Internet of Things

M2M to IoT - Unit - 1

- Sri TDNSSSRAO, SVEC

Contents

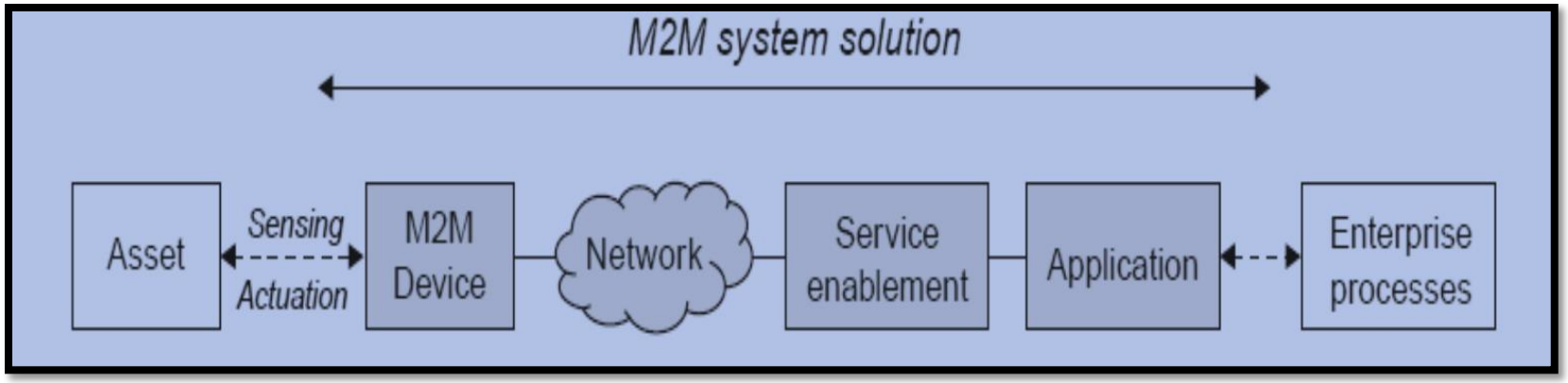
- Introduction from M2M to IoT
  - An Architectural Overview
  - Building architecture
  - Main design principles and needed capabilities
- An IoT architecture outline
- M2M and IoT Technology Fundamentals
- Devices and gateways.

M2M

- M2M has been used throughout the decades as the standard technology in telemetry even before the invention of the Internet itself, as it involved an interaction between two or more machines without human intervention.
- The idea of the Internet of Things, on the other hand, having evolved on the foundations laid down by M2M, aims at offering much more functionality.
- The idea of the Internet of Things, on the other hand, having evolved on the foundations laid down by M2M, aims at offering much more functionality.

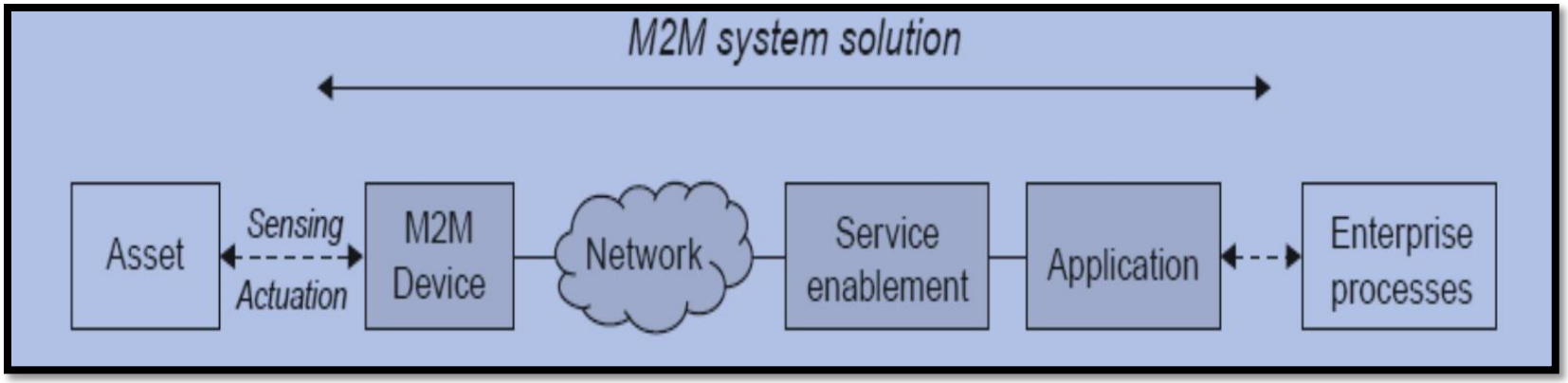
The system components of an M2M solution are as follows:

- M2M Device.
- This is the M2M device attached to the asset of interest, and provides sensing and actuation capabilities. The M2M device is here generalized, as there are a number of different realizations of these devices, ranging from low-end sensor nodes to high-end complex devices with multimodal sensing capabilities.



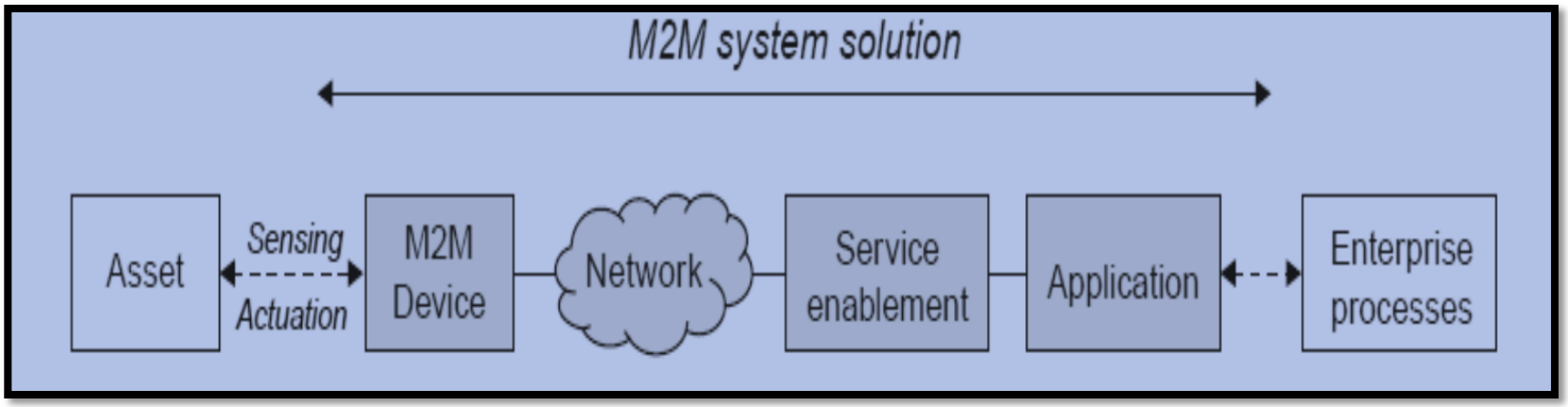
The system components of an M2M solution are as follows:

- Network. The purpose of the network is to provide remote connectivity between the M2M device and the application-side servers. Many different network types can be used, and include both Wide Area Networks (WANs) and Local Area Networks (LANs), sometimes also referred to as Capillary Networks or M2M Area Networks. Examples of WANs are public cellular mobile networks, fixed private networks, or even satellite links.



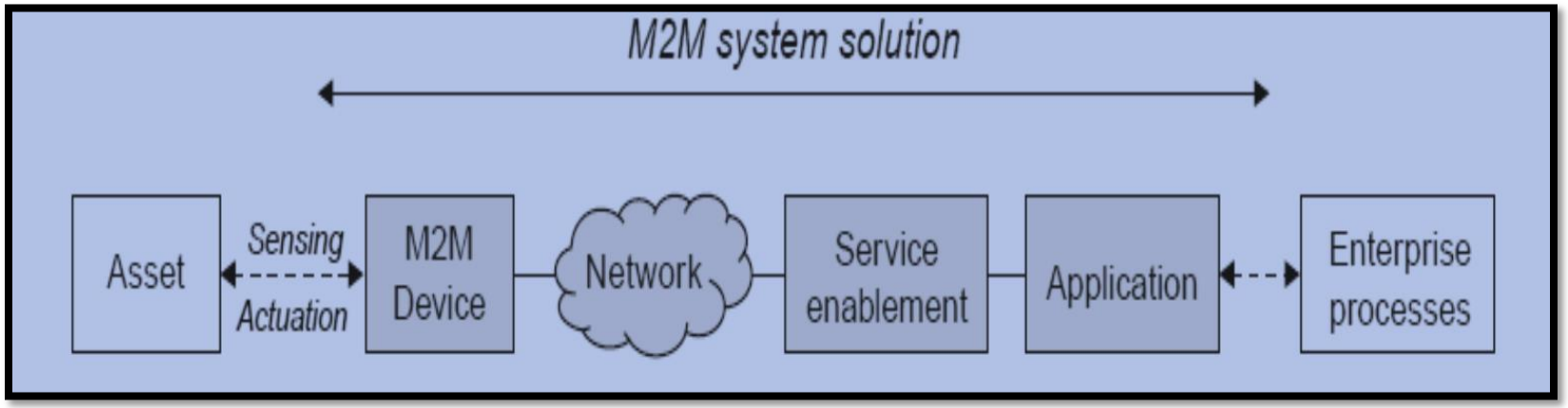
The system components of an M2M solution are as follows:

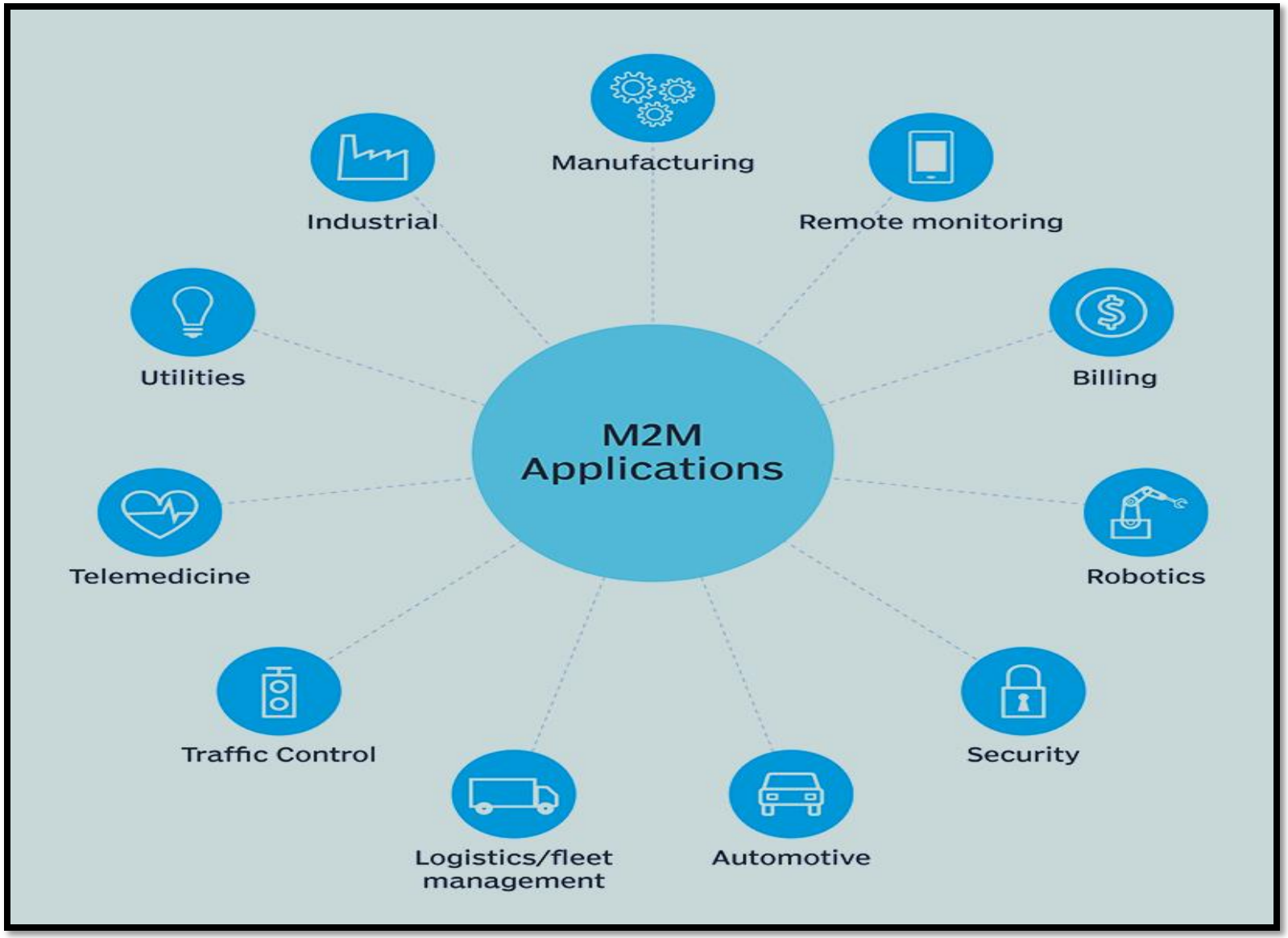
- M2M Service Enablement. Within the generalized system solution outlined above, the concept of a separate service enablement component is also introduced. This component provides generic functionality that is common across a number of different applications. Its primary purpose is to reduce cost for implementation and ease of application development.



