IDE Environment

(Arduino IDE के प्रमुख भाग)

i. **Menu Bar** – **Contains options for File, Edit, Sketch, Tools, and Help.**
ii. **Toolbar** – **Quick-access buttons for Verify, Upload, New, Open, Save, and Serial Monitor.**
iii. **Text Editor** – **Area where you write and edit your code.**
iv. **Output Panel / Console** – **Shows messages, errors, and compilation details.**

1. **मेनू बार** – इसमें File, Edit, Sketch, Tools और Help जैसी विकल्प होते हैं।
2. **टूलबार** – Verify, Upload, New, Open, Save, और Serial Monitor के लिए शॉर्टकट बटन होते हैं।

3. **टेक्स्ट एडिटर** – यहाँ आप कोड लिखते और संपादित करते हैं।
4. **आउटपुट पैनल / कंसोल** – इसमें संदेश, त्रुटियाँ और कंपाइलिंग से संबंधित विवरण दिखते हैं।

➢ LED Programming – एलईडी प्रोग्रामिंग

Required Components (आवश्यक घटक(

For LED programming, you will need:

- **Arduino board**
- **LED**
- **Resistor**
- **Breadboard**
- **Jumper wires**

Once these components are ready, install the Arduino IDE **software on your computer and configure your Arduino board by installing the required device driver.**

एलईडी प्रोग्रामिंग के लिए आपको चाहिए –

- एक Arduino बोर्ड
- एक एलईडी (LED)
- एक रेजिस्टर (Resistor)

- एक ब्रेडबोर्ड (Breadboard)
- जम्पर वायर (Jumper Wires)

जब आपके पास ये सभी घटक हों, तब अपने कंप्यूटर में **Arduino IDE** सॉफ़्टवेयर इंस्टॉल करें और आवश्यक **डिवाइस ड्राइवर** लगाकर Arduino बोर्ड को कॉन्फ़िगर करें।

- **Arduino Board (Arduino बोर्ड)**

The Arduino Uno is a microcontroller board based on the ATmega328.
It includes everything needed to support the microcontroller — just connect it to a computer with a USB cable or power it using an adapter or battery.
It is simple and easy to use for repetitive tasks such as blinking an LED.
The word Uno means "one" in Italian and marks the release of Arduino IDE version 1.0.

Arduino Uno एक माइक्रोकंट्रोलर बोर्ड है जो ATmega328 पर आधारित है।

इसमें माइक्रोकंट्रोलर को सपोर्ट करने के लिए आवश्यक सभी चीजें होती हैं।

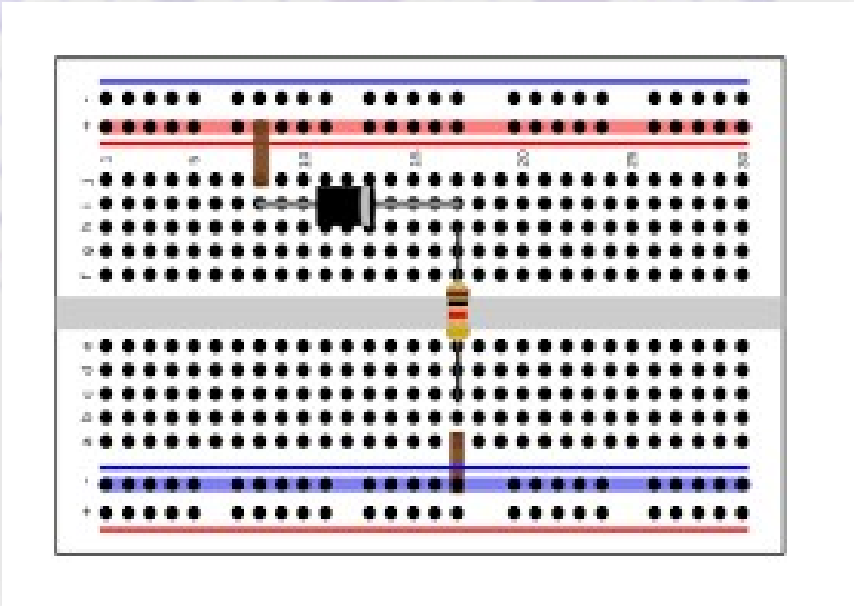
इसे USB केबल से कंप्यूटर से जोड़ा जा सकता है या फिर AC-to-DC एडेप्टर या बैटरी से पावर दी जा सकती है।

यह सरल कार्यों जैसे LED को ऑन-ऑफ करने के लिए बहुत उपयोगी है।

Uno शब्द इटैलियन भाषा का है जिसका अर्थ "एक" होता है — इसे Arduino IDE 1.0 के लॉन्च को दर्शाने के लिए चुना गया।

- **Breadboard (ब्रेडबोर्ड)**

The breadboard is used to make temporary electrical connections between components such as resistors, LEDs, capacitors, etc., before permanently soldering them. It has many small sockets (holes) arranged in a 0.1-inch grid. Some groups of sockets are internally connected, allowing components to be easily inserted and



connected without soldering.

ब्रेडबोर्ड का उपयोग सर्किट के घटकों (जैसे रेजिस्टर, एलईडी, कैपेसिटर आदि) को अस्थायी रूप से जोड़ने के लिए किया जाता है ताकि उन्हें स्थायी रूप से जोड़ने से पहले परीक्षण किया जा सके।

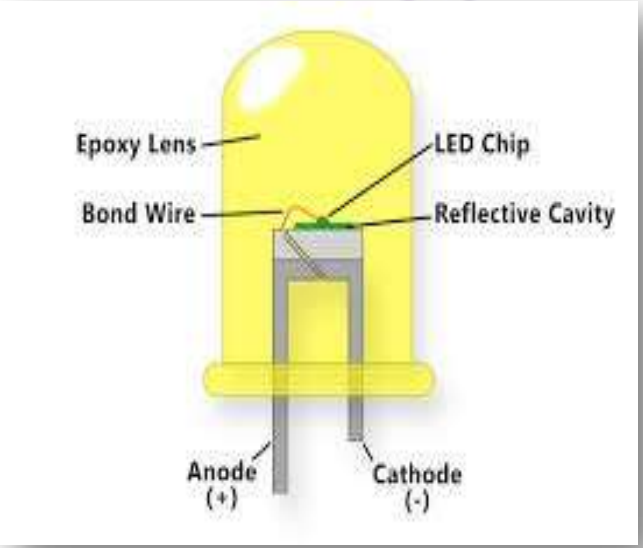
इसमें छोटे-छोटे छेद (holes) होते हैं जो 0.1 इंच की दूरी पर व्यवस्थित रहते हैं।

कुछ सॉकेट आपस में जुड़े होते हैं जिससे बिना सोल्डरिंग के सर्किट को आसानी से जोड़ा जा सकता है।

•LED (लाइट एमिटिंग डायोड)

An LED (Light Emitting Diode) is a diode that emits light when current flows through it.
It has two pins:

- **Cathode (–): Connects to GND (Ground)**
- **Anode (+): Connects to a digital output pin to control the LED state**



एलईडी (लाइट एमिटिंग डायोड) एक ऐसा डायोड

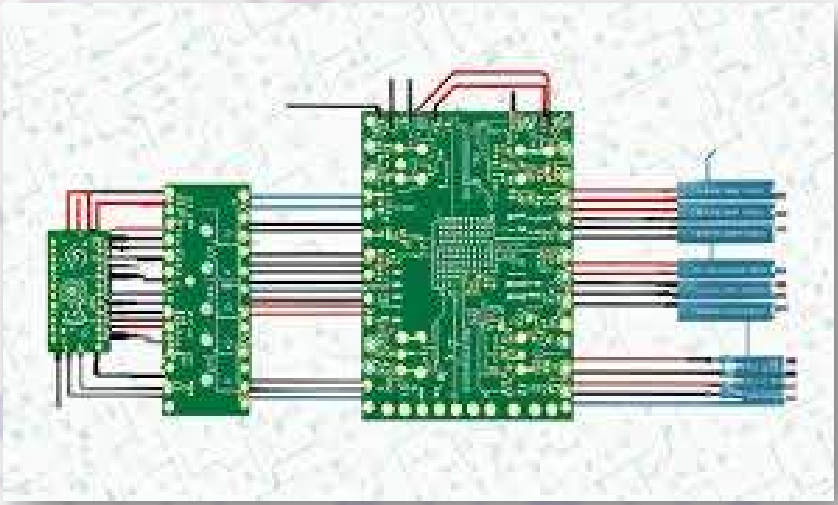
है जो उस पर विद्युत धारा प्रवाहित होने पर प्रकाश उत्सर्जित करता है।

इसमें दो पिन होते हैं:

- **कैथोड (–):** इसे GND (ग्राउंड) से जोड़ा जाता है
- **एनोड (+):** इसे Arduino के आउटपुट पिन से जोड़ा जाता है जिससे एलईडी को नियंत्रित किया जा सके|

• Resistor (रेजिस्टर)

A resistor is a passive electronic component used to limit current or reduce voltage in a circuit. By dropping voltage across it, it protects components like LEDs from excessive current.



रेजिस्टर एक निष्क्रिय (Passive) इलेक्ट्रॉनिक घटक है जिसका उपयोग सर्किट में प्रवाहित होने वाली धारा को सीमित करने या वोल्टेज घटाने के लिए किया जाता है।

यह एलईडी जैसे घटकों को अधिक धारा से बचाता है।

- **Jumper Wires (जम्पर वायर्स)**

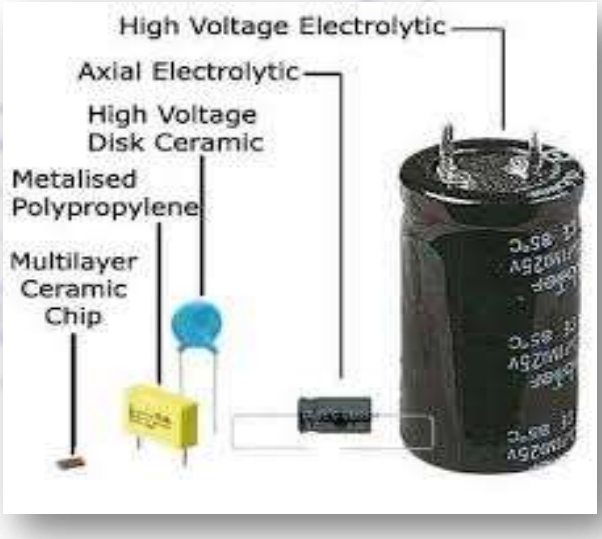
Jumper wires are simple wires with connector pins at both ends.
They are used to connect two points on a breadboard without soldering.

