Choosing an IoT platform:

Domain of Application, Usability, Interoperability, Scalability, Edge Analytics, Security, Recovery

Domain of Application:

* The application in which the intended IoT platform will be deployed is crucial to consider.
* Some IoT platforms are not general enough to be applicable in any domain, designed specifically for certain application areas
* To look at capabilities in terms of acquisition of data from a large variety of device types, communication, network & data protocols
* Specific needs of application domain have to be identified and matched with functionality offerings.

Usability:

* The user-friendliness of IoT platform helps in performing essential tasks, using interfaces & functions of various protocols, visualization, decision-making etc.

Interoperability:

* The ability of IoT platform -
  - To integrate third-party solutions
  - To assimilate data coming via heterogenous communication, network and data protocols, open source tools, ERP tools, etc.

  makes it a highly interoperable IoT solution.

Scalability:

* ability to handle large no of IoT devices.
* should handle unlimited no of devices & their connections seamlessly by approaches such as load balancing & distributed processing.
* check no of devices in terms of connectivity, processing at edge, storage, & subsequent deep analytics.

Edge Analytics:

* IoT platform should support edge analytics, for providing real-time information to the customers
* Ex: Real-time traffic updates, energy estimations in a building, retail, etc. need stream data processing capabilities.
* So IoT platform provide these capabilities; that can be set to trigger alerts/ warnings based on predefined rules/criteria that run on a subset of the incoming stream.

Security:

* Capability of ensuring high-level of security end-to-end by the IoT platform plays a major role in its selection.
* IoT platform needs to be assessed for comprehensive security tools for secure authentication, certification, encryption, identity-based authentication for devices.

Recovery:

* In event of instability of IoT system, how resilient is the system to come back to its original condition is an aspect that needs to be given importance while selecting an IoT platform.
* It needs to provide the ability to have enough redundancy.
* data recovery also considered. It should be able to take periodic backups & restore the data on demand.

Feature	Closed Source	Open Source
Definition.	• Software whose source code is proprietary and not available to public.	Software with publicly-accessible source code that can be modified & redistributed.
Access to code.	Only available to the company (or) authorized users.	Freely available for anyone to view, use & modify
Modification.	Restricted, only the original developers can modify it.	Open for modification by anyone under the respective license
Prototype cost & price.	/ continuous & support.	

middleware provides connection b/w applns, networks, communications & Operating system.
middleware is a software layer, site b/w s/w layer & appln layer.
system software, application softwares.

Features: [Transparency, flexibility, Reusability, Portability, Interoperability, Maintainability]

needs: [device management, Analytics [online, offline], cloud services, Security]

Functional requirements: [Resource discovery, resource management, data, code, event management, security & privacy]

nonfunctional requirements: [Scalability, Reliability, Availability, Timeliness].

Service Oriented middleware: [principle of SOA, characteristics: loose coupling, independent of implementation technology, service composability, service reusability, service discoverability]

3 layers: • sensing layer
↓
middleware components
↓
Enterprise service based applns

main components
nodes: fixed set of operations
Streams - conduit for moving data

Services:
• data storage & management
• QoS, Service discovery & management

Real-time event processing & analysis

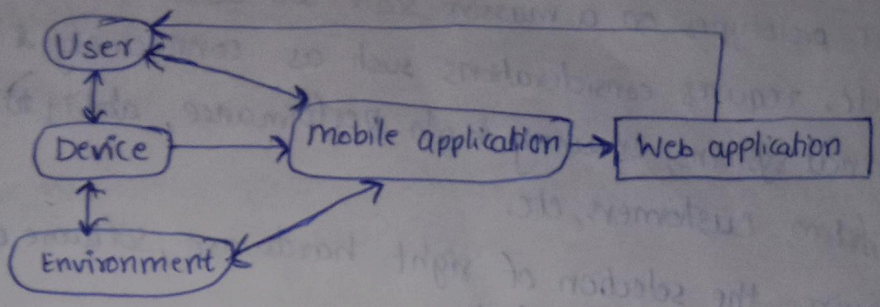
Difference blw Open Source Software (OSS) & Closed Source Software (CSS)