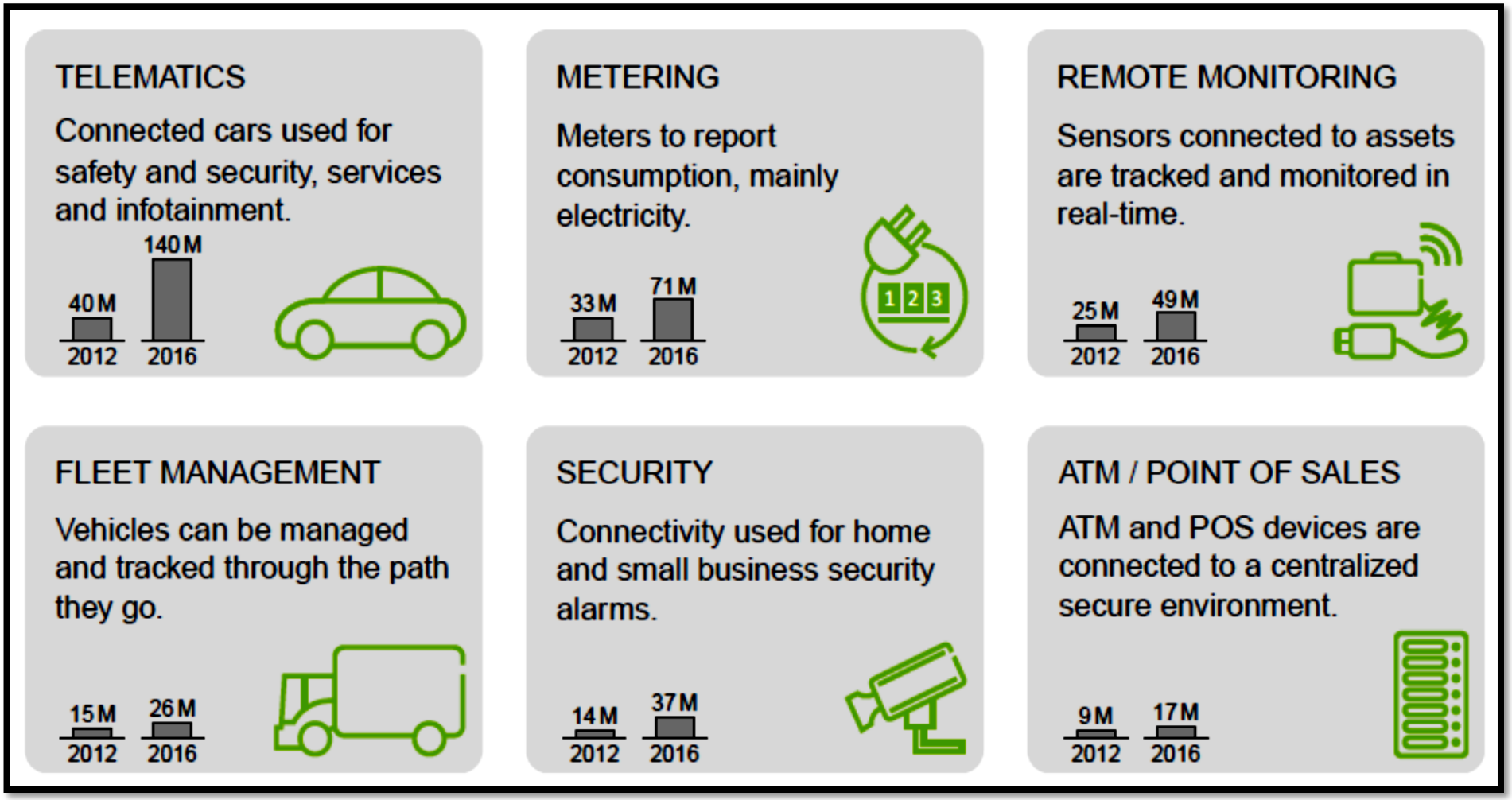
The system components of an M2M solution are as follows:

- M2M Application. The application component of the solution is a realization of the highly specific monitor and control process. The application is further integrated into the overall business process system of the enterprise. The process of remotely monitoring and controlling assets can be of many different types, for instance, remote car diagnostics or electricity meter data management.




M2M Applications
Manufacturing
Industrial
Remote monitoring
Utilities
Billing
Telemedicine
Robotics
Traffic Control
Security
Logistics/fleet management
Automotive

Existing M2M solutions cover numerous industry sectors and application scenarios.



Difference between M2M and IoT

- Machine-to-machine communication, or **M2M**, is exactly as it sounds: two machines "communicating," or exchanging data, without human interfacing or interaction.
  - This includes serial connection, power line connection (PLC), or wireless communications in the industrial Internet of Things (**IoT**).
- **M2M** systems use point-to-point communications between machines, sensors and hardware over cellular or wired networks, while **IoT** systems rely on IP-based networks to send data collected from **IoT**-connected devices to gateways, the cloud or middleware platforms.
- **IoT** is itself **a** subset of **M2M** technology. **IoT** involves communication between machines without human input, making it by definition **a** form of **M2M** communication. However, **IoT** expands the power and potential of **M2M** technology in new ways
- **Four pillars** underpin the ability of **IoT** to operate successfully: device, data, analytics and connectivity.

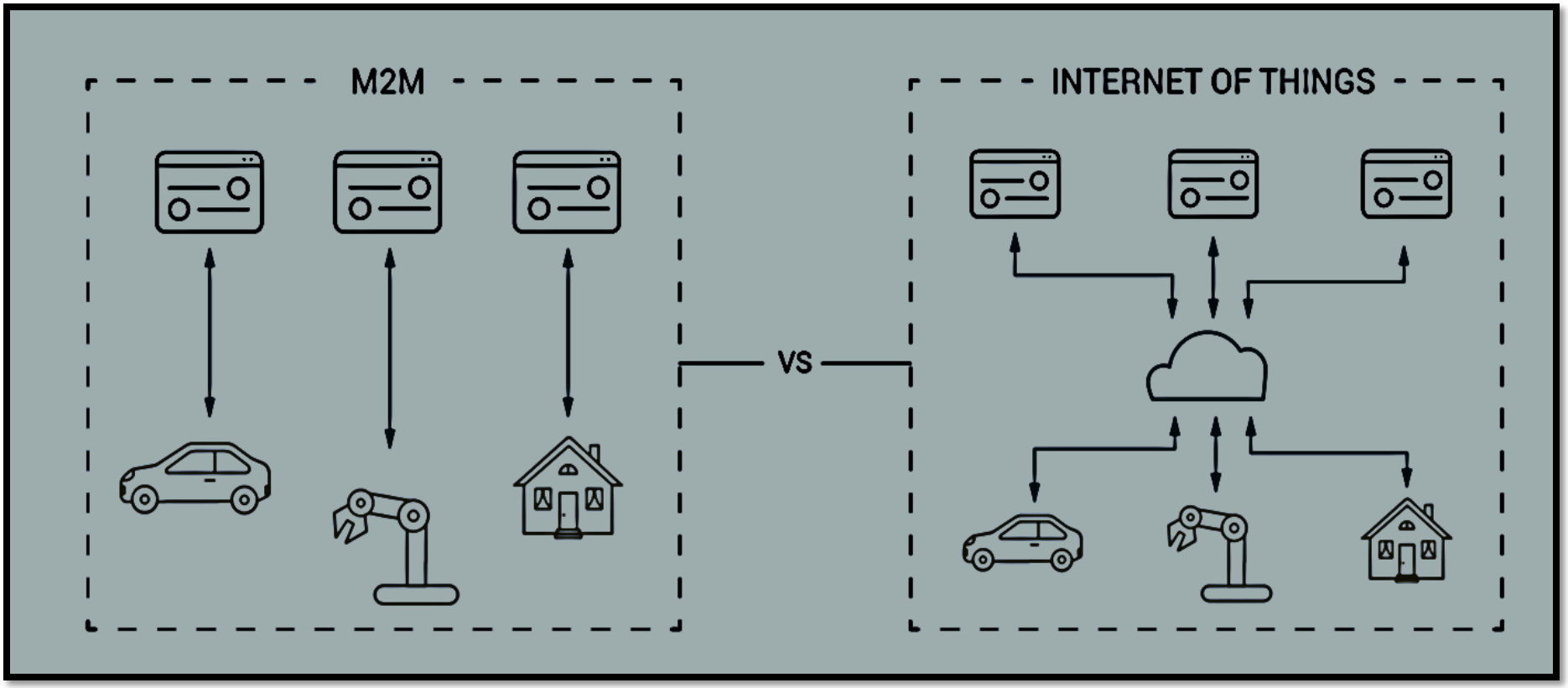
M2M vs. IoT

- While many use the terms interchangeably, M2M and IoT are not the same. IoT needs M2M, but M2M does not need IoT.
- Both terms relate to the communication of connected devices, but
  - M2M systems are often isolated, stand-alone networked equipment.
  - IoT systems take M2M to the next level, bringing together disparate systems into one large, connected ecosystem.
- M2M systems use point-to-point communications between machines, sensors and hardware over cellular or wired networks, while IoT systems rely on IP-based networks to send data collected from IoT-connected devices to gateways, the cloud or middleware platforms.

M2M vs. IoT: What's the difference?

