ASSIGNMENT 1 BRIEF

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number	Unit 43: Internet of Things		
Assignment title			
Academic Year			
Unit Tutor	Ho Hai Van		
Issue date		**Submission date**	
IV name and date			

Submission Format:

Format: This assignment is an Individual assignment and specifically including 1 document: You must use font *Calibri size 12, set number of the pages and use multiple line spacing at 1.3. Margins must be: left: 1.25 cm; right: 1 cm; top: 1 cm and bottom: 1 cm.* The reference follows Harvard referencing system. The recommended word limit is *2.000-2.500 words*. You will not be penalized for exceeding the total word limit. The cover page of the report has to be the Assignment front sheet 1.

Submission: Students are compulsory to submit the assignment in due date and in a way requested by the Tutors. The form of submission will be a soft copy posted on http://cms.greenwich.edu.vn/

Note: The Assignment *must* be your own work, and not copied by or from another student or from books etc. If you use ideas, quotes or data (such as diagrams) from books, journals or other sources, you must reference your sources, using the Harvard style. Make sure that you know how to reference properly, and that understand the guidelines on plagiarism. *If you do not, you definitely get fail.*

Unit Learning Outcomes:

LO1 Analyze what aspects of IoT are necessary and appropriate when designing software applications

LO2 Outline a plan for an appropriate IoT application using common architecture, frameworks, tools, hardware and APIs

LO3 Develop an IoT application using any combination of hardware, software, data, platforms and services.

LO4 Evaluate your IoT application and detail the problem your IoT application solves, the potential impact on people, business, society and the end user and the problems it might encounter when integrating into the wider IoT ecosystem

Assignment Brief and Guidance:

You currently work as a product developer for a new startup where you design IoT products for the consumer, corporate, government and defense clients. As part of your role your manager has tasked you to plan and develop a new IoT product, service or application for a potential client. You are required to identify a target user and conduct tests with this user and include this feedback into multiple iterative versions of your product.

Part 1 (Assignment 1): For the first part, you must:

- Plan an IoT application for a specific target end user and the tests you intend to conduct with this user. This plan will be in the form of a document and will include supporting evidence and material, such as user personas and customer journey maps
- Create multiple iterations of your application and modify each iteration with enhancements gathered from user feedback and experimentation. This will follow the pathway outlined in your plan (log book, …)

Part 2 (Assignment 2): For the first part, you must:

- Show evidence about Developed IoT application using any combination of hardware, software, data, platforms and services (video or images of your IoT system with code snippet)
- Evaluate your IoT application and detail the problem your IoT application solves, the potential impact on people, business, society and the end user and the problems it might encounter when integrating into the wider IoT ecosystem

Learning Outcomes and Assessment Criteria		
Pass	**Merit**	**Distinction**
LO1 Analyse what aspects of IoT are necessary and appropriate when designing software applications		
P1 Explore various forms of IoT functionality. **P2** Review standard architecture, frameworks, tools, hardware and APIs available for use in IoT development.	**M1** Evaluate the impact of common IoT architecture, frameworks, tools, hardware and APIs in the software development lifecycle. **M2** Review specific forms of IoT architecture, frameworks, tools, hardware and APIs for different problem-solving requirements.	**D1** Evaluate specific forms of IoT architecture and justify their use when designing software applications.
LO2 Outline a plan for an appropriate IoT application using common architecture, frameworks, tools, hardware and APIs		
P3 Investigate architecture, frameworks, tools, hardware and API techniques available to develop IoT applications. **P4** Determine a specific problem to solve using IoT.	**M3** Select the most appropriate IoT architecture, frameworks, tools, hardware and API techniques to include in an application to solve this problem. **M4** Apply your selected techniques to create an IoT application development plan.	**D2** Make multiple iterations of your IoT application and modify each iteration with enhancements gathered from user feedback and experimentation.