Aspect	OSS	CSS
Prototyping cost & price	• Free to use, but integration in commercial products can be challenging • Different open-source licenses (GPL, LGPL, BSD) impose varying restrictions.	• Usually requires payment but comes with better functionality. • Some companies offer free trials for a limited period.
Quality support	• Relies on community support, forums and articles. • Limited Professional customer service compared to CSS.	• Offers professional support services, as companies charge for them. • Customer feedback is actively documented for improvements.
Source code availability.	• Source code is fully accessible for modification & customization • Developers can freely share improvements with community.	- Code is proprietary and cannot be modified by users • Only original developers can make changes (or) updates.
Security.	• Bugs are fixed quickly by the open-source community • Open access may expose vulnerabilities to hackers.	• More secure as the code is private & restricted • Dedicated security teams handle vulnerabilities & updates
Usability.	• Documentation is mostly developer-oriented, making it harder for beginners • More flexible for customization but requires technical expertise	• well-documented & user friendly, making it easier for non-developers. • Learning & adoption are usually faster due to structured guidelines

• Customer not clear about requirements

• Designer not known how it works.

Full Stack Prototype:



Prototyping:

- Goal setting before prototype design is a very important step as the constraints for prototyping process can be well understood.
- Various constraints are Dimensions, In-built memory, Hardware ~~customation~~ customization, Power Requirements, Expenditure, Security.

Step-①: Ideation:-

- Every IoT product starts with an idea.
- Have to continue with components originally conceived for prototype at early stage instead of trying to optimize in prototype development process.

Ex: good battery life needed lower powered processor which may not support sensor needed at all times.

Step-②: Prototype:-

- Proof of concept creation becomes an expensive talk.
- Rapid prototyping has become easier and cheaper.
- It enables to develop an early product & verify its feasibility with minimal time & cost.

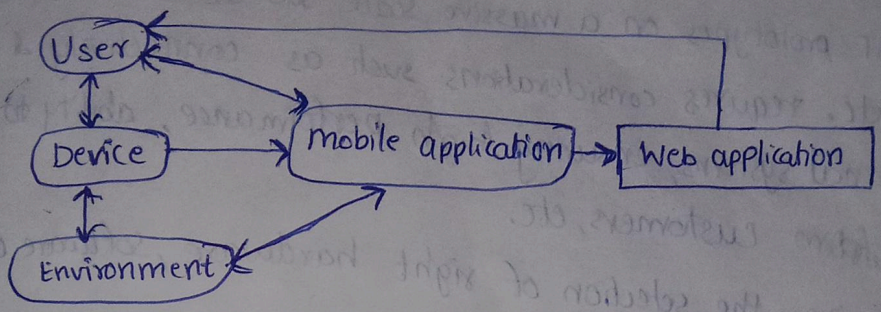
Step-③: Iterate:

- Various processes, vendors/supplies of compliments used in product and other essential supporting services are finalised.
- Certain components may change at later point of time due to the evolving nature of technology.

Step-④: Deploy:

- Conditions of geographical area to be deployed are considered.
- Battery life and connectivity issues have to be taken into account.
- Remote management and frequency of upgrades of device needs due consideration.

Full Stack Prototype:-



Prototyping:

- Goal setting before prototype design is a very important step as the constraints for prototyping process can be well understood.
- Various constraints are Dimensions, In-built memory, Hardware ~~customation~~ customization, Power Requirements, Expenditure, Security.

Step-①: Ideation:-

- Every IoT product starts with an idea.
- Have to continue with components originally conceived for prototype at early stage instead of trying to optimize in prototype development process.

Ex: good battery life needed lower powered processor which may ~~cannot~~ support sensor needed at all times.

Step-②: Prototype:-

- Proof of concept creation becomes an expensive talk.
- Rapid prototyping has become easier and cheaper.
- It enables to develop an early product & verify its feasibility with minimal time & cost.

