ASSIGNMENT 1 FRONT SHEET

Qualification	**TEC Level 5 HND Diploma in Computing**		
Unit number and title	**Unit 43: Internet of Things**		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name		**Student ID**	
Class		**Assessor name**	

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P1	P2	P3	P4	M1	M2	M3	M4	D1	D2

☼ **Summative Feedback:**		☼ **Resubmission Feedback:**
Grade:	**Assessor Signature:**	**Date:**

Internal Verifier's Comments:
Signature & Date:

Contents

TASK 1 – REVIEW AND EVALUATE ABOUT IOT ASPECTS.

1. Review IoT functionality, standard architecture, frameworks, tools, hardware and APIs (P1-P2)

1.1 Definition about IoT:

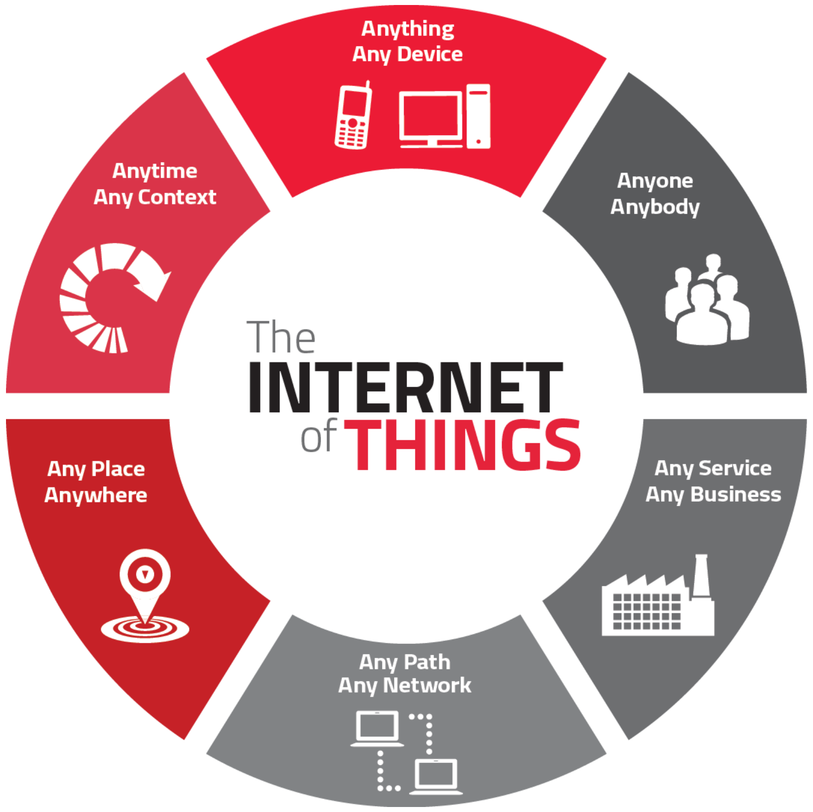


Figure 1: Internet Of Things

The Internet of Things (IoT) is a network that connects physical objects, devices, and appliances, all equipped with sensors, software, and internet connectivity. Through this network, these objects can collect and share data with each other and central systems. IoT includes a wide range of devices, from consumer gadgets like smartwatches to industrial sensors in factories and smart city systems.

The main idea behind IoT is to enable communication between these objects, leading to data gathering, analysis, and informed decision-making. This interconnectedness creates opportunities for automation and efficiency in various sectors such as home automation, healthcare, transportation, and manufacturing.

While IoT offers numerous benefits, it also raises concerns about data privacy, security, and ethical implications. The large amount of data generated by IoT devices can be sensitive and susceptible to misuse if not adequately protected. Addressing these challenges is crucial to ensure responsible and secure implementation and fully realize the potential of IoT.

1.2 How does IoT work?

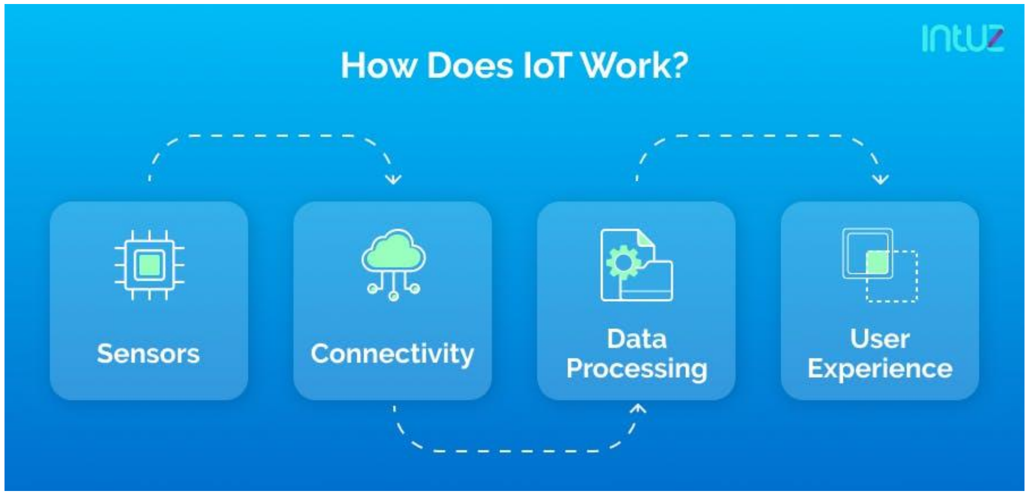
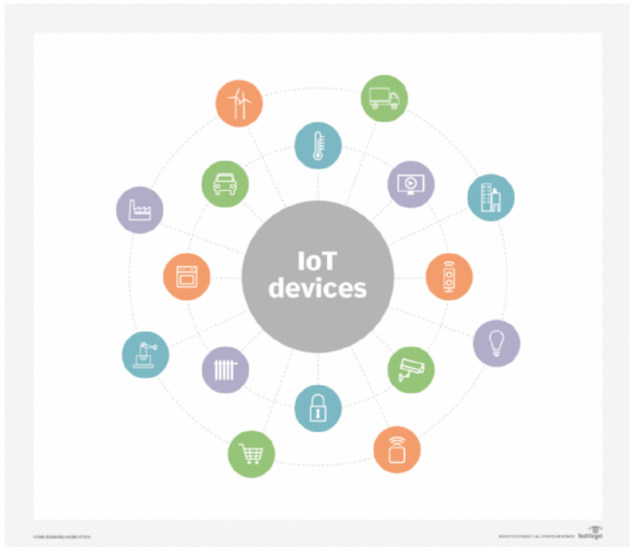


Figure 2: How does IOT work ?

IoT works through a combination of hardware, software, and network technologies that enable objects or devices to connect, communicate, and exchange data over the internet or other communication networks. Here's a simplified overview of how IoT works:

A, Devices and Sensors:

IoT starts with physical devices or objects that are equipped with various sensors and actuators. These sensors can detect changes in the environment, such as temperature, humidity, motion, light, or pressure. Actuators, on the other hand, can perform actions based on received instructions, like turning on a light or adjusting the temperature.



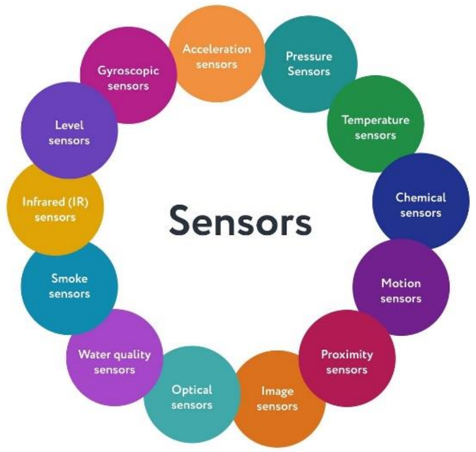


Figure 3: Device and Sensor

Generally, sensors are used in the architecture of IOT devices.

Sensors are used for sensing things and devices etc.

A device that provides a usable output in response to a specified measurement.

The sensor attains a physical parameter and converts it into a signal suitable for processing (e.g. electrical, mechanical, optical) the characteristics of any device or material to detect the presence of a particular physical quantity.

The output of the sensor is a signal which is converted to a human-readable form like changes in characteristics, changes in resistance, capacitance, impedance, etc.

B, Connectivity:

IoT devices need a means of communication to transmit the data they collect. They typically use different communication technologies such as Wi-Fi, Bluetooth, Zigbee, cellular networks, or Low-Power Wide-Area Networks (LPWAN) like LoRaWAN or NB-IoT, depending on the application's requirements.

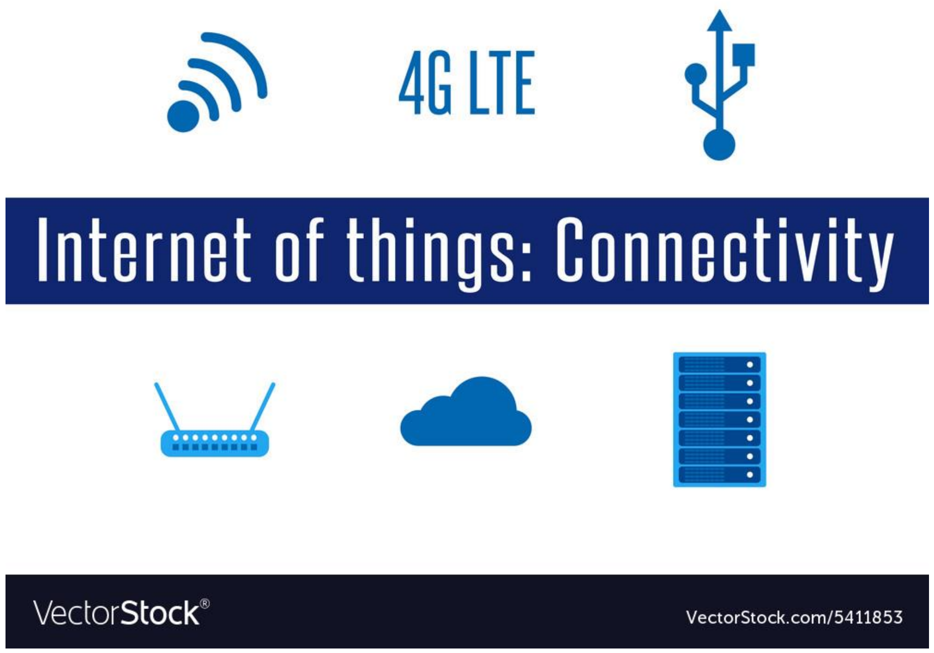


Figure 4: Connectivity IoT

The fragmented nature of IoT deployments mean there are a large number of IoT connectivity standards for organizations to choose from. How to connect to IoT is one of the most important decisions when it comes to IoT. IoT connectivity should be selected based on careful assessment of each deployment's characteristics. For some very high speed, ultra-low latency connectivity is required. This may lead to adoption of 5G or 4G cellular IoT connectivity but this decision must be balanced against the likely cost and the power usage these technologies require. For some simpler deployments, low speed connections that are not always on can be ideal, requiring smaller batteries and delivering IoT connectivity cost effectively.

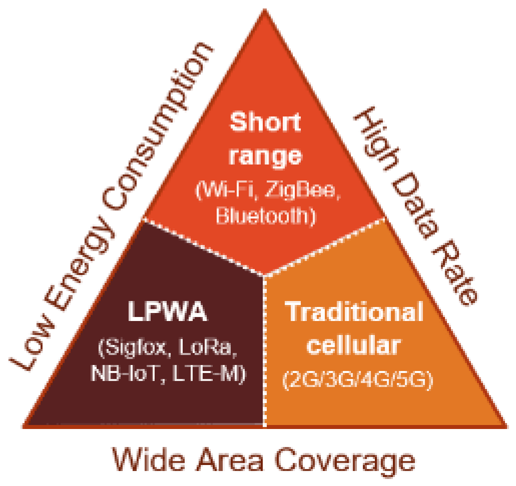


Figure 5: 3 main technical of Connectivity on IoT

The three main technical requirements for any enterprise looking into Internet of Things connectivity are coverage, energy efficiency and data rate. No single technology can excel in all these aspects, as these are trade-offs every radio technology faces. In addition, organizations should consider where their IoT offerings will be provided. If you need global IoT connectivity you should adopt a connectivity technology that is available worldwide.

C, Data Collection and Processing:

The sensors in IoT devices continuously gather data from their surroundings. This data could be temperature readings from a smart thermostat, heart rate measurements from a wearable device, or inventory levels from a smart shelf in a retail store. The collected data is then processed and sometimes analyzed locally on the device itself or sent to a central cloud-based server for further processing.

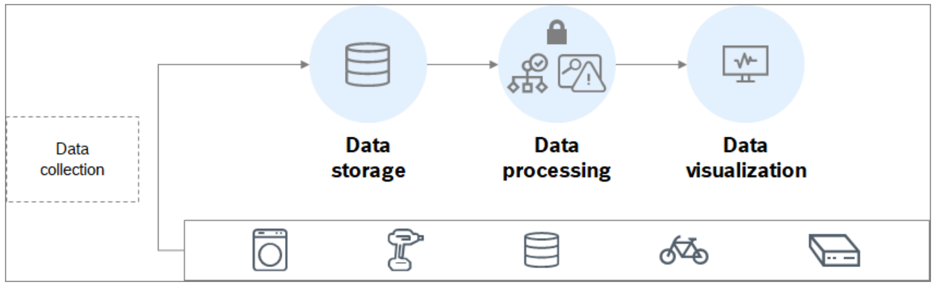


Figure 6: Data Collection and Processing in IoT

There are three common types of IoT data in the commercial real estate industry.

- **Equipment data:** This type of data enables real-time fault detection, runtime-based schedules and predictive maintenance thus saving energy cost, increasing productivity and extending equipment life. Equipment data can be centralized in one platform so that on-site staff, management, and vendors can all be on the same page and data can rollup into executive dashboards.
- **Environmental data:** IoT sensors can be deployed to track a range of data streams within buildings: temperature, air quality, people flow, moisture, or movement. These datasets are primarily used to be proactive for occupant comfort issues and avoid disaster scenarios such as leaks and floods.
- **Submeter data:** Digital Submetering devices connected to the internet help automate the utility submetering process which costs, eliminates errors and generates bills as soon as the billing period ends. Submetering provides transparency to portfolio owners for the recovery rates of their cash outlays.

D, Cloud Infrastructure:

IoT systems often rely on cloud platforms to store and manage the vast amounts of data generated by connected devices. Cloud infrastructure provides scalability, flexibility, and centralized control for data storage and processing. It also enables real-time monitoring, analytics, and visualization of the collected data.

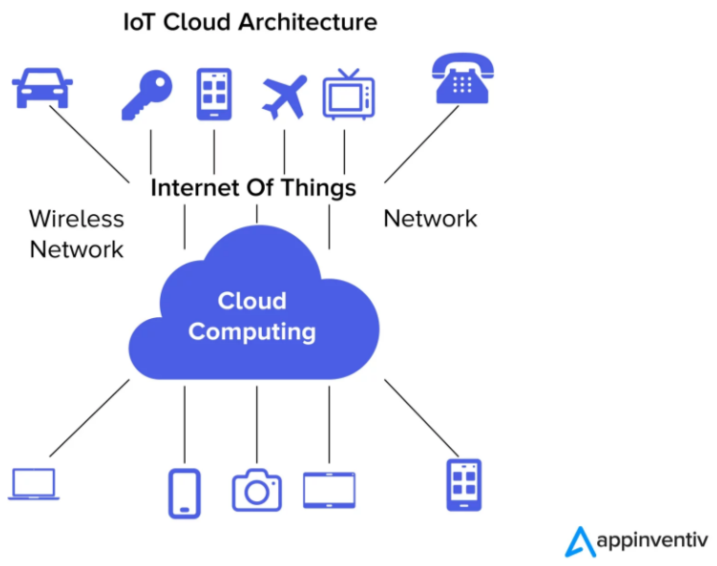


Figure 7: Cloud Infrastructure

It is anticipated that a unique paradigm combining cloud computing and the Internet of Things (IoT cloud-based service) will be disruptive and enable a wide range of application scenarios.

When the capabilities of the IoT and cloud computing tech stacks combine, it creates significant value for both consumers and businesses. This combination is often referred to as an IoT cloud platform.

E, Data Analysis and Action:

Once the data reaches the cloud or edge servers, it can be analyzed using various algorithms and machine learning techniques to extract valuable insights. These insights can be used to trigger specific actions or make informed decisions. For example, an IoT system in a smart home might use data from motion sensors to adjust the lighting and temperature when someone enters a room.

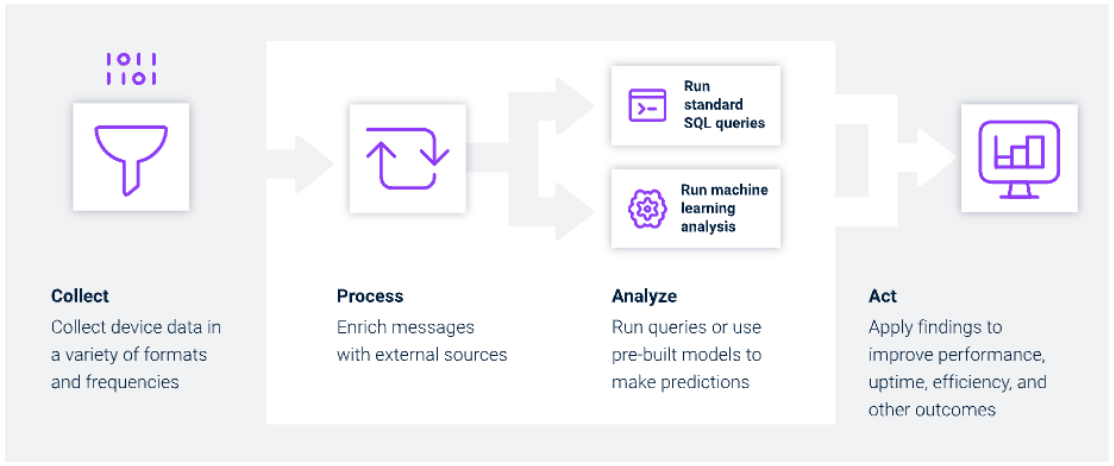

Run standard SQL queries
Run machine learning analysis
Collect
Collect device data in a variety of formats and frequencies
Process
Enrich messages with external sources
Analyze
Run queries or use pre-built models to make predictions
Act
Apply findings to improve performance, uptime, efficiency, and other outcomes

Figure 8: Data Analysis and Action Work in IoT

1.3 Applications of IoT:

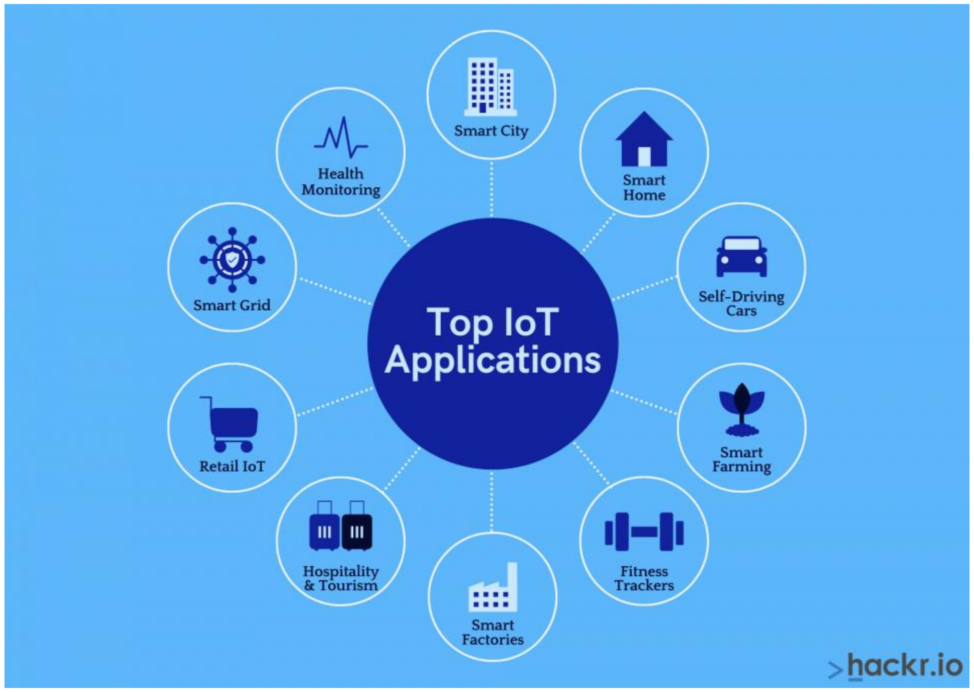


Figure 9: Application in IoT

IoT has a wide range of applications across various industries and domains. Here are some of the key applications of IoT:

A, Smart Home: IoT enables homeowners to create intelligent, automated environments by connecting devices like thermostats, lighting systems, security cameras, door locks, and appliances. Smart homes offer increased energy efficiency, convenience, and enhanced security.