Table of Contents

Table of Figures

Table of Tables

ASSIGNMENT 1 ANSWERS

Introduction

Figure 1: Internet of Things

The Internet of Things (IoT) is a major technology by which we can produce various useful internet applications when the demand for internet application development is rising today. Basically, IoT is a network in which all physical objects are connected to the internet through network devices or routers and exchange data. It allows objects to be controlled remotely across existing network infrastructure. Can say, IoT is a very good and intelligent technique which reduces human effort as well as easy access to physical devices. “Things” in the IoT sense, is the mixture of hardware, software, data, and services. It can refer to a wide variety of devices such as DNA analysis devices for environmental monitoring, electric clamps in coastal waters, Arduino chips in home automation and many other. These devices gather useful data with the help of various existing technologies and share that data between other devices (for examples, Home Automation System which uses Wi-Fi or Bluetooth for exchange data between various devices of home). Four Layers Model of IoT includes Integrated Application, Information Processing, Network Construction, Sensing and Identification.

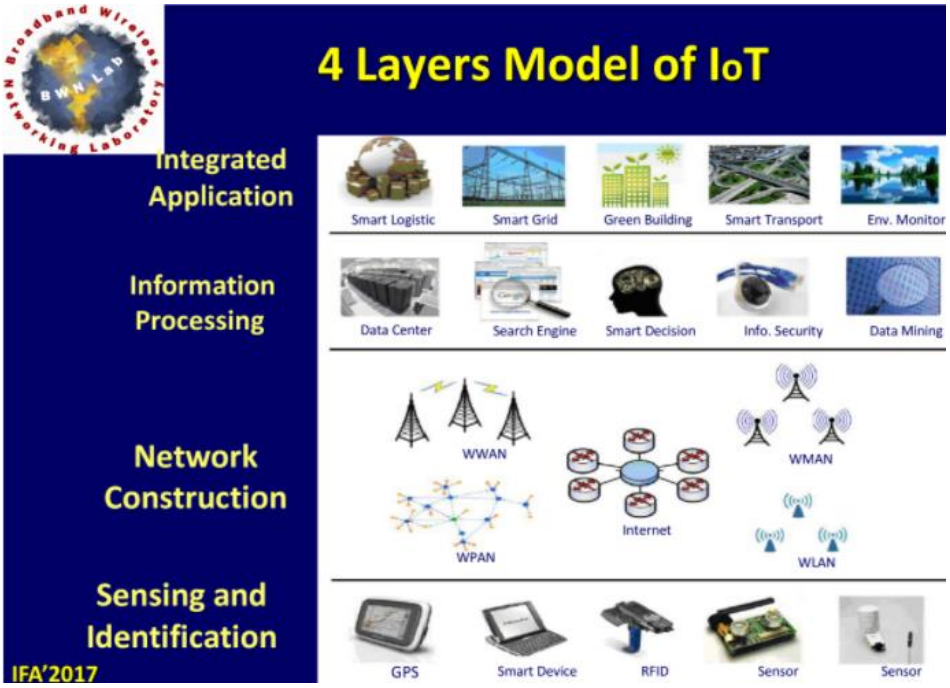

Figure 2: Four Layers Model of IoT

LO1 Analyse what aspects of IoT are necessary and appropriate when designing software applications