जम्पर वायर सामान्य तार होते हैं जिनके दोनों सिरों पर कनेक्टर पिन होती हैं।

इनका उपयोग बिना सोल्डरिंग किए ब्रेडबोर्ड के दो बिंदुओं को जोड़ने के लिए किया जाता है।

- **Capacitor (कैपेसिटर)**

In a DC circuit, a capacitor charges up to the supply voltage and blocks further current because its dielectric is non-conductive.
However, in an AC circuit, current appears to pass through with little or no resistance.



DC सर्किट में कैपेसिटर सप्लाई वोल्टेज तक चार्ज होकर करंट के प्रवाह को रोक देता है क्योंकि

उसका डाइलेक्ट्रिक भाग इंसुलेटर होता है।

लेकिन AC सर्किट में यह करंट को बहुत कम रोध के साथ गुजरने देता है।

❖ Writing Code in Sketch (केच में कोड लिखना)

The first new term in Arduino programming is called a "Sketch".
A Sketch is the program that you upload to the Arduino board.
The Sketch Editor is where you view and edit your code. It supports common text editing shortcuts such as:

- **Ctrl + F: Find**
- **Ctrl + Z: Undo**
- **Ctrl + C: Copy**
- **Ctrl + V: Paste**

Arduino प्रोग्रामिंग में पहली नई शब्दावली है "Sketch" (स्केच)।

एक स्केच वह प्रोग्राम होता है जिसे आप Arduino बोर्ड पर अपलोड करते हैं।

स्केच एडिटर वह स्थान है जहाँ आप अपने कोड को लिखते और संपादित करते हैं।

यह सामान्य शॉर्टकट्स को सपोर्ट करता है, जैसे:

- **Ctrl + F:** खोजने के लिए

- **Ctrl + Z:** पूर्ववत करने के लिए
- **Ctrl + C:** कॉपी करने के लिए
- **Ctrl + V:** पेस्ट करने के लिए

❖ Steps to Write an Arduino Program (Arduino प्रोग्राम लिखने के चरण)

Step 1: Create a New Sketch (नया स्केच बनाना)

- **Select File → New in Arduino IDE.**
  Add a block comment at the top of your sketch to describe what it does and who wrote it.

```
// Example of a single-line comment
a = 100; // This is a single line comment

/* Example of a multi-line comment
   This is a multi-line comment block */
```

Step 2: Maintain Proper Indentation (सही इंडेंटेशन रखें(

Keep your code properly indented and easy to read.
This helps in debugging and understanding the flow of the program.

अपने कोड में सही इंडेंटेशन रखें ताकि उसे पढ़ना और डिबग करना आसान हो।

Step 3: Define Variables (वैरिएबल परिभाषित करें(

Variables are used to store data. Each variable has a name, type**, and** value**.**

Example:-

```
int ledPin = 13; // LED connected to digital pin 13
```

Explanation:

Here, ledPin is an integer variable with value 13, representing the digital pin where the LED is connected.

वैरिएबल डेटा को संग्रहीत करने के लिए उपयोग किए जाते हैं।

ऊपर दिए गए उदाहरण में ledPin एक integer प्रकार का वैरिएबल है, जिसकी वैल्यू 13 है — यानी LED डिजिटल पिन 13 से जुड़ी है।

Step 4: Setup Function (सेटअप फ़ंक्शन(

The setup() function runs once when the Arduino is powered on or reset.
It is used to initialize variables, define pin modes, and start serial communication.

Example:

```
void setup() {
  pinMode(2, OUTPUT); // Initialize digital pin 2 as output
}
```

setup() फ़ंक्शन केवल एक बार चलता है जब Arduino चालू या रीसेट होता है।
इसका उपयोग वैरिएबल सेट करने, पिन मोड निर्धारित करने और सीरियल कम्युनिकेशन प्रारंभ करने के लिए किया जाता है।

Step 5: Loop Function (लूप फ़ंक्शन(

The loop() function runs continuously after setup().
It contains the main logic of your program.

Example:

```
void loop() {
  digitalWrite(2, HIGH); // Turn LED ON
  delay(1000);           // Wait for 1 second
  digitalWrite(2, LOW);  // Turn LED OFF
  delay(1000);           // Wait for 1 second
}
```

loop() फ़ंक्शन setup() के बाद बार-बार चलता रहता है। यहीं पर आपके सर्किट का मुख्य लॉजिक लिखा जाता है। ऊपर दिए गए उदाहरण में LED हर 1 सेकंड में ऑन और ऑफ होती है।

Step 6: User-defined Functions (उपयोगकर्ता द्वारा परिभाषित फ़ंक्शन(

You can create custom functions outside setup() and loop() for specific tasks.
When called, Arduino executes the code inside them and returns to the main program.

आप अपनी आवश्यकता के अनुसार setup() और loop() के बाहर खुद के फ़ंक्शन बना सकते हैं।
जब इन्हें कॉल किया जाता है, Arduino उनके अंदर लिखा कोड चलाता है और फिर मुख्य प्रोग्राम पर लौट आता है।

❖ Compiling and Debugging (कंपाइलिंग और डिबगिंग(

➢ Compiling (कंपाइलिंग(

Compiling converts high-level code into machine language (binary code) that the processor can understand.
If any syntax or logical error occurs, the IDE shows an error message at the bottom.

कंपाइलिंग एक प्रक्रिया है जिसमें हाईलेवल कोड को मशीन - भाषा में बदला जाता है ताकि प्रोसेसर उसे समझ सके।

अगर कोड में कोई गलती)error) हो, तो IDE नीचे एरर संदेश दिखाता है।

➢ Debugging (डिबगिंग(

Debugging means finding and fixing errors in your program.
Arduino IDE doesn't have an in-built debugger, but you can use the Serial Monitor to print messages for testing and tracking code behavior.

डिबगिंग का अर्थ है कोड में त्रुटियाँ ढूँढना और उन्हें ठीक करना।

Arduino IDE में बिल्टइन डिबगर नहीं होता-, लेकिन आप Serial Monitor का उपयोग करके प्रोग्राम के परिणामों और वैरिएबल्स की स्थिति देख सकते हैं।

❖ Uploading the File to Arduino Board

(Arduino बोर्ड पर फाइल अपलोड करना)

Follow these steps to upload your sketch:

1. **Connect your Arduino to the computer using a USB cable.**
   - **The square end of the cable connects to Arduino.**
   - **The flat end connects to the USB port of your computer.**
2. **Select the correct board from the menu**:

- **Go to** Tools → Board → Arduino Uno **(or whichever board you are using).**

3. **Select the correct serial port:**
   - Go to **Tools → Port → (COM3 or COM4)** depending on your system.
4. **Click on the Upload button** (**the right arrow icon on the toolbar) or press** Ctrl + U**.**

Once uploaded successfully, the message "Done uploading" will appear at the bottom of the IDE.

अपना स्केच अपलोड करने के लिए ये चरण अपनाएँ:

1. **USB केबल से Arduino को कंप्यूटर से जोड़ें।**
   - केबल का चौकोर सिरा Arduino में लगेगा।
   - सपाट सिरा कंप्यूटर के USB पोर्ट में।
2. **सही बोर्ड का चयन करें:**
   - **Tools → Board → Arduino Uno** (या जो बोर्ड आप उपयोग कर रहे हैं)।
3. **सही सीरियल पोर्ट चुनें:**
   - **Tools → Port → (COM3 या COM4)** अपने सिस्टम के अनुसार।
4. **Upload बटन** (दाएँ तीर वाला आइकन) पर क्लिक करें या **Ctrl + U** दबाएँ।

सफल अपलोड के बाद IDE के नीचे "Done uploading" लिखा आएगा।

❖ Designing a Simple LED Circuit

(सरल एलईडी सर्किट बनाना)

To make the LED blink, open Arduino IDE and follow these steps:

1. **Go to File → Examples → Basics → Blink**
2. **Select the correct Board and Port.**
3. **Click Upload.**

Once uploaded, the LED connected to **pin 13** on the Arduino Uno will start blinking.

एलईडी को ब्लिंक कराने के लिए Arduino IDE खोलें और ये कदम अपनाएँ:

1. जाएँ File → Examples → Basics → Blink
2. सही Board और Port चुनें।
3. फिर Upload पर क्लिक करें।

अपलोड होने के बाद Arduino Uno के pin 13 पर लगी इनबिल्ट एलईडी ब्लिंक करने लगेगी।

Example: Blink Program (एलईडी ब्लिंक प्रोग्राम)

```
// This program will blink the onboard LED with a delay of 1 second.

void setup() {
  pinMode(13, OUTPUT);   // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // Turn LED ON
  delay(1000);            // Wait for 1 second
  digitalWrite(13, LOW);  // Turn LED OFF
  delay(1000);            // Wait for 1 second
}
```

Explanation:

The LED blinks ON and OFF every second (1000 milliseconds).
You can change the blinking speed by changing the delay value.

ऊपर दिया गया कोड एलईडी को हर 1 सेकंड में ऑन और ऑफ करता है।

अगर आप ब्लिंक की गति बदलना चाहते हैं तो delay(1000) का मान बदल सकते हैं।

❖ Libraries in Arduino

(Arduino में लाइब्रेरीज़)

Libraries add extra functionality to your Arduino projects (e.g., controlling sensors, motors, displays).
To include a library:

- Go to Sketch → Include Library → Add .ZIP Library
  Once added, the library appears at the top of your sketch with a #include line.