M2M	IoT
Machines	Sensors
Hardware-based	Software-based
Vertical applications	Horizontal applications
Deployed in a closed system	Connects to a larger network
Machines communicating with machines	Machines communicating with machines, humans with machines, machines with humans
Uses non-IP protocol	Uses IP protocols
Can use the cloud, but not required to	Uses the cloud
Machines use point-to-point communication, usually embedded in hardware	Devices use IP networks to communicate
Often one-way communication	Back and forth communication
Main purpose is to monitor and control	Multiple applications; multilevel communications
Operates via triggered responses based on an action	Can, but does not have to, operate on triggered responses
Limited integration options, devices must have complementary communication standards	Unlimited integration options, but requires software that manages communications/protocols
Structured data	Structured and unstructured data

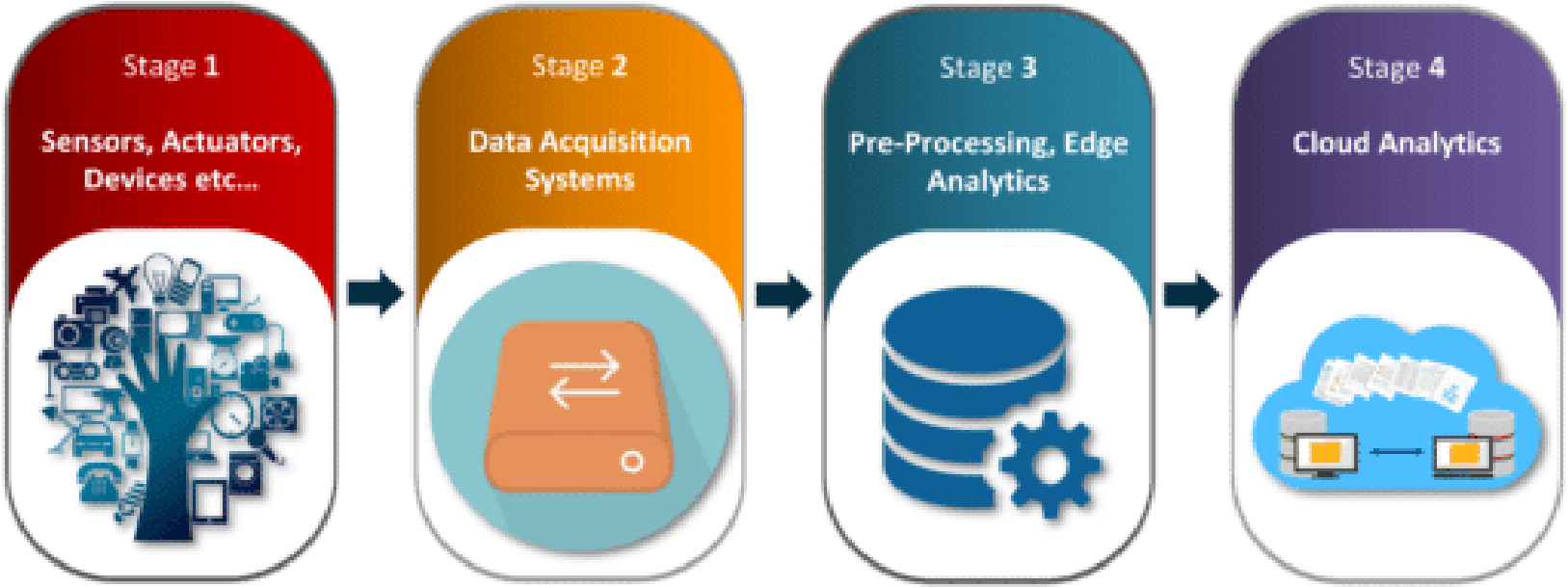
M2M vs IoT



Internet of things- Components

- **A thing,** in the Internet-of-Thing, can be a person or animal or any object that can generate data through a built-in sensor, or any other natural or man-made object that can be assigned an IP address and provided with the ability to transfer data over a network.
- **Sensors:** A sensor is a transducer, whose purpose is to sniff a wide variety of information ranging from Location, Weather/Environment conditions, Grid parameters, Movement on assembly lines, Jet engine maintenance data to Health essentials of a patient and generate output as an electrical or optical signal. The sensors in the IOT are called as a node that will collect information and sent to the outside world, through communication protocols – Bluetooth, BLE, ZigBee, Z-wave, Wi-Fi or through wired communication. These nodes will be forwarding the data to a device called Gateway.
- **IOT Gateway:** The gateway acts as a bridge between these IOT objects and the internet. Gateways can connect to the IoT devices that communicate via specific protocols, store and parse the information and then send them over to cloud servers for processing and analytics. IOT gateways not only abstract the medium of communication but also provide the secure channel required for the transmission of this data. Gateways usually run real-time operation systems (RTOS) or a form of Linux to drive their systems. Hardware and software level encryption is built right into the gateway to provide a secure channel for communication.

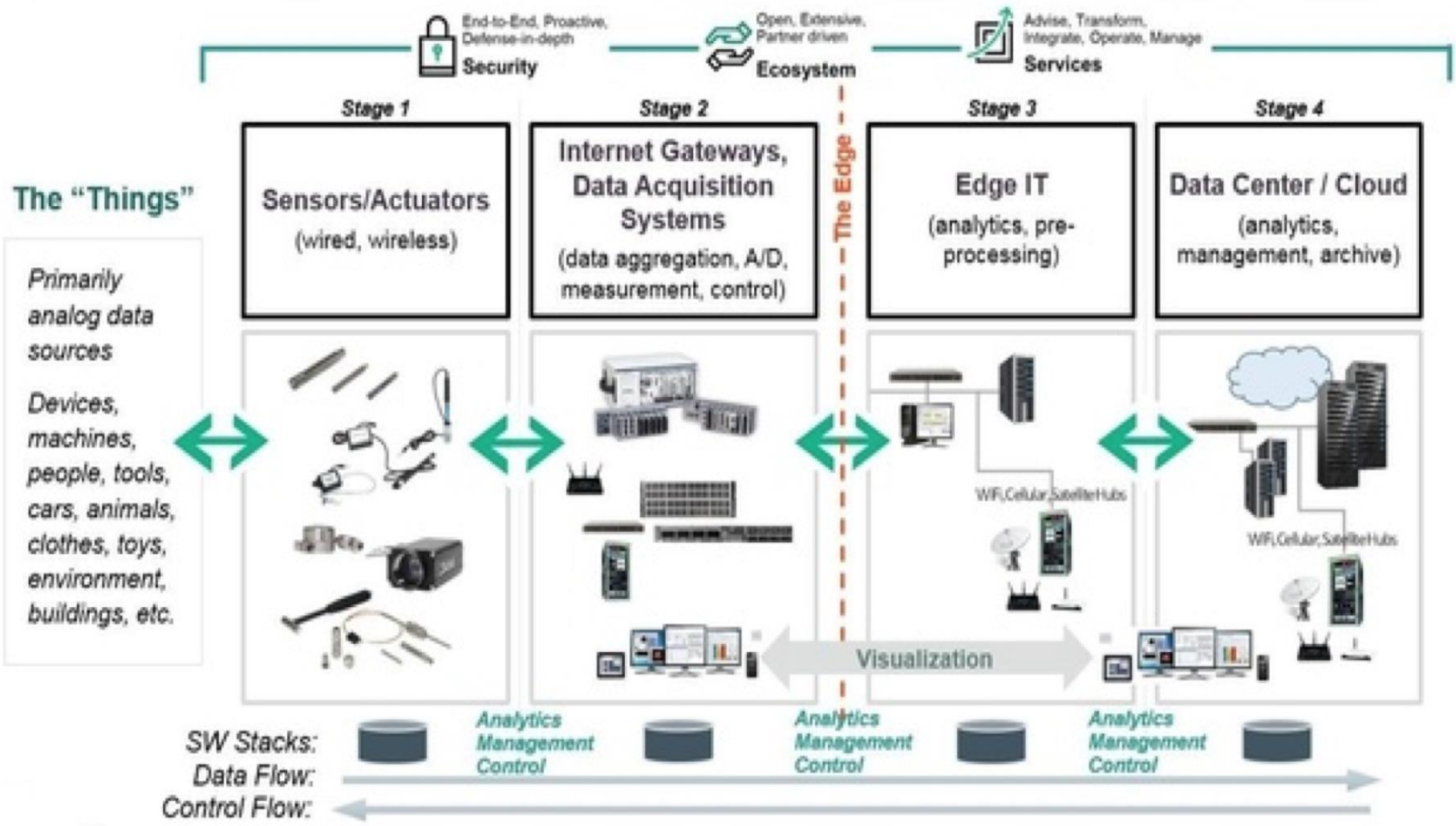
- **Cloud platform and Big data Analytics:** The protocols that support cloud platform are– GPRS, Wi-Fi, CoAP, MQTT, WebSocket, RESTful, etc. Cloud computing is the only platform to support the massive and unpredictable data. Cloud will empower IOT by providing elastic computing power, storage and networking. The massive data generated from IOT can be analyzed in the cloud with big data solutions to gain insights and patterns of usage and behavior of machines and humans. This business intelligence, in turn, allows us to predict forthcoming growth in data demand and deploy additional resources accordingly.
- **These patterns** are then analyzed, and if found relevant, then accordingly the information is sent to the user, to control & monitor their devices (ranging from room thermostat to jet engines & assembly lines) from remote locations. These apps push the important information on your hand-held devices & help to send commands to your Smart Devices.
- **However**, the communication flow can also be in a reverse manner when user/ manufacturer want to actuate any object
  - **User/manufacturer** will give some input in the form of SMS, Push, Email, Call, etc. That information is forwarded to the internet i.e. Cloud
  - **The cloud** then processes the information identifies the particular object through the IP address and pushes the information through the communication protocols to the Gateway.
  - **Gateway** will trigger the actuator that will be responsible for controlling and moving the system or object.



- # Internet of things- Architecture

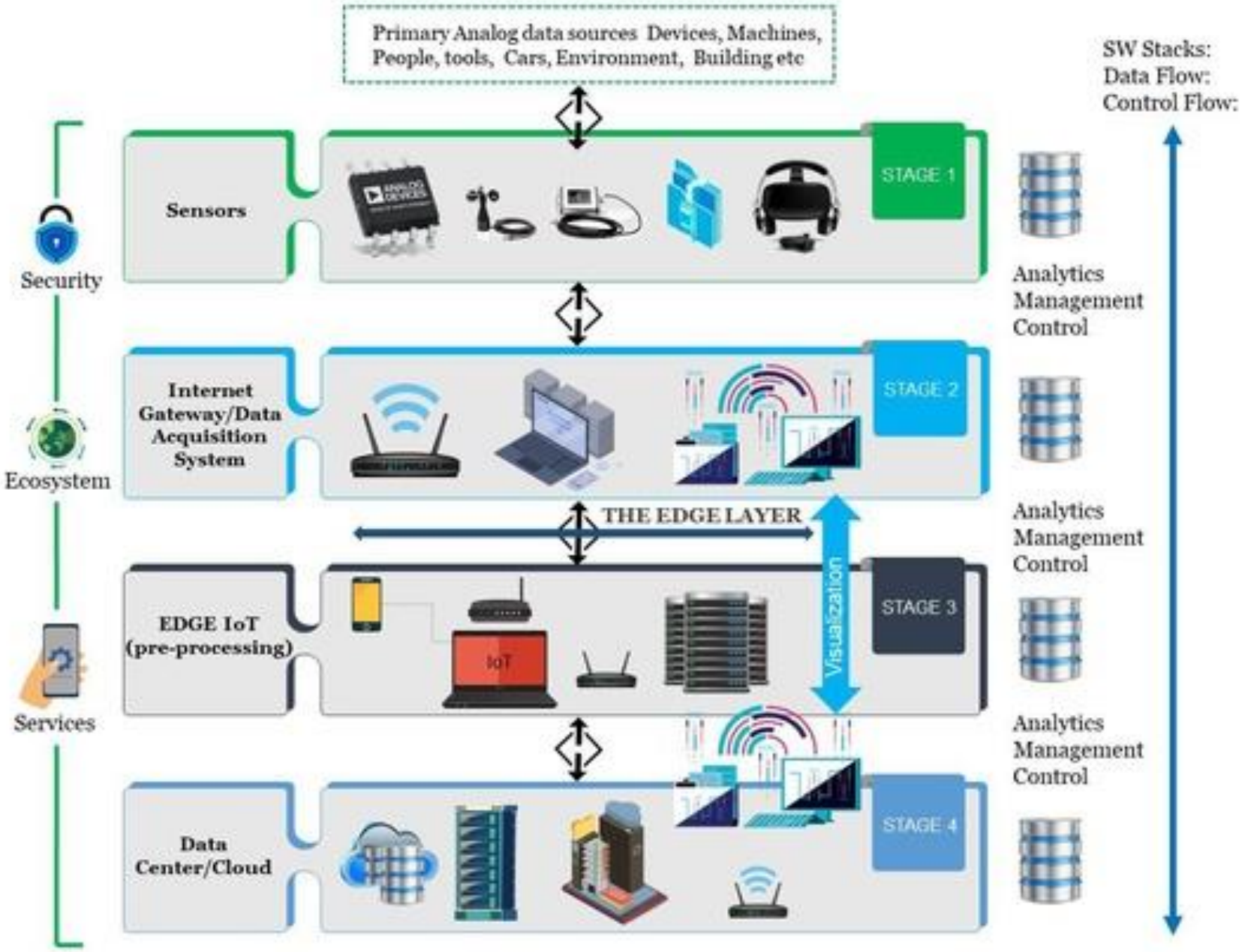
- **Stage 1 (Sensors/Actuators):** A thing in the context of "Internet of Things", should be equipped with sensors and actuators thus giving the ability to emit, accept and process signals.
- **Stage 2 (Data Acquisition Systems):** The data from the sensors starts in analogue form which needs to be aggregated and converted into digital streams for further processing. Data acquisition systems perform these data aggregation and conversion functions.
- **Stage 3 (Edge Analytics):** Once IoT data has been digitized and aggregated, it may require further processing before it enters the data center, this is where Edge Analytics comes in.
- **Stage 4 (Cloud Analytics):** Data that needs more in-depth processing gets forwarded to physical data centers or cloud-based systems.