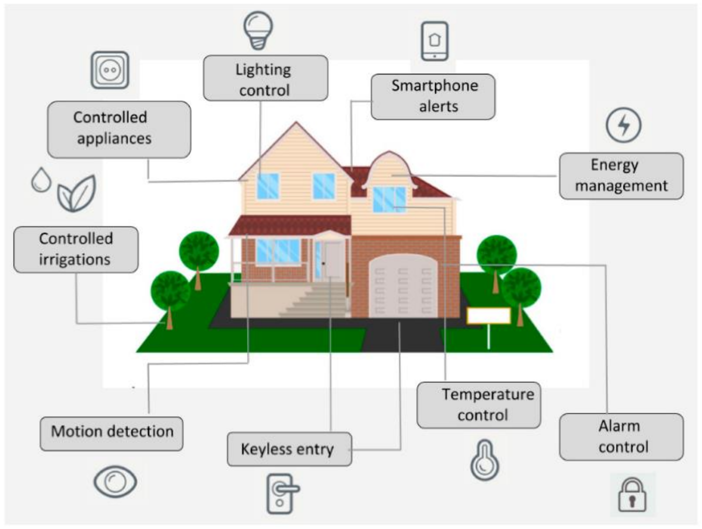


Figure 10: Smart Home in IoT

B, Industrial IoT (IIoT): In industries, IoT is used for remote monitoring and predictive maintenance of machinery and equipment. IIoT enhances operational efficiency, reduces downtime, and enables data-driven decision-making in manufacturing, logistics, and supply chain management.

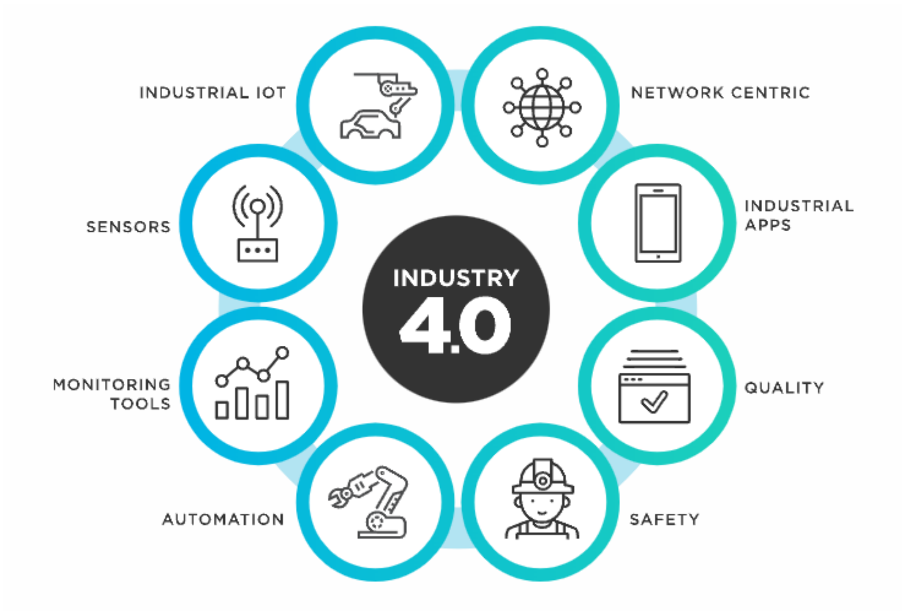


Figure 11: Industrial IoT (IIoT)

C, Healthcare: IoT devices such as wearable fitness trackers, remote patient monitoring devices, and medical sensors help in real-time health monitoring, chronic disease management, and improving overall patient care.

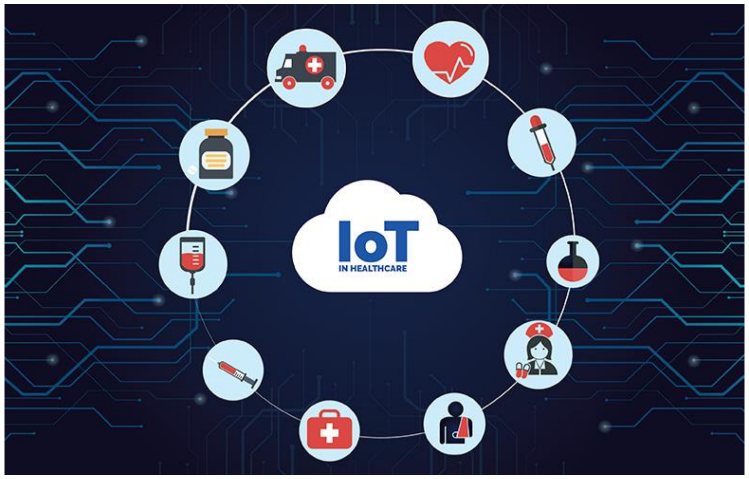


Figure 12: Healthcare in IoT

D, Agriculture: IoT-based solutions are used in smart farming for precision agriculture, where sensors monitor soil moisture, temperature, and other factors, helping farmers optimize irrigation, fertilization, and crop health.

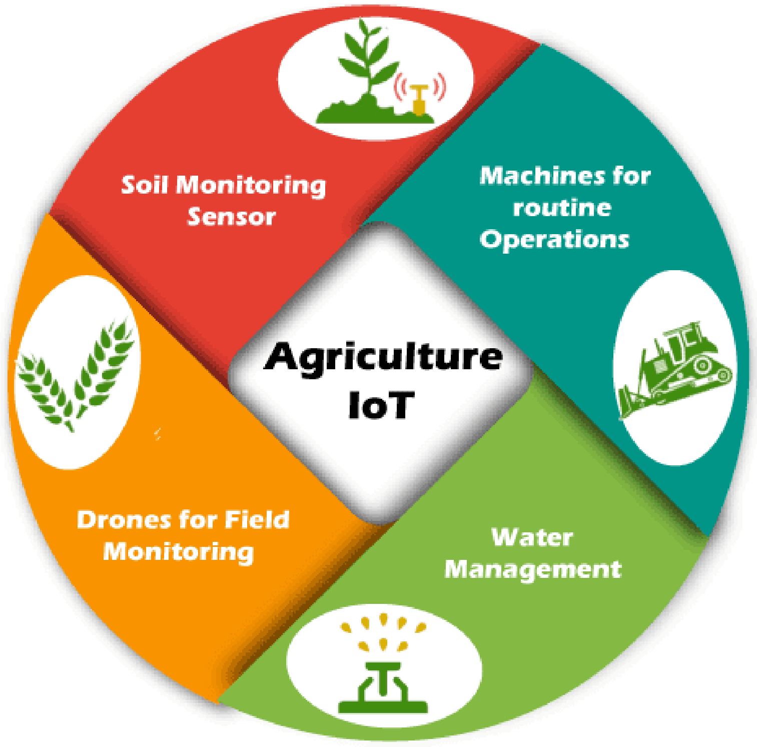


Figure 13: Agriculture in IoTe

E, Smart Cities: IoT technologies are deployed in urban infrastructure for various applications, including traffic management, waste management, smart lighting, environmental monitoring, and public safety.
Environmental Monitoring: IoT sensors are used to monitor air quality, water quality, and weather conditions, providing valuable data for environmental conservation and disaster management.

Figure 14: Smart Cities with IoT

F, Transportation and Logistics: IoT is employed in vehicle tracking, fleet management, smart transportation systems, and logistics optimization, improving efficiency and safety in transportation operations.

Figure 15: Transportation and Logistics in IoT

G, Retail: IoT is used to create personalized shopping experiences, optimize inventory management, and improve customer engagement through beacons, smart shelves, and interactive displays.

Energy Management: IoT devices help in monitoring and optimizing energy usage in buildings, factories, and cities, leading to energy conservation and cost savings.

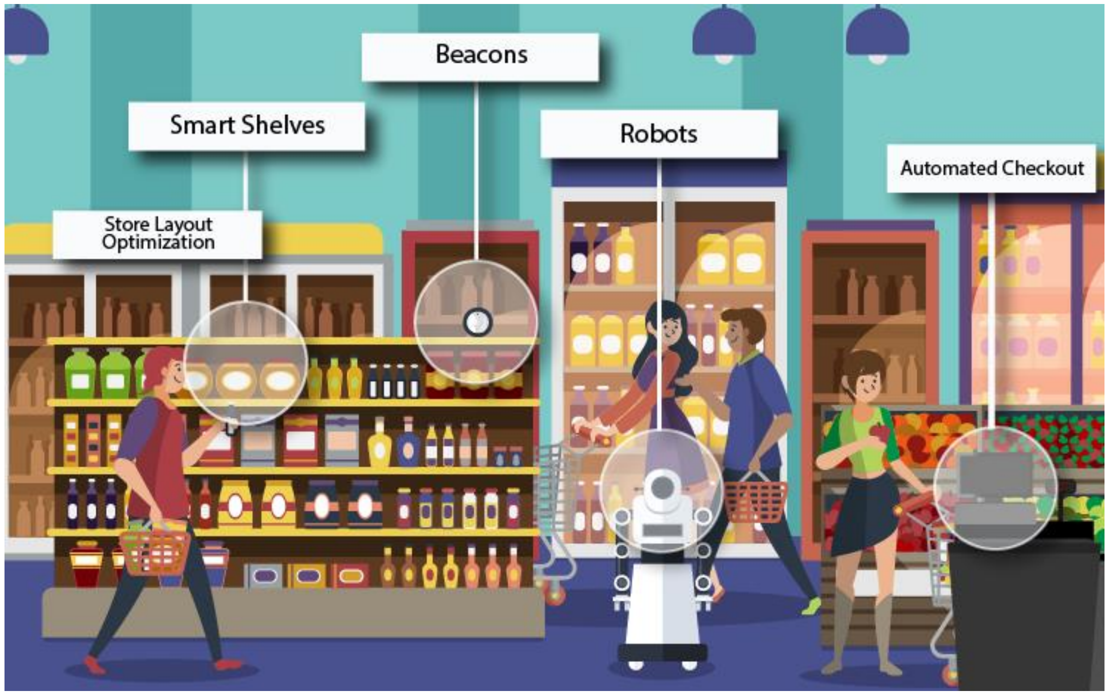


Figure 16: Retail in IoT

I, Wearable Technology: IoT-powered wearable devices, such as smartwatches and fitness trackers, track and analyze user data related to health, fitness, and activity levels.

Smart Grids: IoT technologies are used in electrical grids to improve energy distribution, enable demand-response mechanisms, and promote renewable energy integration.

Home Automation for Assisted Living: IoT can assist elderly or differently-abled individuals to live more independently through smart home devices and wearables that provide safety and health monitoring.

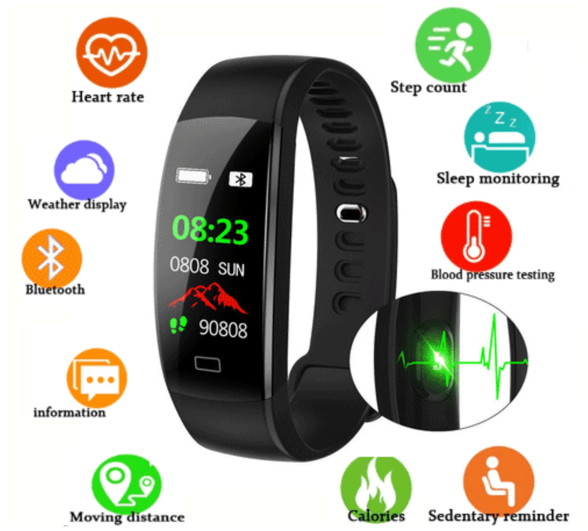


Figure 17: Wearable Technology in IoT

K, Asset Tracking: IoT solutions enable real-time tracking of valuable assets, such as inventory in a warehouse, vehicles, or expensive equipment, helping prevent theft and loss.

Smart Waste Management: IoT-based waste bins can optimize waste collection routes, reduce overflowing bins, and improve waste disposal efficiency.

These are just a few examples of the diverse applications of IoT technology. As the IoT ecosystem continues to evolve, we can expect to see more innovative and transformative use cases across industries and everyday life.

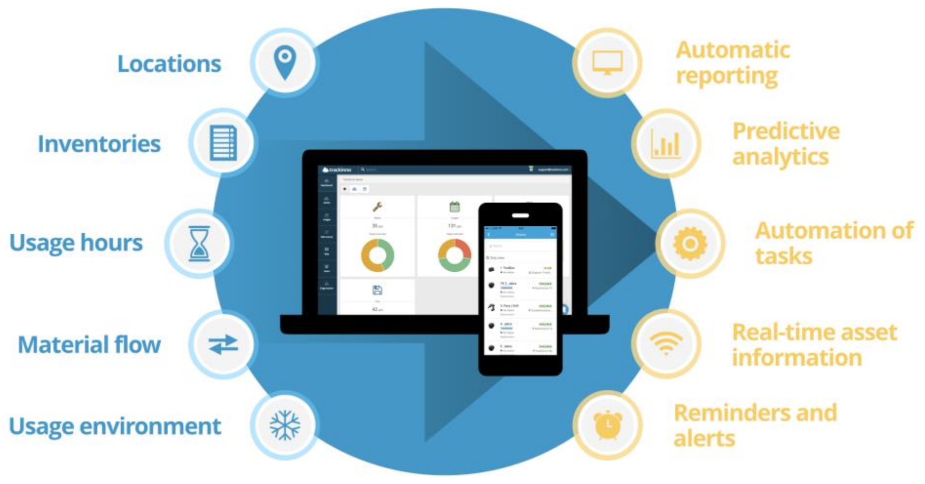


Figure 18: Asset Tracking IoT

1.4 IoT characteristics:

IoT (Internet of Things) has several distinctive characteristics that set it apart from other technologies. Here are some key IoT characteristics:

- **Connectivity:** IoT devices are connected to the internet or other communication networks, allowing them to interact and exchange data with other devices, systems, or cloud platforms. This interconnectedness enables seamless communication and data sharing.
- **Sensing and Actuation:** IoT devices are equipped with various sensors that can detect changes in the environment, such as temperature, humidity, motion, light, or pressure. They also often have actuators that can perform actions based on received instructions, allowing them to interact with the physical world.
- **Data Collection and Analysis:** IoT devices continuously collect data from their surroundings through the embedded sensors. This data can be stored locally on the device or sent to the cloud for further analysis, processing, and insights generation.

- **Automation:** IoT enables automation of processes and actions based on real-time data. Devices can respond to triggers and make decisions without direct human intervention, leading to more efficient and timely operations.
- **Interoperability:** IoT devices and systems often follow standardized communication protocols and data formats to ensure interoperability. This enables devices from different manufacturers to work together and integrate seamlessly into IoT ecosystems.
- **Real-time Processing:** Many IoT applications require real-time or near real-time data processing to enable timely actions and responses. Edge computing is often used to process data closer to the source, reducing latency and enhancing responsiveness.
- **Scalability:** IoT solutions can scale easily to accommodate a growing number of devices and data points. Cloud-based platforms are commonly used to provide the scalability needed to handle the massive amounts of data generated by IoT devices.
- **Security and Privacy:** IoT devices and systems handle sensitive data, making security a critical concern. Secure authentication, encryption, and data privacy measures are essential to protect against unauthorized access and potential data breaches.
- **Ubiquity:** IoT technology can be applied to virtually any physical object, from consumer devices like smartwatches and smart home appliances to industrial equipment and infrastructure components.
- **Energy Efficiency:** Many IoT devices are designed to be energy-efficient, especially those powered by batteries. Optimized power consumption allows devices to operate for extended periods without frequent recharging or replacement.
- **Remote Management:** IoT devices can be remotely managed and updated, reducing the need for physical intervention and maintenance. This capability is especially beneficial for large-scale deployments in diverse locations.
- **Data Analytics and Insights:** The vast amount of data generated by IoT devices can be analyzed using advanced analytics and machine learning algorithms to derive valuable insights, enabling data-driven decision-making.

These characteristics collectively enable IoT to revolutionize various industries and improve the way we live and work by creating intelligent, interconnected systems that enhance efficiency, productivity, and user experiences. However, they also raise challenges in terms of security, privacy, and data management that must be addressed to ensure the responsible and sustainable growth of the IoT ecosystem.

2. Introduce almost standard architectures, frameworks, tools, hardware and APIs that are available for using in IoT development

2.1 Standard architectures:

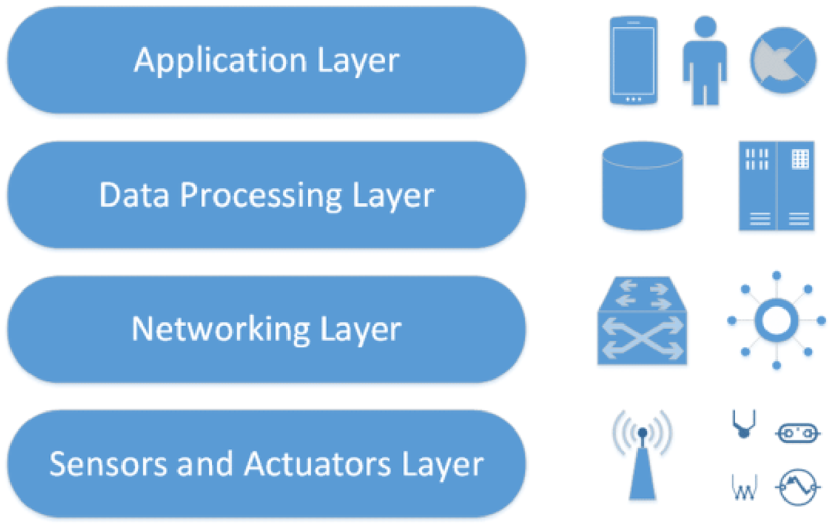


Figure 19: Standard architectures

A, Application Layer:

The application layer defines all applications in which IoT has deployed. It is the interface between the end IoT devices and the network. IoT Applications such as smart homes, smart health, smart cities, etc. It has the authority to provide services to the applications. The services may be different for each application because of services based on the information collected by sensors.

It is applied through a dedicated application at the device end. Such as for a computer, the application layer is applied by the browser. It is the browser that executes application layer protocols like HTTP, HTTPS, SMTP, and FTP. There are many concerns in the application layer out of which security is the key issue.

Common issues and threats of application layers are:

Cross-site scripting:

- It is a type of computer security infirmities that typically found in web applications. It enables attackers to inject client-side scripts such as JavaScript into web pages viewed by other users. By doing so, an attacker can completely change the contents of the application as per his needs and use original information in an illegal way.

- The first method you can use to prevent cross-site scripting from appearing in your applications is by escaping user input. Escaping input means taking the data from an application has received and ensured it's secure before supplying it for the end-user.
- Validating input is another process of ensuring an application is providing the correct data and protecting harmful data from doing harm to the site, database, and users.

Malicious Code Attack:

- It is a particular code in any part of the software system or script that is considered to cause undesired effects, security threats or damage to the system. It is that threat that may not be blocked or controlled by the use of anti-virus software.
- Static Code Analysis (SCA) is an effective method to protect the malicious code from successfully causing any damage to computers. SCA is a debugging method of a computer program that is done by analyzing the code without performing the program. The process supplies an understanding of the code structure and can help to ensure that the code should match with industry standards.
- Today's superior scanners can easily detect malicious code such as a Data Leakage, Time Bombs, Anti Debugging techniques, backdoor Threats, etc.

B, Data Processing Layer:

In three-layer architecture, the data were directly sent to the network layer. Due to sending data directly the chances of getting damages increase. In four-layer architecture, data is sent to this layer that is obtained from a perception layer. Data Processing Layer has two responsibilities it confirms that data is forwarded by the authentic users and prevented from threats.

Authentication is the most commonly used method to verify the users and the data. It is applied by using pre-shared, keys and passwords to the concerned user. The second responsibility of the layer is to send information to the network layer. The medium through which data is transferred from the Data Processing Layer to the network layer can be wireless and wire-based.

Common issues and threats of the Data Processing layer are:

DoS Attack:

- An attacker sends a huge amount of data to make network traffic overloaded. Thus, the huge consumption of system resources exhausts the IoT and makes the user unable to access the system.