लाइब्रेरीज़ आपके Arduino प्रोजेक्ट में अतिरिक्त कार्यक्षमता जोड़ती हैं (जैसे सेंसर, मोटर, डिस्प्ले आदि को नियंत्रित करना)।
लाइब्रेरी जोड़ने के लिए:

जाएँ Sketch → Include Library → Add .ZIP Library
लाइब्रेरी जोड़ने के बाद आपके स्केच की शुरुआत में #include लाइन दिखाई देगी।

❖ Role of Serial Monitor

(सीरियल मॉनिटर की भूमिका)

The Serial Monitor is a part of the Arduino IDE.

It serves as a link between your computer and the Arduino board, allowing you to send and receive data in text form. It is useful for debugging and controlling Arduino through your computer keyboard.

To open Serial Monitor:

- Click on the **magnifying glass icon** on the top-right corner of the IDE, or press **Ctrl + Shift + M**.

It opens a separate window showing communication between Arduino and your computer.

सीरियल मॉनिटर Arduino IDE का एक महत्वपूर्ण हिस्सा है। यह कंप्यूटर और Arduino बोर्ड के बीच संचार का माध्यम है, जिसके द्वारा आप टेक्स्ट के रूप में डेटा भेज या प्राप्त कर सकते हैं।
यह डिबगिंग और Arduino को कीबोर्ड से नियंत्रित करने के लिए बहुत उपयोगी है।

Serial Monitor खोलने के लिए:

IDE के ऊपरी दाएँ कोने में आवर्धक काँच (magnifying glass) आइकन पर क्लिक करें या Ctrl + Shift + M दबाएँ।

यह एक अलग विंडो खोलता है जिसमें Arduino और कंप्यूटर के बीच डेटा ट्रांसफर दिखता है।

❖ Embedded □□□ Language Basics (एंबेडेड □□□ भाषा के मूल सिद्धांत(

Introduction (परिचय)

Embedded C Programming Language, widely used in the development of Embedded Systems, is an extension of the C Programming Language. It uses the same syntax and semantics of C, such as main function, data type declaration, variables, loops, functions, and statements.

एंबेडेड C प्रोग्रामिंग लैंग्वेज, जो एंबेडेड सिस्टम्स के विकास में व्यापक रूप से उपयोग की जाती है, C भाषा का एक विस्तार (extension) है। यह C भाषा की समान संरचना और नियमों का पालन करती है जैसे — main function, data type declaration, variables, loops, functions, और statements।

➢ C Language Introduction (C भाषा का परिचय(

C is a general-purpose programming language created by Dennis Ritchie at Bell **Laboratories in 1972.**
Despite being old, it is still one of the most popular programming languages.
C is strongly associated with **UNIX**, as it was developed to write the UNIX operating system.

C एक सामान्य-उद्देश्यीय प्रोग्रामिंग भाषा है, जिसे 1972 में डेनिस रिची (Dennis Ritchie) द्वारा Bell Laboratories में बनाया गया था।

पुरानी होने के बावजूद यह आज भी एक बहुत लोकप्रिय भाषा है।

C का UNIX ऑपरेटिंग सिस्टम से गहरा संबंध है क्योंकि इसे UNIX लिखने के लिए ही विकसित किया गया था।

➢ Getting Started with C

(C का उपयोग शुरू करना

To start using C, you need two things:

1. **A text editor (like Notepad) —** to write C code.
2. **A compiler (like GCC) —** to translate C code into machine language.

C का उपयोग शुरू करने के लिए आपको दो चीज़ों की आवश्यकता होगी:

1. **एक टेक्स्ट एडिटर** (जैसे Notepad) — कोड लिखने के लिए।
2. **एक कंपाइलर** (जैसे GCC) — कोड को मशीन भाषा में बदलने के लिए।

➢ C IDE Installation (C IDE स्थापित करना(

An IDE (Integrated Development Environment) is used to edit and compile code.
Popular IDEs include Code::Blocks, Eclipse, Sublime Text, and Visual Studio.

They are free and can be used for both editing and debugging C code.

IDE (एकीकृत विकास वातावरण) का उपयोग कोड संपादन और कंपाइल करने के लिए किया जाता है।

लोकप्रिय IDEs हैं — Code::Blocks, Eclipse, Sublime Text, और Visual Studio।

ये सभी निशुल्क हैं और C कोड को एडिट व डीबग दोनों के लिए उपयोग किए जा सकते हैं।

➢ Using Sublime Text for C

(Sublime Text का उपयोग C के लिए(

Steps to Set Up Sublime Text for C Programming:

1. Download Sublime Text Editor
2. Download GCC Compiler from https://sourceforge.net/projects/gcc-win64/
3. Extract GCC and save it in the C drive.
4. Go to the bin folder of GCC and copy its path.

5. Add the path in:
   Settings → System → About → Advanced System Settings → Environment Variables → System Variables → Path → Edit → Add New → Paste Path
6. n Sublime, go to Tools → Build System → New Build System
7. Paste the following code:

```
{
  "cmd": ["gcc $file_name -o ${file_base_name} && ./${file_base_name}"],
  "selector": "source.c",
  "shell": true,
  "working_dir": "$file_path"
}
```

8. Save it and create a new file (e.g., first.c)

9. Write your first code and press **Ctrl + B** to compile & run.

❖ C Syntax (C की सिंटैक्स)

```c
#include <stdio.h>

int main() {
    printf("Hello World!");
    return 0;
}
```

Explanation (व्याख्या):

- **#include <stdio.h> →** Header file library for input/output functions like printf().
- **main() →** The function where program execution starts.
- **printf() →** Used to print text on the screen.
- **return 0; →** Ends the main() function successfully.

New Lines in C (C में नई लाइन जोड़ना)

To print a new line, use the \n character.

Example 1:

```
#include <stdio.h>
int main() {
    printf("Hello World!\n");
    printf("I am learning C.");
    return 0;
}
```

Example 2:

```
#include <stdio.h>
int main() {
    printf("Hello World!\nI am learning C.\nAnd it is awesome!");
    return 0;
}
```

Comments in C (C में टिप्पणियाँ(

Comments make the code easier to understand. They can be used for explanation or to disable code temporarily during testing.

टिप्पणियाँ)Comments) कोड को समझने में आसान बनाती हैं।

इनका उपयोग विवरण देने या टेस्टिंग के दौरान कोड को अस्थायी रूप से निष्क्रिय करने के लिए किया जाता है।

(Single-line Comment)

Begins with //

```
// This is a comment
printf("Hello World!"); // This is also a comment
```

Multi-line Comment

(बहु(पंक्ति टिप्पणी-

Begins with /* and ends with */

```
/* The code below will print "Hello World!"
   to the screen, and it is amazing */
printf("Hello World!");
```

❖ Variables, Data Types, Operators and Expressions(वैरिएबल्स, डेटा टाइप्स, ऑपरेटर और अभिव्यक्तियाँ(

Variables

(वैरिएबल्स)

A *variable* is a named location in memory used to store data.
It can hold different values during program execution.

Rules for naming variables in C:

1. It must begin with a letter or underscore (_).
2. No spaces or special symbols (%, $, # etc.) are allowed.
3. C is case-sensitive → Age and age are different.

वैरिएबल मेमोरी का एक नामांकित स्थान है जहाँ डेटा संग्रहीत किया जाता है।
प्रोग्राम चलने के दौरान इसकी वैल्यू बदली जा सकती है।

वैरिएबल नाम रखने के नियम:

1. नाम की शुरुआत किसी अक्षर या अंडरस्कोर (_) से होनी चाहिए।
2. नाम में स्पेस या विशेष चिह्न (% $ # आदि) न हो।
3. C भाषा case-sensitive है — Age और age अलग-अलग माने जाते हैं।

Example:

```
int age = 20;
float marks = 85.5;
char grade = 'A';
```

Data Types (डेटा टाइप्स)

Data types define the type and size of data a variable can store.

Type	Keyword	Description	Size (bytes)	Example

Type	Keyword	Description	Size (bytes)	Example
Integer	int	Whole numbers	2 or 4	int x = 5;
Float	float	Decimal numbers	4	float y = 3.14;
Double	double	Higher precision decimal	8	double pi = 3.14159;
Character	char	Single character	1	char c = 'A';

डेटा टाइप्स यह निर्धारित करते हैं कि वैरिएबल किस प्रकार का डेटा रखेगा और कितनी मेमोरी लेगा।

प्रकार	कीवर्ड	विवरण	आकार (बाइट्स)	उदाहरण

प्रकार	कीवर्ड	विवरण	आकार (बाइट्स)	उदाहरण
पूर्णांक	int	पूर्ण संख्या	2 या 4	int x = 5;
दशमलव	float	दशमलव संख्या	4	float y = 3.14;
डबल	double	उच्च सटीकता वाला दशमलव	8	double pi = 3.14159;
अक्षर	char	एकल अक्षर	1	char c = 'A';

Constants and Literals

(कॉनस्टैंट्स और लिटरेरल्स)

A constant is a fixed value that cannot be changed during program execution.

Types of Constants:

Integer constants (e.g., 10, 25, -50)

Float constants (e.g., 3.14, -0.99)

Character constants (e.g., 'A', 'b')

String constants (e.g., "Hello World")

You can also use #define to create named constants:

```
#define PI 3.14
```

कॉनस्टैंट ऐसे मान हैं जो प्रोग्राम चलने के दौरान नहीं बदलते।

कॉनस्टैंट्स के प्रकार:

- पूर्णांक कॉनस्टैंट → 10, 25, -50
- दशमलव कॉनस्टैंट → 3.14, -0.99
- वर्ण कॉनस्टैंट → 'A', 'b'
- स्ट्रिंग कॉनस्टैंट → "Hello World"

#define से भी कॉनस्टैंट बनाए जा सकते हैं:

```
#define PI 3.14
```

+ Operators

(ऑपरेटर)

Operators are symbols used to perform operations on variables and values.