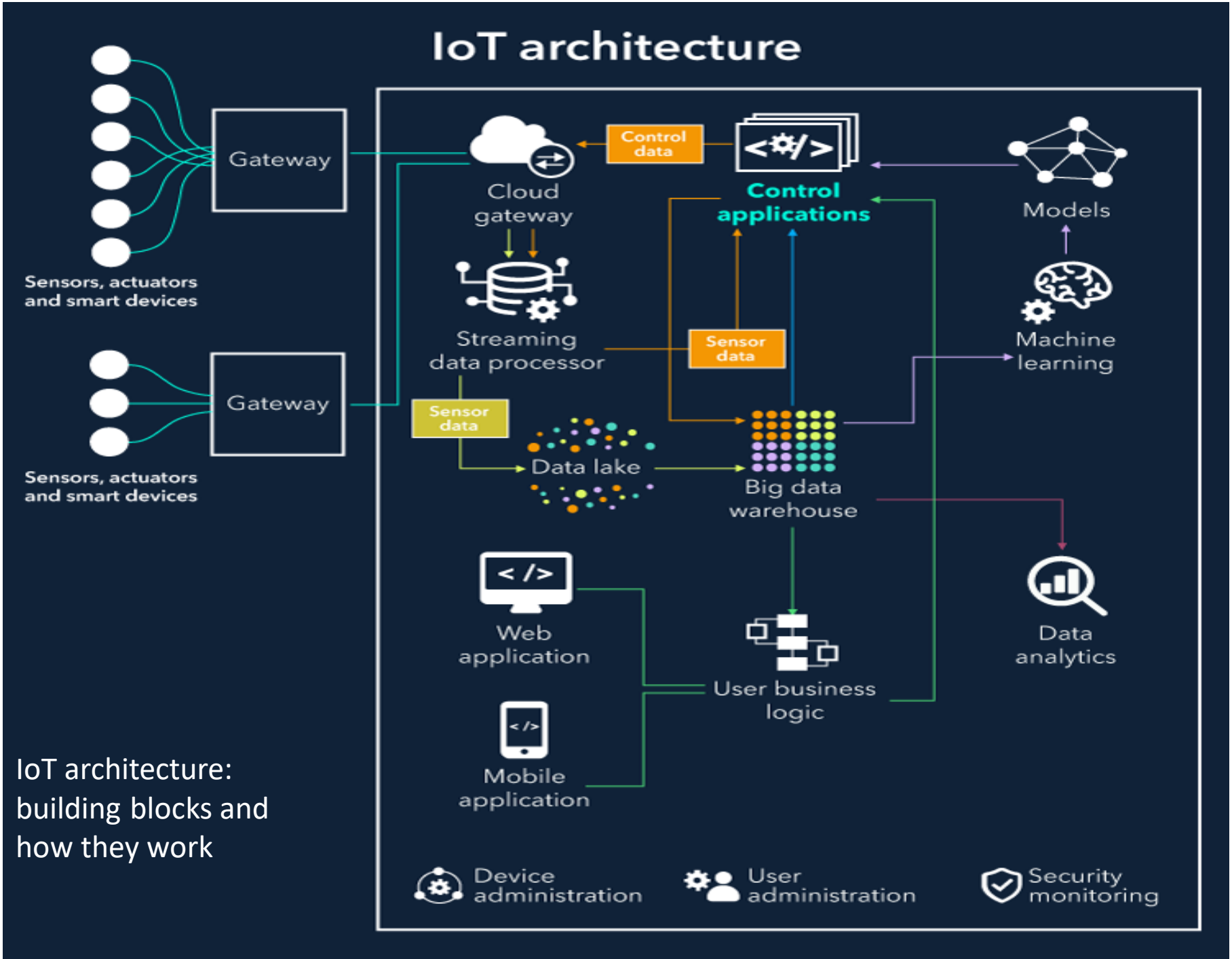
The 4 Stage IoT Solutions Architecture



Edge analytics is simply the process of collecting, analyzing, and creating actionable insights in real-time, directly from the IoT devices generating the data. Automated analytical computation is performed on data at a sensor, network switch or other device instead of waiting for the data to be sent back to a centralized data store.

The 4 stage of IoT solutions Architecture





IoT architecture: building blocks and how they work

IoT architecture components

- **Things** equipped with **sensors** to gather data and **actuators** to perform commands received from the cloud.
- **Gateways** for data filtering, pre-processing and moving it to the cloud and vice versa, – receiving commands from the cloud.
- **Cloud gateways** to ensure data transition between field gateways and central IoT servers.
- **Streaming data processors** to distribute the data coming from sensors among relevant IoT solution's components.
- **Data lake** for storing all the data of defined and undefined value.
- **Big data warehouse** for collecting valuable data.
- **Control applications** to send commands to actuators.
- **Machine learning** to generate the models which are then used by control applications.
- **User applications** to enable users to monitor control their connected things.
- **Data analytics** for manual data processing.

An Architectural Overview: From a reference architecture to a system solution.

