Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

Embedded systems

Lecture 9 :

IOT in Embedded System

Prof.Dr. Mehdi Ebady Manaa

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

Introduction

- The world is now fully embracing the power of the Internet of Things (IoT). As consumers, we are surrounded by the Internet of Things, which has permeated various applications, including smart homes, smart industries, and even the agricultural sector. The IoT ecosystem continues to evolve, offering increasingly advanced and innovative solutions.
- However, for engineers, there can often be confusion when distinguishing between Embedded System programming and IoT programming, given the vast nature of IoT. It’s like navigating a complex puzzle.

What is Internet of Things (IOT)?

- IoT stands for Internet of Things and refers to the interconnected nature of devices that we use in our daily lives. These days "smart" devices can connect to the internet and to other devices to facilitate everyday actions. For example, a smart fridge can detect the items in it as well as expiry dates and provide information to the owner about it. This is a typical use-case for IoT in everyday use.
- An embedded system is a tiny computer that was built for a custom purpose. Such a computer usually has a microprocessor or microcontroller, which is an integrated circuit that typically contains processor, memory, and I/O (input/output) peripherals on one chip.
- The defining factor of an embedded system is that it can carry out some form of digital processing. This is what distinguishes it from plain hardware devices that contain only some circuitry and possibly a battery.
- An embedded system often has sensors that let it monitor environmental factors.

Internet of Things (IOT)