- Deploy an antivirus program and firewall will restrict the bandwidth usage to authenticated users only. Server Configuration is another method that can help reduce the probability of being attacked.

Malicious Insider Attack:

- It comes from the inside of an IoT environment to access private information. It is conducted by an authorized user to access the information of another user.
- The practices such as an encryption of data, implementing proper password management practices, installing antivirus will helps to keep data safe from such threats.

<u>C, Network Layer:</u>

This layer is also known as a transmission layer. It acts like a bridge that carries and transmits data gathered from physical objects through sensors. The medium can be wireless or wire-based. It also connects the network devices and networks to each other. Hence, it is extremely sensitive to attacks from the attackers. It has important security issues regarding integrity and authentication of data that is being transmitted to the network.

Common issues and threats of the Network layer are:

Main-in-The-Middle Attack:

- MiTM attack is an attack where the attacker privately intercepts and modifies the communication between the sender and receiver who assume they are directly communicating with each other. It leads to a serious threat to online security because they give the attacker the pathway to capture and control data in real-time
- Secure/Multipurpose Internet Mail Extensions encrypts emails which ensures that only intended users can read will prevent data from MITM Attacks.

Storage Attack:

- The crucial information of users is saved on storage devices or on the cloud. Both the storage devices and the cloud can be attacked by the attacker and the user’s information may be modified to incorrect details.
- By making regular backups of files, by running anti-virus software and using a system with strong passwords so that data access is restricted are the ways by which we can protect data from the attacker.

Exploit Attack:

- An exploit is any unethical or illegal attack in a form of software, blocks of data or a sequence of commands. It takes benefit of security infirmities in an application, system or hardware. It usually occurs with the goal of getting control of the system and steals information stored on the network. By installing all software patches, security releases and all updates for your software are few preventive measures against attack.

D. Perception layer/Sensor layer:

The sensor layer has the responsibility to recognize things and gather the data from them. There are many types of sensors connected to the objects to gather information such as RFID, sensors and 2-D barcode. The sensors are selected as per the requirement of applications. The data that is collected by these sensors can be about location, changes in the air, environment, etc. However, they are the main aim of attackers who wish to use them to replace the sensor with their own.

Hence, most of the threats are related to sensors are

Eavesdropping:

- It is an unauthorized real-time attack where personal communications, such as phone calls, fax transmissions, text messages are intercepted by an attacker. It tries to take crucial information that is transferred over a network. Preventive measures such as Access control, continuous supervision/observation of all devices, thorough inspection by a qualified technical countermeasures specialist of all components need to be ensured.

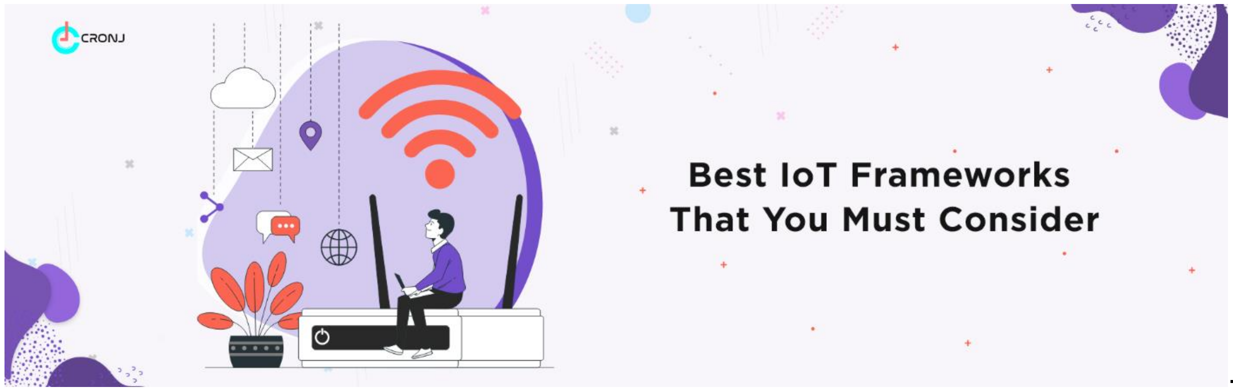
Replay Attack:

- It is also known as a playback attack. It is an attack in which an attacker intrudes on the conversation between the sender and receiver and extracts authentic information from the sender. The added risk of replay attacks is that a hacker doesn’t even need improved skills to decrypt a message after seizing it from the network.
- This attack can be prevented by using Timestamps on all messages. This protects hackers from resending messages sent longer ago than a particular length of time. Another preventive measure to avoid becoming a victim is to set a password for each transaction that’s only used once and discarded.

Timing Attack:

- It is usually utilized in devices that have weak computing abilities. It allows an attacker to find vulnerabilities and withdraw secrets maintained in the security of a system by observing how long it takes the system to respond to various queries, input or other algorithms.
- To prevent this attack we can simply make a constant-time comparison either by using timer functions. You should have tests that make sure that compiler optimizations don't place timing infirmities back into your code.
- In this way, IoT 4 layer architecture can fulfill the requirements of IoT regarding security and privacy. We have described different research about layered architectures of IoT and also outline the security attacks based on the layers that can affect the performance of IoT.
- hIOTron shares practical approaches and tactics that will readily useful in the industry. hIOTron's IoT Course helps weigh in many factors that come handy in building successful IoT products.

2.2 Framework:



-

Today, the Internet has become a major part of the world. It finds application in every corner of the globe. With the increasing use of the Internet, more appliances and devices are now being connected to the web, forming the 'Internet of Things,' commonly known by its acronym 'IoT'. The IoT is not just a simple technology. It is a complex framework that includes several different technologies, designed to work together in tandem. In this article, you will learn what the IoT framework is in detail, as well as the Open-source IoT frameworks.

In typical scenarios where large data is generated and transmitted across multiple devices, there is a crucial focal point where all the data is gathered and consolidated. This specific point plays a vital role in the network as it facilitates the combination of all the data, enabling a comprehensive understanding of the generated information.

However, achieving seamless data transmission and generation is not a coincidence; rather, it is made possible through the implementation of the Internet of Things Framework (IoT framework). But what exactly is the IoT framework?

The Internet of Things (IoT) Framework can be described as an interconnected ecosystem comprising various devices that communicate with one another over the Internet. These connected devices are designed to exchange and sense data autonomously, requiring minimal human intervention.

The IoT framework plays a crucial role in enabling smooth communication among these connected devices over the Internet. It is aptly named the 'Internet of Things' framework because it facilitates the interaction of 'Things' (devices) over the Internet.

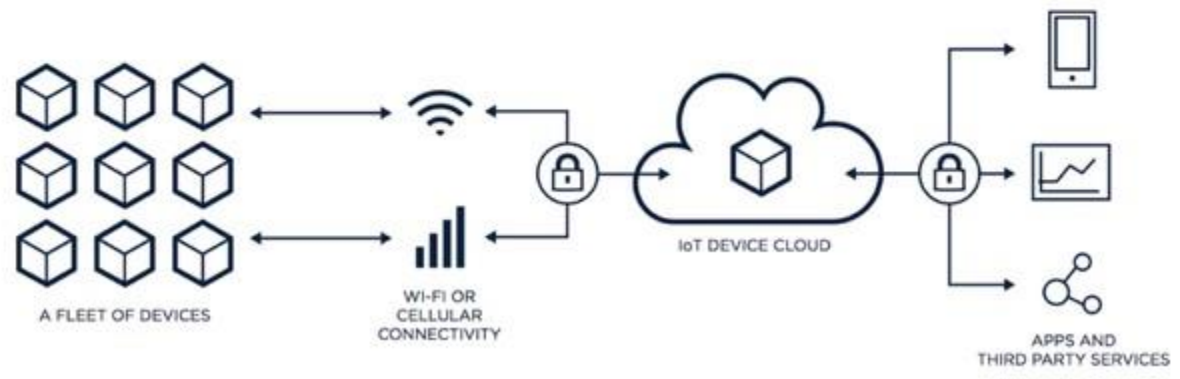


Figure 20: IoT Framework Overview

The IoT framework holds significant importance in today's technology-driven world, with applications spanning across almost every industry. One prominent area where it is extensively utilized is in the creation of smart homes.

Furthermore, the concept of the IoT framework is extended to the development of various physical objects, including thermostats, electrical appliances, security and alarm systems, and even vending machines, among numerous other objects.

Open Source IoT Frameworks:

To grasp the concept of an IoT framework open source, consider the following three key points:

- **Consumer Freedom:** Every consumer desires the freedom to use technology devices of their choice without being restricted to products from a single vendor. For example, some smartwatches only work when paired with smartphones from the same brand.
- **Easy Integration for Dealers:** IoT device vendors want a seamless integration of their products with various technology ecosystems, enabling compatibility with a wide range of devices.
- **Application Development Flexibility:** Application developers wish to support multiple devices without having to write specific code for each vendor's products.

An Open source framework presents a solution to address these challenges. It enables scalability and high flexibility levels, granting consumers the freedom to choose, vendors smoother integration opportunities, and developers the ability to create applications without vendor-specific codes.

The beauty of IoT framework open sources lies in their accessibility, as they are generally free to download, easy to install, and simple to launch, facilitating widespread adoption and usage.

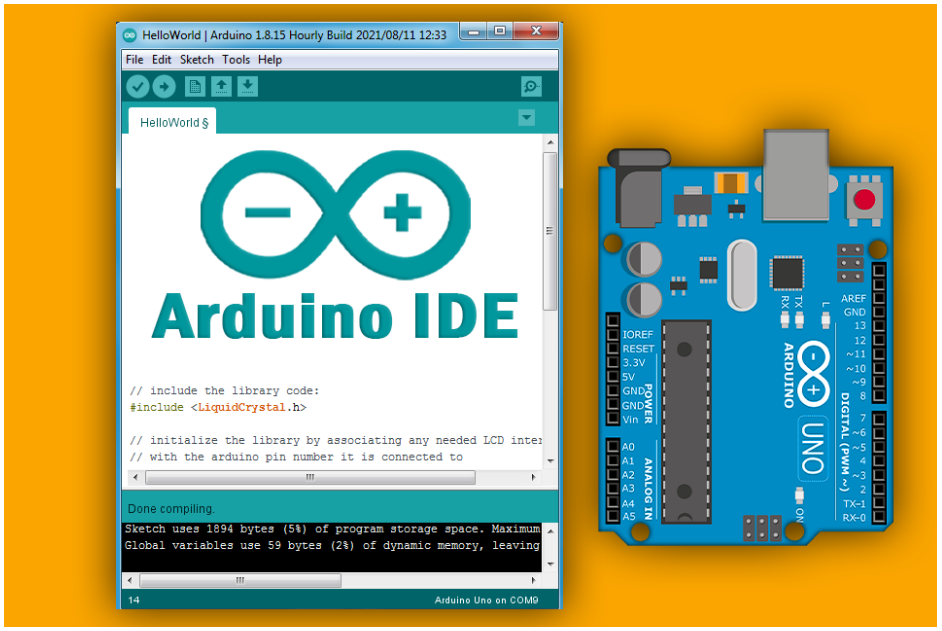
Example:

Arduino stands out as one of the top recommended open-source IoT frameworks, particularly for individuals looking to create a computer system capable of sensing and exerting control over the surrounding environment.

Arduino's open-source nature encompasses both hardware and software components, making it exceptionally user-friendly and accessible. Its simplicity makes it an ideal choice for various IoT projects.

The functioning of the Arduino open-source framework relies on several hardware specifications, often employed in interactive electronics, enabling seamless operation and interaction with the physical world.

2.3 Tools:



```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD inter
// with the arduino pin number it is connected to
```

```
Sketch uses 1894 bytes (5%) of program storage space. Maximum
Global variables use 59 bytes (2%) of dynamic memory, leaving
```

Figure 21: Arduino Tools

Arduino is an advanced and versatile IoT tool designed to connect various equipment from different frameworks seamlessly. Equipped with a wide array of libraries and models to aid IoT app developers, Arduino offers the ideal combination of features. For those seeking to create a computer system capable of superior control and sensing abilities in the physical world, Arduino is the perfect choice, surpassing typical stand-alone computing machines.

As an open-source prototyping platform, Arduino offers a straightforward and user-friendly IoT solution. It provides an Integrated Development Environment and a dedicated programming language. By operating with a set of hardware specifications applicable to numerous interactive electronic devices, Arduino ensures a seamless integration of IoT software and hardware. It comes with pre-equipped, cost-effective devices that effortlessly interact with the environment, making it a practical and accessible choice for IoT projects.

2.4 Hardware

There is a wide range of hardware options available for IoT development, catering to various application requirements and use cases. Here are some popular hardware components commonly used in IoT projects:

Microcontrollers and Single-Board Computers (SBCs):

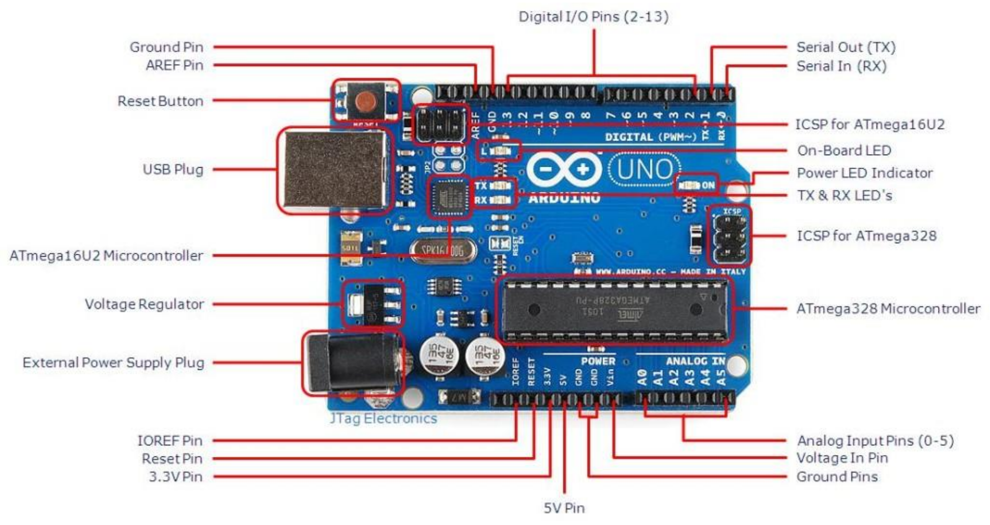


Figure 22: Arduino boards

- **Arduino:** Arduino boards are widely used in IoT prototyping due to their simplicity, ease of use, and a vast community of developers. They come with various models and specifications, offering different connectivity options like Wi-Fi, Bluetooth, and LoRa.

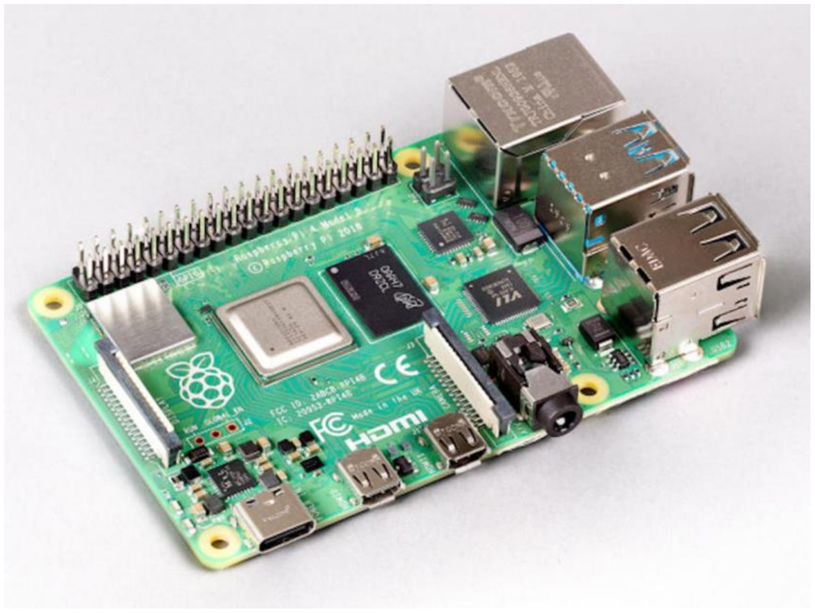


Figure 23: Reaspberry board

- **Raspberry Pi:** Raspberry Pi is a popular SBC that provides more processing power than traditional microcontrollers. It can run full-fledged operating systems and is commonly used in IoT projects that require more computational capabilities.

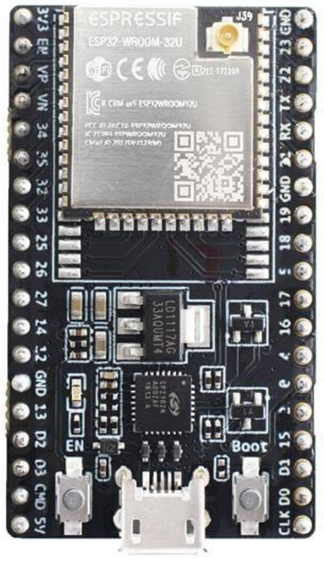


Figure 24: ESP32 board

- **ESP8266 and ESP32:** These low-cost and power-efficient Wi-Fi enabled microcontrollers are highly popular in IoT projects for their built-in Wi-Fi capabilities and GPIO pins for sensor connections.

Sensors:

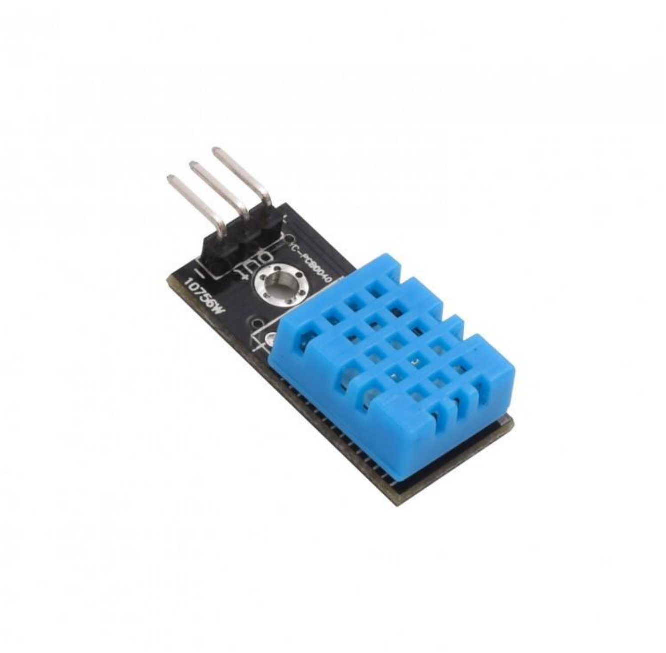


Figure 25: Temperature and Humidity Sensors

- **Temperature and Humidity Sensors:** These sensors measure ambient temperature and humidity levels, essential for climate control and environmental monitoring applications.

Figure 26: Motion Sensors

- **Motion Sensors:** Motion detectors, like PIR (Passive Infrared) sensors or accelerometers, detect movement and are commonly used in security and automation systems.

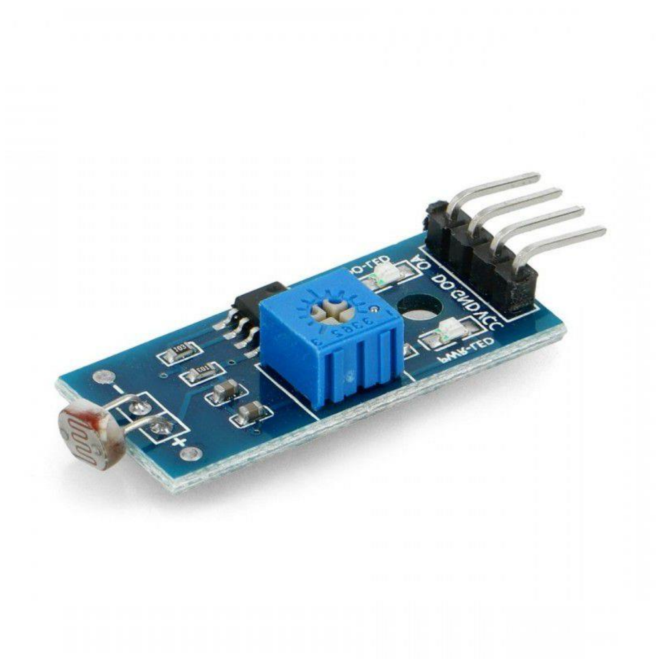


Figure 27: Light Sensors

- **Light Sensors:** Light sensors measure ambient light levels and are used in applications like automatic lighting control and smart energy management.