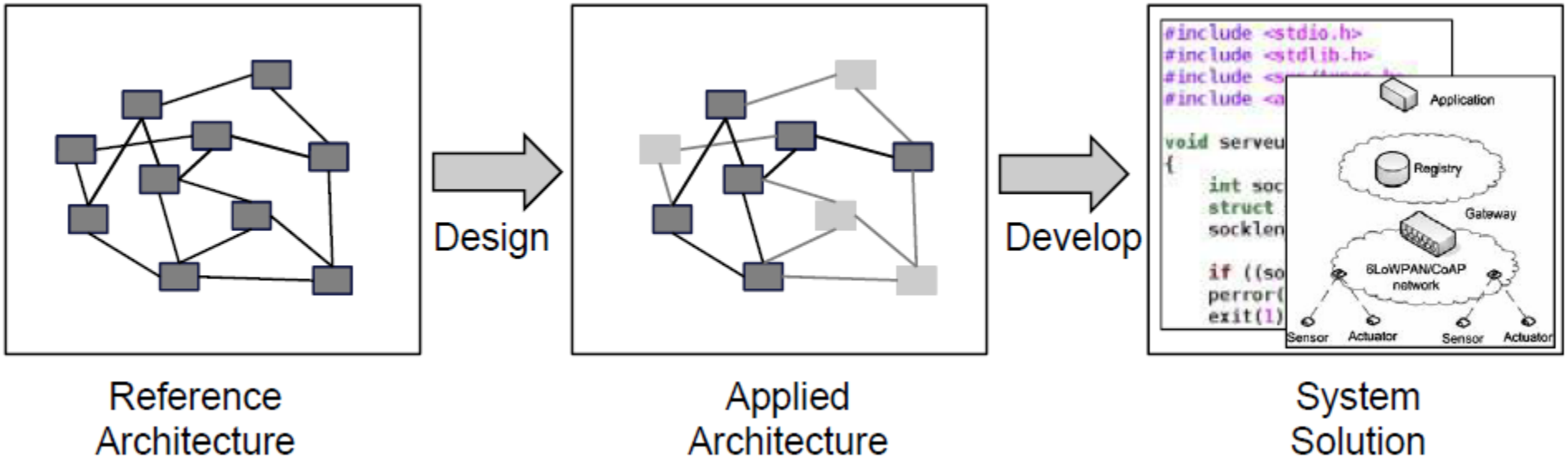
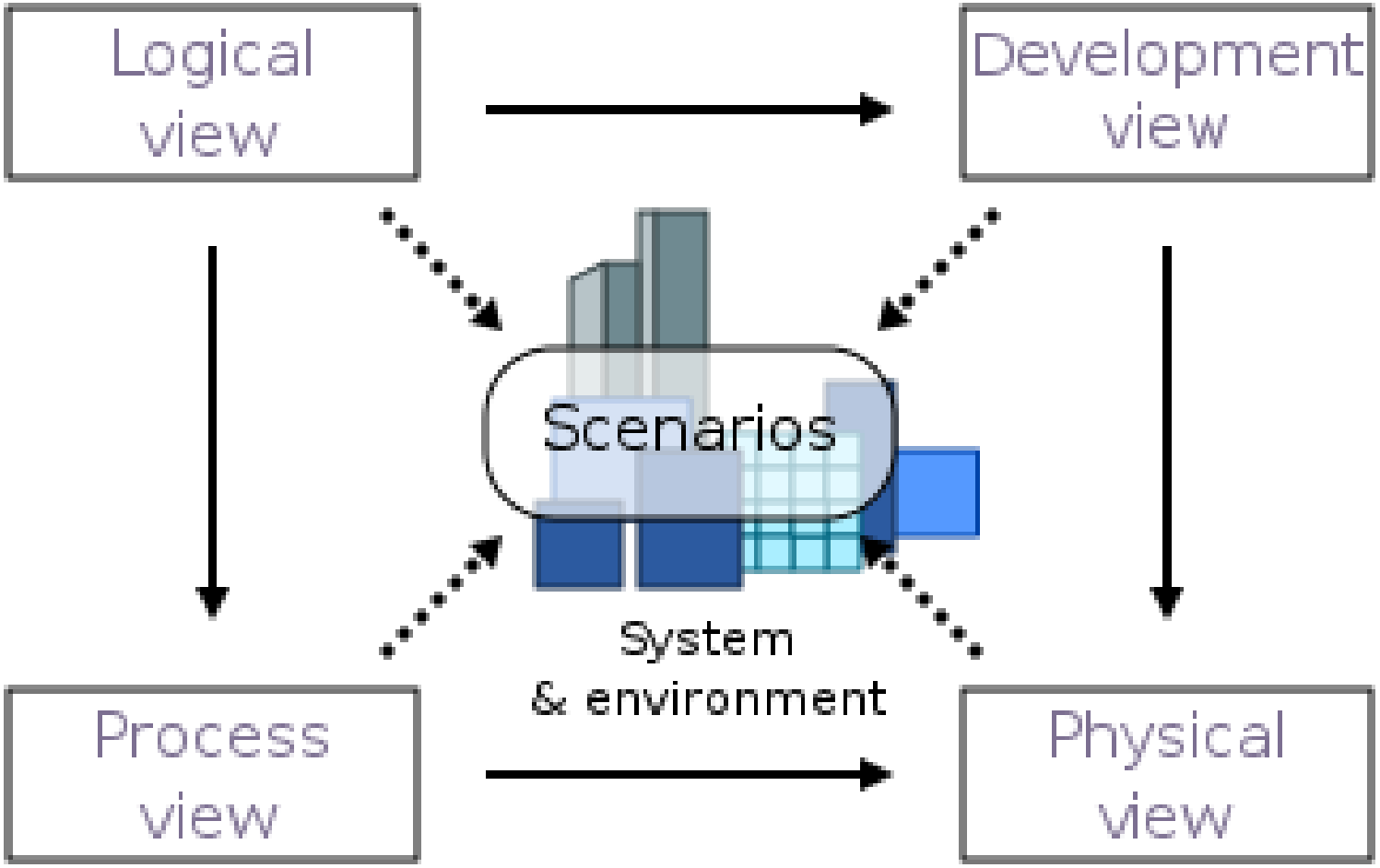
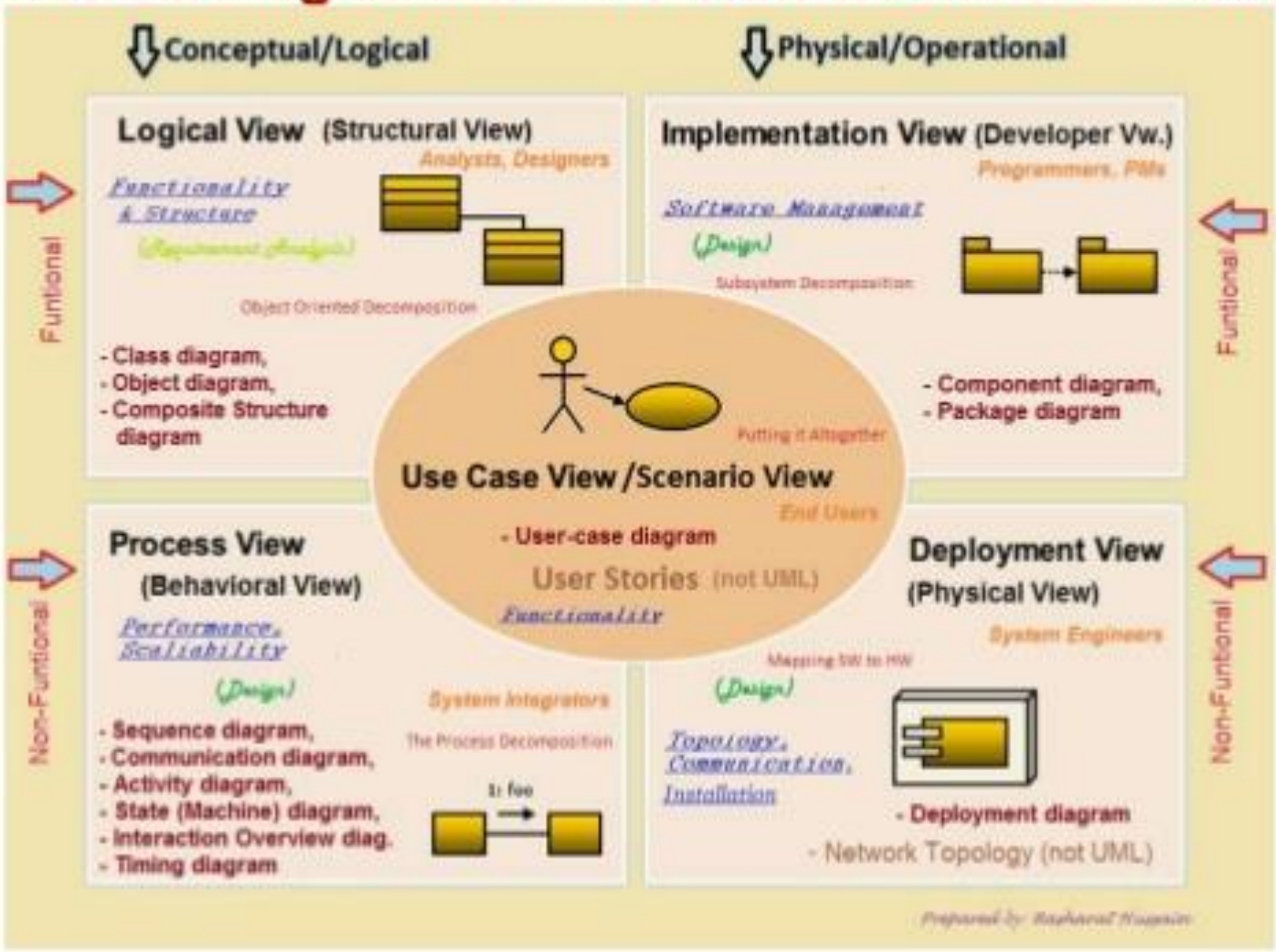
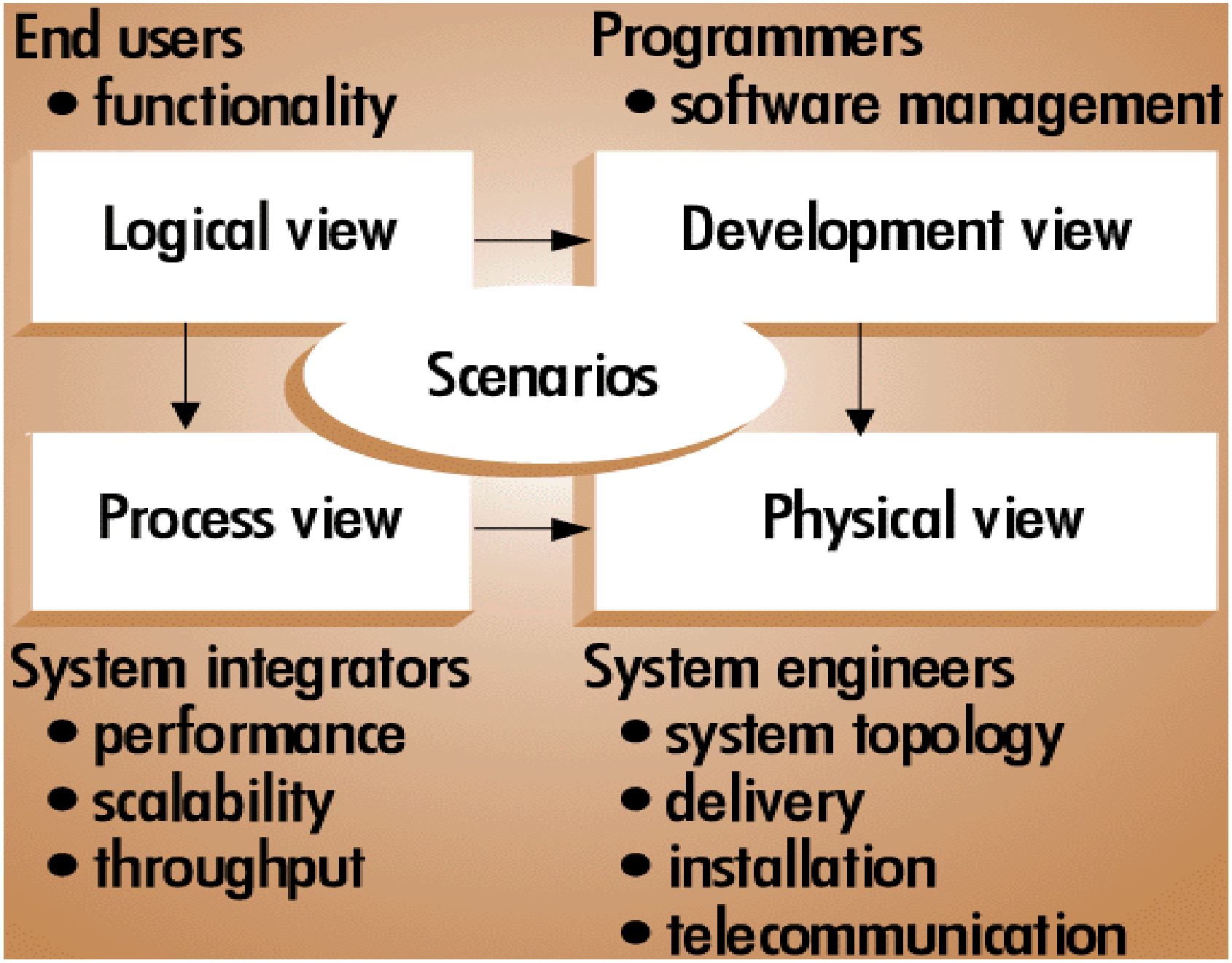


Illustration of the 4+1 Architectural View Model



UML2.0 diagrams in '4+1 View Model' Architecture




End users
• functionality
Programmers
• software management
Logical view
Development view
Scenarios
Process view
Physical view
System integrators
• performance
• scalability
• throughput
System engineers
• system topology
• delivery
• installation
• telecommunication

Architectural Overview

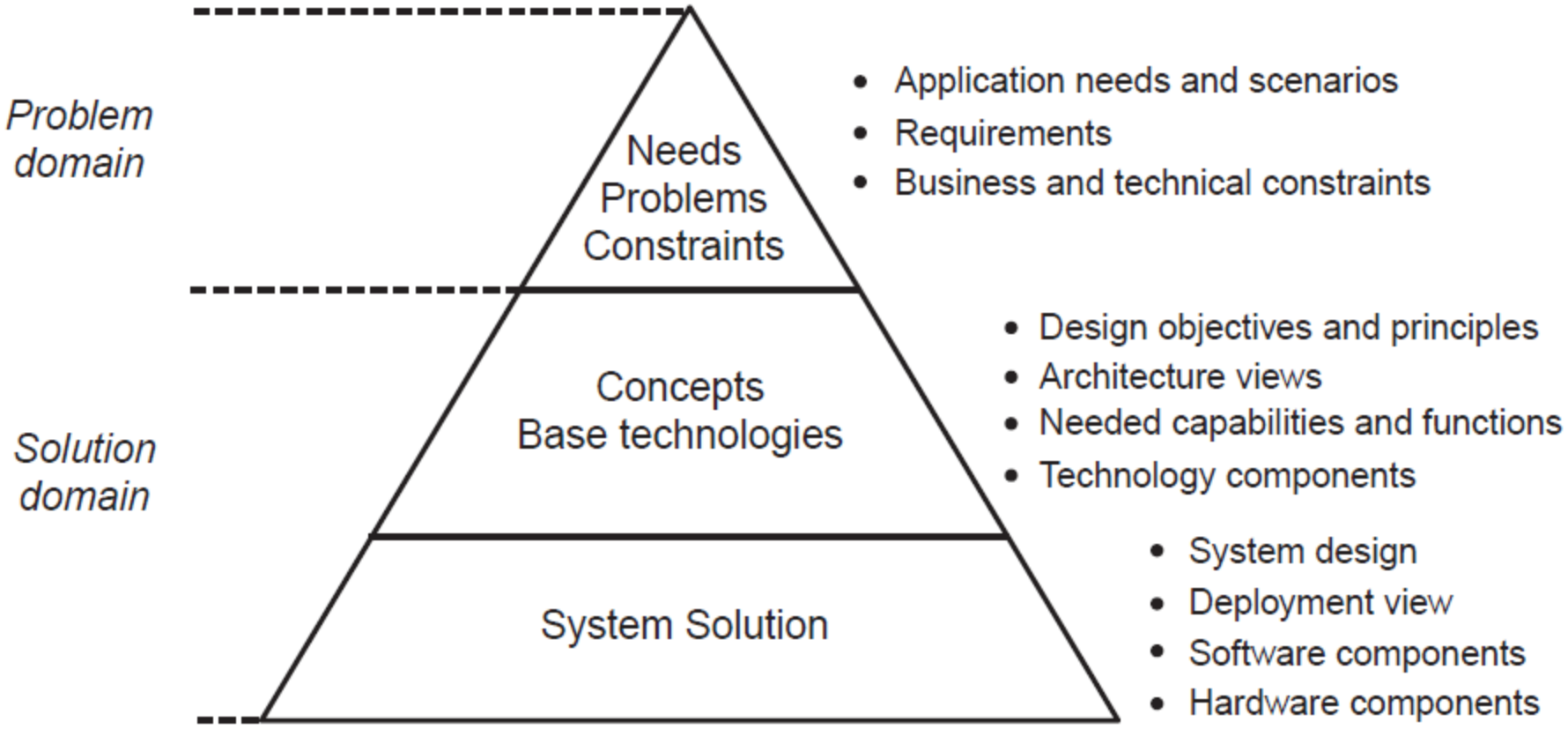
- An architecture can be described in several different views to capture specific properties that are of relevance to model, and the views chosen in are:
- **Functional view (*Logical view):*** Description of what the system does, and its main functions.
- **Deployment view (*Physical view):*** Description of the main real world components of the system such as devices, network routers, servers, etc.
- **Process view (Behavioral view):** deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the run time behavior of the system.
- **Implementation view(Development view):** Used to capture the architectural decisions made for the implementation.
- **Scenarios(use case diagram):** in which your system or application interacts with people, organizations, or external systems.

4+1 is a View Model

- **4+1** is a view model used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineers, and project managers. The four views of the model are logical, development, process and physical view. In addition, selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view.
- ***Logical view***: The logical view is concerned with the functionality that the system provides to end-users. UML diagrams are used to represent the logical view, and include class diagrams, and state diagrams.
- ***Process view***: The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the run time behavior of the system. The process view addresses concurrency, distribution, integrator, performance, and scalability, etc. UML diagrams to represent process view include the sequence diagram, communication diagram, activity diagram.