P1 Explore various forms of IoT functionality

1. **Internet of underwater things**

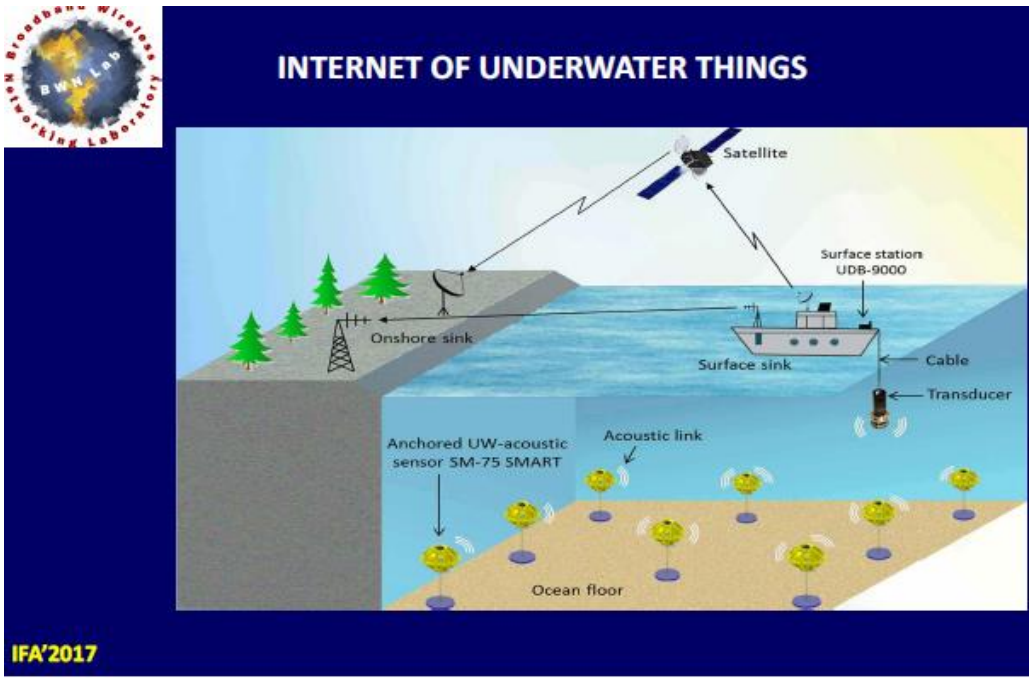

Figure 3: Internet of Underwater Things

Internet of Underwater Things (IoUT) was first discussed in 2012, following the successful establishment of IoT. IoUT is the application of IoT to water bodies. Say other, IoUT is the network of interconnected smart objects underwater. The aim of this technology is to digitally connect all of the earth's water bodies like oceans, lakes, streams, and rivers.

IoUT when in full operation will have a lot of advantages to the human race like environmental monitoring - a network of smart underwater objects will enable thorough underwater environmental monitoring, this will enable water quality monitoring, pollution detection, and control, oil and gas pipeline monitoring, etc. Next, IoUT also ensure disaster prevention by detecting early signs of a tsunami and other natural disasters. Besides, IoUT can also be useful in the educational sector for research purposes with a better experience in aquatic education, underwater data collection, and archeological expeditions.

Lots of smart underwater objects have been created in recent years. Some of these are autonomous underwater vehicles(AUV), remotely operated underwater vehicles (ROV), autonomous surface vehicles, and more. With these inventions, underwater wireless sensors networks have been deployed, and have been a largely successful network system. In this network system, underwater sensors are the major components and the sensors used are nodes with acoustic modems. These sensors are connected to the smart objects underwater through nodes, and relay information such as water quality, temperature, and pressure through acoustic modems. While this is a promising prospect, it can not yet be put to full use, due to the challenges facing IoUT.