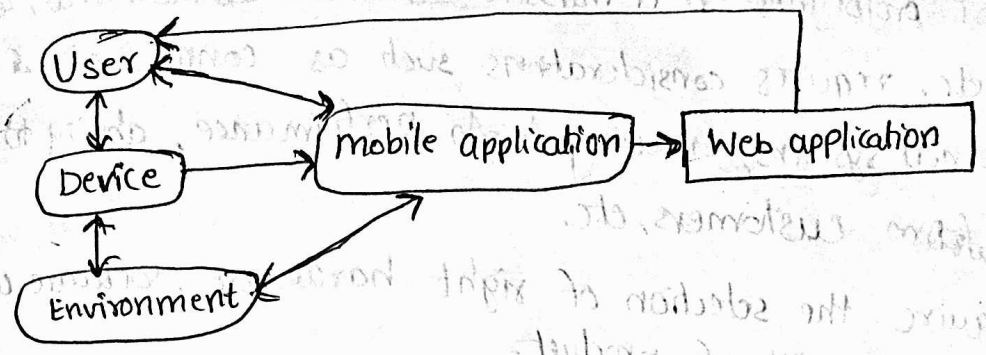
Step-③: Iterate:

- Various processes, vendors/supplies of compliments used in product and other essential supporting services are finalised.
- Certain components may change at later point of time due to the evolving nature of technology.

Step-④: Deploy:

- Conditions of geographical area to be deployed are considered.
- Battery life and connectivity issues have to be taken into account.
- Remote management and frequency of upgrades of device needs due consideration.

Full Stack Prototype:



Prototyping:

- Goal setting before prototype design is a very important step as the constraints for prototyping process can be well understood.
- Various constraints are Dimensions, In-built memory, Hardware customization, Power Requirements, Expenditure, Security.

Step-①: Ideation:

- Every IoT product starts with an idea.
- Have to continue with components originally conceived for prototype at early stage instead of trying to optimize in prototype development process.

Ex: good battery life needed lower powered processor which may not support sensor needed at all times.

Step-②: Prototype:

- Proof of concept creation becomes an expensive talk.
- Rapid prototyping has become easier and cheaper.
- It enables to develop an early product & verify its feasibility with minimal time & cost.

Step-③: Iterate:

- Various processes, vendors/supplies of compliments used in product and other essential supporting services are finalised.
- Certain components may change at later point of time due to the evolving nature of technology.

Step-④: Deploy:

- Conditions of geographical area to be deployed are considered.
- Battery life and connectivity issues have to be taken into account.
- Remote management and frequency of upgrades of device needs due consideration.

Step (5): Scale

- Scaling IoT prototypes on a massive scale such as industrial, commercial, residential, etc. requires considerations such as connectivity & interface with external systems, ensuring high performance, ability to capture feedback from customers, etc.
- These require the selection of right hardware, software components in assuring scalability of product.

Selection of physical Devices:

- We must take into consideration other objects that prototype will be using such as for specific task if users will be wearing, masks, gloves, eyeprotectors
- Prototype to be tested should be subjected to natural environment for which it is designed for target consumers, geographical location, either interactive / static, transaction online, offline, realtime etc.

Sketches & Diagrams:

- Oldest methods of prototyping. No artistic skills & little efforts are required.
- System/process or structure of ideas can be illustrated by sketches.
- Sketches are useful to understand complex use cases. Behaviour maps, journey maps, system flow diagrams, & many other mapping methods are used for understanding complex situations.

Physical Design Considerations:

① Different modules for IoT prototyping:

- Processor modules - for executing device firmware
- Sensory modules - sense surrounding ~~information~~ environment for information collection.
- Power modules - To supply power.
- Input/Output modules - facilitates users to interact with product physically for providing input or accessing output.
- Communication module - for device communication, to facilitate the device with internet.
- Action modules - to control surrounding objects using device

② Selection of Embedded platforms: Microcontroller, SOC

Processor module: Processor board is the main part of prototype of any IoT device. IoT device processor is generally a lower power microcontroller board.

main things to consider while selecting processor are:

- user friendly development environment, hardware capabilities of selected processor, memory (memory size) ~~is to be studied~~

③ Common Open Source Microcontroller Boards for IoT prototyping:

- Raspberry Pi, Arduino Board, BeagleBoard, Intel.