1. Arithmetic Operators (गणितीय ऑपरेटर):

Operator	Description	Example	Result
+	Addition	10 + 5	15
–	Subtraction	10 – 5	5
*	Multiplication	10 * 5	50
/	Division	10 / 5	2
%	Modulus (Remainder)	10 % 3	1

2. Relational Operators (संबंध ऑपरेटर):

Operator	Meaning	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

Logical Operators (तार्किक ऑपरेटर):

Operator	Description	Example
&&	Logical AND	(a > 0 && b > 0)
!	Logical NOT	!(a > b)

Expressions (अभिव्यक्तियाँ(

An *expression* is a combination of operands (variables or constants) and operators that produces a value.

Example:

```
int x = 10 + 5 * 2;  // Expression = 10 + (5 * 2) = 20
```

अभिव्यक्ति (Expressions) वैरिएबल्स और ऑपरेटरों का ऐसा समूह है जो एक मान उत्पन्न करता है।

उदाहरण:

```
int x = 10 + 5 * 2;  // परिणाम = 20
```

Assignment Operator (असाइनमेंट ऑपरेटर)

Used to assign values to variables.

```
x = 10;  // assign 10 to x
y = x + 5;  // assign (x + 5) to y
```

असाइनमेंट ऑपरेटर (=) का उपयोग किसी वैरिएबल को मान देने के लिए किया जाता है।

❖ Variables in C

Variables are containers for storing data values, like numbers and characters. In C, there are different types of variables (defined with different keywords), for example:

- int **- stores integers (whole numbers), without decimals, such as 123 or -123**
- float **- stores floating point numbers, with decimals, such as 19.99 or -19.99**
- char **- stores single characters, such as 'a' or 'B'. Characters are surrounded by single quotes**

वेरिएबल्स डेटा वैल्यूज़ को स्टोर करने के लिए कंटेनर होते हैं, जैसे नंबर और कैरेक्टर। C में, विभिन्न प्रकार के वेरिएबल्स होते हैं (विभिन्न कीवर्ड्स के साथ परिभाषित), उदाहरण के लिए:

- **int** - पूर्ण संख्याएँ (whole numbers) स्टोर करता है, बिना दशमलव के, जैसे 123 या -123

- **float** - फ्लोटिंग पॉइंट नंबर स्टोर करता है, दशमलव के साथ, जैसे 19.99 या -19.99
- **char** - सिंगल कैरेक्टर स्टोर करता है, जैसे 'a' या 'B'। कैरेक्टर सिंगल कोट्स में होते हैं।

Syntax:

```
<type> <variableName> = value;
```

Example 1:

```
int myNum = 15;
```

Example 2:

```
// Declare a variable
int myNum;
// Assign a value to the variable
myNum = 15;
```

Format Specifiers in C:

Format specifiers are used together with the printf() function to tell the compiler what type of data the variable is storing. It is basically a placeholder for the variable value. A format specifier starts with a percentage sign %, followed

by a character. For example, to output the value of an int variable, use the format specifier %d surrounded by double quotes (""), inside the printf() function:

प्रारूप विनिर्देशक का उपयोग printf() फ़ंक्शन के साथ किया जाता है ताकि कंपाइलर को यह बताया जा सके कि चर किस प्रकार के डेटा को संग्रहीत कर रहा है। यह मूल रूप से चर मान के लिए एक प्लेस होल्डर है। एक प्रारूप विनिर्देशक प्रतिशत चिह्न % से शुरू होता है, उसके बाद एक वर्ण होता है। उदाहरण के लिए, int चर के मान को आउटपुट करने के लिए, printf() फ़ंक्शन के अंदर डबल कोट्स ("") से घिरे प्रारूप विनिर्देशक %d का उपयोग करें:

Example:

// Create variables int myNum = 15; // Integer (whole number) float myFloatNum = 5.99; // Floating point number char myLetter = 'D'; // Character

// Print variables printf("%d\n", myNum); printf("%f\n", myFloatNum); printf("%c\n", myLetter); `

Declare Multiple Variables

```
int x = 5, y = 6, z = 50;printf("%d", x + y + z);
```

❖ Identifiers – आईडेंटिफायर

Identifiers are names given to program elements, such as variables, arrays, functions, structures, unions and labels etc. An identifier can be composed only of uppercase, lowercase letters, underscore and digits, but should start only with an alphabet or an underscore.

आइडेंटिफायर प्रोग्राम एलिमेंट्स को दिए गए नाम हैं, जैसे वेरिएबल, ऐरे, फंक्शन, स्ट्रक्चर, यूनियन और लेबल आदि। एक आइडेंटिफायर केवल अपरकेस, लोअरकेस अक्षर, अंडरस्कोर और अंकों से बन सकता है, लेकिन केवल एक अल्फाबेट या अंडरस्कोर से शुरू होना चाहिए।

- Valid identifiers: total, sum, average, x, y, mark_1, x1

Note: Underscore character is usually used as a link between two words in long identifiers.

नोट: अंडरस्कोर कैरेक्टर आमतौर पर लंबे आइडेंटिफायर में दो शब्दों के बीच एक लिंक के रूप में उपयोग किया जाता है|

❖ Built-in Data Types –

बिल्टइन डेटा टाइप-

Data types are the definition you use to assign the type to the variable that determines what value it is going to

store and manipulate. Whenever you declare a variable it requires a data type to be declared with. Every data type comes with a certain size and range when declared, assigns that size and range to the variable automatically.

डेटा टाइप वह परिभाषा है जिसका उपयोग आप वेरिएबल के प्रकार को असाइन करने के लिए करते हैं जो यह निर्धारित करता है कि वह किस वैल्यू को स्टोर और मैनिपुलेट करने वाला है। जब भी आप एक वेरिएबल को डिक्लेयर करते हैं तो इसके साथ एक डेटा टाइप की आवश्यकता होती है। हर डेटा टाइप एक निश्चित आकार और सीमा के साथ आता है जब डिक्लेयर किया जाता है, उसका आकार और सीमा को वेरिएबल को स्वचालित रूप से असाइन करता है।

Format Specifier	**Data Type**	**Size**	**Description**
%d or %i	int	2 or 4 bytes	whole numbers
%f	float	4 bytes	6-7 decimal digits
%lf	double	8 bytes	15 decimal digits

Format Specifier	Data Type	Size	Description
%c	char	1 byte	single character
%s	strings		Used for strings

Example:

```
// Create variables
int myNum = 5;              // Integer
(whole number)
float myFloatNum = 5.99; // Floating
point number
double myNum = 19.99;       // double
char myLetter = 'D';        // Character
char myText[] = "Hello"; // String

// Print variables
printf("%d\n", myNum);
printf("%f\n", myFloatNum);
printf("%c\n", myLetter);
printf("%s", myText);
`
```

C Decimal Precision (C दशमलव परिशुद्धता)

If you want to remove the extra zeros (set decimal precision), you can use a dot (.) followed by a number that specifies how many digits that should be shown after the decimal point:

यदि आप अतिरिक्त शून्यों को हटाना चाहते हैं (दशमलव परिशुद्धता निर्धारित करना), तो आप एक बिंदु (.) के बाद एक संख्या का उपयोग कर सकते हैं जो निर्दिष्ट करता है कि दशमलव बिंदु के बाद कितने अंक दिखाए जाने चाहिए:

```
`float myFloatNum = 3.5;`

`printf("%f\n", myFloatNum);` // Default will show 6 digits after the decimal point
`printf("%.1f\n", myFloatNum);` // Only show 1 digit
`printf("%.2f\n", myFloatNum);` // Only show 2 digits
`printf("%.4f", myFloatNum);` // Only show 4 digits
```

The sizeof Operator

#include <stdio.h> int main() { int myInt; float myFloat; double myDouble; char myChar;

```
printf("%lu\n", sizeof(myInt));
printf("%lu\n", sizeof(myFloat));
```

```
printf("%lu\n", sizeof(myDouble));
printf("%lu\n", sizeof(myChar));
return 0;

}
```

Constants and Literals

What are Constants?

If you don't want others (or yourself) to change existing variable values, you can use the const keyword. This will declare the variable as "constant", which means unchangeable and read-only:

यदि आप नहीं चाहते कि दूसरों (या आप स्वयं) मौजूदा वेरिएबल मानों को बदलें, तो आप const कीवर्ड का उपयोग कर सकते हैं। यह वेरिएबल को "स्थिर" के रूप में घोषित करेगा, जिसका अर्थ है अपरिवर्तनीय और केवल पढ़ने के लिए:

Example 1:

```
`const int myNum = 15;` // myNum will always be 15
`myNum = 10;` // error: assignment of read-only variable 'myNum'
```

Example 2:

`const int minutesPerHour;`
`minutesPerHour = 60;` // error

What are Literals?

Literals can be defined as data that is given in a variable or constant. For example 9, -1.5, "Abhinav", 'DigitalRK' are literals. Literals are fixed numeric or non-numeric values.