Figure 28: Proximity Sensors

- **Proximity Sensors:** Proximity sensors detect the presence or absence of objects within a certain range and are used in applications like touchless interfaces and obstacle detection.

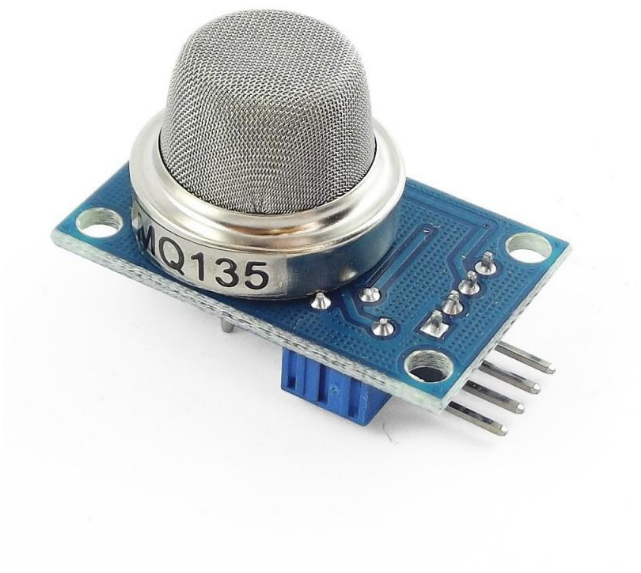


Figure 29: Gas and Air Quality Sensors

- **Gas and Air Quality Sensors:** These sensors measure gas concentration levels and air quality parameters for applications in environmental monitoring and safety systems.

Communication Modules:

- **Wi-Fi Modules:** Wi-Fi modules enable IoT devices to connect to Wi-Fi networks, providing high-speed data transmission and internet connectivity.
- **Bluetooth Modules:** Bluetooth modules are used for short-range wireless communication between IoT devices and smartphones or other Bluetooth-enabled devices.
- **Zigbee Modules:** Zigbee is a low-power, low-data-rate communication protocol suitable for IoT applications with a focus on low power consumption and reliability.
- **Cellular Modules:** Cellular modules, such as GSM, 3G, or LTE, enable IoT devices to connect to cellular networks, providing wide-area connectivity for remote or mobile applications.

Gateways:

- IoT Gateways: IoT gateways act as intermediaries between IoT devices and the cloud or data center. They enable data preprocessing, edge computing, and protocol translation.

Actuators:

- **Relays:** Relays are used to control high-power devices or electrical loads, allowing IoT systems to switch devices on or off remotely.
- **Servo Motors:** Servo motors are used for precise control and movement in applications like robotics and home automation.

Displays:

- **OLED Displays:** OLED displays are compact and power-efficient, making them suitable for small IoT devices requiring visual output.
- **LCD Displays:** LCD displays provide larger screen sizes and are commonly used in IoT devices that need more extensive user interfaces.

Power Sources:

- **Batteries:** Batteries are widely used to power portable and wireless IoT devices, offering flexibility and ease of use.
- **Solar Panels:** Solar panels can be integrated with IoT devices to provide sustainable and renewable power, especially for remote and outdoor applications.

These are just some of the many hardware components available for IoT development. Developers can choose the appropriate hardware based on factors like power requirements, connectivity, processing capabilities, and the specific needs of their IoT projects.

2.5 API

When it comes to IoT security, several APIs (Application Programming Interfaces) and frameworks are available to help developers implement robust security measures in their IoT applications. These APIs address various aspects of IoT security, including authentication, encryption, access control, and secure communication. Here are some commonly used IoT security APIs:

TLS (Transport Layer Security) API:

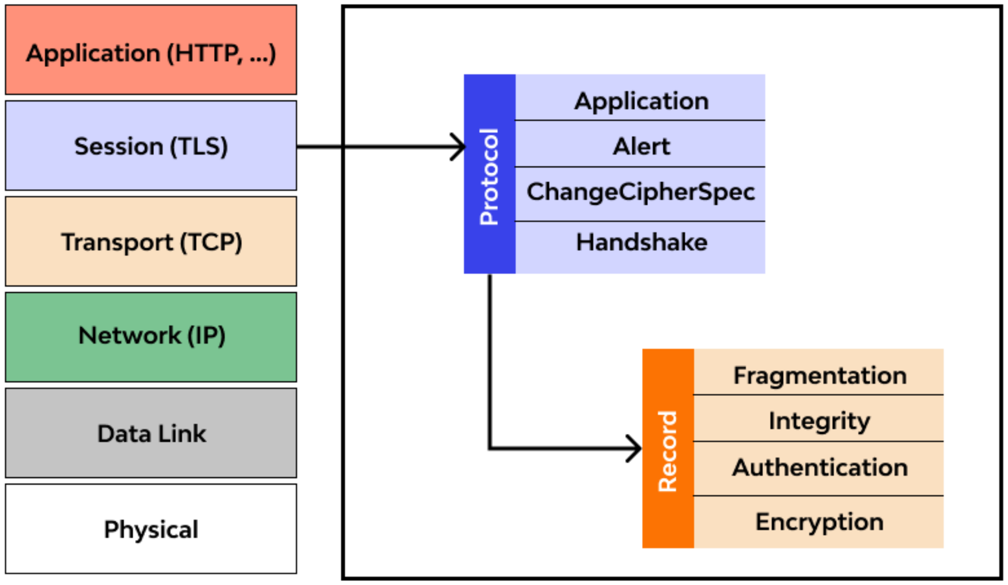


Figure 30: Tls API

- TLS is a cryptographic protocol that provides secure communication over a network. It ensures data privacy and integrity during transmission between IoT devices and servers. Implementing TLS in IoT applications helps prevent eavesdropping and man-in-the-middle attacks.

MQTT API:

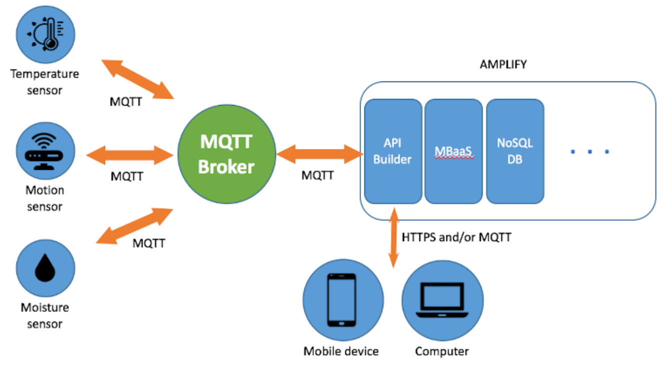


Figure 31: MQTT API

- MQTT (Message Queuing Telemetry Transport) is a lightweight and efficient messaging protocol widely used in IoT. MQTT offers support for authentication and encryption, making it a secure choice for communication between IoT devices and brokers or gateways.

CoAP (Constrained Application Protocol) API:

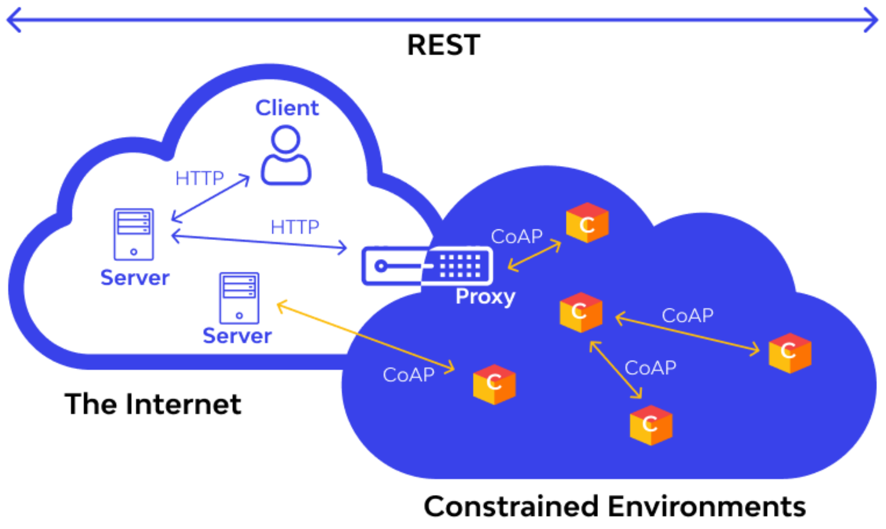


Figure 32: CoAP (Constrained Application Protocol) API

- CoAP is a specialized protocol designed for constrained IoT devices and networks. It provides lightweight communication with built-in support for security mechanisms like Datagram Transport Layer Security (DTLS) for secure transport.

OAuth 2.0 API:

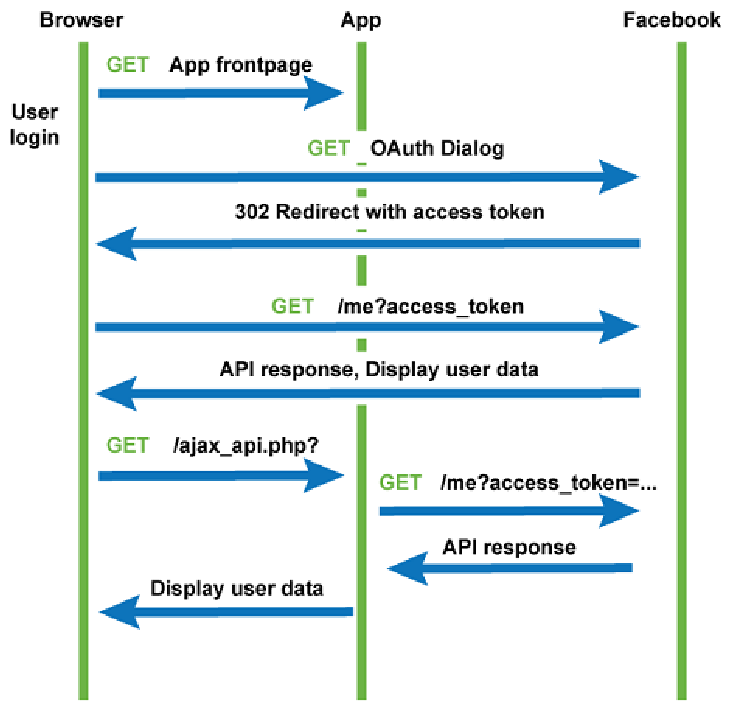


Figure 33: OAuth 2.0 API

- OAuth 2.0 is an industry-standard authorization framework used to grant secure, token-based access to resources. It enables secure authorization and access control for IoT devices trying to interact with cloud services or APIs.

PKI (Public Key Infrastructure) API:

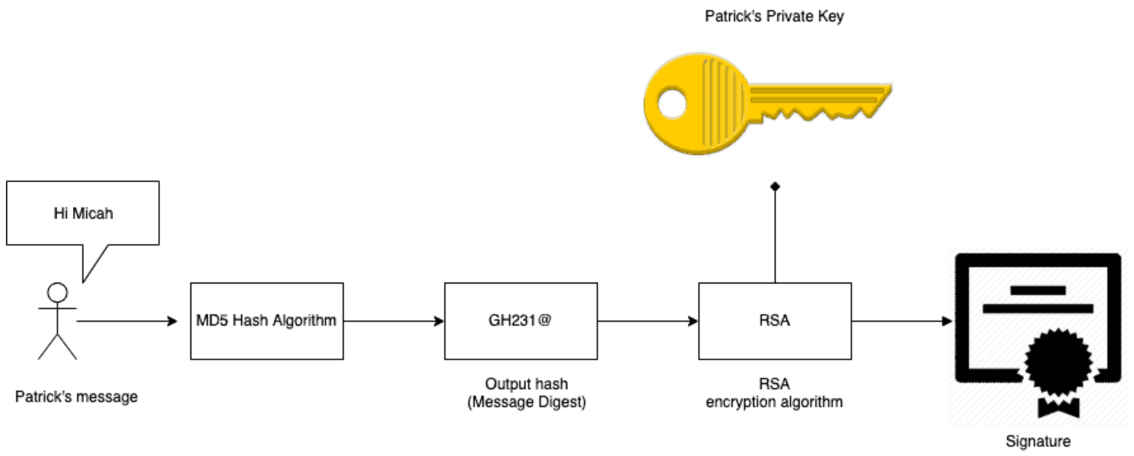


Figure 34: PKI API

- PKI is a set of protocols and standards that enable secure communication through the use of public and private key pairs. PKI APIs help manage digital certificates, verify identities, and establish secure connections in IoT applications.

ECDH (Elliptic Curve Diffie-Hellman) API:

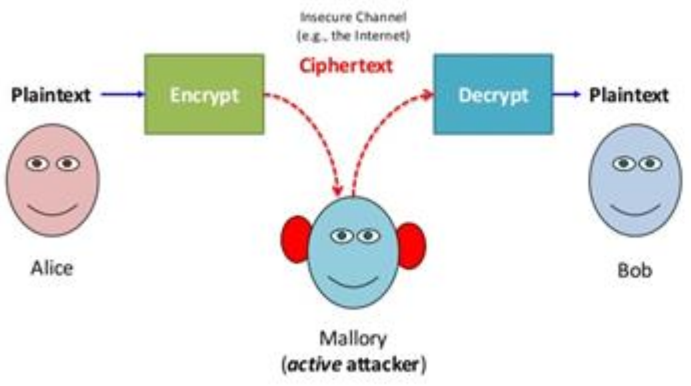


Figure 35: ECDH (Elliptic Curve Diffie-Hellman) API

- ECDH is a key exchange algorithm used in IoT security to establish shared secret keys between two communicating parties. This API is essential for secure communication and data encryption.

Device Provisioning API:

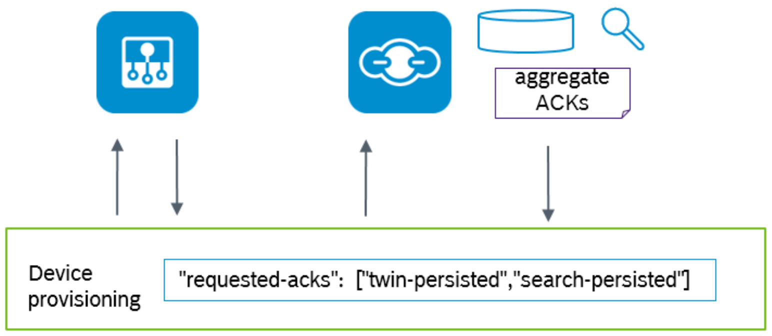


Figure 36: Device Provisioning API

- Device provisioning APIs enable secure onboarding of IoT devices into the network or cloud platform. They help authenticate new devices, assign unique identifiers, and establish secure connections during the initial setup process.

Security Token Service (STS) API:

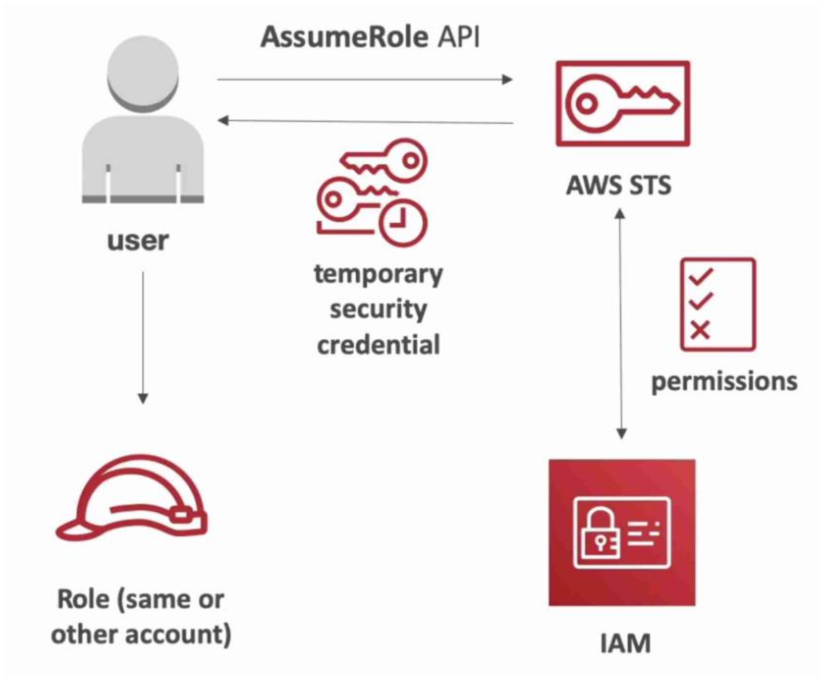


Figure 37: Security Token Service (STS) API

- STS APIs provide temporary security tokens to IoT devices for secure access to cloud services or resources. These tokens have a limited validity period, adding an extra layer of security to IoT interactions.

Secure Element API:

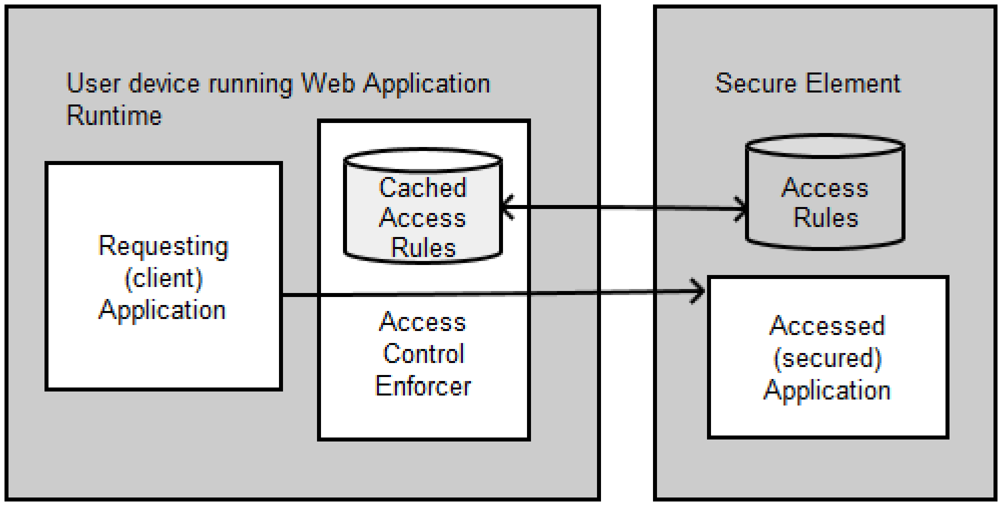

User device running Web Application Runtime
Requesting (client) Application
Cached Access Rules
Access Control Enforcer
Secure Element
Access Rules
Accessed (secured) Application

Figure 38: Secure Element API

- Secure Element APIs are used to interact with hardware security modules or secure chips embedded in IoT devices. These elements store sensitive data, such as cryptographic keys, in a tamper-resistant manner, ensuring better protection against attacks.

IoT Security Frameworks:

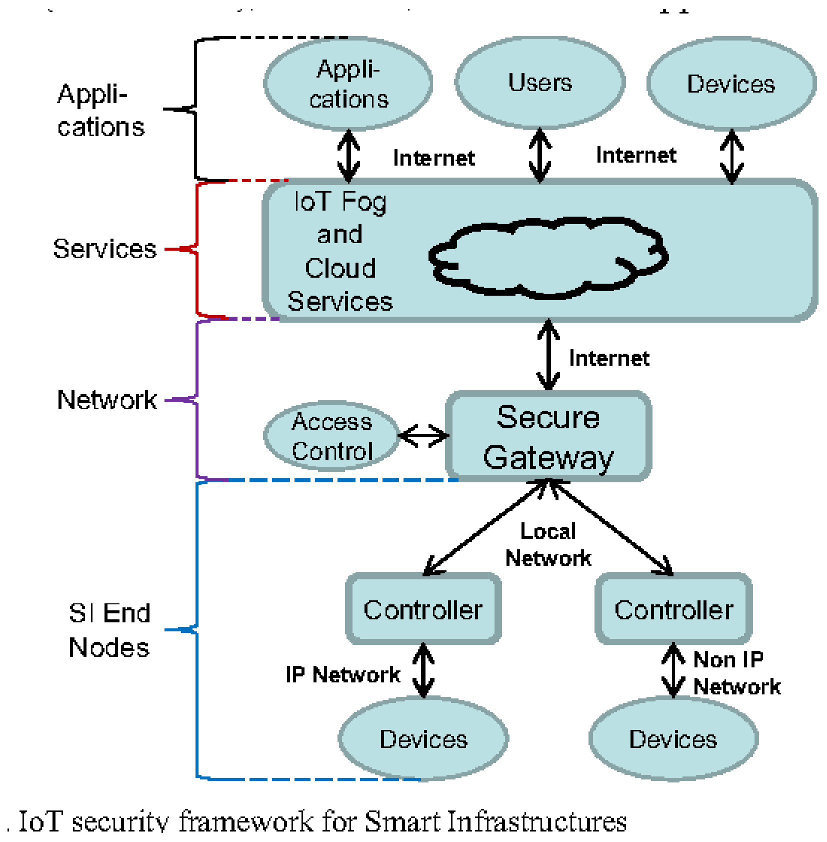


Figure 39: IoT Security Frameworks

Apart from individual APIs, there are comprehensive IoT security frameworks that provide a holistic approach to securing IoT applications. Some popular frameworks include:

- PSA (Platform Security Architecture) by Arm
- IoTSF (IoT Security Foundation) Security Compliance Framework

Developers should choose the appropriate security APIs and frameworks based on the specific requirements of their IoT projects. Implementing robust security measures is crucial to protect IoT devices, data, and communication from potential threats and attacks.