2. Internet of underground things

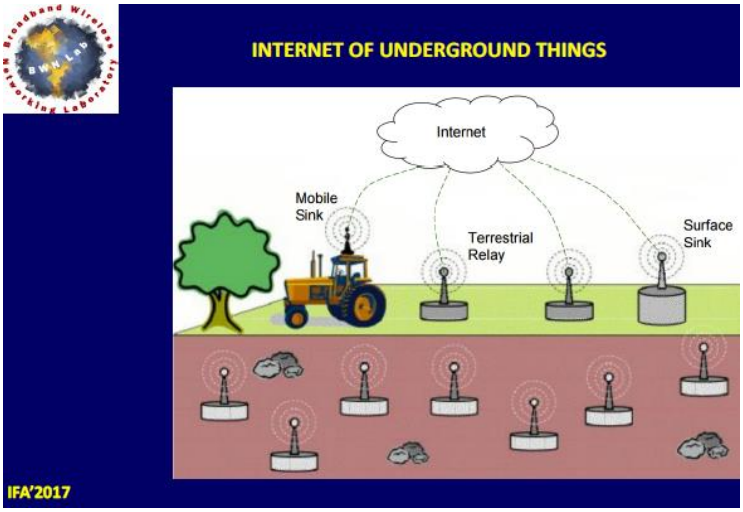

Figure 4: Internet of Underground Things

Internet of Underground Things (IoUT) is an enabling technology that can provide smart oil and gas fields, smart agriculture fields, and smart seismic quality control. However, the implementation of IoUT is a challenging task due to the harsh underground environment which requires low power and small size underground sensors, long-range communication technology, efficient networking solutions, and accurate localization techniques. IoUT is the communication media where the sensors (underground things) are buried and communicate through the soil. Note that there is also a possibility to place the underground things in an open underground space, such as mines and tunnels. In this case, the network setup is underground, but the communication between the devices takes place through the air and hence terrestrial wireless technologies, such as radio frequency (RF) and visible light (VLC) are used.

3. Internet of battlefield things

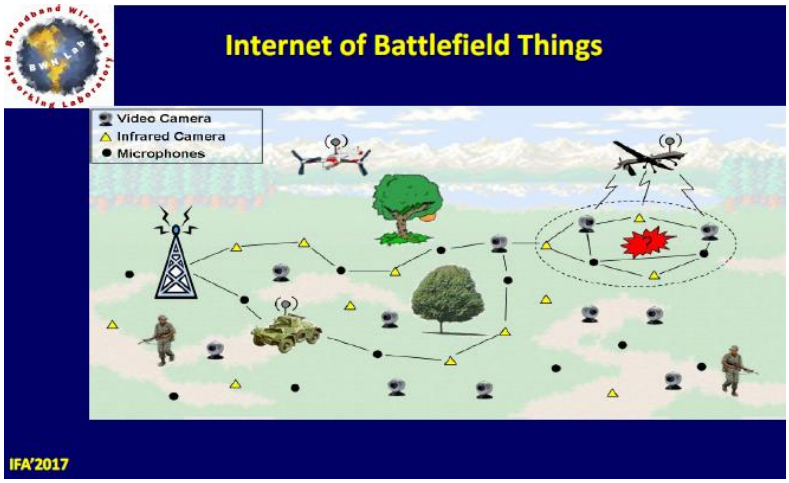

Figure 5: Internet of Battlefield Things

The rapid emergence of the Internet of Things is propelled by the logic of two irresistible technological arguments are machine intelligence and networked communications. Things are more useful and effective when they are smarter, and even more so when they can talk to each other. Exactly the same logic applies to things that populate the world of military battles. They too can serve the human warfighters better when they possess more intelligence and more ways to coordinate their actions among themselves. We call this the Internet of Battle Things (IoBT). In some ways, IoBT is already becoming a reality but 20-30 years from now it is likely to become a dominant presence in warfare.

In other words, the battlefield of the future will be densely populated by a variety of entities ("things") – some intelligent and some only marginally so – performing a

broad range of tasks like sensing, communicating, acting, and collaborating with each other or with human warfighters. They will include sensors, munitions, weapons, vehicles, robots, and human-wearable devices. Its capabilities will include selectively collecting and processing information, acting as agents to support sensemaking, undertaking coordinated defensive actions, and unleashing a variety of effects on the adversary. They will do all this collaboratively, continually communicating, coordinating, negotiating, and jointly planning or executing their activities.