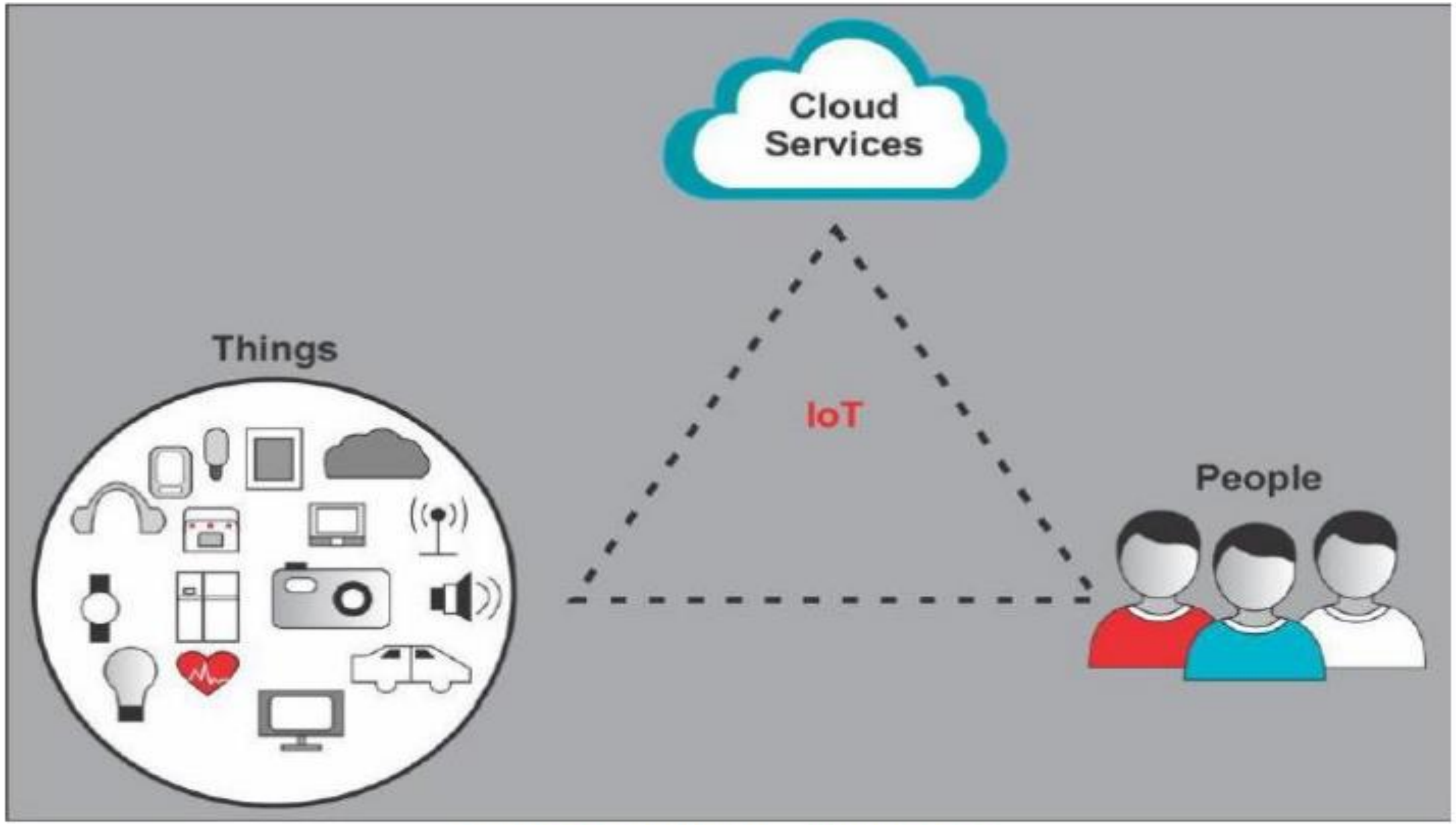


Figure: Main components of IoT

Internet of Things (IOT)

- IOT is considered as a scenario of accessing any information from anywhere and accessible to everyone. This is described as follows:
- Anything: Eventually, any device, appliance or entity will be seamlessly connected to the Internet. Connectivity will not be the main feature of the device, but will extend the device’s capabilities.
- **Anywhere:** Any conceived wireless connectivity framework should be abstract enough to run from any location - both geographically and from a network topology perspective. The former refers to Internet-based ubiquity; the latter, refers to the ability to clone the framework into intranet environments where Internet access is limited or undesired. Acknowledging the structure of the Internet beyond the public domain is important to enable the expansion of the IoT paradigm.
- **Anyone:** Currently, not all things are connected to the IoT. But an IoT ecosystem that is easy to use and secure is not that far away. This will make the IoT accessible to anyone. Anyonewill be able to connect their product to the Internet, and also customize it to their personalpreferences.

What is an Embedded System in IoT?

•Embedded Systems are a combination of hardware and software designed for specific tasks.

•In IoT, they consist of:

•**Customized Hardware** tailored to specific system requirements.

•**Embedded Firmware** (software) that powers the hardware to deliver desired functionality.

•The distinction between Embedded Systems and IoT lies in their scope: Embedded Systems focus on integrated hardware-software solutions, while IoT involves interconnected devices and data exchange.

What is an Embedded System in IoT?

- Compared to General Purpose System (GPS), Embedded Systems (ES) are constrained in terms of resources. This means it will have lesser computing power, memory and lesser peripherals compared to a General Purpose System (GPS). For example your laptop might have 4GB of RAM (which is a GPS) whereas your wearable device (ES) might hardly have 200 MB of RAM. This makes programming ES much different and challenging than GPS.
- Let us take an example of mobile phones of a particular brand. They will release mobile phones in different models. Each of them will have a specific hardware specification (storage space, screen type, resolution, memory size, camera pixels etc…).
- The software requirements (Version of OS say Android, Number of applications, Additional security features) purely depend on the hardware capability. What applies to model X will not be applicable for model Y, which exactly is nothing but the customized nature of Embedded System.

Embedded System - With and Without OS

- Taking a step further, Embedded systems can be with or without an Operating System (OS) running in it. The devices that run those functions are elementary and the control which is not challenging are the **embedded system without operating system.**
- For example, think about a simple temperature sensing device. The functionality of Embedded Software is very simple. It has to read the sensor data and display them in the output, say Seven Segment Display (SSD).
- The programming logic and coding is very simple in nature with a single **task / application** running in the system. Hence this can be a system running without an OS. From a hardware perspective a simple **micro-controller connected with peripheral devices** would be sufficient for this.