3. Evaluate the impact of common IoT architecture, frameworks, tools, hardware and APIs in the software development lifecycle (M1-M2)

3.1 Impact of Common IoT Architecture, Frameworks, Tools, Hardware, and APIs in the Software Development Lifecycle (SDLC)

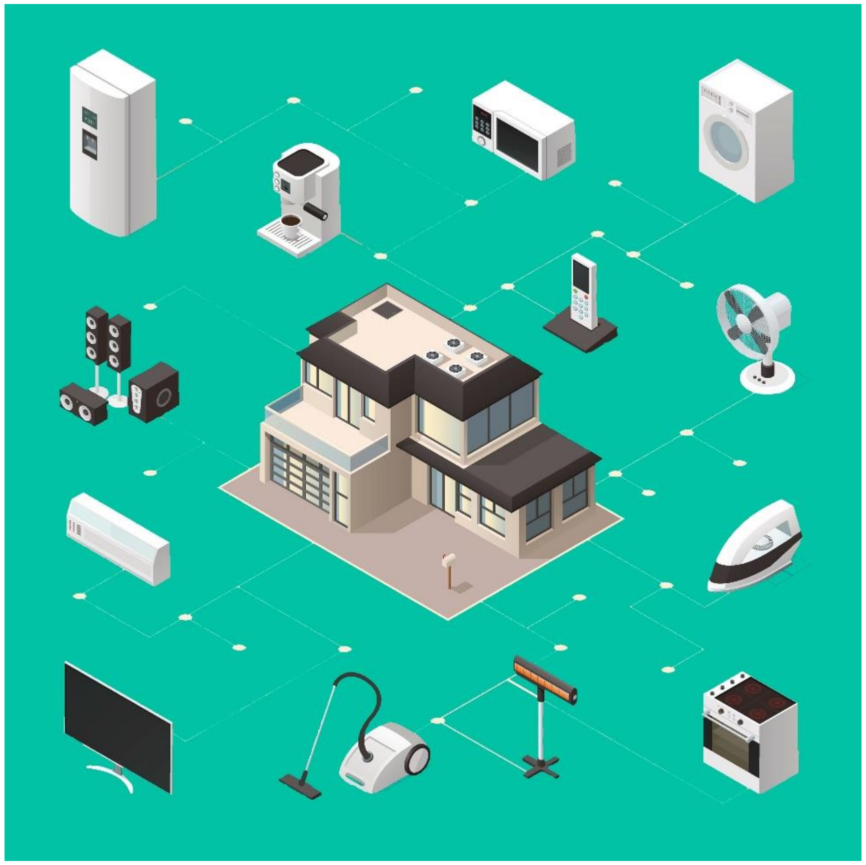
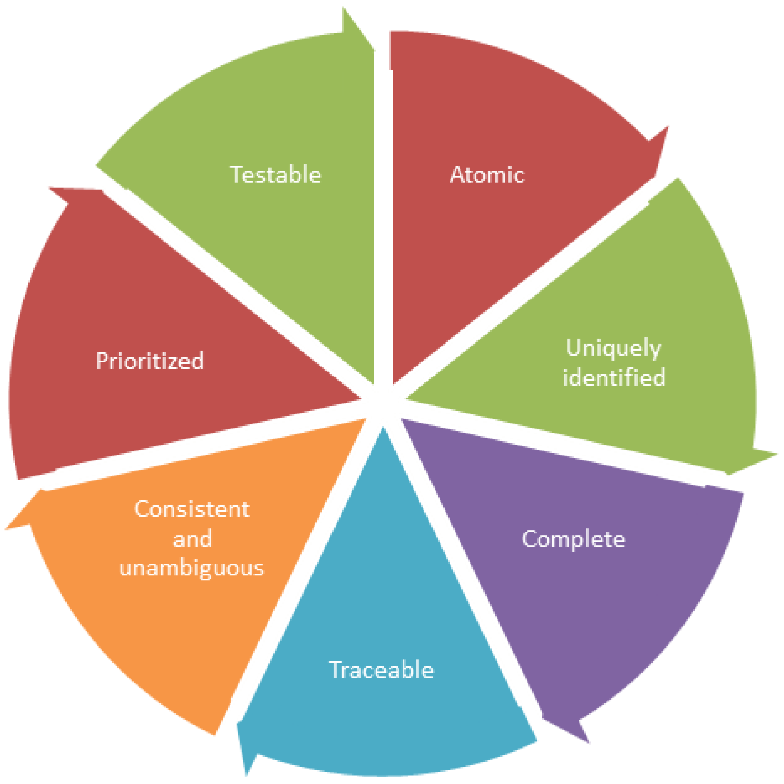


Figure 40: Internet of Things

The Internet of Things (IoT) has significantly transformed the software development lifecycle, introducing various elements that need to be considered at each phase of the process. Let's explore how common IoT architecture, frameworks, tools, hardware, and APIs impact the different phases of the SDLC:



- **Requirements Analysis:** The IoT's unique nature necessitates gathering requirements related not only to traditional software functionalities but also to hardware components, data acquisition, communication protocols, and integration with various IoT devices and sensors. Stakeholder collaboration becomes essential to understand their specific needs and use cases.

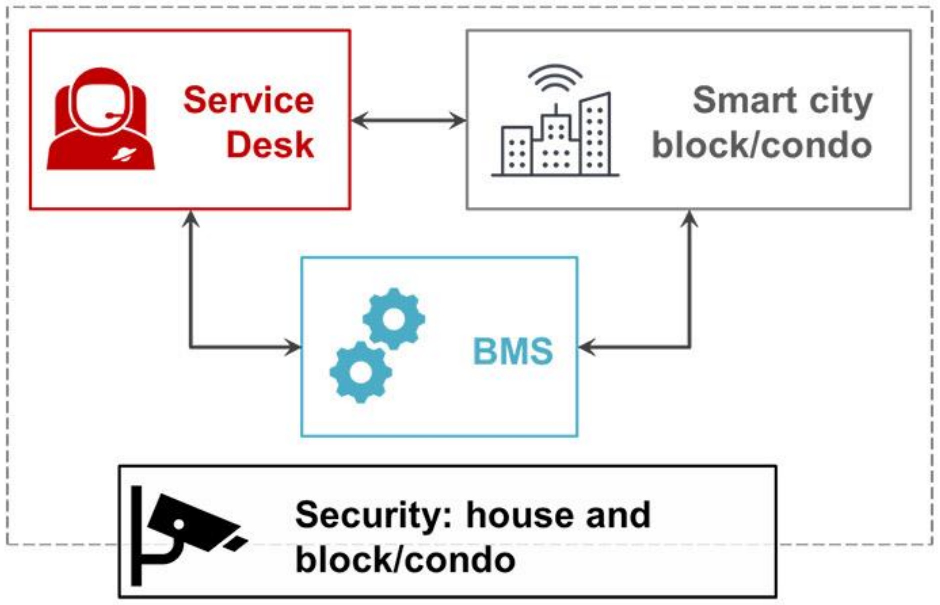
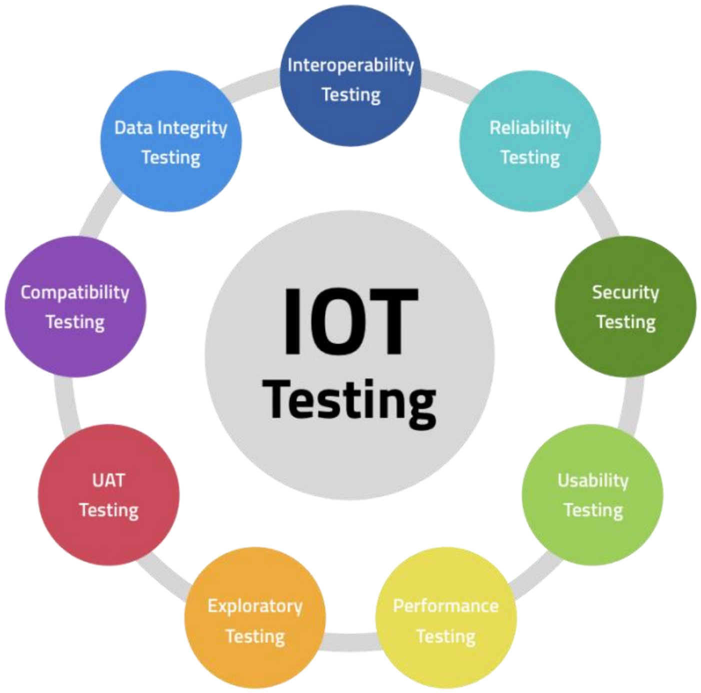


Figure 41: Digital transformation and feasibility study

- **Feasibility Study:** The feasibility study should consider the compatibility of the chosen IoT architecture with existing infrastructure, as well as the scalability and performance requirements. This phase is crucial to evaluate the practicality of the IoT solution and its potential to meet business objectives.

- **Design:** IoT architecture and frameworks play a critical role in shaping the design phase. Developers must choose the appropriate architecture, like centralized, decentralized, or hybrid, to meet scalability, data processing, and latency requirements. Selecting the right framework simplifies development by providing pre-built modules for IoT functionalities.

- **Implementation:** IoT development involves coding not only for software applications but also for embedded systems and hardware components. Common APIs enable seamless integration between software and hardware, making it easier for developers to interact with sensors, devices, and cloud platforms.



- **Testing:** IoT testing is multifaceted due to the interconnected nature of devices and software. Frameworks that support automated testing and simulators for virtual environments become indispensable. Extensive testing should be conducted to ensure security, interoperability, and performance across various IoT components.

3.2 Impact of Common IoT Architecture, Frameworks, Tools, Hardware, and APIs on IoT Security

IoT security is a critical aspect that demands attention throughout the development lifecycle. Common IoT architecture, frameworks, tools, hardware, and APIs can have both positive and negative effects on IoT security:

- **Standardization Benefits:** Common IoT architecture and frameworks that adhere to industry security standards can improve security by ensuring consistent security practices across the ecosystem. Following best practices and guidelines reduces vulnerabilities.
- **Security Frameworks and Tools:** Utilizing security frameworks and tools specifically designed for IoT can enhance security measures. These tools may offer features such as encryption, authentication, and secure bootstrapping for devices, making it harder for attackers to exploit vulnerabilities.
- **Hardware-based Security:** IoT hardware with built-in security features like Trusted Platform Modules (TPMs) and Hardware Security Modules (HSMs) can add an extra layer of protection to sensitive data and cryptographic operations.

Challenges in IoT Security:

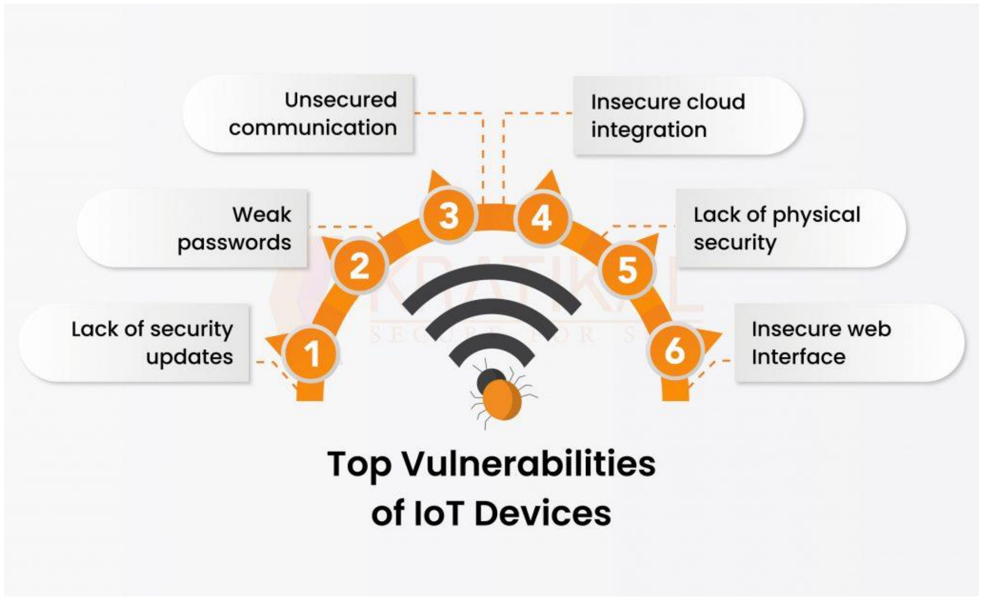
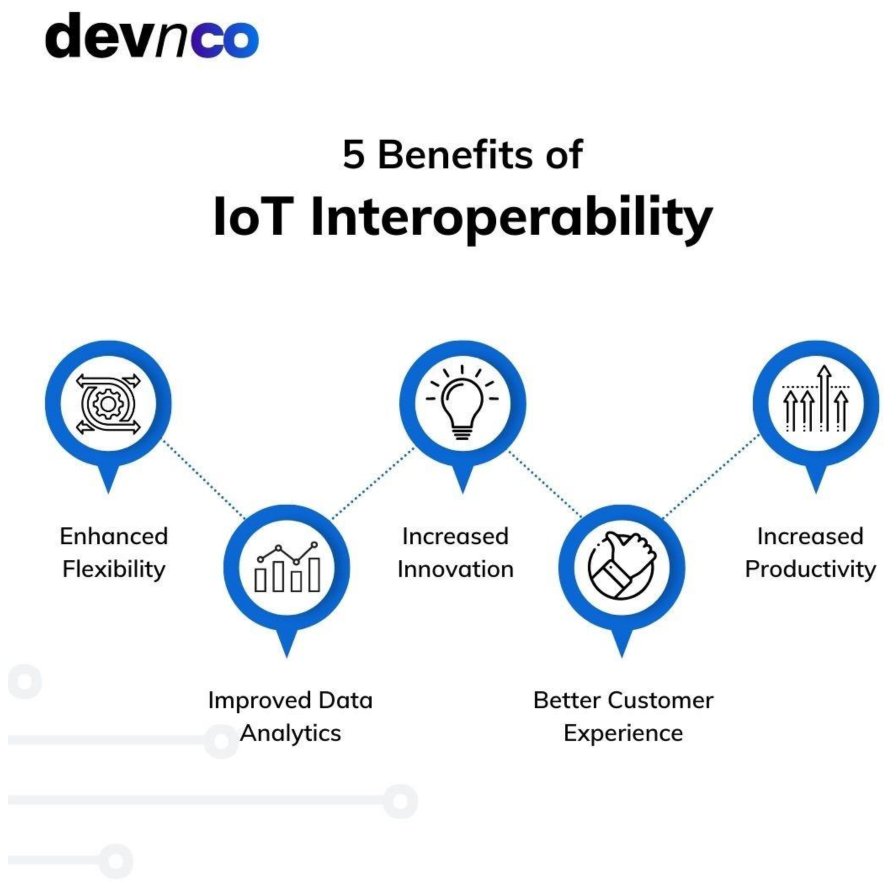


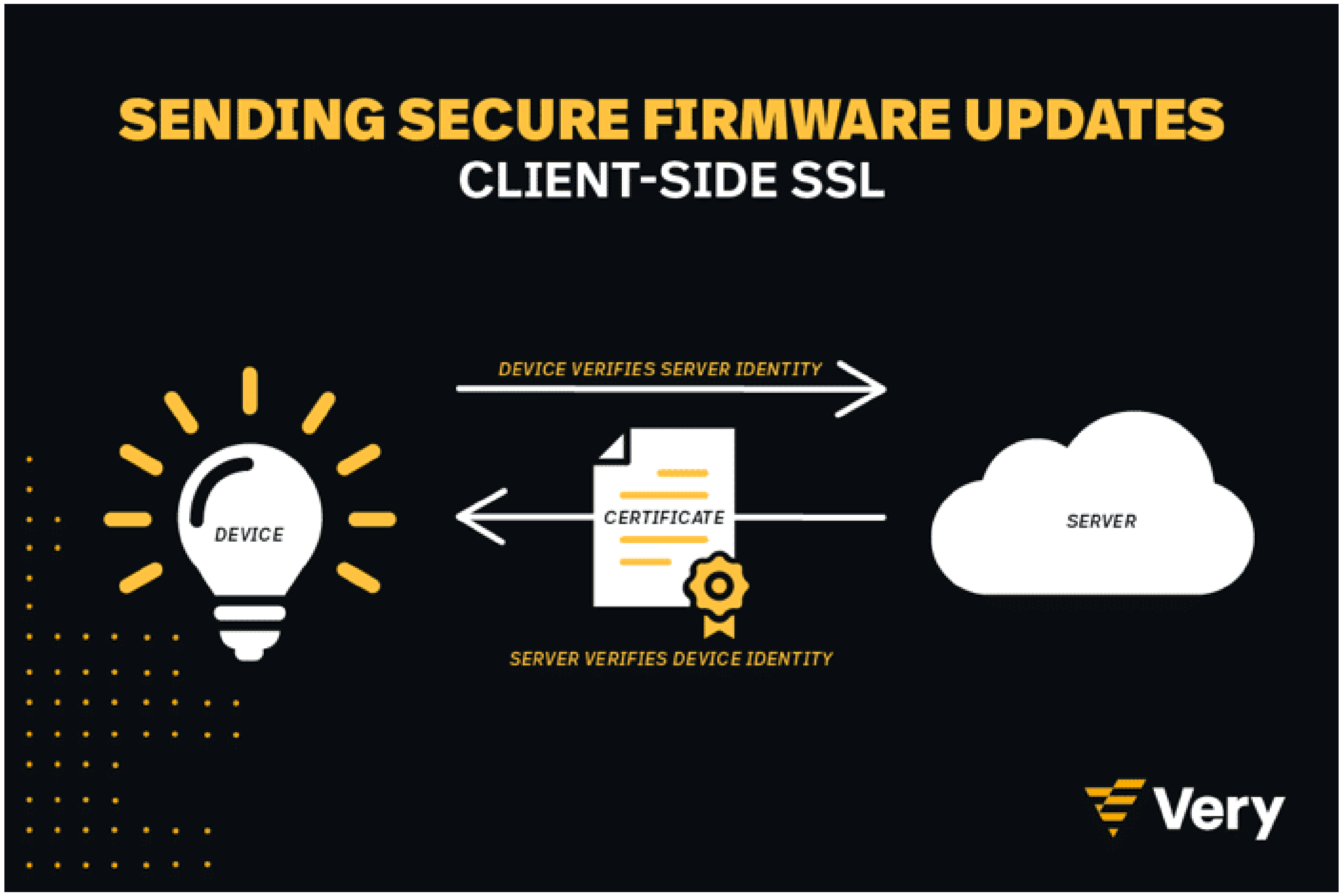
Figure 42: Top Vulnerabilities of IoT

- **Device Vulnerabilities:** Many IoT devices have limited computing power and memory, making it challenging to implement robust security measures. This limitation could lead to security vulnerabilities that attackers could exploit.