- ***Development view***: The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the Package diagram.
- ***Physical view***: The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. UML diagrams used to represent the physical view include the deployment diagram.
- ***Scenarios***: The description of an architecture is illustrated using a small set of use cases, or scenarios, which become a fifth view. The scenarios describe sequences of interactions between objects and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. This view is also known as the **use case view**.

Problem and Solution domain partitioning



Model for Reference Architecture

- Creating a model for the reference architecture, one needs to establish overall objectives for the architecture as well as design principles that come from major features of the resulting system solution.
- An overall objective might be to decouple application logic from communication mechanisms, and typical design principles might then be to design for protocol interoperability and to design for encapsulated service descriptions.
- Objectives and principles have to be derived from a deeper understanding of the actual problem domain, and is typically done by identifying recurring problems or type solutions, and extracting common design patterns.
- The problem domain establishes the foundation for the subsequent solutions. It is common to partition the architecture work and solution work into two domains.

- The top level of the triangle is referred to here as the “problem domain” (“domain model” in software engineering). The problem domain is about understanding the applications of interest, for example, developed through scenario building and use case analysis in order to derive requirements.
- These constraints can be technical, like limited power availability in wireless sensor nodes, or non-technical, like constraints coming from legislation or business. These real-world design constraints are to be analyzed.
- The lower level is referred to as the solution domain. Design objectives and principles are established, conceptual views are refined, required functions are identified, and where logical partitions of functionality and information are described. Logical architecture is defined, or network architecture in the form of a network topology diagram is produced. Other suitable technology components such as operating systems and protocols or protocol stacks at this level.
- The actual system solution is finally captured by a system design that typically results in actual software and hardware components, as well as information on how these are to be configured, deployed, and provisioned.

Main design principles and needed capabilities

- Within existing work for deriving requirements and creating architectures or reference models for IoT and M2M, three primary sources can been identified.
- European 7th Framework Program research projects, SENSEI (2013)
- IoT-A (2013)
- ETSI M2M TC 2013

Main design principles and needed capabilities

- Comparing these different approaches, a common feature is the focus on a horizontal system approach.
- Clear separation of the underlying communication networks and related technologies from capabilities that enable services.
- Clear desire to define uniform interfaces towards the devices that provide sensing and actuation, including abstraction of services the devices provide.

Main design principles and needed capabilities

- The overall design objective of IoT architecture shall be to target a horizontal system of real-world services that are open, service-oriented, secure, and offer trust.
- Design for reuse of deployed IoT resources across application domains.
- Design for a set of support services that provide open service-oriented capabilities and can be used for application development and execution.
- Design for different abstraction levels that hide underlying complexities and heterogeneities.
- Design for sensing and actors taking on different roles of providing and using services across different business domains and value chains.
- Design for ensuring trust, security, and privacy.

Main design principles and needed capabilities

- Design for scalability, performance, and effectiveness.
- Design for evolvability, heterogeneity, and simplicity of integration.
- Design for simplicity of management.
- Design for different service delivery models.
- Design for lifecycle support. The lifecycle phases are: planning, development, deployment, and execution. Management aspects include deployment efficiency, design time tools, and run-time management.

Main design principles and needed capabilities

- From these design principles, and taking into consideration detailed use cases and target applications, it is possible to identify requirements that form the basis for a more detailed architecture design.
- Different sets of requirements have been identified in the already-referenced work.

Devices in IoT

- A device can be characterized as having several properties, including: Microcontroller: 8-, 16-, or 32-bit working memory and storage.
- Power Source: Fixed, battery, energy harvesting, or hybrid.
- Sensors and Actuators: Onboard sensors and actuators, or circuitry that allows them to be connected, sampled, conditioned, and controlled.
- Communication: Cellular, wireless, or wired for LAN and WAN communication.
- Operating System (OS): Main-loop, event-based, real-time, or full featured OS.
- Applications: Simple sensor sampling or more advanced applications.
- User Interface: Display, buttons, or other functions for user interaction.
- Device Management (DM): Provisioning, firmware, bootstrapping, and monitoring.
- Execution Environment (EE): Application lifecycle management and Application Programming Interface (API).

Device types

- Grouping of devices into two categories.
- **Basic Devices**: Devices that only provide the basic services of sensor readings and/or actuation tasks, and in some cases limited support for user interaction. LAN communication is supported via wired or wireless technology, thus a gateway is needed to provide the WAN connection.
- **Advanced Devices**: In this case the devices also host the application logic and a WAN connection. They may also feature device management and an execution environment for hosting multiple applications. Gateway devices are most likely to fall into this category.

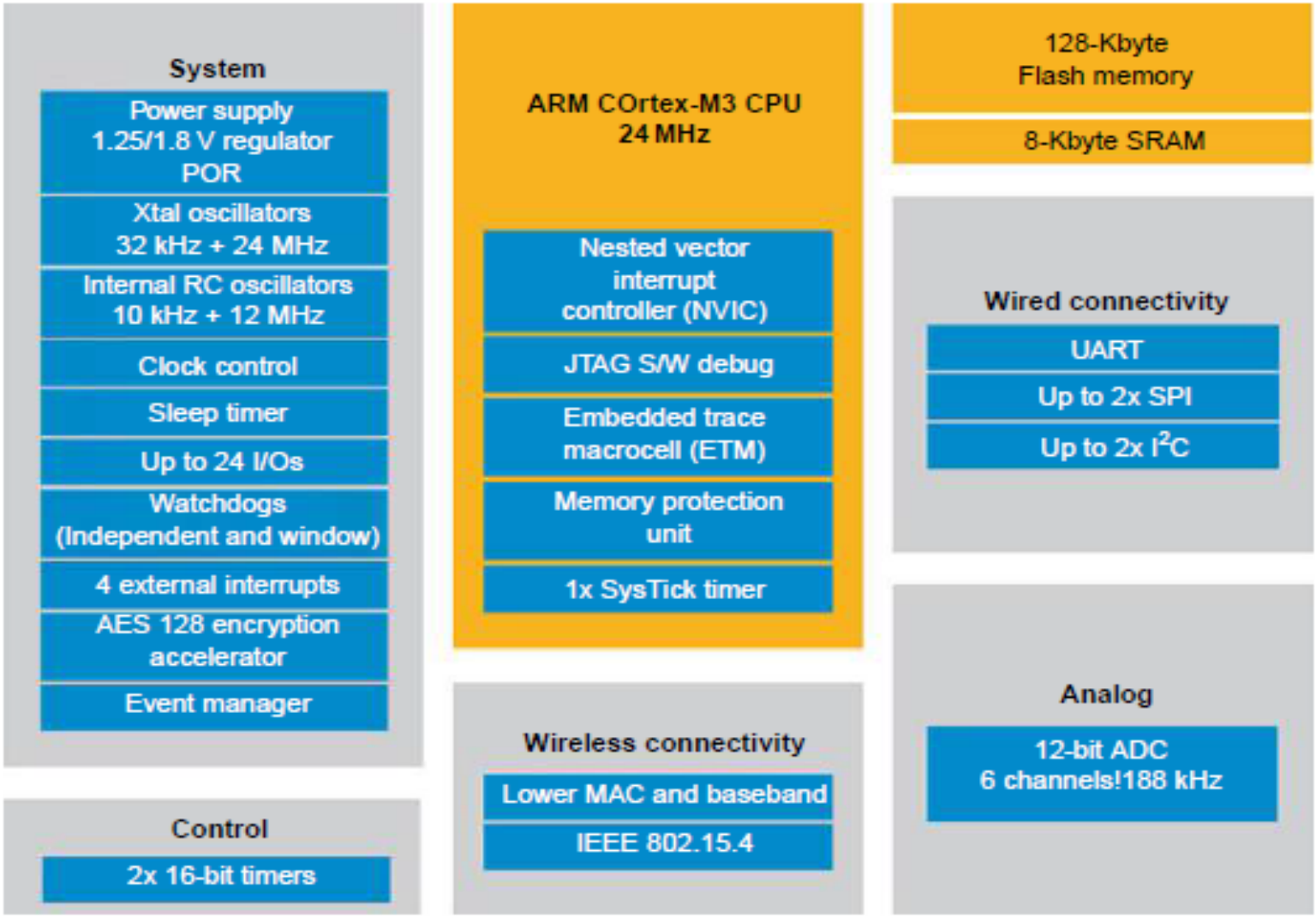
Deployment scenarios for devices

- Deployment can differ for basic and advanced deployment scenarios. Example deployment scenarios for basic devices include:
- **Home Alarms**: Such devices typically include motion detectors, magnetic sensors, and smoke detectors. A central unit takes care of the application logic that calls security and sounds an alarm if a sensor is activated when the alarm is armed. The central unit also handles the WAN connection towards the alarm central. These systems are currently often based on proprietary radio protocols.
- **Smart Meters**: The meters are installed in the households and measure consumption of, for example, electricity and gas. A concentrator gateway collects data from the meters, performs aggregation, and periodically transmits the aggregated data to an application server over a cellular connection. By using a capillary network technology

Deployment scenarios for devices

- **Building Automation Systems (BASs):** Devices include thermostats, fans, motion detectors, and boilers, which are controlled by local facilities, but can also be remotely operated. Standalone Smart Thermostats uses Wi-Fi to communicate with web services.
- **Examples for advanced devices include**:
- Onboard units in cars that perform remote monitoring and configuration over a cellular connection.
- Robots and autonomous vehicles such as unmanned aerial vehicles that can work both autonomously or by remote control using a cellular connection.
- Video cameras for remote monitoring over 3G and LTE.
- Oil well monitoring and collection of data points from remote devices.
- Connected printers that can be upgraded and serviced remotely.

Example of a microcontroller with integrated STM32W-RFCKIT



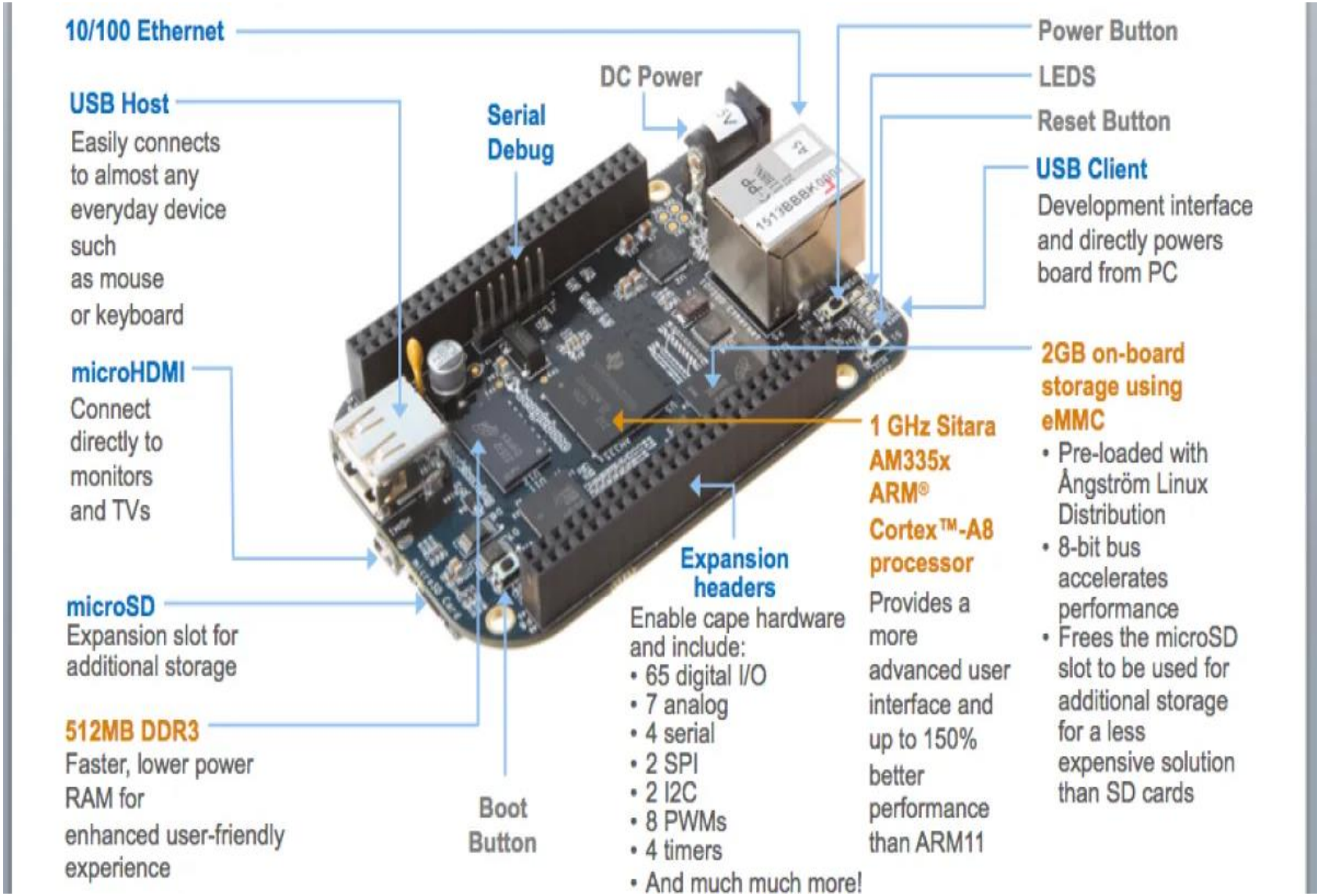
Embedded Boards for IoT

- **Mediatek - Linkit One**
- **Particle Photon**
- **Tessel**
- **Adafruit Flora**
- **Udoo Neo**
- **Intel Edison**
- **Beaglebone Black**
- **Raspberry Pi**