Embedded System - With and Without OS

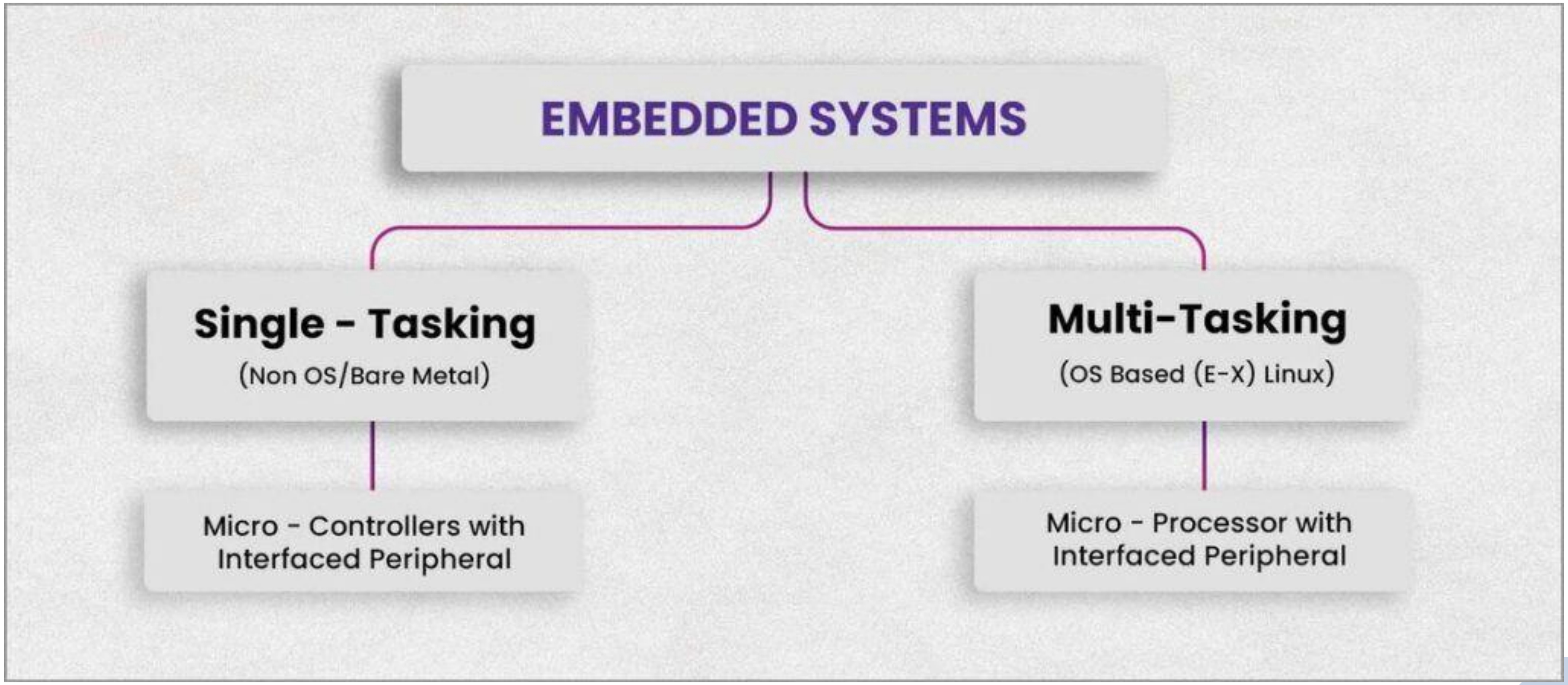

EMBEDDED SYSTEMS
Single - Tasking
(Non OS/Bare Metal)
Multi-Tasking
(OS Based (E-X) Linux)
Micro - Controllers with Interfaced Peripheral
Micro - Processor with Interfaced Peripheral