④ Selection of sensors & actuators:

Sensor modules: choosing right sensor involves many factors such as availability, accuracy, precision, measurement range, power consumption, cost and supplier.

Action modules: several action modules that allow the device to control the surroundings. Ex: servo motor, DC motor controllers, sound amplifiers & relays

⑤ Communication modules:

Wi-Fi, BLE, Cellular, etc.

⑥ Power modules:- (battery operated (or) plugged-in).

⑦ Input/Output modules:- LEDs, switches, touch-pads, LCD displays, etc.

Rapid Prototyping Techniques:

- 3D printing, Computerized Numerical Control (CNC) machining, Turning, Milling, Drilling, Casting.

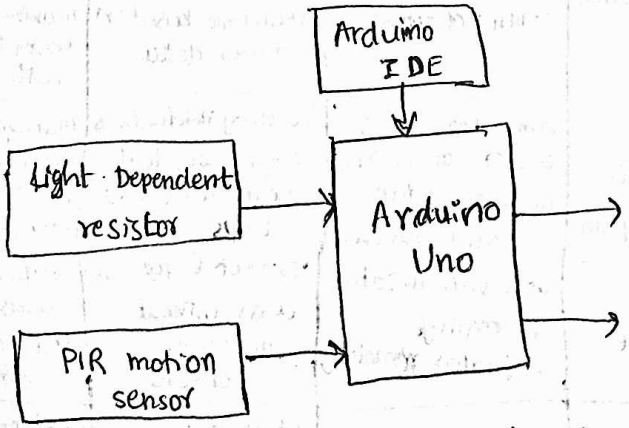
Prototype Stage to Production Stage:-

- Sourcing, NPR release with Tooling & Sampling, Production Launch.

NPR → Non Production Release

Prototyping Logical Design:

- Unambiguous business Strategy.
- Optimising design.
- Selecting appln specific power model.
- Device Security aspects
- Adapting to network technology evolution.
- Support for multiple platforms
- Adopt well proven technologies.



- Block diagram of Automation of Smart Light & Smart fan.

Aspect	IaaS (Infrastructure level)	PaaS (Service-level)	SaaS (Application-level)
Key benefits	cost effective and scalable.	Accelerates application development	Eliminates the need for local software installation
Common use cases	Hosting website and data storage	Developing custom applications.	Email services & CRM.
Control	high	moderate.	Low.
Target Users	Developers.	Business users.	System managers
Access.	Runtime environment	User functionalities.	VMs & storage.
Examples	AWS.	Google App Engine	Salesforce CRM.

Criteria.	SEaaS (Sensing as a Service)	SFaaS (Sensing & Actuator	SEaaS. (Sensor Event	SenaaS (Sensor as a
Functionality.	Provides raw sensor data for use in appln.	Includes both sensing & actuation capabilities.	Processes event-based data from IoT sensors & triggers notifications.	Virtualizes sensors, exposing their capabilities as a service.
Data Vs Action.	Focuses on collecting & sharing sensor data.	Allows both data collection & remote control of actuators.	Detects and processes events in real-time based on sensor-data.	Abstracts sensor hardware, making it easier to integrate with software.
Example Use cases.	Selling temperature, air quality, (or) traffic data in a marketplace.	Remote monitoring & control of industrial machines (or) fire alarm systems	Sending notifications when water levels rise (or) air quality drops	Managing large-scale sensor networks with privacy & access control.
Virtualization	Provides data but does not virtualize sensors	Uses virtualization to manage actuators remotely.	Detects & processes data without virtualizing sensors.	Virtualizes sensors to hide technical complexities
User Interaction.	Users can buy (or) sell sensing data.	Users can monitor & also remotely trigger actions.	Users subscribe to alerts or notifications.	Developers can integrate sensors without worrying about communication protocols.