4. Internet of space things

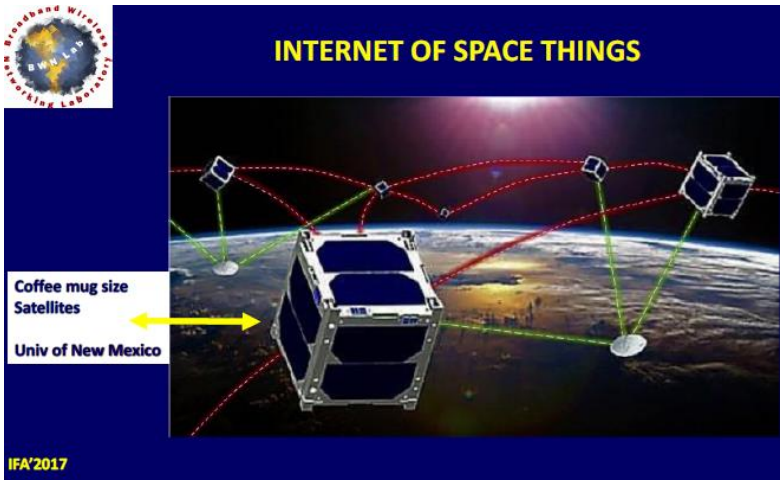

Figure 6: Internet of Space Things

The Internet of Things (IoT) has been recognized as a key driver of 5G wireless communications, with a projected 50 billion endpoints by 2020 ranging from connected temperature sensors to unmanned aerial vehicles. The long term success of IoT is tied to its pervasiveness, an area where the heterogeneous connectivity solutions of today fall short by a large margin. The true potential of IoT can only be realized when it is augmented with a ubiquitous connectivity platform capable of functioning even in the most remote of locations. To this end, this project focuses on the development of a novel cyber-physical system spanning ground, air, and space, called the Internet of Space Things/CubeSats (IoST). IoST expands the functionalities of traditional IoT, by not only providing an always-available satellite network backhaul, but also by contributing real-time satellite-captured information and, more importantly, performing integration of on the ground data and satellite information to enable new applications. The fundamental building block for IoST is a new generation of nano-satellites known as CubeSats, which are augmented with SDN and NFV solutions.

5. Internet of nano-things

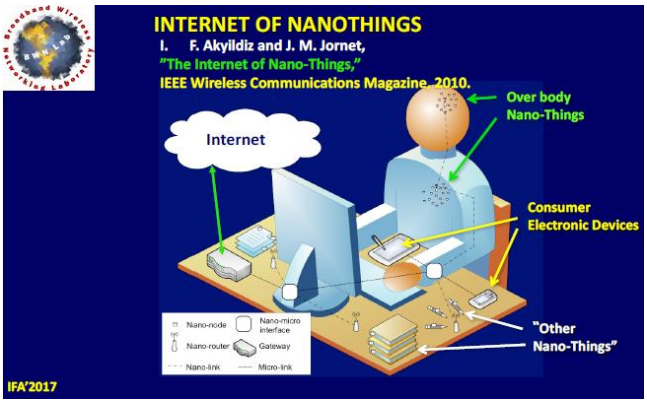

Figure 7: Internet of Nano-Things

The Internet of Nano Things (IoNT) adds a new scale in IoT incorporating nano-sensors in the devices, which in turn allows it to connect and communicate through the nanotechnology network with the internet. This new paradigm of communication networks will have an impact in almost all fields of our society, from health to the protection of the environment. Therefore, there are nano-scale properties that require new solutions for communication and that must be provided by the information and communication sciences. The focus of this work is based on the study of the current state of IoNT, which links two approaches where there have been significant advances in recent years such as IoT and nanotechnologies. The aim of this work is to investigate the state of the art and analyze trends in the use of IoNT, its application, and future challenges in different fields of social interest. Some applications of IoNT like Multimedia, Biomedical, Military, Industrial, Environmental, etc.