Figure : Embedded Systems Classification

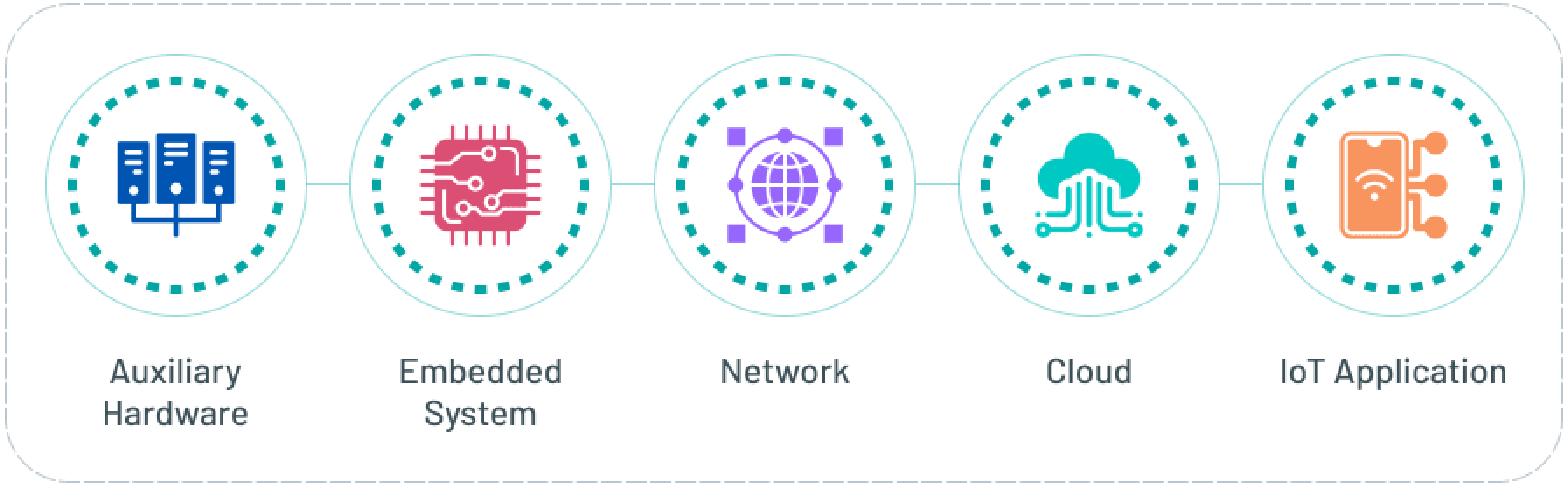
Embedded System - With and Without OS

- On the contrary, take another example, say a home set-top box. This device has to do **multiple tasks or applications** in terms of streaming, storing, channel handling, subscription information, user information and many other things.
- In order to achieve them an Operating System is required which can enable multi-tasking by managing resources efficiently. From a hardware perspective this will have more **powerful microprocessors with peripheral devices with higher capacity**.
- These operating systems are typically called Embedded **Operating Systems (EOS)**. Examples of EOS include Linux, Android and Symbian. If the nature of the application has a higher time constraint (ex: Robots) then the EOS requires additional features. Such operating systems are called **Real Time Operating Systems (RTOS).** Some of the examples of RTOS include FreeRTOS, Mbed OS and VxWorks.

Understanding IoT Embedded Systems

- To grasp the concept of IoT embedded systems, we need to understand what they are and their key components. IoT embedded systems are the combination of physical objects and embedded systems technology that enable them to connect and communicate with each other and the internet. These systems consist of sensors, microcontrollers, and communication protocols that facilitate data collection, processing, and transmission. Examples of IoT applications include smart homes, healthcare systems, and industrial automation processes, which rely on the integration of physical devices with embedded systems.

The Role of Embedded Systems in IoT



The Role of Embedded Systems in IoT

- POS terminals
- Point-of-Sale terminals and self-service checkout terminals are a common use of an IoT embedded system. These touch-enabled IoT embedded systems connect to payment gateways via the internet to process payments. These devices have a scanner attached to them, and can also receive input from a pin pad and card reader.
- Vending machines
- An IoT embedded system in the form of a vending machine can send messages alerting employees on the ground when items need to be refilled. The devices receive input from a coin slot that should be able to read the type of coin inserted. Further inputs include a touch screen and credit card reader.
- Ticket scanners
- Event venues can have a ticket scanner IoT embedded system in place that reads ticket information from visitors, compares it with a local or online database, and then engages a turnstile to let the visitor in.
- These IoT embedded systems can be further programmed to send attendance data to an analytics platform, which can then process the data and provide detailed insight to event organizers.

Advancements in IoT Embedded Systems

- Recent advancements in technology have propelled the progress of IoT embedded systems. One significant advancement is the improvement in IoT hardware, including microcontrollers, sensors, and actuators. These components have become smaller, more powerful, and more energy-efficient, allowing for more sophisticated and efficient IoT applications. Additionally, software innovations such as operating systems, machine learning, and edge computing have enabled more intelligent and autonomous IoT systems. Integration with artificial intelligence and machine learning algorithms has further enhanced the capabilities and decision-making capabilities of IoT embedded systems.
- Case studies of successful IoT embedded system implementations provide valuable insights into the practical applications of these advancements. For example, in the healthcare industry, IoT embedded systems have revolutionized patient monitoring, enabling remote monitoring and early intervention. In the industrial sector, IoT embedded systems have improved efficiency and productivity by enabling real-time monitoring and predictive maintenance.