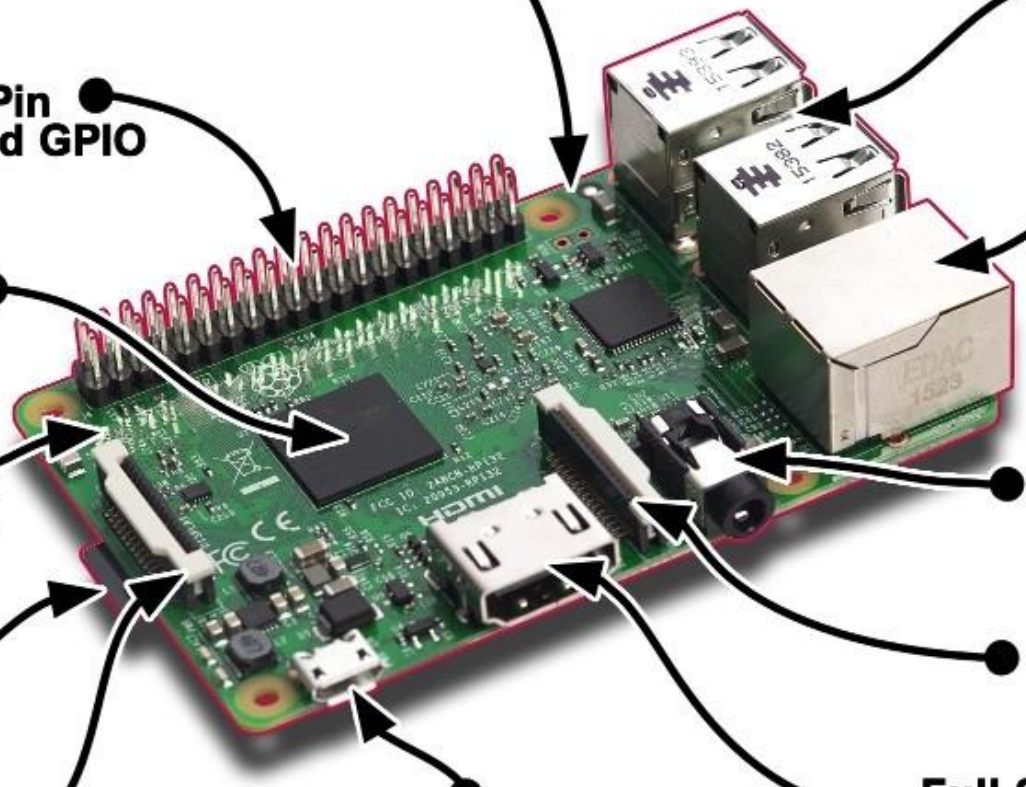
LinkIt ONE


LinkIt ONE Hardware Dev Kit (HDK)
(Front View)
GPIO
Headphone Jack
SD / SPI Switch
USB Jack
Mass Storage / Normal mode switch
Battery Jack
Seeed "GROVE" system
SPI
USB / Battery Power switch
Reset
Power Out
A/D & GPIO
labs.mediatek.com/linkithdk

Beaglebone Black



Raspberry Pi 3 Model B



Dimensions
85.6mm x 56mm x 21mm

4 x USB 2 Ports

40 Pin Extended GPIO

10/100 LAN Port

Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz

3.5mm 4-pole Composite Video and Audio Output Jack

On Board Bluetooth 4.1 Wi-Fi

CSI Camera Port

MicroSD Card Slot

Full Size HDMI Video Output

DSI Display Port

Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps

Gateways

- A gateway is a network node that forms a passage between two networks operating with different transmission protocols and serves as a translator between different protocols, e.g. between IEEE 802.15.4 or IEEE 802.11, to Ethernet or cellular.
- Different types of gateways, which can work on different levels in the protocol layers. Most often a gateway refers to a device that performs translation of the physical and link layer, but **application layer gateways** (ALGs) are also common. The latter is preferably avoided because it adds complexity and is a common source of error in deployments.
- ALGs include the ZigBee Gateway Device, which translates from ZigBee to SOAP and IP, or gateways that translate from Constrained Application Protocol (CoAP) to HyperText Transfer Protocol/Representational State Transfer (HTTP/REST).
- LAN technologies, such as 802.11 and Z-Wave, the gateway is used for inclusion and exclusion of devices. Typically works by activating the gateway into inclusion or exclusion mode and by pressing a button on the device to be added or removed from the network.

Data management

- Typical functions for data management include performing sensor readings and caching this data, as well as filtering, concentrating, and aggregating the data before transmitting it to back-end servers.
- Local applications: Examples of local applications that can be hosted on a gateway include closed loops, home alarm logic, and ventilation control, or the data management function.
- The benefit of hosting this logic on the gateway instead of in the network is to avoid downtime in case of WAN connection failure, minimize usage of costly cellular data, and reduce latency.

Device management

- Device management (DM) is an essential part of the IoT and provides efficient means to perform many of the management tasks for devices:
- Provisioning: Initialization (or activation) of devices in regards to configuration and features to be enabled.
- Device Configuration: Management of device settings and parameters.
- Software Upgrades: Installation of firmware, system software, and applications on the device.
- Fault Management: Enables error reporting and access to device status.

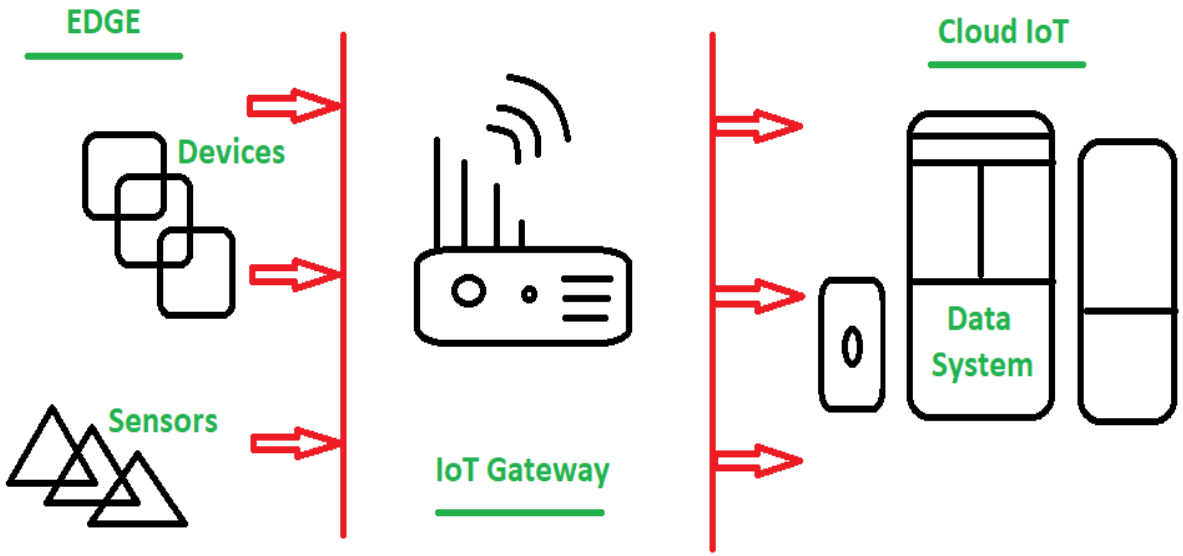
Advanced devices

- Features that can characterize an advanced device are the following:
- A powerful CPU or microcontroller with enough memory and storage to host advanced applications, such as a printer offering functions for copying, faxing, printing, and remote management.
- A more advanced user interface with, for example, display and advanced user input in the form of a keypad or touch screen.
- Video or other high bandwidth functions.
- The barrier for new developers will further be reduced thanks to the consolidation of software and interfaces, e.g. it will be possible to interact with a device using simple HTTP/REST and to easily install a Java application on a device, resulting in an increased number of developers.

Advanced devices

- Developments in hardware and network technologies, entirely new device classes and features are expected, such as: Battery-powered devices with ultra-low power cellular connections.
- Devices that harvest energy from their environment.
- Smart bandwidth management and protocol switching, i.e. using adaptive RF mechanisms to swap between, for example, Bluetooth LE and IEEE 802.15.4.
- Multi-radio/multi-rate to switch between bands or bit rates (slower bit rate implies better sensitivity at longer range).
- Microcontrollers with multicore processors.
- Novel software architectures for better handling of concurrency.
- The possibility to automate the design of integrated circuits based on business-level logic and use case.
- All these improvements that the IoT brings will remove the final barriers that have been holding back the market for M2M.

Internet of Things (IoT) Gateways



- **Gateway** provides bridge between different communication technologies which means we can say that a Gateway acts as a medium to open up connection between cloud and controller(sensors / devices) in IoT. Gateways establish device to device or device to cloud communication. A gateway can be a typical hardware device or software program.
- Enables a connection between sensor network and Internet along with enabling IoT communication, it also performs many other tasks like, performs protocol translation, aggregating all data, local processing and filtering of data before sending it to cloud, locally storing data and autonomously controlling devices based on some inputted data, providing additional device security.

Internet of Things (IoT) Gateways

- As IoT devices work with low power consumption (Battery power) in other words they are energy constrained so if they will directly communicate to cloud/internet, it won't be effective in terms of power.
- So they communicate with Gateway first using short range wireless transmission modes/network like ZigBee, Bluetooth, etc as they consume less power or they can also be connected using long range like Cellular and WiFi etc.