6. Internet of bio-nanothings

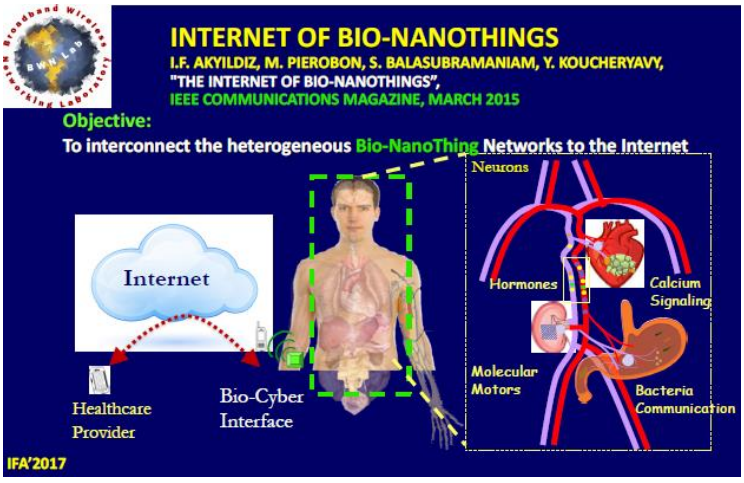

Figure 8: Internet of Bio-NanoThings

While research and development continue for general IoT devices, there are many application domains where very tiny, concealable, and non-intrusive Things are needed. The properties of recently studied nanomaterials, such as graphene, have inspired the concept of Internet of NanoThings (IoNT), based on the interconnection of nanoscale devices. Despite being an enabler for many applications, the artificial nature of IoNT devices can be detrimental where the deployment of NanoThings could result in unwanted effects on health or pollution. The novel paradigm of the Internet of Bio-Nano Things (IoBNT) is introduced by stemming from synthetic biology and nanotechnology tools that allow the engineering of biological embedded computing devices. Based on biological cells, and their functionalities in the biochemical domain, Bio-NanoThings promise to enable applications such as intra-body sensing and actuation networks, and environmental control of toxic agents and pollution. The IoBNT stands as a paradigm-shifting concept for communication and network engineering, where novel challenges are faced to develop efficient and safe techniques for the exchange of information, interaction, and networking within the biochemical domain, while enabling an interface to the electrical domain of the Internet.

P2 Review standard architecture, frameworks, tools, hardware and APIs available for use in IoT development