Challenges Faced by IoT Embedded Systems

- Despite the advancements, there are several challenges that IoT embedded systems face. One significant challenge is security concerns. The interconnected nature of IoT devices increases the risk of data breaches and privacy issues. Addressing these challenges requires implementing robust security measures, including encryption, authentication, and secure communication protocols.
- Scalability and interoperability are also challenges for IoT embedded systems. As the number of IoT devices increases, managing and coordinating their interactions becomes more complex. The lack of standardized communication protocols can hinder interoperability between devices from different manufacturers. Addressing these challenges requires the development and adoption of IoT standards and alliances.
- Power consumption and energy efficiency are vital considerations in IoT embedded systems. As devices become more numerous and widespread, optimizing energy usage becomes crucial to prolong battery life and reduce environmental impact. Techniques such as power management, sleep modes, and using renewable energy sources help improve energy efficiency in IoT embedded systems.

Challenges Faced by IoT Embedded Systems

- Reliability and maintenance are additional challenges. IoT devices need to be reliable and perform consistently in various environments. Additionally, the maintenance of IoT systems can be challenging due to their distributed nature and the need for over-the-air updates. Implementing reliable hardware and software, as well as developing efficient maintenance strategies, can help overcome these challenges.
- Compliance with regulatory requirements is another hurdle for IoT embedded systems. Different regions and industries have specific regulations and standards that need to be met. Therefore, ensuring compliance while developing and deploying IoT solutions is a critical consideration.

How does Internet of Thing (IoT) Work?

- The working of IoT is different for different IoT echo system (architecture). However, the key concept of there working are similar. The entire working process of IoT starts with the device themselves, such as smartphones, digital watches, electronic appliances, which securely communicate with the IoT platform. The platforms collect and analyze the data from all multiple devices and platforms and transfer the most valuable data with applications to devices.

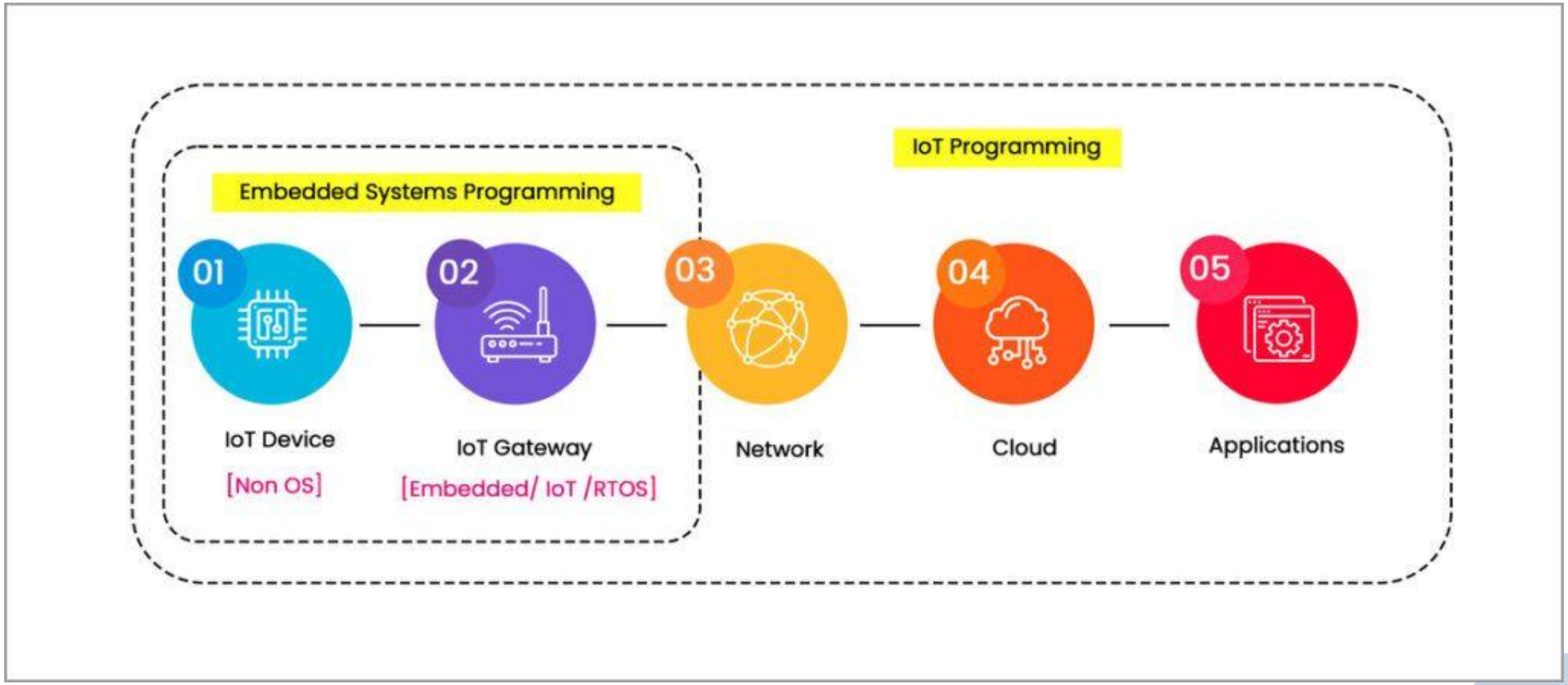
Key Differences between Embedded Systems and IoT