लिटरल्स को एक डेटा के रूप में परिभाषित किया जा सकता है जो एक वेरिएबल या कांस्टेंट में दिया जाता है। उदाहरण के लिए 9, -1.5, "Abhinav", 'DigitalRK' सभी लिटरल्स हैं। लिटरल्स फिक्स्ड न्यूमेरिक या नॉन-न्यूमेरिक वैल्यूज़ होते हैं।

Example:

In the following example myNum is called Identifiers, int is data type and 5 is Literals

निम्नलिखित उदाहरण में myNum को आइडेंटिफायर कहा जाता है, int डेटा प्रकार है और 5 लिटरल्स है

`int myNum = 5;`

Operators in C

Operators are used to perform operations on variables and values. C divides the operators into the following groups:

ऑपरेटर का उपयोग चर और मान पर संचालन करने के लिए किया जाता है। C ऑपरेटर को निम्नलिखित समूहों में विभाजित करता है:

- Arithmetic operators
- Assignment operators
- Comparison operators or Relational Operators
- Logical operators
- Bitwise operators

(Arithmetic operators) _ अ्रथमेटिक ऑपरेटर्स

The Arithmetical operators are the operators which help you perform the arithmetical operation in your program. Example: Addition, Subtraction, Divide and Multiply etc are the Arithmetic operators…

अर्थमेटिक ऑपरेटर वे ऑपरेटर होते हैं जो आपके प्रोग्राम में अर्थमेटिक ऑपरेशन करने में आपकी सहायता करते हैं। उदाहरण : जोड़, घटाना, विभाजन और गुणा आदि अर्थमेटिक ऑपरेटर होते हैं।

Types of Arithmetic Operators in C

The C Arithmetic Operators are of two types based on the number of operands they work. These are as follows:

सी अंकगणतीय ऑपरेटर ऑपरेण्ड की संख्या के आधार पर दो प्रकार के होते हैं। ये इस प्रकार हैं:

1. Binary Arithmetic Operators
2. Unary Arithmetic Operators

---

1. Binary Arithmetic Operators in C

The C binary arithmetic operators operate or work on two operands.

सी बाइनरी अंकगणतीय ऑपरेटर दो ऑपरेण्ड पर कार्य करते हैं।

Example :-

```c
// Working of arithmetic operators
#include <stdio.h>
int main()
{
    int a = 10, b = 4, res;
    // printing a and b
    printf("a is %d and b is %d\n", a, b);

    res = a + b; // addition
    printf("a + b is %d\n", res);

    res = a - b; // subtraction
    printf("a - b is %d\n", res);

    res = a * b; // multiplication
    printf("a * b is %d\n", res);
    res = a / b; // division
    printf("a / b is %d\n", res);

    res = a % b; // modulus for remainder
    printf("a % b is %d\n", res);

    return 0;
}
```

2. Unary Arithmetic Operators in C

The unary arithmetic operators operate or work with a single operand.

एकल अंकगणतीय ऑपरेटर एकल ऑपरेण्ड के साथ कार्य करते हैं।

Increment Operator in C

The '++' operator is used to increment the value of an integer. It can be used in two ways:

'++' ऑपरेटर का उपयोग किसी पूर्णांक के मान को बढ़ाने के लिए किया जाता है। इसका उपयोग दो तरीक़ों से किया जा सकता है:

1. Pre-increment

```
#include <stdio.h>
int main()
{
    int a = 10, b = 4, res;
    res = ++a + ++b; // Pre Increment
    printf("Result is %d\n", res);
    return 0;
}
```

2. Post-increment

```
#include <stdio.h>
int main()
{
    int a = 10, b = 4, res;
    res = a++ + b++;
    printf("Result is %d\n", res); // Post Increment
    res = a + b; // Increased
    printf("Result is %d\n", res);
    return 0;
}
```

Decrement Operator in C

The '--' operator is used to decrement the value of an integer. Just like the increment operator, the decrement operator can also be used in two ways:

'--' ऑपरेटर का उपयोग किसी पूर्णांक के मान को घटाने के लिए किया जाता है। इन्क्रीमेंट ऑपरेटर की तरह ही, डिक्रीमेंट ऑपरेटर का भी दो तरीक़ों से उपयोग किया जा सकता है:

1. Pre-Decrement

```
int a = 10, b = 4, res;
res = --a + --b; // Pre Decrement
printf("Result is %d\n", res);
```

2. Post-Decrement

```
int a = 10, b = 4, res;
res = a-- + b--;
printf("Result is %d\n", res); // Post Decrement
res = a + b; // Increased
printf("Result is %d\n", res);
```

Assignment Operators

An assignment operator assigns a value to a variable based on the value of another expression:

असाइनमेंट ऑपरेटर किसी अन्य अभिव्यक्ति के मान के आधार पर किसी चर को मान निर्दिष्ट करता है:

Example :-

```c
#include <stdio.h>
int main()
{
    int a = 25, b = 5;
    // using operators and printing results
    printf("a = b: %d\n", a = b);
    printf("a += b: %d\n", a += b);
    printf("a -= b: %d\n", a -= b);
    printf("a *= b: %d\n", a *= b);
    printf("a /= b: %d\n", a /= b);
    printf("a %%= b: %d\n", a %= b);
    return 0;
}
```

Relational (Comparison) Operators in C

The relational operators in C are used for the comparison of the two operands. All these operators are binary operators that return true or false values as the result of comparison.

C में रिलेशनल ऑपरेटर का उपयोग दो ऑपरेण्ड की तुलना के लिए किया जाता है। ये सभी ऑपरेटर बाइनरी ऑपरेटर हैं जो तुलना के परिणामस्वरूप सत्य या असत्य मान लौटाते हैं।

Example :-

```c
#include <stdio.h>
int main()
{
    int a = 25, b = 5;
    // using operators and printing results
    printf("a < b : %d\n", a < b);  // Output : 0
    printf("a > b : %d\n", a > b);  // Output : 1
    printf("a <= b: %d\n", a <= b); // Output : 0
    printf("a >= b: %d\n", a >= b); // Output : 1
    printf("a == b: %d\n", a == b); // Output : 0
    printf("a != b : %d\n", a != b); // Output : 1
    return 0;
}
```

Note: Here, 0 means false and 1 means true.

User Input

To get user input, you can use the scanf() function:

उपयोगकर्ता इनपुट प्राप्त करने के लिए, आप `scanf()` फ़ंक्शन का उपयोग कर सकते हैं:

```c
// declare a blank variable
int myNum;

// Ask the user to type a number
printf("Type a number: \n");

// Get and save the number the user types
scanf("%d", &myNum);

// Output the number the user typed
printf("Your number is: %d", myNum);
```

❖ Conditional Statements and Loops

कंडीशनल स्टेटमेंट और लूप्स

You are aware that C language expression statements and repetitive statements are used. These statements are used to assign values and to execute repetitive tasks. A programming language will be incomplete if it does not have a decision making statement. These decision making statements are also called conditional, execution and selection statements.

आप जानते हैं कि C भाषा में एक्सप्रेशन स्टेटमेंट और दोहराव वाले स्टेटमेंट्स का उपयोग किया जाता है। ये स्टेटमेंट्स वैल्यू को असाइन करने और दोहराए जाने वाले कार्य को निष्पादित करने के लिए

उपयोग किए जाते हैं। प्रोग्रामिंग भाषा इनकम्प्लीट होगी अगर इसमें निर्णय लेने के स्टेटमेंट नहीं होंगे। इन निर्णय लेने वाले स्टेटमेंट्स को कंडीशनल, एक्सिक्यूशन और सेलेक्शन स्टेटमेंट भी कहा जाता है।

Conditional Statements _ कंडीशनल स्टेटमेंट

In any programming language, there is a need to perform different tasks based on the condition, that means if the condition will be true then the block of code will be executed else not. For this kind of thing we are facilitated to use various conditional statements to place conditions.

किसी भी प्रोग्रामिंग लैंग्वेज में कंडीशन के आधार पर अलग-अलग टास्क परफॉर्म करने की ज़रूरत होती है, यानी अगर कंडीशन सही होगी तो कोड के ब्लॉक को एक्सिक्यूट किया जाएगा वरना नहीं। इस तरह की चीज़ के लिए हमें विभिन्न सशर्त बयानों का उपयोग करने की सुविधा दी जाती है।

Type of Statements

- **if Statement**
- **if and else Statement**
- **else if Statement**

- **Switch Statement**

The if Statement

यदि कोई शर्त सत्य है तो निष्पादित किए जाने वाले C कोड के ब्लॉक को निर्दिष्ट करने के लिए if कथन का उपयोग करें।

Use the if statement to specify a block of C code to be executed if a condition is true.

Example :-

```
if (20 > 18) {
    printf("20 is greater than 18");
}
```

We can also test variables:

Example :-

```
int x = 20;
int y = 18;

if (x > y) {
    printf("x is greater than y");
}
```

The else Statement

यदि शर्त गलत है तो निष्पादित किए जाने वाले कोड के ब्लॉक को निर्दिष्ट करने के लिए अन्य (else) कथन का उपयोग करें।

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax :-

```
if (condition) {
    // block of code to be executed if the condition is true
} else {
    // block of code to be executed if the condition is false
}
```

Example :-

Check whether an integer is odd or even

```
int i = 20;

if (i % 2 == 0) {
    printf("%d is even Number", i);
} else {
    printf("%d is Odd Number", i);
}
```

The else if Statement

यदि पहली शर्त गलत है तो नई शर्त निर्दिष्ट करने के लिए अन्यथा यदि (else if) कथन का उपयोग करें।

Use the else if statement to specify a new condition if the first condition is false.

Syntax :-

```
if (condition1) {
    // block of code to be executed if condition1 is true
} else if (condition2) {
    // block of code to be executed if condition1 is false and condition2 is true
} else {
    // block of code to be executed if condition1 and condition2 are false
}
```

Example :-

```
int myNum = 2;

if (myNum > 0) {
    printf("The value is a positive number.");
} else if (myNum < 0) {
    printf("The value is a negative number.");
} else {
    printf("The value is 0.");
}
```

ShortHand If...Else (Ternary Operator)

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements.

एक शॉर्ट-हैंड if else भी है, जिसे टर्नरी ऑपरेटर के रूप में जाना जाता है क्योंकि इसमें तीन ऑपरेण्ड होते हैं। इसका उपयोग कोड की कई पंक्तियों को एक पंक्ति से बदलने के लिए किया जा सकता है। इसका उपयोग अक्सर सरल if else कथनों को बदलने के लिए किया जाता है।

Example :-

```
int a = 20;
int b = 18;

(a > b) ? printf("a is Greater.") : printf("b is Greater.");
```

Switch Statement

Instead of writing many if..else statements, you can use the switch statement. The switch statement selects one of many code blocks to be executed.