Selected Cloud Service Providers (*)

* when it comes to selecting cloud platforms for IoT, scalability, cost & functionality are impt. parameters to be considered.
* Two open-source platforms; Kaa IoT & ThingSpeak.
  Two most popular platforms; Google & Oracle Cloud.

(1) Kaa IoT platform:

- Open-Source, multipurpose middleware platform for end-to-end IoT application development.
- It reduces cost, market time & risk as open-source.
- many unique features;
  - open source, Handles millions of devices, Reduced development time, Ease of implementation, Reduced marketing time.
- Advantages;
  Data Security, Easy to use, third party integration.
- Limitations: Application based deployment is not possible.

(2) ThingSpeak IoT platform:

- Open Source platform for collecting & storing sensor data on the cloud.
- It provides you with apps to analyze & virtualize your data in MATLAB.
- Arduino, Raspberry Pi & Beaglebone boards can be used to send sensor data.
- Features;
  - App integration.
  - collection of data from private channels
  - Feature supporting scheduling of event.
  - Visualization & analytic functionality of MATLAB.
- Advantages;
  - Easy Virtualization / Free channel hosting / Python, Node.js & Ruby support.

(3) Google Cloud:

- One of the most popular cloud platforms available. end-to-end platform for IoT solutions.
- Facilitates easy connection, storage & management of IoT data.
- main focus on making smart things in easy & fast way.
- Per minute use pricing is offered by Google cloud.
- Main features:
  - Efficient & scalable.
  - Very large storage capacity.
  - Sever maintenance cost is less.
  - Analyses big data.

- Advantages:
  - Lesser access time than many other platforms
  - Fastest input/output
  - Provides integration with other Google Services.
- Limitations:
  - Limited programming language choices.
  - Many components are using Google Technologies.

(4) Oracle cloud:

- Offers real-time IoT data analysis, end-point management and high-speed messaging when users can get real-time notifications on their devices.
- Oracle IoT cloud service is a PaaS. Cloud-based offering helps in making critical business decisions.
- Features:
  - Real time feedback
  - Security & scalability
  - Analytics
  - Reduced marketing time
- Advantages
  - High speed messaging, Event storing capability, Device visualization.

(*) REST-BASED WEB SERVICES for IoT:

- In resource-centric infrastructures such as IoT, REST (Representational state transfer) style of web services are preferred due to lightweight, simple, portability, reliability & ability to directly transmit data via HTTP.
- The representation of state of IoT resource anemometer is the value '20 mbs/sec'.
- The interaction b/w client & server is stateless as opposed to stateful web services such as those based on Simple Object access protocol (SOAP).
- Advantage is including the usage of URIs for IoT resources.
- It is based on Service Oriented Architecture (SOA), it inherits the basic features of loose coupling, reusability, scalability, integrity, etc.
- Constrained Application Protocol (CoAP) is specialized web transfer protocol, that works with constrained nodes & constrained networks in the Internet.
- It is used to connect the low powered devices to IoT.
- CoAP is based on architectural principles of REST & runs over Universal Datagram (UDP) protocol.

Database Management System (DBMS)	Data stream Management Systems (DSMS)
Data level	
• Static/stored data, time of capture (or) storage of data is unimportant • Access to data can be either sequential or random • Data updation is relatively slow, usually in relatively small increments • Data is consistent and has high precision	• Continuously changing data, time of event capture and time of processing are important. • Data access is sequential and is usually based on the timestamp of incoming events • High frequency & volume of data updation • Data is noisy due to imperfections in data capturing process
Processing level	
• Processing on data available currently in database. • Asynchronous data processing, no need to process (or) deliver results in real time • Processing on relatively large portions of data • Data value remains consistent over time.	• Processing on current as well as historical data, which could be data that arrived few seconds, hours, or days before • Synchronous, real-time processing of data, Requirement for delivery of results rapidly • Incremental processing approach • Data can become stale and no longer be usable after a set period of time.
Query level	
• Queries can be dynamic & complex. • Queries are precise. • Multiple rounds of data access may be done to answer a query	• Queries are pre-determined and run on time varying data. Complexity is in the capturing and processing of event patterns. • Queries can be approximate and can return approximate results • One time access to data, after which the data is no longer available

Feature	cold storage	Hot storage
Accessibility	Slow, data is stored offline / in low-accessibility environments.	Fast, data is stored online and readily accessible.
Use case	long term storage for rarely accessed data	Frequently accessed data and real-time applications
Security and cost	More secure as it's offline and lower cost.	Less secure due to continuous online availability & higher cost

Apache, Storm, Millwheel, Aapache Samza, Apache Flink

- Name, graduation
- Desired role
- strength at what skill, weakness
- Company's requirement.