Internet of Things (IoT) Gateways

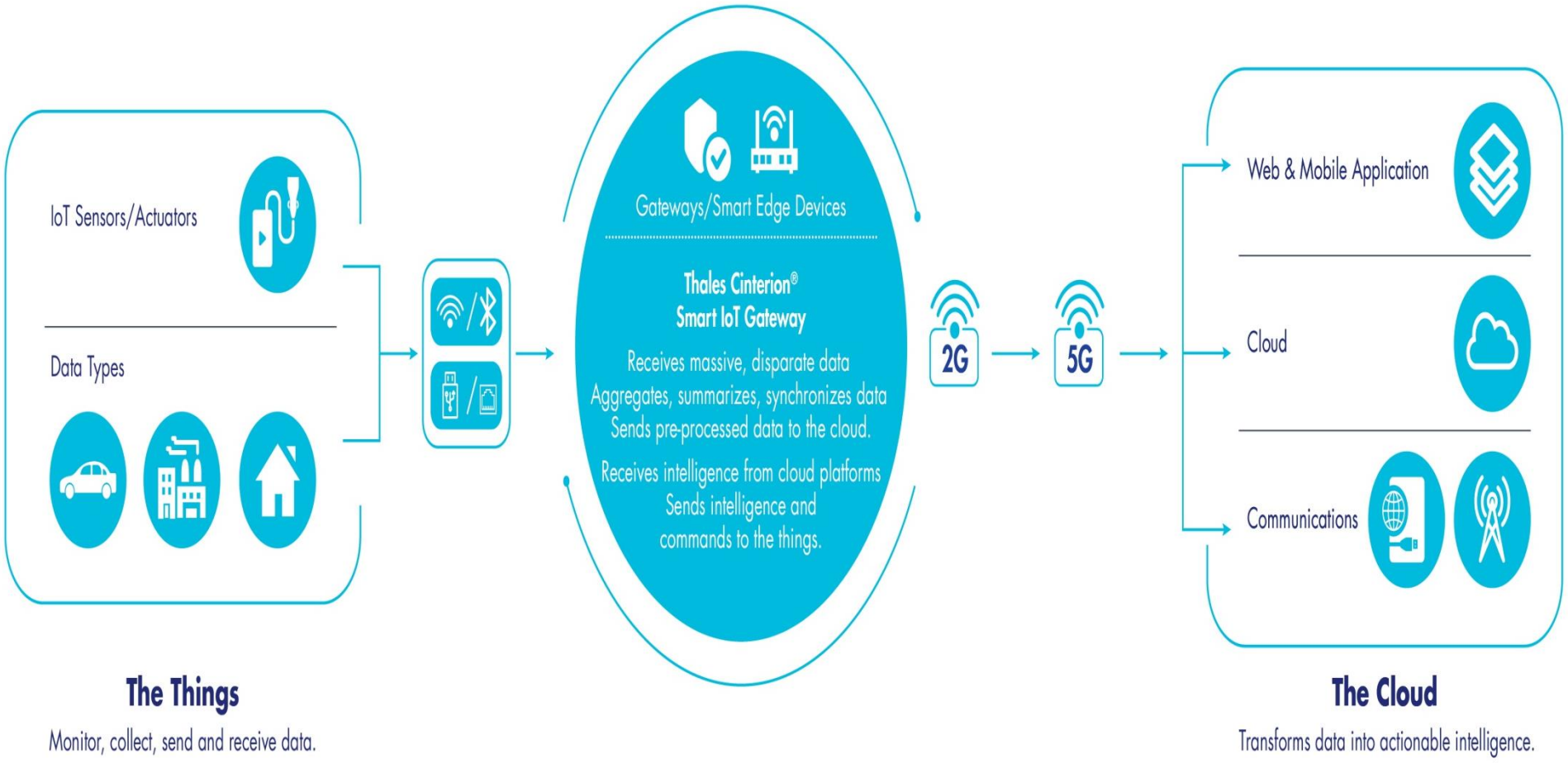
- Gateway links them to Internet/ cloud by converting data into a standard protocol like MQTT. using ethernet, WiFi/cellular or satellite connection. And in mostly Gateway is Mains powered unlike sensor nodes which are battery powered. In practice there are multiple Gateway devices.
- Smartphone can also work as a basic IoT gateway when we use multiple radio technologies like WiFi, Bluetooth, Cellular network of smart phone to work on any IoT project in sending and receiving data at that time this also acts as a basic IoT Gateway.

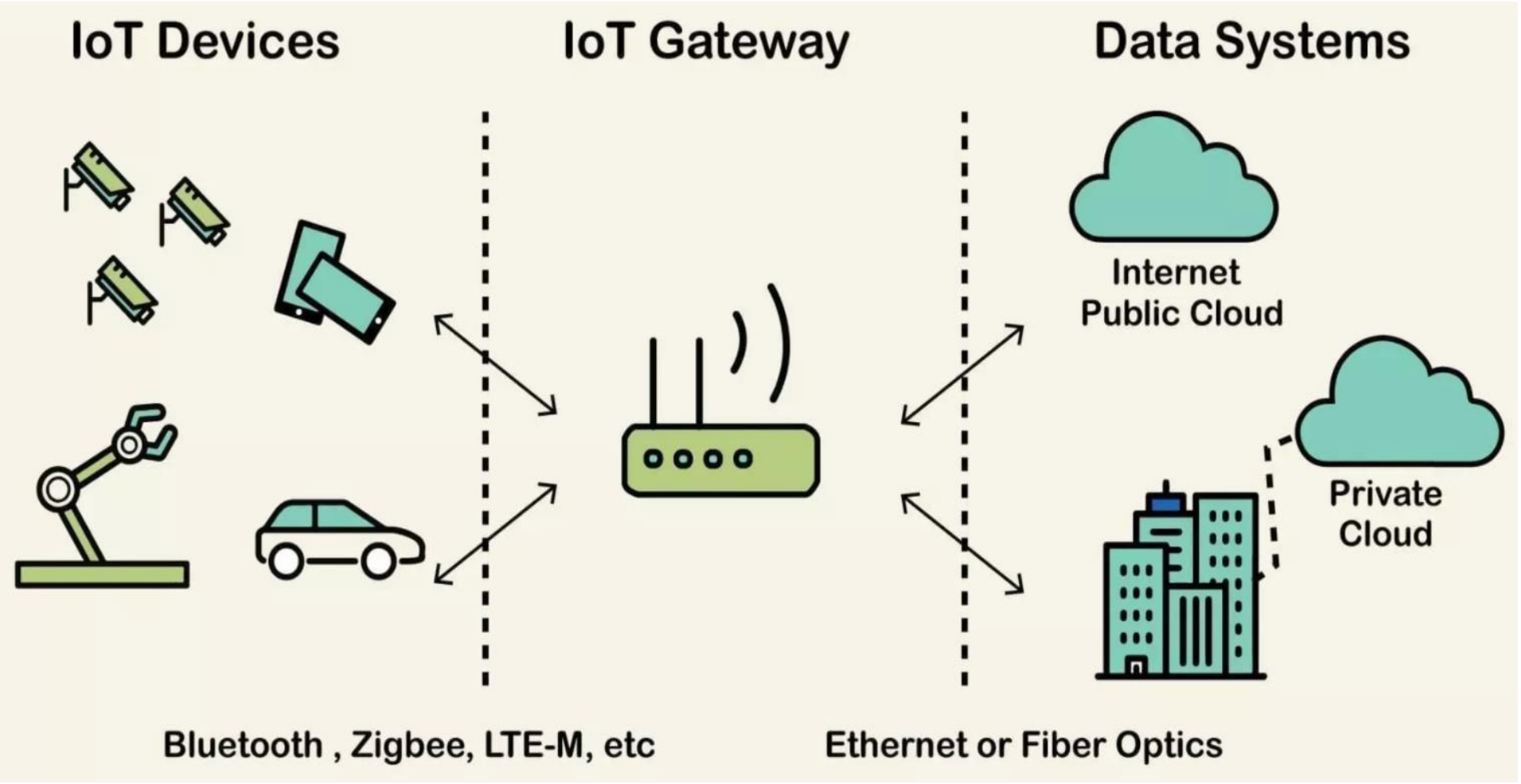
Key functionalities of IoT Gateway

- Establishing communication bridge
- Provides additional security.
- Performs data aggregation.
- Pre processing and filtering of data.
- Provides local storage as a cache/ buffer.
- Data computing at edge level.
- Ability to manage entire device.
- Device diagnostics.
- Adding more functional capability.
- Verifying protocols.

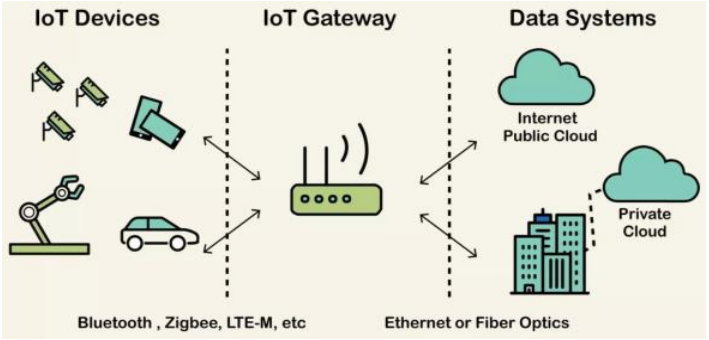
Working of IoT Gateway

- Receives data from sensor network.
- Performs Pre processing, filtering and cleaning on unfiltered data.
- Transports into standard protocols for communication.
- Sends data to cloud.
- IoT Gateways are key element of IoT infrastructure as Gateways establish connection for communication and also performs other task as described above. So IoT Gateway is one of most essential thing when we start think about an IoT ecosystem.

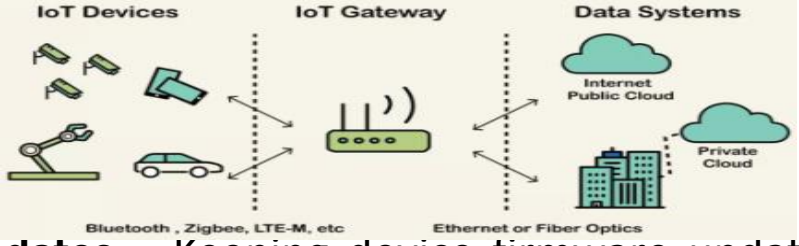

IoT Sensors/Actuators
Data Types
Gateways/Smart Edge Devices
Thales Cinterion®
Smart IoT Gateway
Receives massive, disparate data
Aggregates, summarizes, synchronizes data
Sends pre-processed data to the cloud.
Receives intelligence from cloud platforms
Sends intelligence and
commands to the things.
2G
5G
Web & Mobile Application
Cloud
Communications
The Things
Monitor, collect, send and receive data.
The Cloud
Transforms data into actionable intelligence.


IoT Devices
IoT Gateway
Data Systems
Internet
Public Cloud
Private
Cloud
Bluetooth , Zigbee, LTE-M, etc
Ethernet or Fiber Optics

The art and science of IoT gateway architecture



- **Device layer** - IoT gateway hardware comprises a microprocessor or controller depending on processing speed and memory required, a connectivity module (cellular, Wi-Fi, Bluetooth, etc.), IoT sensors, and circuitry.
- **Operating system** – The OS is software that runs gateway hardware and other programs on the device. Choice of an OS such as Java, Linux, RTOS, etc., depends on the gateway's application.
- **Hardware abstraction** – The abstraction layer allows the software to be developed and controlled independently of the hardware. This adds flexibility and agility to application design and makes software updates and evolution easier.
- **Sensor and actuator drivers** – This layer serves as the interface between the device and sensors and modules. Specific stacks are integrated depending on what the application demands.
- **Device management and configuration** – IoT gateways need to keep track of all the connected devices and sensors it communicates with. This layer tracks and manages the configurations, settings, and properties of sensors and connected devices within its ecosystem.
- **Security** – Security is a key consideration in gateway architecture. This layer ensures that gateways have trusted identities, strong encryption, and crypto authentication schemes. It provides a secure boot to protect devices from intrusion and ensure data integrity and confidentiality.



- **Firmware Over the Air Updates** – Keeping device firmware updated and enabling security patches and fixes to defend against ever-evolving threats is paramount to maintaining device integrity. This layer ensures that Firmware Over The Air (FOTA) updates are managed securely and efficiently to preserve device memory, power, and network bandwidth.
- **Communication protocols** – IoT gateway protocols are selected according to the amount and frequency of data communicated to the cloud. Gateways need to connect via a cellular module (5G/4G/3G), Ethernet, and/or Wi-Fi, but the underlying communication protocol layer is typically TCP IP protocol.
- **Data management** – IoT gateways manage data from sensors and connected devices and data coming from the cloud. The data management layer controls streaming, filtering, and data storage, and it provides data traffic control to minimize delays and ensure device fidelity.
- Cloud connectivity manager - This layer is responsible for seamless, secure connectivity with cloud platforms and device and cloud authentication.
- **Custom software applications** - IoT gateways integrate custom software to manage specific application needs. This layer interacts with all other layers to efficiently, securely, and efficiently manage data needs specific to the IoT application.
- **Gateway data transfer** – This layer controls the gateway's connection to the Internet using either a 5G/4G/3G/GPRS modem or IoT module, ethernet, or Wi-Fi. It also analyses and determines which data needs to be communicated to the cloud and which data should be cached for processing offline to save processing power and data plan fees.

END