कई if..else कथन लिखने के बजाय, आप स्विच कथन का उपयोग कर सकते हैं। स्विच कथन निष्पादित किए जाने वाले कई कोड ब्लॉकों में से एक का चयन करता है।

Syntax :-

```
switch (expression) {
    case x:
        // code block
        break;
    case y:
        // code block
        break;
    default:
        // code block
}
```

Example :-

```c
int day = 4;

switch (day) {
    case 1:
        printf("Monday");
        break;
    case 2:
        printf("Tuesday");
        break;
    case 3:
        printf("Wednesday");
        break;
    case 4:
        printf("Thursday");
        break;
    case 5:
        printf("Friday");
        break;
    case 6:
        printf("Saturday");
        break;
    case 7:
        printf("Sunday");
        break;
}
```

❖ Loops in C

जब तक एक निर्दिष्ट स्थिति पूरी हो जाती है, Loops कोड के एक ब्लॉक को निष्पादित कर सकते हैं। Loops उपयोगी होते हैं क्योंकि वे समय बचाते हैं,

त्रुटियाँ कम करते हैं और कोड को अधिक पठनीय बनाते हैं।

Loops can execute a block of code as long as a specified condition is reached. Loops are handy because they save time, reduce errors, and they make code more readable.

Type of loop in C

- While loop
- Do while loop
- For loop
- Nested loop
- Infinite loop

While Loop in C

जब तक निर्दिष्ट स्थिति true है, while लूप कोड के एक ब्लॉक के माध्यम से लूप करता है:

The while loop loops through a block of code as long as a specified condition is true:

Syntax :-

```
while (condition) {
    // code block to be executed
}
```

Example 1 :-

```
int i = 0;
while (i < 5) {
    printf("%d\n", i);
    i++;
}
```

Example 2 :-

```
int i = 0;
while (i < 5) {
    printf("goalyaan\n");
    i++;
}
```

Note:

Do not forget to increase the variable used in the condition (i++), otherwise the loop will never end!

कंडीशन (i++) में प्रयुक्त वेरिएबल को बढ़ाना न भूलें, अन्यथा लूप कभी खत्म नहीं होगा!

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

do/while लूप, while लूप का एक प्रकार है। यह लूप एक बार कोड ब्लॉक को निष्पादित करेगा, यह जांचने से पहले कि क्या स्थिति सत्य है, उसके बाद यह लूप को तब तक दोहराएगा जब तक कि स्थिति सत्य है।

Syntax :-

```
do {
    // code block to be executed
} while (condition);
```

Example :-

```
#include <stdio.h>
int main() {
    int i = 0;
    // do while loop
    do {
        printf("Goalyaan\n");
        i++;
    } while (i < 3);
    return 0;
}
```

Break and Continue in While Loop

आप while लूप में break और continue का भी उपयोग कर सकते हैं:

You can also use break and continue in while loops.

Break Example :-

```
int i = 0;
while (i < 10) {
    if (i == 4) {
        break;
    }
    printf("%d\n", i);
    i++;
}
```

Continue Example :-

```
int i = 0;
while (i < 10) {
    if (i == 4) {
        i++;
        continue;
    }
    printf("%d\n", i);
    i++;
}
```

For Loop in C

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop.

जब आपको पता हो कि आपको कोड के किसी ब्लॉक को कितनी बार लूप करना है, तो while लूप के स्थान पर for लूप का उपयोग करें।

Syntax :-

```
for (expression1; expression2; expression3) {
    // code block to be executed
}
```

Example 1 :-

```
int i;
for (i = 0; i < 5; i++) {
    printf("%d\n", i);
}
```

Example 2 :-

```
int i;
for (i = 0; i < 5; i++) {
    printf("goalyaan\n");
}
```

Break and Continue in For Loop

Break :-

You have already seen the break statement used in the switch statement. It was used to "jump out" of a switch statement. The break statement can also be used to jump out of a loop.

आपने पहले ही switch स्टेटमेंट में इस्तेमाल किए गए break स्टेटमेंट को देखा है। इसका इस्तेमाल switch स्टेटमेंट से "jump out" करने के लिए किया जाता था। break स्टेटमेंट का इस्तेमाल लूप से बाहर निकलने के लिए भी किया जा सकता है।

Example :-

```
int i;
for (i = 0; i < 10; i++) {
    if (i == 4) {
        break;
    }
    printf("%d\n", i);
}
```

Continue :-

यदि कोई continue स्थिति होती है, तो जारी कथन एक iteration को तोड़ देता है और अगले iteration के साथ जारी रहता है।

The continue statement breaks one iteration (in the loop) if a specified condition occurs, and continues with the next iteration.

Example :-

```
int i;
for (i = 0; i < 10; i++) {
    if (i == 4) {
        continue;
    }
    printf("%d\n", i);
}
```

Nested Loops - नेस्टेड लूप्स

It is also possible to place a loop inside another loop. This is called a nested loop. The "inner loop" will be executed one time for each iteration of the "outer loop".

एक लूप को दूसरे लूप के अंदर रखना भी संभव है। इसे नेस्टेड लूप कहा जाता है। "आंतरिक लूप" को "बाहरी लूप" की प्रत्येक पुनरावृत्ति के लिए एक बार निष्पादित किया जाएगा।

Syntax :-

```
for (initialization; condition; increment) {
    for (initialization; condition; increment) {
        // statement of inside loop
    }
    // statement of outer loop
}
```

Example 1 :-

```
int i, j;
// Outer loop
for (i = 1; i <= 2; ++i) {
    printf("Outer: %d\n", i); // Executes 2 times
    // Inner loop
    for (j = 1; j <= 3; ++j) {
        printf(" Inner: %d\n", j); // Executes 6 times (2 * 3)
    }
}
```

Example 2 :-

```
int i, j;
for (i = 1; i <= 5; i++) {
    for (j = 1; j <= i; j++) {
        printf("%d ", j);
    }
    printf("\n");
}
```

Example 3 :-

```
int i, j;
for (i = 1; i <= 5; i++) {
    for (j = 1; j <= i; j++) {
        printf("* ");
    }
    printf("\n");
}
```

Infinite Loops - इनफिनाइट लूप

An infinite loop is a looping construct that does not terminate the loop and executes it forever. It is also called an indefinite loop or endless loop. It either has continuous output or no output.

एक अनंत लूप वह संरचना है जो कभी समाप्त नहीं होती और लूप को हमेशा के लिए चलाती रहती है। इसे अनिश्चितकालीन लूप या अंतहीन लूप भी कहा जाता है।

Example 1 :-

```
for (;;) {
    printf("GoalYaan");
}
```

Example 2 :-

```
int i = 0;
while (i < 5) {
    printf("goalyaan\n");
}
```

(Arrays) सारणियाँ :-

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. To create an array, define the data type (like int) and specify the name of the array followed by square brackets [].

Arrays का उपयोग एक ही वेरिएबल में कई मानों को संग्रहीत करने के लिए किया जाता है।

Example :-

```
int marks[] = {25, 50, 75, 100};
printf("%d", marks[0]); // Outputs 25
```

Set Array Size :-

Another common way to create arrays is to specify the size and add elements later.

```
int myNumbers[4];
myNumbers[0] = 25;
myNumbers[1] = 50;
myNumbers[2] = 75;
myNumbers[3] = 100;
printf("%d", myNumbers[0]);
```

Change an Array Element :-

```
int marks[] = {25, 50, 75, 100};
marks[0] = 33;
printf("%d", marks[0]);
```

Loop through an Array :-

You can loop through array elements with the for loop.

```
int marks[] = {25, 50, 75, 100};
for (int i = 0; i < 4; i++) {
    printf("%d\n", marks[i]);
}
```

Functions in C

फ़ंक्शन कोड का एक ब्लॉक है जो केवल तभी रन करता है जब इसे कॉल किया जाता है।

आप किसी फ़ंक्शन में डेटा, जिसे parameters के रूप में जाना जाता है, पास कर सकते हैं।

फ़ंक्शन्स का उपयोग कुछ कार्य करने के लिए किया जाता है, और वे कोड का पुनः उपयोग करने के लिए महत्वपूर्ण हैं: कोड को एक बार परिभाषित करें, और इसे कई बार उपयोग करें।

A function is a block of code which only runs when it is called.
You can pass data, known as parameters, into a function.
Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

Predefined Functions

For example, main() is a function, which is used to execute code, and printf() is a function; used to output/print text to the

screen.

उदाहरण के लिए, main() एक फ़ंक्शन है, जिसका उपयोग कोड निष्पादित करने के लिए किया जाता है, और printf() एक फ़ंक्शन है; जिसका उपयोग स्क्रीन पर टेक्स्ट आउटपुट/प्रिंट करने के लिए किया जाता है:

Example :-

```
int main() {
    printf("Hello World!");
    return 0;
}
```

User Defined Function

To create (often referred to as declare) your own function, specify the name of the function, followed by parentheses () and curly brackets {}:

अपना स्वयं का फ़ंक्शन बनाने (जिसे अक्सर घोषित करना कहा जाता है) के लिए, फ़ंक्शन का नाम निर्दिष्ट करें, उसके बाद कोष्ठक () और घुमावदार कोष्ठक {} लगाएं।

Syntax :-

```
returnType functionName(parameter1, parameter2, parameter3) {
    // code to be executed
}
```

Return Type

The void keyword indicates that the function should not return a value. If you want the function to return a value, you can use

a data type (such as int or float) instead of void, and use the return keyword inside the function.

void कीवर्ड यह संकेत देता है कि फ़ंक्शन को कोई मान नहीं लौटाना चाहिए। यदि आप चाहते हैं कि फ़ंक्शन कोई मान लौटाए, तो आप void के बजाय डेटा प्रकार (जैसे int या float) का उपयोग कर सकते हैं, और फ़ंक्शन के अंदर return का उपयोग कर सकते हैं।

Using void Type

Example 1 :-

```
#include <stdio.h>

// Create a function
void myFunction() {
    printf("I just got executed!");
}

int main() {
    myFunction(); // call the function
    return 0;
}
```

Example 2 :-

```
void calculateSum(int x, int y) {
    int sum = x + y;
    printf("The sum of %d + %d is: %d", x, y, sum);
}

int main() {
    calculateSum(5, 8); // call the function
    return 0;
}
```

Example 3 :-

```
void myFunction(char name[], int age) {
    printf("Hello %s. You are %d years old.\n", name, age);
}

int main() {
    myFunction("Liam", 3);
    myFunction("Jenny", 14);
    myFunction("Anja", 30);
    return 0;
}
```

Return Values

Example 1 :-

```
int myFunction(int x) {
    return 5 + x;
}

int main() {
    printf("Result is: %d", myFunction(3));
    return 0;
}
```

Example 2 :-

```
int myFunction(int x, int y) {
    return x + y;
}

int main() {
    printf("Result is: %d", myFunction(5, 3));
    return 0;
}
```

Standard Library of C Functions in Arduino IDE

The Arduino IDE provides access to a large group of standard C libraries through the use of the #include directive. This gives programmers access to pre-made functions, as well as libraries written specifically for Arduino.