IoT system Layers and its Specifications for Security controls,

IoT Layer	Components	Working of Layer	Security Issues	Security parameters	Counter-measures
Perception Layer	Smart Card, RFID tag, Sensors.	Collection of Information	Terminal Security, Sensor network security	Authentication, confidentiality	Certification & access control
Network Layer	Wireless (or) wired network, computer, components	Transmission of information.	Information transmission security.	Integrity, availability, confidentiality	Hop-by-hop data encryption
Application Layer	Intelligent devices.	Analysis of information, control decision-making	Information processing	Safety of IoT, privacy	End-end encryption.

Denial of Service (DoS)

① Physical Layer:

- The actual transmissio in this layer with signals, modulation
- Various attacks-

Jamming :- The by cre comm

Node tampering:

② Link layer:-

- Various data strea
- Data frame detect control functionali
- possible attacks

Collision:- Battery exha

③ Network Layer:

- While routing pac

Spoofing: Relayi attac ban ava

Feature	Cold storage	Hot storage
Accessibility	Slow, data is stored offline / in low-accessibility environments.	Fast, data is stored online and readily accessible.
Use case	long term storage for rarely accessed data.	Frequently accessed data and real-time applications
Security and cost	More secure as it's offline and lower cost.	Less secure due to continuous online availability & higher cost

Apache, Storm, Millwheel, Aapache samza, Apache Flink.

- Name, graduation
- Desired role
- strength at what skill, weakness
- Company's requirement.

SWOT

IoT System Layers and its Specifications for Security controls

IoT Layer	Components	Working of Layer.	Security Issues	Security Parameters	Counter-measures
Perception Layer.	Smart Card, RFID tag, Sensors.	Collection of Information.	Terminal Security, Sensor network security	Authentication, confidentiality	Certification & access control
Network Layer	Wireless (or) wired network, computer, components	Transmission of information.	Information transmission security.	Integrity, availability, confidentiality.	Hop-by-hop data encryption
Application Layer	Intelligent devices.	Analysis of information, control decision-making	Information processing	Safety of IoT, privacy.	End-to-end encryption.

Denial of Service (DoS) Attacks:-

① Physical Layer:

- The actual transmission and reception of data is carried out in this layer with selection & generation of carrier frequency signals, modulation & demodulation, encryption & decryption techniques.
- Various attacks-

Jamming:- The intruder exhausts the bandwidth of channel by creating fake traffic to prevent node-to-node communication of WSN.

Node Tampering: The WSN node is tampered with by eavesdropping to steal and tamper important information. Thus availability of correct information is affected.

② Link layer:-

- Various data streams are multiplexed here.
- Data frame detection and error control and medium access control functionalities are provided here.
- possible attacks -

Collision:- When 2 packets trasmitted on same frequency, the collision of packets happens due to sharing transmission medium simultaneously & data packets are corrupted. so, retransmission is required to remove the error.

Battery exhaustion:- When a DoS attack generates large amount of data traffic, the transmission channel is available only for a few devices in an IoT network. So, large no of requests are sent result in exhausting battery.

③ Network Layer:

- While routing packets in WSN, possible attacks are -

Spoofing: Relaying & misdirection of traffic is done in this attack by sending useless messages to exhaust the bandwidth of the channel to affect network availability of legitimate user.

- Selective forwarding: The compromised node is used for forwarding the messages to only selected nodes to enable the intention of attacker to perform malicious activity.

Homing: search is done in traffic to find out nodes, which do key management and search cluster heads for taking control for malicious activities.