- **1. Scope and Connectivity:**
- Embedded systems work independently, focusing on specific tasks without needing internet connection, prioritizing reliability. On the other hand, the Internet of Things and **embedded systems** relies on internet connectivity, allowing devices to communicate, share data, and be controlled globally for diverse applications.
- **2. Data Handling and Processing:**
- Embedded systems handle data locally, crucial for tasks needing low latency, like a car’s airbag system. In contrast, IoT devices collect data and send it to central servers or the cloud for in-depth analysis and advanced processing, expanding their capabilities.

Key Differences between Embedded Systems and IoT

- **3. Flexibility and Upgradability:**
- Embedded systems are less flexible and hard to change after manufacturing. IoT devices, on the other hand, can be easily updated remotely, allowing for continuous improvement and adaptation to new features and security measures.
- **4. Security Considerations:**
- Embedded systems are generally more secure because they are localized, but they can be vulnerable to physical attacks. IoT and embedded system devices, connected to the internet, face more security challenges, including cyberattacks. To protect them, strong security measures like encryption and regular updates are essential.
- **5. Applications and Use Cases:**
- Embedded systems excel in real-time control for applications like medical devices and industrial automation. IoT, known for connectivity and data-sharing, is widespread in applications like smart homes, smart cities, agricultural monitoring, and asset tracking.

Difference Between Embedded System Programming and IoT Programming



Difference Between Embedded System Programming and IoT Programming

•**Embedded System Programming as a Subset of IoT Programming**:
•IoT programming encompasses embedded system programming since IoT devices and gateways are built on embedded systems.
•**IoT Devices**:
•Lightweight and cost-effective, designed for specific tasks (e.g., street light monitoring).
•Typically do not run a full operating system, functioning as simple embedded systems.
•**IoT Gateways**:
•More powerful than IoT devices, capable of connecting multiple IoT devices to the cloud.
•Require operating systems (e.g., Android Things, Amazon FreeRTOS) tailored for IoT-specific needs like security.
•**Programming Perspective**:
•Both IoT devices and gateways use similar programming languages (e.g., Assembly, C, C++) to implement functionality.
•IoT developers must integrate IoT gateways with the cloud using networking protocols like TCP/IP.
•**Expanded Role of Embedded Developers**:
•Developers move beyond product creation to building end-to-end IoT solutions by integrating devices, gateways, and cloud systems.