Arduino IDE #include निर्देश के उपयोग से मानक C पुस्तकालयों के एक बड़े समूह तक पहुँच प्रदान करता है।

void setup() Function

In Arduino, the void setup() function is used to initialize variables, set pin modes, and start using libraries at the beginning of a program.

Arduino में, void setup() फ़ंक्शन का उपयोग वेरिएबल्स को प्रारंभ करने, पिन मोड सेट करने और लाइब्रेरीज़ शुरू करने के लिए किया जाता है।

Example :-

```
int button = 5; // button connected to pin 5
int LED = 6;    // LED connected to pin 6

void setup() {
    pinMode(button, INPUT);  // set the digital pin as input
    pinMode(LED, OUTPUT);    // set the digital pin as output
}
```

pinMode() Function

The pinMode() function is used to configure a specific pin to behave either as an input or an output.

pinMode() फ़ंक्शन का उपयोग किसी पिन को इनपुट या आउटपुट के रूप में सेट करने के लिए किया जाता है।

Syntax :-

```
void setup() {
    pinMode(pin, mode);
}
```

void loop() Function

In Arduino, the void loop() function is a required part of every sketch that contains the main code. It runs repeatedly until the Arduino is powered off or reset.

Arduino में, void loop() फ़ंक्शन हर स्केच का आवश्यक हिस्सा होता है जो तब तक चलता है जब तक Arduino बंद या रीसेट नहीं हो जाता।

Example :-

```
void loop() {
    if (digitalRead(button) == HIGH) { // if button pressed
        digitalWrite(LED, HIGH); // turn on LED
        delay(500);
        digitalWrite(LED, LOW);  // turn off LED
        delay(500);
    }
}
```

digitalWrite() Function

The digitalWrite() function is used to write a HIGH or LOW value to a digital pin.

digitalWrite() फ़ंक्शन का उपयोग डिजिटल पिन पर HIGH या LOW मान लिखने के लिए किया जाता है।

Syntax :-

```
void loop() {
    digitalWrite(pin, value);
}
```

digitalRead() Function

Reads the value from a specified digital pin, to check either HIGH or LOW.

उच्च या निम्न की जाँच करने के लिए निर्दिष्ट डिजिटल पिन से मान पढ़ता है।

Syntax :-

```
digitalRead(pin);
```

analogRead() Function

By using the **analogRead()** function, we can read the status of any sensor applied to one of the pins.

analogRead() फ़ंक्शन का उपयोग करके, हम किसी सेंसर की स्थिति पढ़ सकते हैं।

Syntax :-

```
analogRead(pin);
```

delay() Function

Pauses the program for the specified amount of time (in milliseconds). There are 1000 milliseconds in one second.

प्रोग्राम को निर्धारित समय (मिलीसेकंड में) के लिए रोक देता है।

Syntax :-

```
delay(ms);
```

Communication Functions - कम्यूनिकेशन फ़ंक्शन

The Arduino's primary communication method allows it to send and receive data with a computer or other devices via a serial connection.

Arduino की प्राथमिक संचार विधि इसे कंप्यूटर या अन्य उपकरणों के साथ सीरियल कनेक्शन के माध्यम से डेटा भेजने और प्राप्त करने की अनुमति देती है।

Serial Communication

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (UART/USART).

Arduino बोर्ड और कंप्यूटर या अन्य उपकरणों के बीच संचार के लिए उपयोग किया जाता है।

USART = Universal Synchronous Asynchronous Receiver Transmitter

UART = Universal Asynchronous Receiver/Transmitter

Serial.begin() Function

Everything begins with the Serial.begin() function. This function in the setup() routine is executed only once, that is, when the Arduino is starting.

सब कुछ Serial.begin() से शुरू होता है। यह केवल एक बार चलता है जब Arduino शुरू होता है।

Example :-

```
void setup() {
    Serial.begin(9600); // set up serial library baud rate
}
```

Serial.print() and Serial.println()

Both print data to the serial port as human-readable ASCII text. The println() version also adds a carriage return and a new line.

दोनों डेटा को सीरियल पोर्ट पर ASCII टेक्स्ट के रूप में प्रिंट करते हैं। println() नया लाइन जोड़ता है।