Wormhole: Through tunnelling of bits of data on a low latency link, a relocation of data packet, is carried out by changing the data position on network.

Sybil: An attacker replicates a single node with multiple identities to other nodes in sybil attack.

Acknowledgement flooding: Intruder spoofs the wrong acknowledgement to the sensor network node.

(4) Transport Layer:

- Transport layer of WSN due to its architecture avoids congestion by reducing traffic & provides reliable transmission.
- possible attacks -

Desynchronizing: Fake messages are generated at endpoint nodes asking for retransmission of message, though no error. These messages increase traffic & consume battery.

Flooding: Traffic is generated by useless messages to create congestion for affecting availability of transmission facility to authentic user.

(5) Application Layer:

- In application layer of IoT, while carrying out software services, this layer also provides traffic management service.
- Path-based DoS attacks are dominant here.
- Other DoS attacks:
  node subversion, node malfunction, false node, message corruption, neglect & greed, black holes & interrogation.

Best Practices for securing IoT devices:

- **Tamper Resistant Hardware:**
  - → This device should be kept in isolation and only designated persons should get the physical access
  - → Physical endpoint security can be ensured by camera covers, small plastic devices, port locks to provide secure cover to webcam, USB & Ethernet ports.
  - → Implements strong boot password is best practices for device security.
  - → Open ports, open password prompts, comm^n without encryption.
- **Firmware updates / Patch updates:**
  - → Time to time firmware upgrade to remove loop holes
  - → Formakes system to be more efficient & reliable.
- **Dynamic Testing:**
  - → for assigning min baseline of sec, testing of IoT system devices is essential.
  - → static testing is not useful but Dynamic testing for finding vulnerabilities for checking the hlw & slw items used dynamic testing should be done.
- **Strong Authentication Technique practices:**
  - → strong username, password credentials should be used for authentication purpose.
  - → should be unique.
  - → adaptive authentication like context-aware authentication (CAA) is advised.
- **Use of Secure protocols & encryption:**
  - → Zigbee, Z-wave, 6LoWPAN, NFC, Wi-Fi, Neul, Sigfox.
  - → Depending on the kind of protocol & device comp. compability, encryption technique is used.
- **Network division into segments:**
  - → Network divison creates diff. security zones with diff. security policies for each zone.
  - → established based on kind of application, information & threat level resulting in preventing internal & external unauthorized access
- Sensitive Information protection.
- Encouraging Ethical hacking & discouraging Harbour for unethical practices
- Need for IoT security & privacy certification Board.

Data Science Process

Data acquisition:

- data can be obtained from variety of IoT devices (sensors, actuators, etc) either in online mode (or) data that is acquired from these devices & stored, secured & managed in local / cloud repository (both.
- Data is highly heterogenous, can be in several forms -

unstructured: not fit into row/column, no predefined model, schema associated with data. Can be text, images, videos, time-series data etc. Data storage is challenge due to differing requirements for storage of each of these types of data

structured: form of data has predefined record length & associated data model. Rarely IoT data is available like that.

- Data can be classified on its state -

static data: usually in high volume & communicated using communication protocols such as MQTT & CoAP, then integrated by IoT services for further processing & storage.

Streaming data: captured usually at gateways. Since this data is in motion, analytics are tuned to work on a subset of data at a time. This real-time processing is useful to send warnings, alerts, etc.

Data Understanding, preprocessing & preparation:

Importing data: various ways to import include reading from tables, excel sheets, clipboard, csv, fixed width formatted data, etc.

Data cleaning: involves rectifying improper data having missing values, formatting issues, malformed records issues, outliers, etc.

Data merging: combining datasets from different sources to create a unified dataset, further used for processing. Useful for data integration purpose when there is heterogeneity in datasets.

Data standardisation: Is required bcoz the variables are measured at different scales. The common way to do standardization is to bring the data into a normal distribution, that is, Gaussian with zero mean & unit variance.

Data scaling: is process of scaling features b/w predefined max & min.

Data normalization: Used for scaling the sample data values so that it has unit norm. This is usually performed using either L1 (or) L2 norms