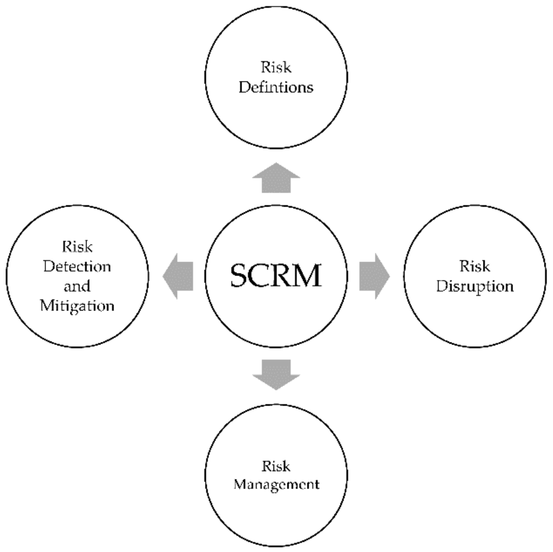


- **Interoperability:** With a wide variety of IoT devices and protocols, ensuring interoperability while maintaining security becomes a complex task. Incompatible security implementations may create loopholes for attackers.

- **Data Privacy:** IoT devices often collect and transmit sensitive user data. Ensuring the privacy of this data requires implementing strong data encryption and access control mechanisms.


SENDING SECURE FIRMWARE UPDATES
CLIENT-SIDE SSL
DEVICE VERIFIES SERVER IDENTITY
DEVICE
CERTIFICATE
SERVER
SERVER VERIFIES DEVICE IDENTITY
Very

- **Firmware Updates:** The ability to patch security vulnerabilities quickly and efficiently is crucial in IoT. However, updating firmware in resource-constrained devices can be problematic.

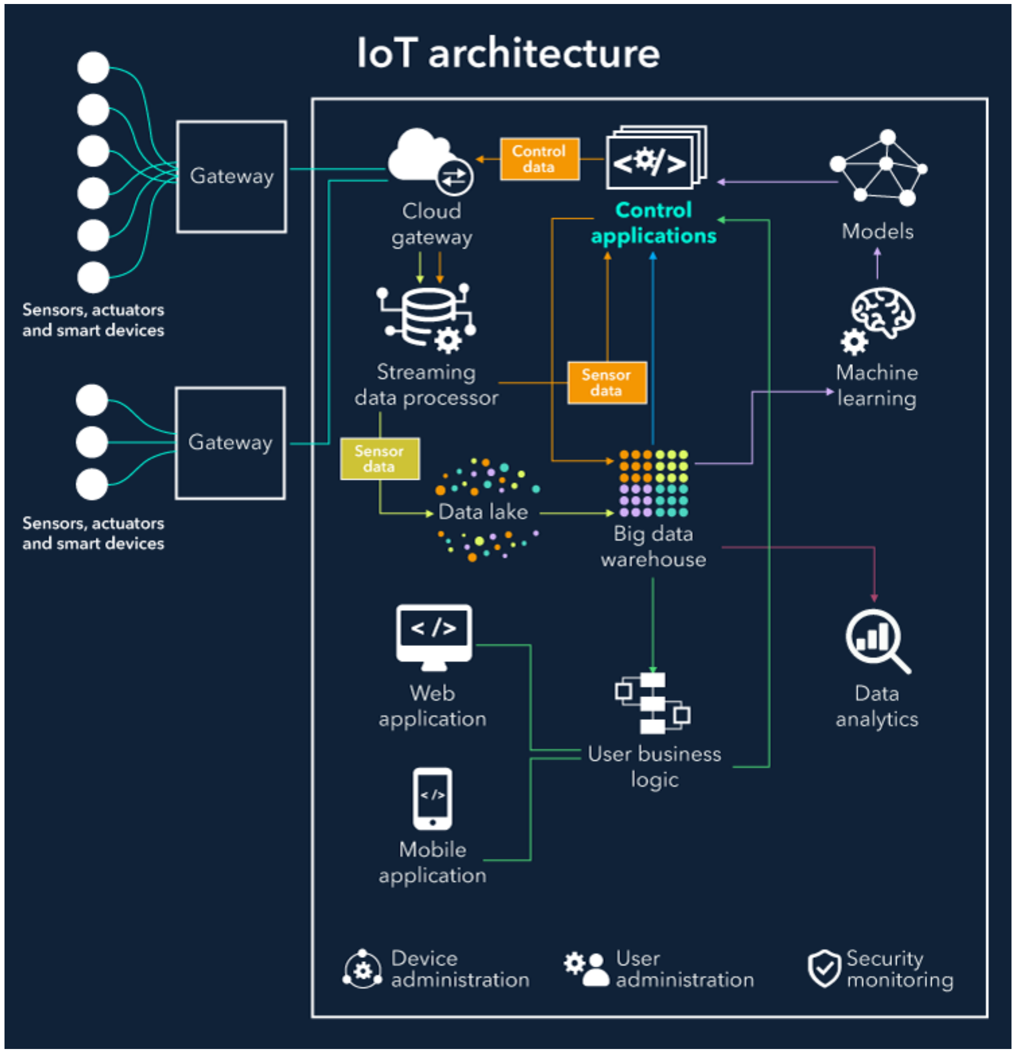


- **Supply Chain Risks:** The IoT ecosystem involves multiple vendors and suppliers, making it susceptible to supply chain attacks where malicious components could be introduced.

- **Lack of Security Awareness:** IoT development teams might not prioritize security, leading to poorly implemented security measures or overlooking potential threats.

In conclusion, while common IoT architecture, frameworks, tools, hardware, and APIs can streamline the software development lifecycle and offer security advantages, it's essential to be aware of the unique security challenges presented by the IoT landscape. Implementing robust security measures and staying up-to-date with industry best practices are critical to safeguarding IoT systems from potential threats.

4. Evaluate specific forms of IoT architecture and justify their usage when designing software applications.



When designing software applications for IoT, various forms of IoT architecture can be considered based on the specific requirements and constraints of the application. Each architecture has its strengths and weaknesses, and choosing the right one is crucial for the successful implementation of the IoT system. Below are specific forms of IoT architecture and their justifications for usage:

A. Centralized IoT Architecture:

In a centralized IoT architecture, all data processing, analysis, and decision-making occur on a centralized server or cloud platform. IoT devices act as data collectors and transmit data to the server for processing. This architecture is suitable for applications where data analysis and decision-making require significant computational power, storage, and real-time processing capabilities.

Justification:

- Centralized processing allows for easier scalability and maintenance as all updates and changes can be made at the central server.
- It is suitable for applications with large-scale data processing requirements, such as industrial monitoring and control systems.

B, Decentralized IoT Architecture:

In a decentralized IoT architecture, data processing and analysis are distributed across multiple edge devices and gateways. Each device can perform local data processing and make autonomous decisions based on predefined rules. Data aggregation may occur at higher-level gateways before transmitting it to the cloud for further analysis.

Justification:

- Decentralized architecture reduces reliance on the central server and minimizes latency by enabling real-time data analysis at the edge.
- It is suitable for applications with low-latency requirements, such as real-time monitoring and response systems.

C, Hybrid IoT Architecture:

A hybrid IoT architecture combines elements of both centralized and decentralized approaches. It strikes a balance between local data processing at the edge and cloud-based processing for more complex analytics. Some decisions are made locally, while others are passed to the cloud for further analysis.

Justification:

- Hybrid architectures are versatile and can adapt to varying data processing needs, optimizing resource utilization and performance.
- They are suitable for applications with dynamic data processing requirements, such as smart cities, where different IoT devices have varying computing capabilities.

D, Peer-to-Peer (P2P) IoT Architecture:

In a P2P IoT architecture, IoT devices directly communicate with each other without the need for a central server. They share data and make decisions collaboratively, reducing reliance on a centralized infrastructure.

Justification:

- P2P architecture offers increased resilience and fault tolerance as it eliminates single points of failure.
- It is suitable for applications in remote or disconnected environments, where internet connectivity is limited or unreliable.

E, Hierarchical IoT Architecture:

Hierarchical IoT architecture divides the IoT system into multiple layers, each responsible for specific functions, data processing, and communication. Data flows hierarchically from the edge devices to the higher-level gateways or cloud platforms.

Justification:

- Hierarchical architecture simplifies system management and organization, making it easier to monitor and maintain various layers independently.
- It is suitable for applications with complex data processing requirements, such as smart grid systems.

F, Event-Driven IoT Architecture:

In an event-driven IoT architecture, actions are triggered based on specific events or conditions. IoT devices monitor events and react accordingly, reducing continuous data transmission and processing.

Justification:

- Event-driven architecture is power-efficient as IoT devices can remain in a low-power state until an event occurs, reducing energy consumption.

- It is suitable for applications with intermittent data generation, such as home automation systems.

In conclusion, the choice of IoT architecture depends on the specific requirements, scalability, real-time processing needs, and resource constraints of the software application. By evaluating the advantages and justifications of each architecture, developers can make informed decisions to design effective and efficient IoT applications.

TASK 2 – PLAN AN APPROPRIATE IOT APPLICATION

1. Investigate architecture, frameworks, tools, hardware and API techniques available to develop IoT applications. (P3)

1.1 Standard architectures:

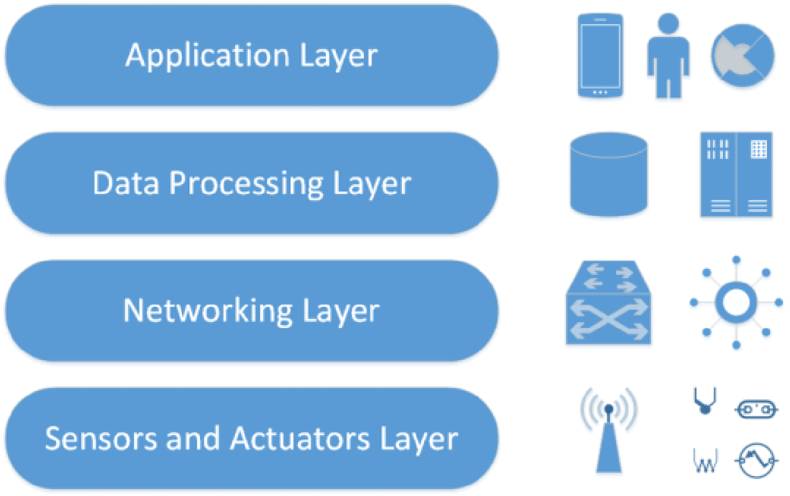


Figure 43: Standard architectures

<u>A, Application Layer:</u>

The application layer defines all applications in which IoT has deployed. It is the interface between the end IoT devices and the network. IoT Applications such as smart homes, smart health, smart cities, etc. It has the authority to provide services to the applications. The services may be different for each application because of services based on the information collected by sensors.

It is applied through a dedicated application at the device end. Such as for a computer, the application layer is applied by the browser. It is the browser that executes application layer protocols like HTTP, HTTPS, SMTP, and FTP. There are many concerns in the application layer out of which security is the key issue.

Common issues and threats of application layers are:

Cross-site scripting:

- It is a type of computer security infirmities that typically found in web applications. It enables attackers to inject client-side scripts such as JavaScript into web pages viewed by other users. By doing so, an attacker can completely change the contents of the application as per his needs and use original information in an illegal way.

- The first method you can use to prevent cross-site scripting from appearing in your applications is by escaping user input. Escaping input means taking the data from an application has received and ensured it's secure before supplying it for the end-user.
- Validating input is another process of ensuring an application is providing the correct data and protecting harmful data from doing harm to the site, database, and users.

Malicious Code Attack:

- It is a particular code in any part of the software system or script that is considered to cause undesired effects, security threats or damage to the system. It is that threat that may not be blocked or controlled by the use of anti-virus software.
- Static Code Analysis (SCA) is an effective method to protect the malicious code from successfully causing any damage to computers. SCA is a debugging method of a computer program that is done by analyzing the code without performing the program. The process supplies an understanding of the code structure and can help to ensure that the code should match with industry standards.
- Today's superior scanners can easily detect malicious code such as a Data Leakage, Time Bombs, Anti Debugging techniques, backdoor Threats, etc.

B, Data Processing Layer:

In three-layer architecture, the data were directly sent to the network layer. Due to sending data directly the chances of getting damages increase. In four-layer architecture, data is sent to this layer that is obtained from a perception layer. Data Processing Layer has two responsibilities it confirms that data is forwarded by the authentic users and prevented from threats.

Authentication is the most commonly used method to verify the users and the data. It is applied by using pre-shared, keys and passwords to the concerned user. The second responsibility of the layer is to send information to the network layer. The medium through which data is transferred from the Data Processing Layer to the network layer can be wireless and wire-based.

Common issues and threats of the Data Processing layer are:

DoS Attack:

- An attacker sends a huge amount of data to make network traffic overloaded. Thus, the huge consumption of system resources exhausts the IoT and makes the user unable to access the system.

- Deploy an antivirus program and firewall will restrict the bandwidth usage to authenticated users only. Server Configuration is another method that can help reduce the probability of being attacked.

Malicious Insider Attack:

- It comes from the inside of an IoT environment to access private information. It is conducted by an authorized user to access the information of another user.
- The practices such as an encryption of data, implementing proper password management practices, installing antivirus will helps to keep data safe from such threats.

C, Network Layer:

This layer is also known as a transmission layer. It acts like a bridge that carries and transmits data gathered from physical objects through sensors. The medium can be wireless or wire-based. It also connects the network devices and networks to each other. Hence, it is extremely sensitive to attacks from the attackers. It has important security issues regarding integrity and authentication of data that is being transmitted to the network.

Common issues and threats of the Network layer are:

Main-in-The-Middle Attack:

- MiTM attack is an attack where the attacker privately intercepts and modifies the communication between the sender and receiver who assume they are directly communicating with each other. It leads to a serious threat to online security because they give the attacker the pathway to capture and control data in real-time
- Secure/Multipurpose Internet Mail Extensions encrypts emails which ensures that only intended users can read will prevent data from MITM Attacks.

Storage Attack:

- The crucial information of users is saved on storage devices or on the cloud. Both the storage devices and the cloud can be attacked by the attacker and the user’s information may be modified to incorrect details.
- By making regular backups of files, by running anti-virus software and using a system with strong passwords so that data access is restricted are the ways by which we can protect data from the attacker.

Exploit Attack:

- An exploit is any unethical or illegal attack in a form of software, blocks of data or a sequence of commands. It takes benefit of security infirmities in an application, system or hardware. It usually occurs with the goal of getting control of the system and steals information stored on the network. By installing all software patches, security releases and all updates for your software are few preventive measures against attack.

D. Perception layer/Sensor layer:

The sensor layer has the responsibility to recognize things and gather the data from them. There are many types of sensors connected to the objects to gather information such as RFID, sensors and 2-D barcode. The sensors are selected as per the requirement of applications. The data that is collected by these sensors can be about location, changes in the air, environment, etc. However, they are the main aim of attackers who wish to use them to replace the sensor with their own.

Hence, most of the threats are related to sensors are

Eavesdropping:

- It is an unauthorized real-time attack where personal communications, such as phone calls, fax transmissions, text messages are intercepted by an attacker. It tries to take crucial information that is transferred over a network. Preventive measures such as Access control, continuous supervision/observation of all devices, thorough inspection by a qualified technical countermeasures specialist of all components need to be ensured.

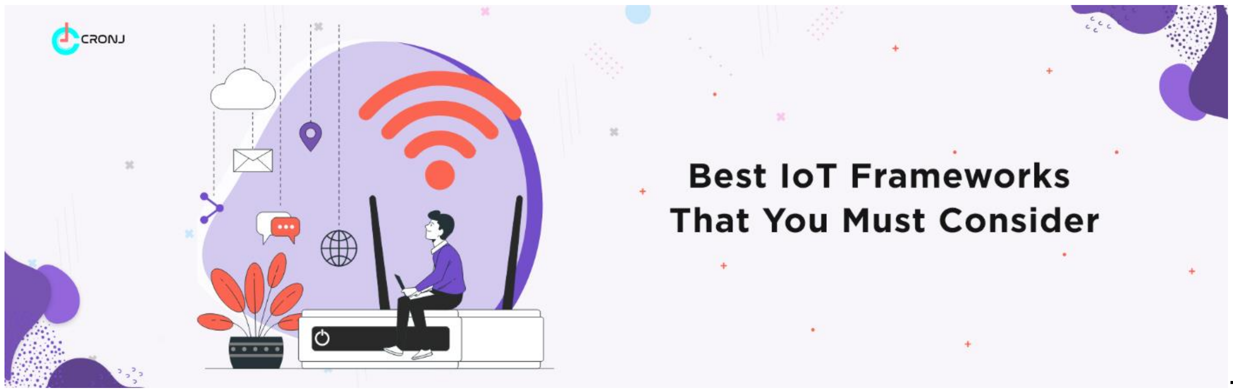
Replay Attack:

- It is also known as a playback attack. It is an attack in which an attacker intrudes on the conversation between the sender and receiver and extracts authentic information from the sender. The added risk of replay attacks is that a hacker doesn’t even need improved skills to decrypt a message after seizing it from the network.
- This attack can be prevented by using Timestamps on all messages. This protects hackers from resending messages sent longer ago than a particular length of time. Another preventive measure to avoid becoming a victim is to set a password for each transaction that’s only used once and discarded.

Timing Attack:

- It is usually utilized in devices that have weak computing abilities. It allows an attacker to find vulnerabilities and withdraw secrets maintained in the security of a system by observing how long it takes the system to respond to various queries, input or other algorithms.
- To prevent this attack we can simply make a constant-time comparison either by using timer functions. You should have tests that make sure that compiler optimizations don't place timing infirmities back into your code.
- In this way, IoT 4 layer architecture can fulfill the requirements of IoT regarding security and privacy. We have described different research about layered architectures of IoT and also outline the security attacks based on the layers that can affect the performance of IoT.
- hIOTron shares practical approaches and tactics that will readily useful in the industry. hIOTron's IoT Course helps weigh in many factors that come handy in building successful IoT products.

1.2 Framework:



-

Today, the Internet has become a major part of the world. It finds application in every corner of the globe. With the increasing use of the Internet, more appliances and devices are now being connected to the web, forming the 'Internet of Things,' commonly known by its acronym 'IoT'. The IoT is not just a simple technology. It is a complex framework that includes several different technologies, designed to work together in tandem. In this article, you will learn what the IoT framework is in detail, as well as the Open-source IoT frameworks.

In typical scenarios where large data is generated and transmitted across multiple devices, there is a crucial focal point where all the data is gathered and consolidated. This specific point plays a vital role in the network as it facilitates the combination of all the data, enabling a comprehensive understanding of the generated information.

However, achieving seamless data transmission and generation is not a coincidence; rather, it is made possible through the implementation of the Internet of Things Framework (IoT framework). But what exactly is the IoT framework?

The Internet of Things (IoT) Framework can be described as an interconnected ecosystem comprising various devices that communicate with one another over the Internet. These connected devices are designed to exchange and sense data autonomously, requiring minimal human intervention.

The IoT framework plays a crucial role in enabling smooth communication among these connected devices over the Internet. It is aptly named the 'Internet of Things' framework because it facilitates the interaction of 'Things' (devices) over the Internet.

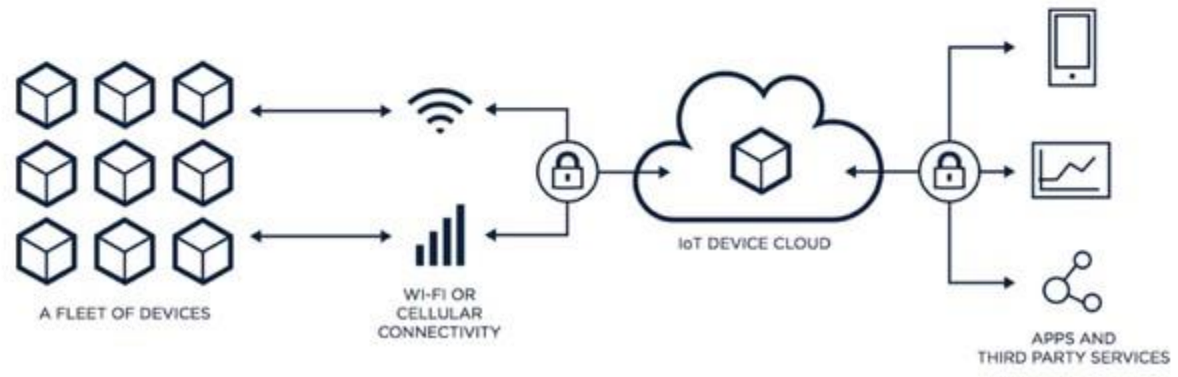


Figure 44: IoT Framework Overview

The IoT framework holds significant importance in today's technology-driven world, with applications spanning across almost every industry. One prominent area where it is extensively utilized is in the creation of smart homes.