Example :-

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.print("Welcome to GoalYaan");
    Serial.println("Welcome to GoalYaan"); // for new line
}
```

Chapter 5 _ Security and Future of IoT Ecosystem

(IoT पारिस्थितिकी तंत्र की सुरक्षा और भविष्य)

◈ Need of Security in IoT _ Why Security?

IoT में सुरक्षा की आवश्यकता – क्यों ज़रूरी है सुरक्षा?

Every connected device creates opportunities for attackers. These vulnerabilities are broad, even for a single small device. The risks include data transfer, device access, malfunctioning devices, and always-on/always-connected devices.

हर कनेक्टेड डिवाइस हमलावरों के लिए एक अवसर बनाता है। ये कमजोरियाँ व्यापक होती हैं, यहाँ तक कि एक छोटे उपकरण के लिए भी। जोखिमों में डेटा ट्रांसफर, डिवाइस एक्सेस, डिवाइस खराब होना और हमेशा चालू/कनेक्टेड रहना शामिल हैं।

The main challenge is the limitation of security in low-cost devices and the growing number of devices creating more attack opportunities.

मुख्य चुनौती कम लागत वाले उपकरणों की सुरक्षा सीमाएँ और उपकरणों की बढ़ती संख्या है जो हमलों के अधिक अवसर पैदा करती है।

◈ Privacy for IoT Enabled Devices

IoT सक्षम उपकरणों में गोपनीयता

1. Secure the IoT Network

   Protect the IoT network by using antivirus, anti-malware, firewalls, and intrusion

detection systems.

IoT नेटवर्क को सुरक्षित रखें — एंटीवायरस, एंटी-मालवेयर, फायरवॉल, और घुसपैठ रोकथाम सिस्टम का उपयोग करें।

2. Authenticate the IoT Devices

Use two-factor authentication, digital certificates, and biometrics.

IoT डिवाइस को प्रमाणित करें — दो-कारक प्रमाणीकरण, डिजिटल प्रमाणपत्र और बायोमेट्रिक्स का उपयोग करें।

3. Use IoT Data Encryption

Encrypt data at rest and in transit using cryptographic algorithms.

IoT डेटा एन्क्रिप्शन का उपयोग करें – स्थिर और ट्रांजिट डेटा को एन्क्रिप्ट करें।

4. Use IoT PKI Security Methods

Use X.509 digital certificates and public/private key management.

IoT PKI सुरक्षा विधियों का उपयोग करें – X.509 डिजिटल प्रमाणपत्र और सार्वजनिक/निजी कुंजी प्रबंधन लागू करें।

5. Use IoT Security Analytics

Use tools that can detect IoT-specific attacks that traditional firewalls can't.

IoT सुरक्षा एनालिटिक्स का उपयोग करें – ऐसे टूल्स का उपयोग करें जो IoT-विशिष्ट हमलों का पता लगा सकें।

6. Use IoT API Security Methods

Secure data movement between devices and systems using REST-based APIs.

IoT API सुरक्षा विधियों का उपयोग करें – केवल अधिकृत डिवाइस और ऐप्स को API तक पहुँचने दें।

◈ IoT Security for Consumer Devices

उपभोक्ता उपकरणों के लिए IoT सुरक्षा

IoT security requires Identification, Authentication, Authorization, and Monitoring.

IoT सुरक्षा के लिए **पहचान, प्रमाणीकरण, प्राधिकरण, और निगरानी** आवश्यक हैं।

1. Identification – पहचान

Devices must have secure identifiers like serial numbers.

उपकरणों के पास सुरक्षित पहचानकर्ता होने चाहिए जैसे सीरियल नंबर।

2. Authentication – प्रमाणीकरण

Establish a trusted connection between devices

and networks.

उपकरण और नेटवर्क के बीच एक विश्वसनीय कनेक्शन स्थापित करें।

3. Authorization _ प्राधिकरण

Define what actions each device is allowed to perform.

प्रत्येक डिवाइस को अनुमति प्राप्त कार्यों को परिभाषित करें।

4. Monitoring _ निगरानी

Detect unusual data patterns to prevent breaches.

असामान्य डेटा पैटर्न की पहचान करें और सुरक्षा उल्लंघन रोकें।

◈ Security Levels _ सुरक्षा स्तर

1. **Low-Level Attack _ निम्न स्तर का हमला**

Attack attempt fails.

हमला करने का प्रयास विफल हो जाता है।

2. **Medium-Level Attack _ मध्यम स्तर का हमला**

Attacker only listens but doesn't change data.

हमलावर केवल सुनता है, डेटा नहीं बदलता।

3. **High-Level Attack _ उच्च स्तर का हमला**

Attacker modifies or corrupts data.

हमलावर डेटा में बदलाव करता है या उसे भ्रष्ट करता है।

4. **Extremely High-Level Attack _ अत्यधिक उच्च स्तर का हमला**

Unauthorized access, illegal operation, or DDoS attack.

अनधिकृत प्रवेश, अवैध कार्य या DDoS हमला।

◆ Protecting IoT Devices – IoT उपकरणों की सुरक्षा

1. **Keep Changing Your Password _ पासवर्ड बदलते रहें**

Regularly change passwords for all devices.

सभी उपकरणों के पासवर्ड समय-समय पर बदलें।

2. **Update Your Device _ अपने उपकरण अपडेट रखें**

Enable automatic updates and install security patches.

ऑटो अपडेट सक्षम करें और सुरक्षा पैच इंस्टॉल करें।

3. **Reduce Use of Cloud _ क्लाउड का उपयोग कम करें**

Avoid unnecessary cloud storage to prevent hacking.

हैकिंग से बचने के लिए क्लाउड का कम उपयोग करें।

4. **Use Secondary Network** _ अलग नेटवर्क का उपयोग करें

Use a separate Wi-Fi for IoT devices.

IoT उपकरणों के लिए अलग वाई-फाई नेटवर्क का उपयोग करें।

◈ Botnet – बॉटनेट

A Botnet is a network of Internet-connected devices (computers, servers, mobiles, IoT) used for spam, fraud, or DDoS attacks.

बॉटनेट इंटरनेट से जुड़े उपकरणों (कंप्यूटर, सर्वर, मोबाइल, IoT) का समूह होता है, जिसका उपयोग स्पैम, धोखाधड़ी या DDoS हमलों के लिए किया जाता है।

The term comes from Robot + Network.

यह शब्द **Robot** और **Network** से मिलकर बना है।

Future in IoT Ecosystem – IoT ईकोसिस्टम का भविष्य

The future of IoT has the potential to be limitless. The IoT is considered the future of the Internet. In future, the IoT will completely change our living styles and business models. It will allow people and devices to communicate anytime, anywhere, using any network or service.

IoT का भविष्य असीमित संभावनाओं से भरा हुआ है। इसे इंटरनेट का भविष्य माना जाता है। भविष्य में IoT हमारी जीवनशैली और व्यवसाय मॉडल को पूरी तरह बदल देगा। यह लोगों और उपकरणों को किसी भी समय, कहीं भी, किसी भी नेटवर्क या सेवा का उपयोग करते हुए संवाद करने की अनुमति देगा।

The main goal of IoT is to create a superior world for human beings.

IoT का मुख्य उद्देश्य मानवों के लिए एक बेहतर दुनिया बनाना है।

◈ Key Advantages of Future IoT Ecosystem – भविष्य के IoT पारिस्थितिकी तंत्र के लाभ

1. **Artificial Intelligence will become smarter than ever (AI पहले से ज्यादा स्मार्ट हो जाएगा)**

   In the IoT environment, devices like smart lights, coffee makers, and home hubs collect user data and analyze habits for better business decisions.

   IoT वातावरण में, घरेलू उपकरण जैसे लाइटिंग सिस्टम, कॉफी मेकर, स्मार्ट होम हब उपयोगकर्ता की आदतों और पैटर्न का डेटा एकत्र करते हैं, जिसे बेहतर व्यावसायिक निर्णयों के लिए विश्लेषित किया जाता है।

2. **High-speed Internet will push the growth further (हाई-स्पीड इंटरनेट विकास को बढावा देगा)**

Companies will launch high-speed data transmission technologies that will make networks faster and improve IoT performance.

कंपनियाँ हाई-स्पीड डेटा ट्रांसमिशन तकनीकें लाएँगी, जिससे नेटवर्क और तेज़ होंगे और IoT का प्रदर्शन बेहतर होगा।

3. **Security concerns will increase (सुरक्षा चिंताएँ बढ़ेंगी)**

As the number of IoT devices increases, so will the risk of security and privacy issues.

IoT उपकरणों की संख्या बढ़ने से, सुरक्षा और गोपनीयता संबंधी चिंताएँ भी बढ़ेंगी।

4. **Cities will become smarter (शहर और स्मार्ट बनेंगे)**

IoT will enable smart cities to automate data collection from cameras, kiosks, and parking systems, saving time and money.

IoT के माध्यम से स्मार्ट शहरों में कैमरा, पार्किंग स्टेशन, और अन्य सेंसर से डेटा स्वचालित रूप से एकत्र किया जा सकेगा।

◈ Need of Powerful Core for Building Secure Algorithms – सुरक्षित एल्गोरिद्म के निर्माण के लिए शक्तिशाली कोर की आवश्यकता

To protect user privacy and prevent IoT data breaches, encryption and key management must be used.

उपयोगकर्ता की गोपनीयता की सुरक्षा के लिए, मानक क्रिप्टोग्राफिक एल्गोरिद्म और एन्क्रिप्शन तकनीकों का उपयोग आवश्यक है।

Encrypt data both at rest and during transfer.

डेटा को स्टोरेज और ट्रांसफर दोनों समय पर एन्क्रिप्ट करें।

◈ What is Cryptography – क्रिप्टोग्राफी क्या है?

Cryptography is the technique of securing information through the use of codes, allowing only authorized persons to access it.

क्रिप्टोग्राफी एक तकनीक है जिसमें कोड का उपयोग कर सूचना को सुरक्षित बनाया जाता है, ताकि केवल अधिकृत व्यक्ति ही इसे समझ सके।

◈ Types of Cryptography – क्रिप्टोग्राफी के प्रकार

1. Symmetric Key Cryptography – सममित कुंजी क्रिप्टोग्राफी

Both sender and receiver use the same key to encrypt and decrypt data.
भेजने वाला और प्राप्तकर्ता दोनों एक ही कुंजी का उपयोग करते हैं।

Examples: **DES (Data Encryption Standard)** and **AES (Advanced Encryption Standard)**.

2. Asymmetric Key Cryptography – असममित कुंजी क्रिप्टोग्राफी

Uses two keys: a public key for encryption and a private key for decryption.
Even if the public key is known, only the private key holder can decrypt it.
उदाहरण: RSA (Rivest_Shamir_Adleman) Algorithm.

◈ Examples for New Trends – IoT में नई तकनीकी प्रवृत्तियाँ (AI, ML Integration)

Artificial Intelligence (AI) and Machine Learning (ML) are software techniques that make computers intelligent — they can analyze data and make decisions like humans.

AI और ML ऐसी तकनीकें हैं जो कंप्यूटर को इंसानों जैसी बुद्धिमत्ता देती हैं।

◈ Subsets of Artificial Intelligence – कृत्रिम बुद्धिमत्ता के उपसमूह

1. **Machine Learning (मशीन लर्निंग):**
   Computer programs automatically learn from data without human help.
   कंप्यूटर प्रोग्राम स्वयं सीख सकते हैं बिना किसी मानव सहायता के।

2. **Deep Learning (डीप लर्निंग):**
   Learns from large amounts of unstructured data (like text, images, videos).
   बड़ी मात्रा के डेटा से सीखने की क्षमता रखता है।

3. **Neural Networks (न्यूरल नेटवर्क):**
   Computer systems inspired by the human brain; made up of interconnected layers called neurons.
   मानव मस्तिष्क की तरह कार्य करने वाले एल्गोरिद्म का समूह।

4. **Natural Language Processing (प्राकृतिक भाषा संसाधन):**
   Machines understand, interpret, and respond to human language.
   मशीनें मानव भाषा को पढ़ती और समझती हैं।

5. **Computer Vision (कंप्यूटर विजन):**

Computers analyze and understand images to make decisions.

कंप्यूटर छवियों को पहचानने और समझने में सक्षम होते हैं।

◈ Machine Learning (ML) – मशीन लर्निंग

Machine Learning is a subset of AI that allows systems to improve automatically through data and experience.

मशीन लर्निंग AI की शाखा है जो सिस्टम को डेटा और अनुभव से खुद सीखने में सक्षम बनाती है।

It builds models using training data to make predictions or decisions.

यह प्रशिक्षण डेटा के आधार पर भविष्यवाणियाँ और निर्णय लेती है।

The term Machine Learning was coined in 1959 by Arthur Samuel (IBM).

मशीन लर्निंग" शब्द की शुरुआत 1959 में आर्थर सैमुअल (IBM) ने की थी।

◈ Difference between AI and ML – AI और मशीन लर्निंग में अंतर

Artificial Intelligence (AI)	Machine Learning (ML)
Enables a machine to simulate human behavior.	Allows a machine to learn from data automatically.
एआई मशीन को मानव व्यवहार की नकल करने की क्षमता देता है।	एमएल मशीन को डेटा से स्वतः सीखने की क्षमता देता है।
Goal: Build smart systems like humans.	Goal: Learn from data to produce accurate results.
उद्देश्य: इंसानों जैसी स्मार्ट सिस्टम बनाना।	उद्देश्य: डेटा से सीखकर सटीक परिणाम देना।
AI performs any human-like task.	ML performs specific tasks through training.
एआई किसी भी इंसानी कार्य को कर सकता है।	एमएल विशेष कार्यों के लिए प्रशिक्षित होता है।
Includes ML and Deep Learning.	Deep Learning is a part of ML.
एआई में एमएल और डीप लर्निंग दोनों शामिल हैं।	डीप लर्निंग एमएल का उपभाग है।
Broad scope and applications.	Limited scope compared to AI.

Artificial Intelligence (AI)	Machine Learning (ML)
एआई का दायरा बहुत व्यापक है।	एमएल का दायरा सीमित है।
Examples: Siri, Alexa, chatbots, expert systems, humanoid robots.	Examples: Google search, recommender systems, Facebook tagging.