Types of IoT embedded systems

- An embedded system for IoT will have some type of software installed on it to allow it to perform its function. This could be simply firmware or an embedded operating system. It is the software that empowers the IoT embedded system to communicate with other IoT embedded system devices.
- In the context of embedded systems, we also talk of embedded devices. These terms are sometimes used interchangeably. However, embedded devices are usually part of a larger embedded system for IoT use cases. To differentiate: An embedded device is a highly specialized device intended for one or very few specific purposes. It is also called a Single Purpose Device or Dedicated Device. These devices are IoT embedded systems that have been created specifically to operate one particular program or perform a task that is specific to a certain type of business.
- Examples of single-purpose IoT embedded systems include:
  - Kiosk devices
  - POS solutions
  - Self-checkout systems
- Embedded systems for IoT use cases are widely used in retail, consumer products, automotive, healthcare, and many other industries.

embedded systems are of three types:

- An embedded system may be a standalone system or a segment of a larger system. It is mainly planned for a special purpose or process within a bigger system. In general, embedded systems are of three types:
- 1. Small-Scale This system can be developed with an 8 or 16-bit microcontroller. It can be worked with a battery. For creating a small-scale embedded system, mostly cross-assembler, assembler, an editor, and IDE programming tools are used.
- 2. Medium Scale It is developed with 16 or 32-bit microcontrollers. It delivers both software and hardware complexity. To build this type of system C, source code, C++, Java, and other engineering tools, are used.
- 3. Sophisticated Embedded Systems This type of system involves lots of software and hardware complexity. You may need configuration processors, IPS, scalable processors, PLAs, and ASIPS. For the creation of this system, you require software and hardware co-design & elements that need to combine in the final system.

Where are embedded systems IoT devices used?

- IoT embedded systems are used in a variety of applications. You can find them in your smartwatch, smartphones, wearables, home automation systems, smart thermostats, healthcare devices, smart lights, smart grids, among others.

Applications of Embedded Systems in IoT

Embedded systems in IoT are fundamental for collecting, processing, and transmitting data, supporting the functionality of IoT across various applications. Key areas include:

1. **Smart Homes**: Automating functions like lighting, temperature, and security with energy-efficient and remotely controlled systems.
2. **Industrial Automation**: Real-time monitoring and control of machinery for smoother production, anomaly detection, and productivity enhancement.
3. **Healthcare**: Monitoring vital signs, managing chronic conditions, and enabling real-time data sharing with healthcare providers.
4. **Agriculture**: Precision farming through monitoring environmental factors to optimize irrigation and fertilization.
5. **Transportation**: Enhancing vehicle performance, logistics, traffic management, and route optimization.

Embedded systems are critical for IoT's evolution, fostering efficiency, productivity, and improved quality of life across industries.

Examples of Embedded Systems in the IoT:

- **Smart Home Devices**: Embedded systems are used in smart home devices such as thermostats, lighting systems, and security systems. These devices are capable of communicating with each other and with the cloud, and can be controlled by a smartphone or other device.
- **Medical Devices**: Embedded systems are used in medical devices such as pacemakers, insulin pumps, and blood glucose monitors. These devices are capable of monitoring the patient's condition and transmitting data to healthcare providers.
- **Industrial Automation**: Embedded systems are used in industrial automation systems such as assembly lines, robotics, and process control systems. These systems are capable of monitoring and controlling industrial processes, improving efficiency and productivity.
- Embedded systems are essential to the functioning of the Internet of Things. They provide the intelligence that enables devices to communicate with each other and with the cloud. Embedded systems are responsible for sensor integration, communication, data processing, security, and power management. Examples of embedded systems in the IoT include smart home devices, medical devices, and industrial automation systems. As the IoT continues to grow, the role of embedded systems will become increasingly important.

Some Possible Challenges of Embedded Systems in IoT

- Embedded systems in IoT are transformative but face several challenges that must be addressed to ensure optimal performance and functionality:
- 1. Power Consumption: Energy efficiency is crucial as many systems rely on battery power. Increasing device numbers in IoT networks amplify this challenge.
- 2. Security: Embedded systems are vulnerable to cyber threats because they collect sensitive data and communicate with devices. Ensuring security involves encryption, authentication, and access control, which becomes complex with larger IoT networks.
- 3. Interoperability: Compatibility between diverse devices and communication protocols is critical but challenging due to the heterogeneity of IoT systems.
- 4. Scalability: As IoT networks grow, embedded systems must support increasing demand, requiring robust hardware, software, and communication architecture.
- 5. Real-time Performance: Achieving low-latency communication and processing is essential for real-time functions, which becomes more difficult as the network expands.

Thank You