Furthermore, the concept of the IoT framework is extended to the development of various physical objects, including thermostats, electrical appliances, security and alarm systems, and even vending machines, among numerous other objects.

Open Source IoT Frameworks:

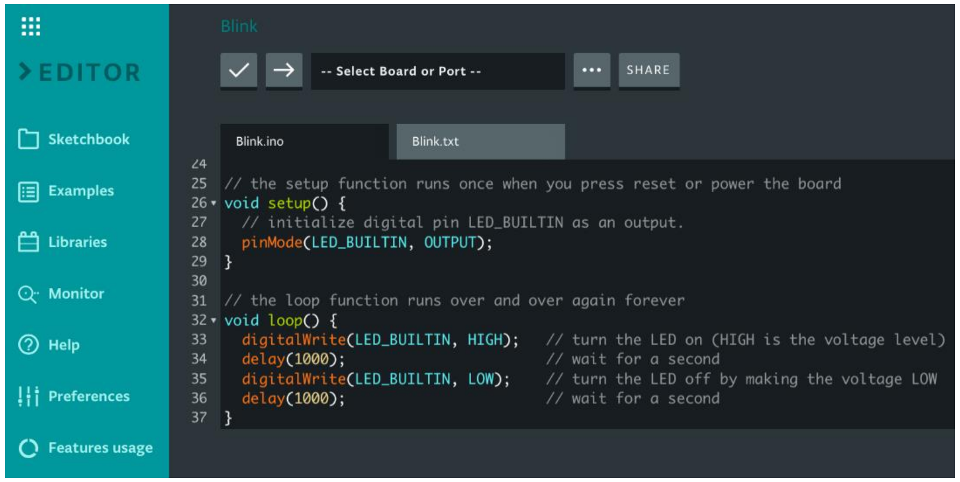
To grasp the concept of an IoT framework open source, consider the following three key points:

- **Consumer Freedom:** Every consumer desires the freedom to use technology devices of their choice without being restricted to products from a single vendor. For example, some smartwatches only work when paired with smartphones from the same brand.
- **Easy Integration for Dealers:** IoT device vendors want a seamless integration of their products with various technology ecosystems, enabling compatibility with a wide range of devices.
- **Application Development Flexibility:** Application developers wish to support multiple devices without having to write specific code for each vendor's products.

An Open source framework presents a solution to address these challenges. It enables scalability and high flexibility levels, granting consumers the freedom to choose, vendors smoother integration opportunities, and developers the ability to create applications without vendor-specific codes.

The beauty of IoT framework open sources lies in their accessibility, as they are generally free to download, easy to install, and simple to launch, facilitating widespread adoption and usage.

Example:



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

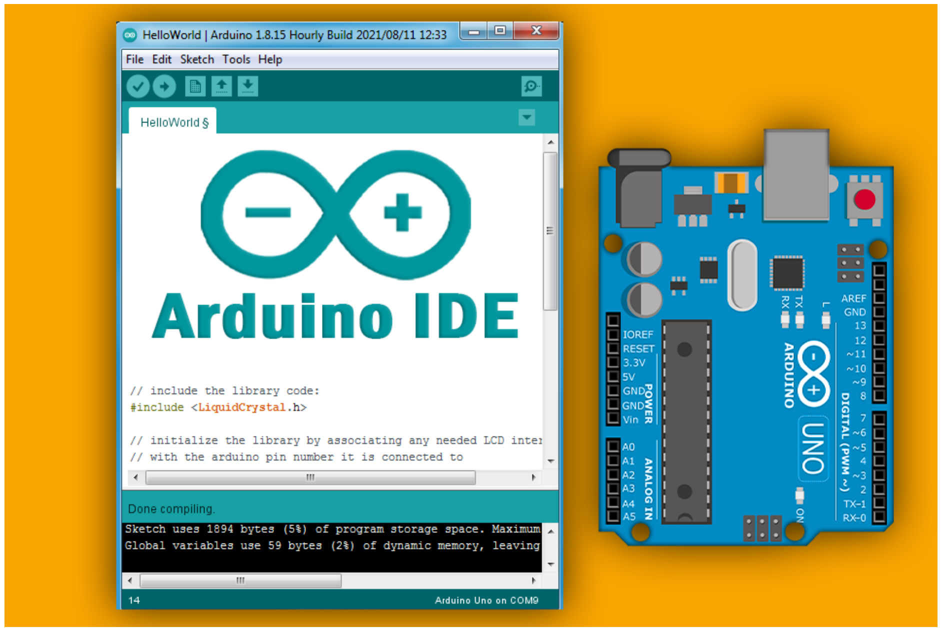
Figure 45: Arduino Framework

Arduino stands out as one of the top recommended open-source IoT frameworks, particularly for individuals looking to create a computer system capable of sensing and exerting control over the surrounding environment.

Arduino's open-source nature encompasses both hardware and software components, making it exceptionally user-friendly and accessible. Its simplicity makes it an ideal choice for various IoT projects.

The functioning of the Arduino open-source framework relies on several hardware specifications, often employed in interactive electronics, enabling seamless operation and interaction with the physical world.

1.3 Tools:



```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD inter
// with the arduino pin number it is connected to
```

```
Sketch uses 1894 bytes (5%) of program storage space. Maximum
Global variables use 59 bytes (2%) of dynamic memory, leaving
```

Figure 46: Arduino Tools

Arduino is an advanced and versatile IoT tool designed to connect various equipment from different frameworks seamlessly. Equipped with a wide array of libraries and models to aid IoT app developers, Arduino offers the ideal combination of features. For those seeking to create a computer system capable of superior control and sensing abilities in the physical world, Arduino is the perfect choice, surpassing typical stand-alone computing machines.

As an open-source prototyping platform, Arduino offers a straightforward and user-friendly IoT solution. It provides an Integrated Development Environment and a dedicated programming language. By operating with a set of hardware specifications applicable to numerous interactive electronic devices, Arduino ensures a seamless integration of IoT software and hardware. It comes with pre-equipped, cost-effective devices that effortlessly interact with the environment, making it a practical and accessible choice for IoT projects.

1.4 Hardware

There is a wide range of hardware options available for IoT development, catering to various application requirements and use cases. Here are some popular hardware components commonly used in IoT projects:

Microcontrollers and Single-Board Computers (SBCs):

- **ESP32:** These low-cost and power-efficient Wi-Fi enabled microcontrollers are highly popular in IoT projects for their built-in Wi-Fi capabilities and GPIO pins for sensor connections.

Sensors:

- **Temperature and Humidity Sensors:** These sensors measure ambient temperature and humidity levels, essential for climate control and environmental monitoring applications.
- **Motion Sensors:** Motion detectors, like PIR (Passive Infrared) sensors or accelerometers, detect movement and are commonly used in security and automation systems.

Communication Modules:

- **Wi-Fi Modules:** Wi-Fi modules enable IoT devices to connect to Wi-Fi networks, providing high-speed data transmission and internet connectivity.
- **Bluetooth Modules:** Bluetooth modules are used for short-range wireless communication between IoT devices and smartphones or other Bluetooth-enabled devices.
- **Servo Motors:** Servo motors are used for precise control and movement in applications like robotics and home automation.

Displays:

- **OLED Displays:** OLED displays are compact and power-efficient, making them suitable for small IoT devices requiring visual output.
- **LCD Displays:** LCD displays provide larger screen sizes and are commonly used in IoT devices that need more extensive user interfaces.

Power Sources:

- **Batteries:** Batteries are widely used to power portable and wireless IoT devices, offering flexibility and ease of use.
- **Solar Panels:** Solar panels can be integrated with IoT devices to provide sustainable and renewable power, especially for remote and outdoor applications.

These are just some of the many hardware components available for IoT development. Developers can choose the appropriate hardware based on factors like power requirements, connectivity, processing capabilities, and the specific needs of their IoT projects.

1.5 API

When it comes to IoT security, several APIs (Application Programming Interfaces) and frameworks are available to help developers implement robust security measures in their IoT applications. These APIs address various aspects of IoT security, including authentication, encryption, access control, and secure communication. Here are some commonly used IoT security APIs:

TLS (Transport Layer Security) API:

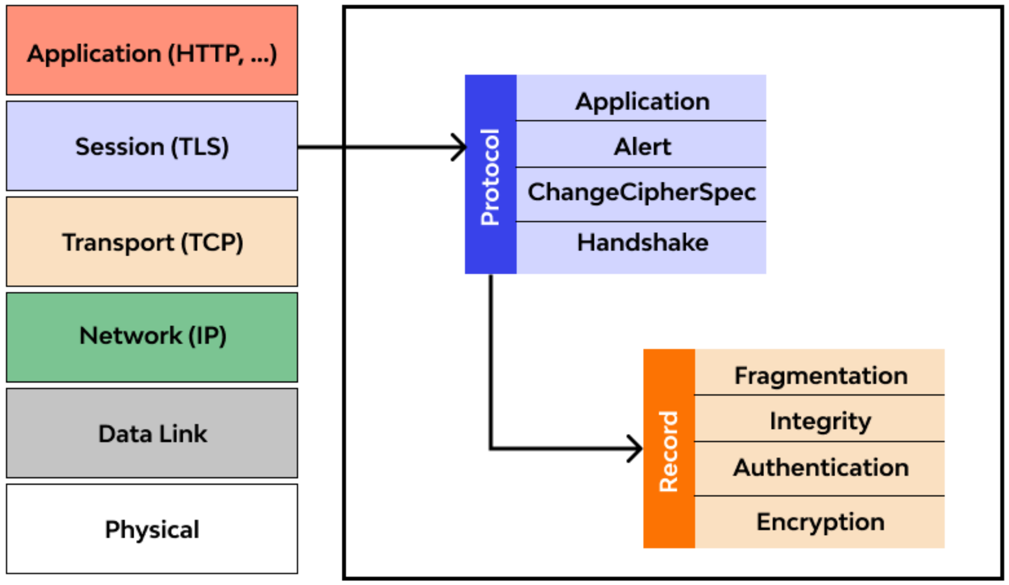
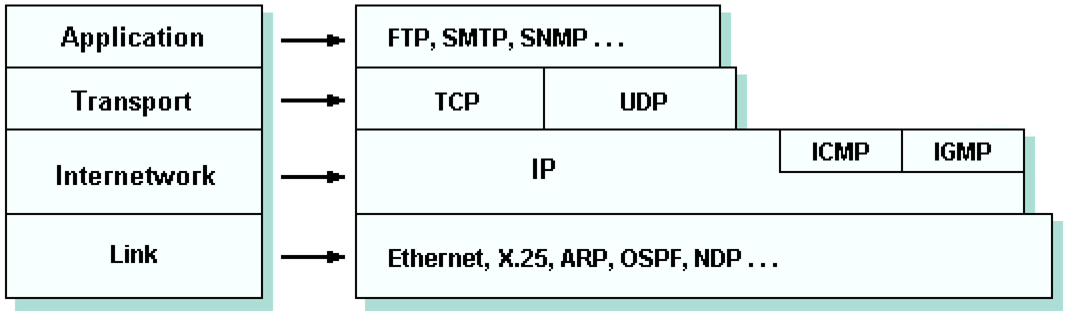


Figure 47: Tls API

- TLS is a cryptographic protocol that provides secure communication over a network. It ensures data privacy and integrity during transmission between IoT devices and servers. Implementing TLS in IoT applications helps prevent eavesdropping and man-in-the-middle attacks.

Understanding Transmission Control Protocol (TCP)



The Transmission Control Protocol (TCP) provides a good background for understanding how Transport Layer Security (TLS) works. TCP is a layer of abstraction of a reliable network running over an unreliable channel. Host-to-host routing and addressing are provided by IP (Internet Protocol).

For network communication, the TCP/IP protocol suite presents a four-layered model. Using well-defined interfaces, each layer communicates with the others and has its own responsibilities.

A three-way handshake bootstraps any TCP connection. Clients initiate the TCP three-way handshake by sending a TCP packet to the server. In this case, the packet is called the SYN packet. The SYN flag is set in the TCP packet. As part of the SYN packet, the client selects a random sequence number, the source port number, the destination port number, the TCP Segment Len field, and other fields. The TCP Segment Len field indicates how much application data this packet carries. The TCP Segment Len field will be zero for all messages sent during the TCP three-way handshake since no exchange has yet begun.

When the server receives the initiate message from the client, it too picks a random sequence number and passes it back to the client. The packet is known as the SYN-ACK packet. In order to achieve error control and ordered delivery, TCP packets must be uniquely identified. Clients and servers exchange sequence numbers to keep that promise. By numbering the packets, both sides of the communication channel will know which packets get lost during transmission and which packets are duplicates, and how to order a random set of packets.

Upon receiving the SYN-ACK packet from the server, the client sends a TCP packet to acknowledge the handshake. A packet such as this is called the ACK packet. This includes the source port, destination port, the initial client sequence number + 1 as the new sequence number, and the acknowledgment number. The acknowledgment number is calculated by adding 1 to the sequence number in the SYN-ACK packet. The TCP Segment Len field is zero because we are still in the three-way handshake.

2. Determine a specific problem to solve using IoT. (P4)

A, Scenario:

At TTP Company, there was a growing concern about the reliability of the traditional manual employee attendance system. The management sought to implement a more efficient and secure solution that not only tracks employees' time of entry and exit but also enhances the overall security of the company. To address these issues, the company decided to introduce a Biometric Employee Attendance and Access Control System that integrates fingerprint scanning technology with door access control.

B, Solution:

To address these challenges and enhance the security measures at the company, the management decides to implement a cutting-edge solution: integrating fingerprint-based door access with automated attendance tracking.

Proposed Solution: Fingerprint Door Access with Integrated Attendance Tracking

C, Standard Architectures

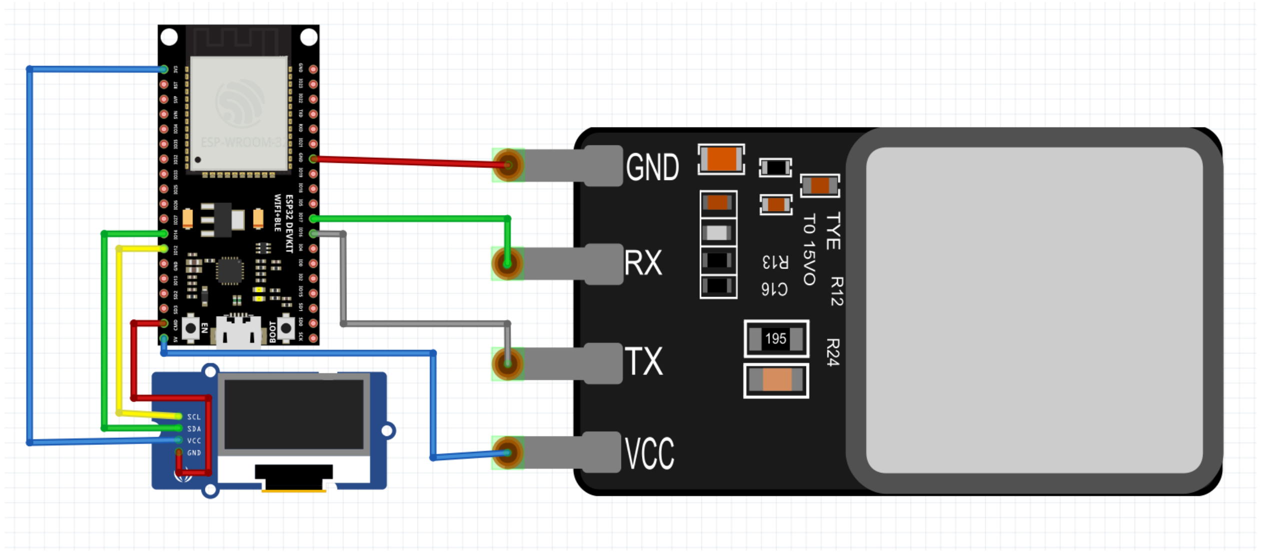


Figure 48: Architectures of my Project

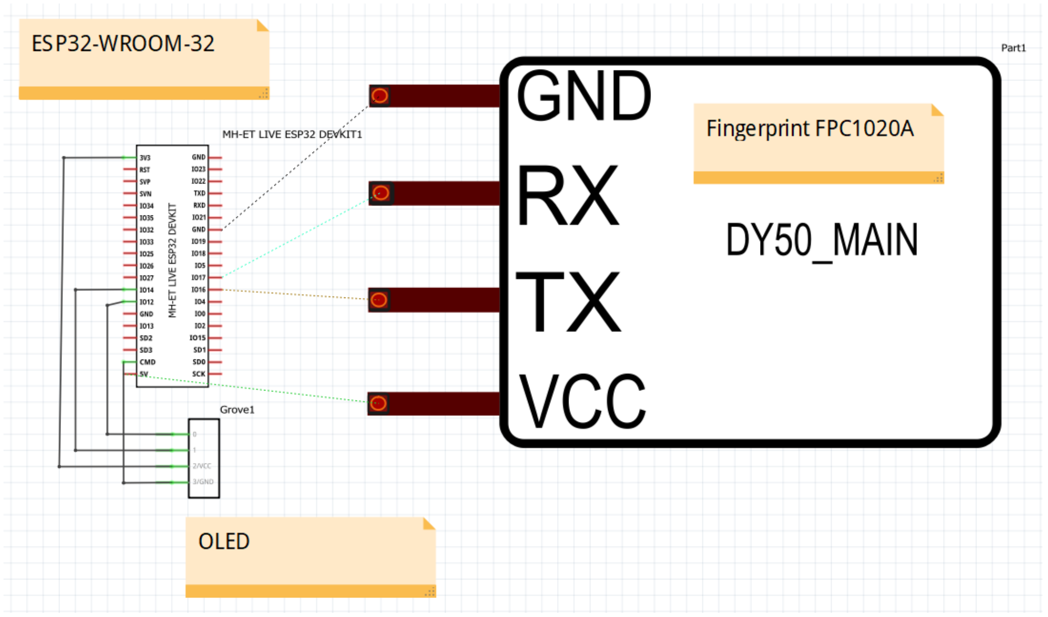
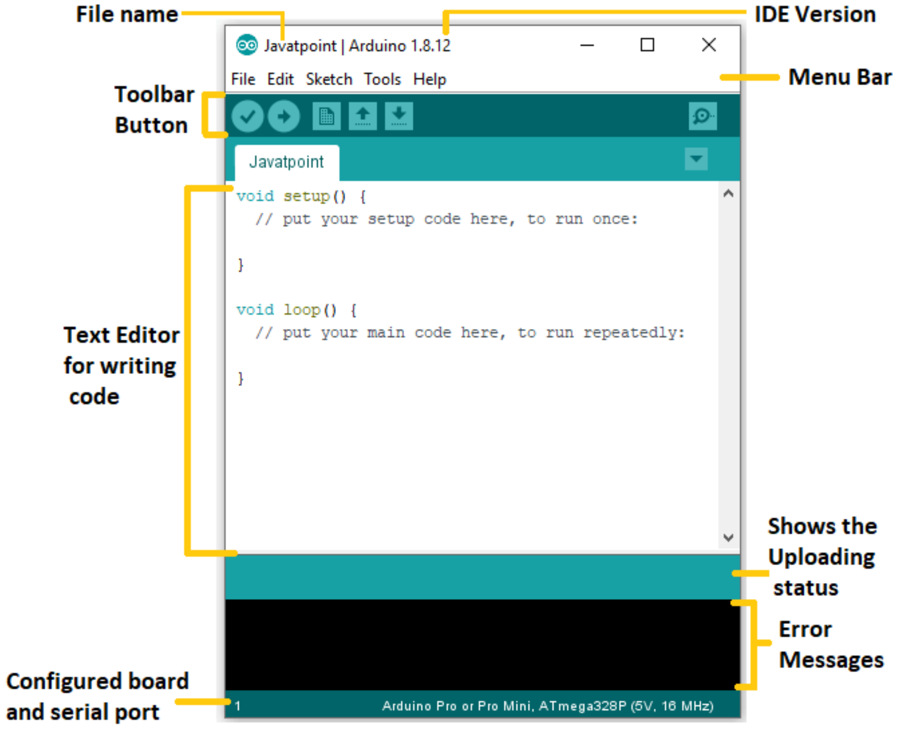


Figure 49: electrical circuit diagram

The biometric door access system follows a standard client-server architecture. The ESP32 acts as the client device responsible for interfacing with the fingerprint sensor and OLED display, while serving as the gateway to the server, which controls the door lock via the servo motor. The server could be a dedicated machine or a cloud-based platform that handles user authentication and access control.

D, Framework:


File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 50: Arduino IDE Platform

The system is built on the **Arduino IDE platform**, utilizing the Arduino framework for programming the ESP32 microcontroller. The Arduino framework simplifies the development process and provides a rich set of libraries and functions, making it ideal for prototyping and quick project iterations.

E, Tools:

Arduino IDE: The primary integrated development environment (IDE) used for writing, compiling, and uploading code to the ESP32 microcontroller.

Git/GitHub: Version control system and repository for collaborative development and project management.

FPC1020A Library: A custom library for interfacing with the FPC1020A fingerprint sensor to capture and process fingerprint data.

OLED Display Library: A library used to control the small OLED display for visual feedback to the user.

Servo Library: A library to control the servo motor for opening and closing the door lock.

F, Hardware:

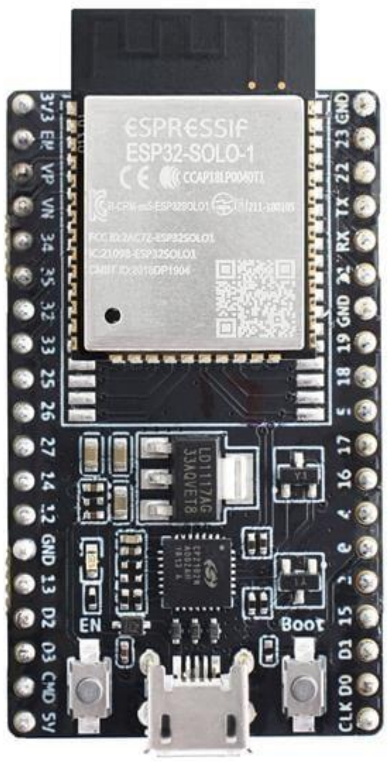


Figure 51: ESP32-WROOM-32

ESP32 Development Board: The main control unit responsible for running the program and managing communication with other components.

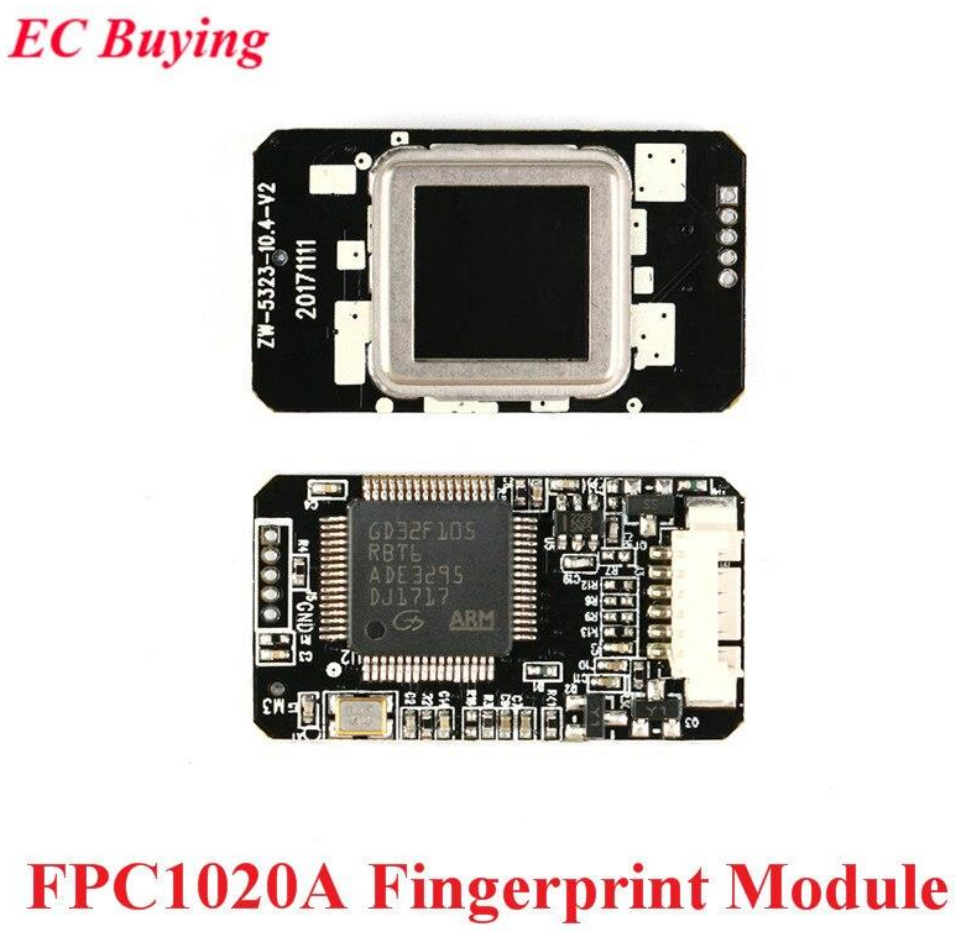


Figure 52: FPC1020A Fingerprint Sensor

FPC1020A Fingerprint Sensor: A capacitive fingerprint sensor that captures and processes fingerprint data for identification.

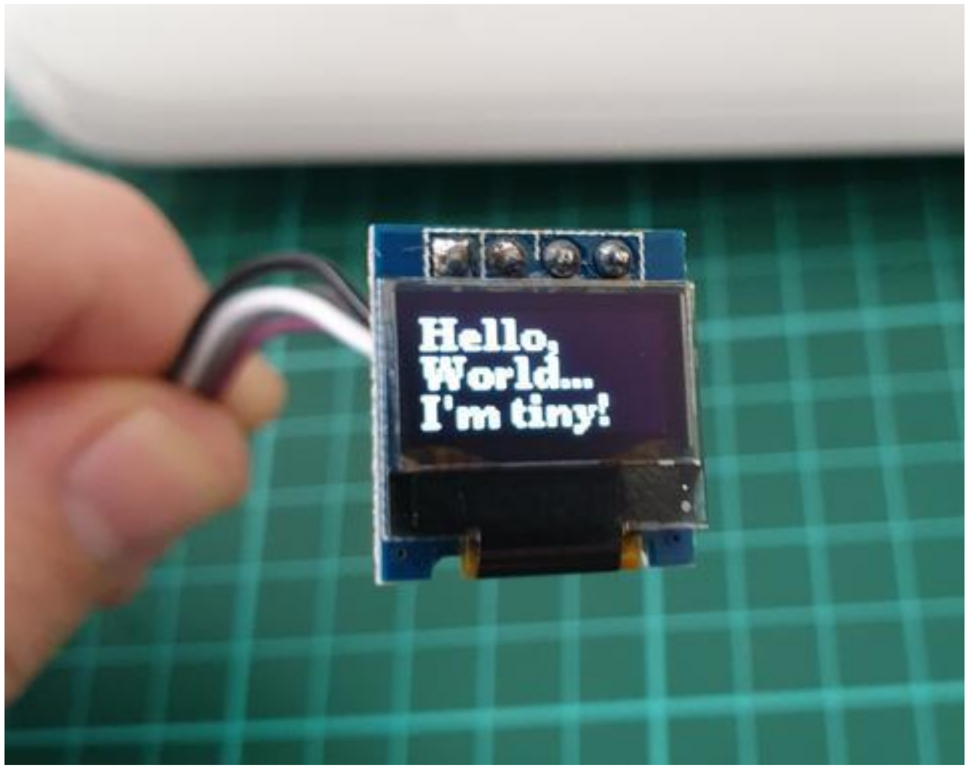


Figure 53: Small OLED Display

Small OLED Display: A visual feedback interface to display relevant information, such as success/failure of fingerprint authentication.

Servo Motor: A mechanical actuator that controls the physical door lock, granting or denying access based on authentication results.

G, API:

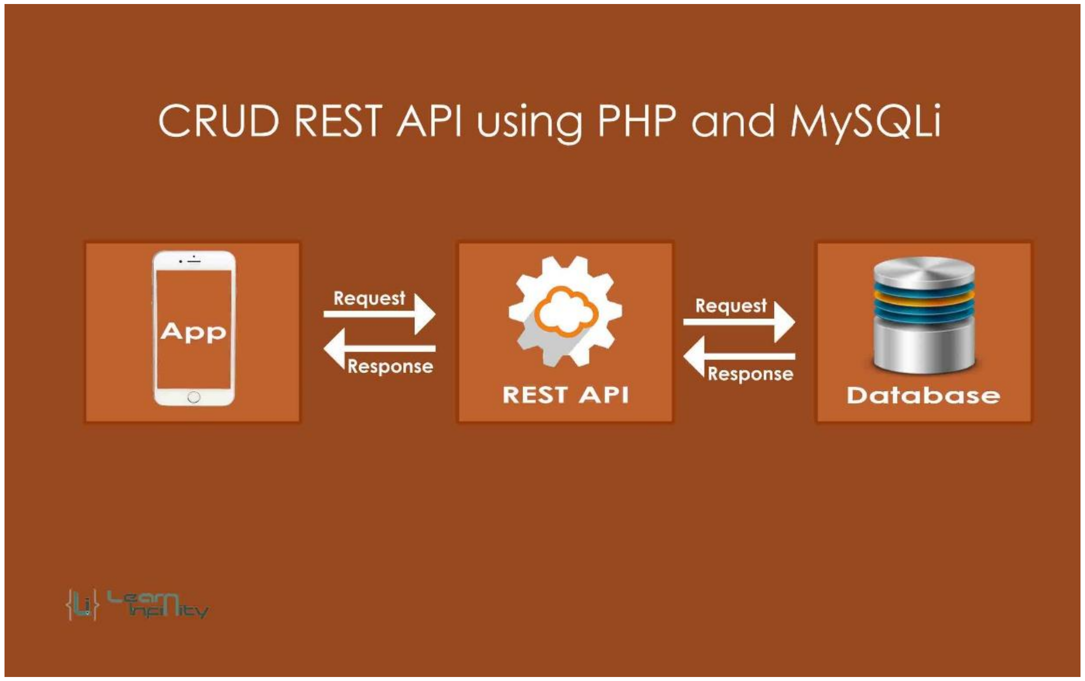


Figure 54: PHP and MySQL API

The PHP API acts as a bridge between the ESP32 client and the MySQL database. It provides endpoints for user registration, fingerprint enrollment, fingerprint verification, and access log recording. The API processes incoming requests from the ESP32, performs necessary database operations, and sends back appropriate responses to the client for feedback.

H, Conclusion:

The development of a biometric door access system using an ESP32 microcontroller, FPC1020A fingerprint sensor, OLED display, and servo motor offers an efficient and secure solution for companies seeking enhanced security and automated attendance tracking. The utilization of the Arduino IDE and libraries simplifies the implementation process, while the custom APIs enable seamless communication between the hardware components and the ESP32. The integration of biometric authentication and door access control will result in improved security measures and streamlined employee attendance tracking.

3. Select the most appropriate IoT architecture, frameworks, tools, hardware and API techniques to include in an application to solve this problem.

To develop an IoT application for the biometric door access system, we need to carefully select the most appropriate IoT architecture, frameworks, tools, hardware, and API techniques. Here's a detailed discussion of each component:

A, IoT Architecture:

- For the biometric door access system, a suitable IoT architecture would be the client-server architecture. The ESP32 acts as the client, responsible for capturing fingerprint data and communicating with the server. The server, built using PHP and MySQL, handles user authentication, stores fingerprint templates, and controls the door lock. This architecture is efficient, scalable, and ensures centralized control and management.

Advantages:

- **Scalability:** The client-server architecture allows easy scalability by adding more ESP32 devices for additional doors or expanding the system to cover more users.
- **Centralized Management:** All data and access control logic are managed centrally, simplifying administration and updates.

Disadvantages:

- **Single Point of Failure:** If the server experiences downtime or becomes inaccessible, the entire system may be affected.

B, Framework:

Using a PHP framework like Laravel or CodeIgniter is beneficial for structured development, built-in security features, and standardized practices. It helps in modularizing code, enforcing code reusability, and implementing authentication mechanisms.

Advantages:

- **Organized Development:** A PHP framework provides a structured approach to development, making it easier to manage and maintain the codebase.
- **Security:** PHP frameworks come with built-in security features and protection against common vulnerabilities.

C, Tools:

- **Arduino IDE:** Necessary for programming the ESP32 and handling communication with hardware components.
- **Postman:** Useful for testing and debugging PHP API endpoints during development.

D, Hardware:

- **ESP32 Development Board:** The ESP32 is cost-effective, offers built-in Wi-Fi capabilities, and has sufficient processing power for the task.
- **FPC1020A Fingerprint Sensor:** A reliable and widely used fingerprint sensor with advanced capabilities for accurate identification.

E, API Techniques:

Develop a RESTful API to facilitate communication between the ESP32 client and the server. REST APIs are lightweight, flexible, and can handle HTTP requests effectively. JSON can be used as the data format for communication.

Advantages:

- **Lightweight:** RESTful APIs are lightweight and suitable for resource-constrained IoT devices like ESP32.
- **Platform-Agnostic:** Since it uses standard HTTP methods, it can be accessed from any platform or programming language.

F, Feasibility:

- **Economic Feasibility:**The proposed solution is economically feasible due to the availability of cost-effective hardware components like the ESP32 and FPC1020A. Developing the application using open-source tools and frameworks further reduces development costs.
- **Technical Feasibility:** The technical feasibility is high as the chosen components and technologies are readily available and have proven compatibility with each other. The Arduino IDE supports programming the ESP32, and PHP with MySQL is a common technology stack for web development.
- **Organizational Feasibility:** The organizational feasibility depends on the company's existing technical expertise and resources. If the company has experience in PHP and MySQL development, adopting this solution will be more feasible. Adequate training and support may be required for the deployment and maintenance of the biometric door access system.

In conclusion, the selected IoT architecture, frameworks, tools, hardware, and API techniques align well with the requirements of the biometric door access system. The solution is economically viable, technically feasible, and can be made organizationally feasible with appropriate support and training. It offers an efficient and secure approach to employee authentication and door control, making it a suitable choice for the company's needs.

4. Apply your selected techniques to create an IoT application development plan.

A, Project Overview:

The IoT application aims to develop a biometric door access system using ESP32 with a fingerprint sensor (FPC1020A) and a small OLED display. The system will integrate fingerprint-based authentication with door access control, enhancing security and attendance tracking at the company premises.

B, Project Scope:

Develop a client-server architecture using ESP32 as the client and PHP with MySQL as the server.

Implement a RESTful API for communication between the ESP32 client and the server.

Design a user-friendly interface on the OLED display for visual feedback.

Integrate the servo motor to control the door lock.

Develop a web-based administration panel for user management and access control.

C, Project Timeline:

Phase	**Task**	**Timeline (days)**
Planning	Requirement gathering and scope definition	**0.5**

Hardware Setup	Procure ESP32, FPC1020A, OLED display, and servo	**2**
Software Setup	Install Arduino IDE, PHP, MySQL, and libraries	**2**
ESP32 Programming	Fingerprint sensor integration	**0.5**
	OLED display interface	**0.5**
Server Setup	PHP API development	**0.5**
	MySQL database setup and integration	**0.5**
Client-Server	Establish communication between ESP32 and server	**0.5**
Testing	Test hardware and software integration	**2**
User Interface	Design OLED display interface	**2**
Security	Implement secure authentication mechanisms	**0.5**
Integration	Integrate all components and perform testing	**1**
Total Duration		**12.5**

D, Resources:

- **Human Resources:**
  - IoT Developer: Responsible for ESP32 programming and hardware integration.
  - PHP Developer: In charge of API development and server-side scripting.
  - UI/UX Designer: Designing the user interface for the OLED display and web admin panel.
  - Database Administrator: Setting up and maintaining the MySQL database.
- **Budget:**
  - Hardware Costs: ESP32, FPC1020A, OLED display, servo motor, cables, etc. **10$**
  - Software Costs: License fees, if any, for specific development tools. **50$**
  - Personnel Costs: Salaries and benefits for the development team. **1000$**
  - Miscellaneous Costs: Training, testing, and deployment expenses. **10$**
- **Time Commitment:** The development team will work full-time for 25 weeks as per the project timeline.

E, Risk Management:

- **Internet Dependency:** Address the internet dependency by implementing a fail-safe mechanism for offline authentication during network disruptions.
- **Security:** Implement encryption and secure storage for fingerprint templates and user data to prevent unauthorized access and data breaches.
- **Compatibility:** Ensure compatibility and seamless communication between hardware components and software libraries to avoid technical challenges.

F, Quality Assurance:

- Conduct rigorous testing at various stages, including unit testing, integration testing, and user acceptance testing.
- Regular code reviews to maintain coding standards and reduce potential vulnerabilities.
- Address feedback and continuously improve the system during development.

G, Deployment and Training:

- Deploy the biometric door access system to the company premises after successful testing.
- Provide comprehensive training to employees for proper usage and maintenance of the system.

H, Project Management:

- Use project management tools and methodologies to track progress and allocate resources efficiently.
- Regular team meetings and status updates to ensure effective communication and collaboration.

By following this detailed development plan, the IoT application for the biometric door access system can be successfully completed, providing the company with an enhanced security solution for attendance tracking and access control.

5, Multiple Iterations Plan for Biometric Door Access System:

A, Iteration 1: Basic Biometric Door Access System

- Develop a basic biometric door access system using the selected architecture, hardware, and API techniques.
- Implement fingerprint-based authentication using the FPC1020A sensor and ESP32.
- Integrate OLED display to provide visual feedback on successful/failed authentication.
- Control the servo motor to unlock the door upon successful fingerprint authentication.
- Create a simple PHP API for user registration and access control.

B, Iteration 2: Enhancing Communication Security

- Implement secure communication protocols, such as HTTPS, between the ESP32 client and the server to protect data transmission.
- Encrypt fingerprint templates and user data stored in the database to prevent unauthorized access.
- Use secure authentication mechanisms, such as token-based authentication, to enhance user login security.

C, Iteration 3: Implementing Biometric Data Security

- Store fingerprint templates as encrypted hashes in the database to protect biometric data.
- Use salting and hashing techniques to further strengthen the security of stored biometric data.

D, Iteration 4: Intrusion Detection and Alarm

- Add intrusion detection capabilities to the system, using sensors (e.g., motion sensors) to detect unauthorized access attempts.
- If an intrusion is detected, trigger an alarm to notify security personnel or send alerts to authorized personnel.

E, Iteration 5: Access Logs and Audit Trail

- Implement a comprehensive logging system to record access events and user activities.
- Store access logs securely and enable auditing capabilities for tracking access patterns and detecting anomalies.

F, Iteration 6: Device Identity Verification

- Implement device identity verification to ensure that only trusted and authorized devices can connect to the server.

- Utilize digital certificates or device keys to authenticate and validate the ESP32 client before allowing access.

G, Iteration 7: Continuous Security Monitoring

- Set up continuous security monitoring and real-time alerts to detect potential security breaches or abnormal behavior.
- Integrate security monitoring tools and SIEM (Security Information and Event Management) systems.

H, Iteration 8: Regular Security Assessments and Updates

- Conduct regular security assessments, including penetration testing, to identify vulnerabilities and weaknesses.
- Perform timely updates and patch management to ensure the system stays protected against emerging threats.

I, Iteration 9: Employee Awareness Training

- Provide regular security awareness training to employees to educate them about best security practices and potential threats.
- Emphasize the importance of protecting their biometric data and following secure access procedures.

By following this iterative approach and continuously improving the IoT application's security at each stage, the Biometric Door Access System can offer a robust and highly secure solution for employee authentication and door control, ensuring the protection of sensitive data and access control integrity.

REFERENCE:

https://www.hiotron.com/iot-architecture-layers

https://www.freecodecamp.org/news/introduction-to-iot-internet-of-things/

https://www.geeksforgeeks.org/sensors-in-internet-of-thingsiot/

https://www.vectorstock.com/royalty-free-vector/iot-connectivity-vector-5411853

https://www.enertiv.com/resources/faq/what-is-iot-data-collection

https://appinventiv.com/blog/iot-and-cloud-computing-benefits-business/

https://www.mdpi.com/1424-8220/21/11/3784

https://nectarbits.com/blog/ten-popular-iot-development-tools-suggested-by-nectarbits/

https://www.encata.net/projects/digital-transformation-and-feasibility-study-for-a-large-construction-company

https://securityboulevard.com/2022/12/dont-let-your-ignorance-make-you-vulnerable-to-iot-attacks

https://securityboulevard.com/2022/12/dont-let-your-ignorance-make-you-vulnerable-to-iot-attacks/