1. IoT standard architecture

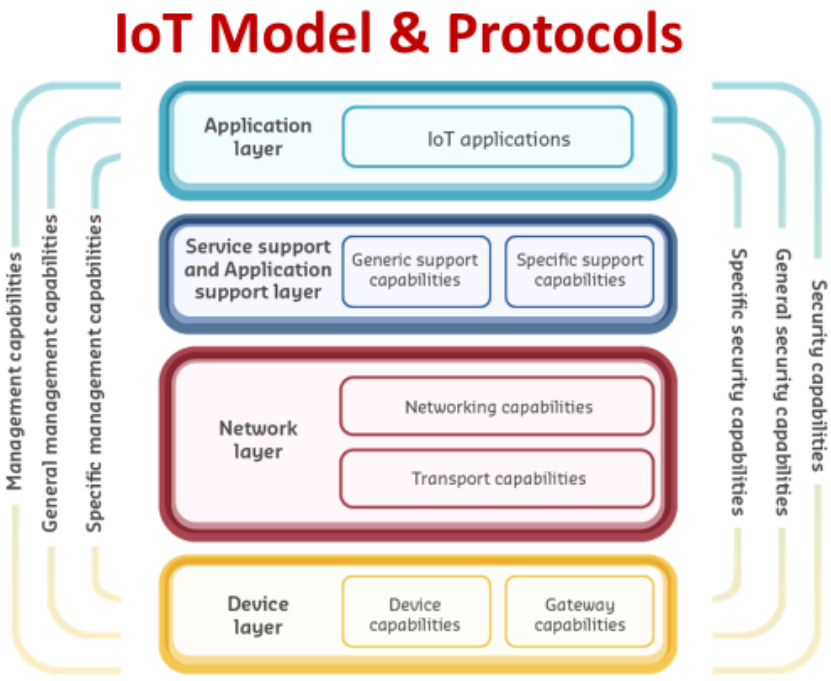

Figure 9: IoT Standard Architecture

While every IoT system is different, the foundation for each Internet of Things architecture as well as its general data process flow is roughly the same. First of all, it consists of the Things, which are objects connected to the Internet which by means of their embedded sensors and actuators are able to sense the environment around them and gather the information that is then passed on to IoT gateways. The next stage consists of IoT data acquisition systems and gateways that collect the great mass of unprocessed data, convert it into digital streams, filter, and pre-process it so that it is ready for analysis. The third layer is represented by edge devices responsible for further processing and enhanced analysis of data. This layer is also where visualization and machine learning technologies may step in. After that, the data is transferred to data centers which can be either cloud-based or installed locally. This is where the data is stored, managed, and analyzed in-depth for actionable insights.

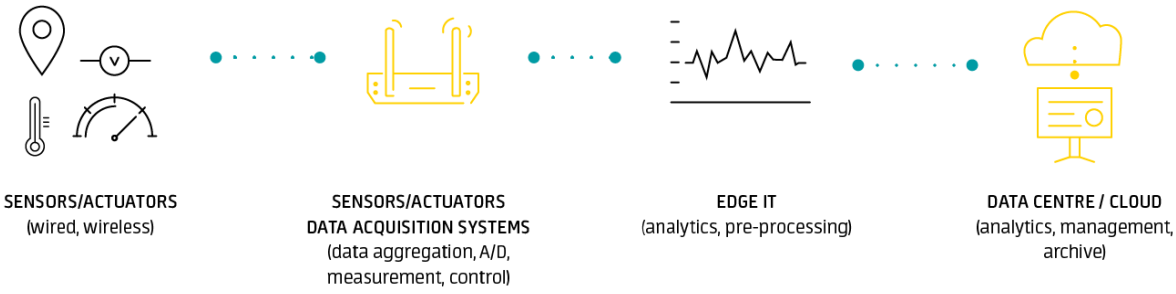

Figure 10: Four Layers of IoT Architecture

Below are described in detail four layers of IoT architecture:

Figure 11: Things, Sensors and Controllers

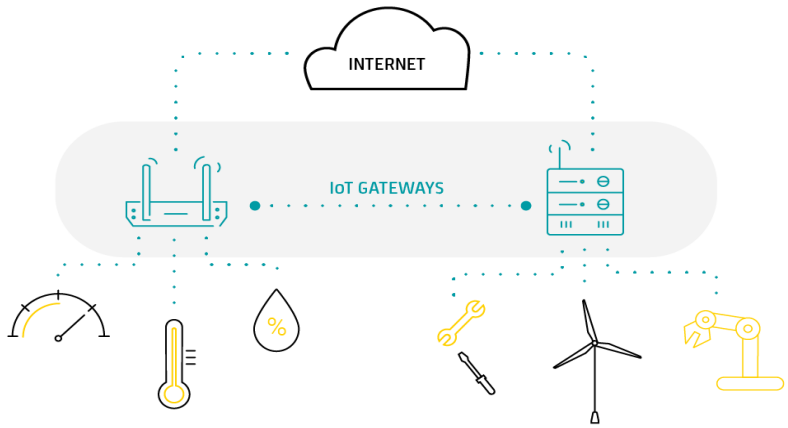

Figure 12: Gateways and Data Acquisition

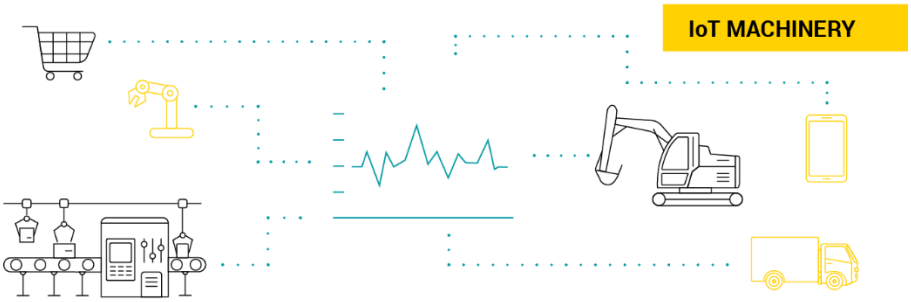

Figure 13: Edge Analytics

Figure 14: Data Centre / Cloud Platform

2. IoT standard framework

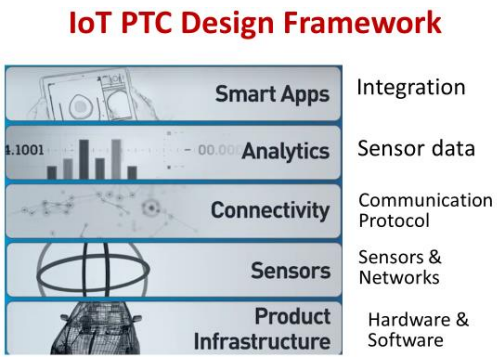

Figure 15: IoT Standard Framework

Standards frameworks help simplify programming chores. Sometimes the choices are dictated by a chosen platform. There are a number of vendors that provide hardware and the software support, and developers will be hard pressed to work outside of these confines. There are other open solutions that do not necessarily address an end-to-end approach, but rather, leave providing significant portions of the environment to the programmer. They also may not address issues such as system and power management, remote updates, and so on. Typically IoT frameworks are built on top of or otherwise utilize standard communication protocols like TCP/IP. They may be more specific in support such as using IPv6 over Low power Wireless Personal Area Networks (6LoWPAN). They may also use higher level protocols like Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), or Extensible Messaging and Presence Protocol (XMPP). It is also possible to use something like Object Management Group's (OMG) Data Distribution Service (DDS).

3. IoT standard tool – Tinkercad

Tinkercad is an easy, browser-based 3D design and modeling tool for all. Tinkercad allows users to image anything and then design it in minutes.

(Link: www.tinkercad.com)

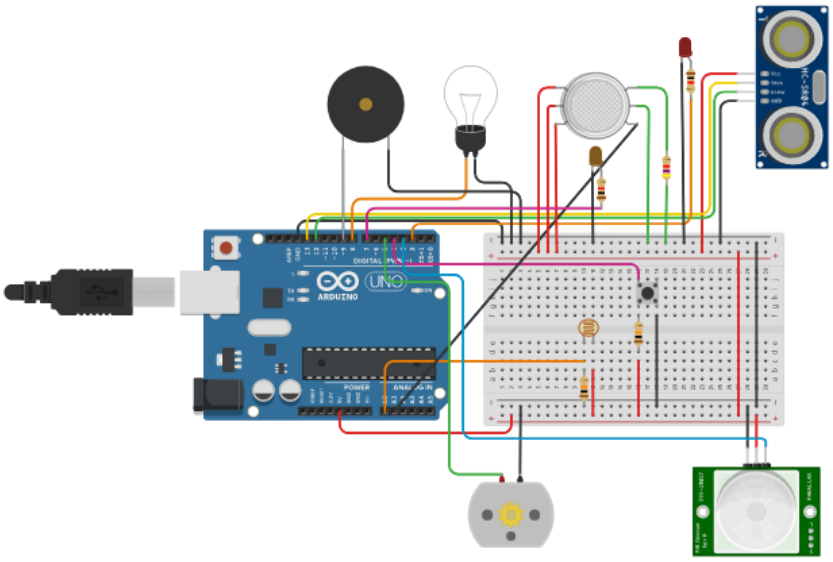

Figure 16: Example about Smart Home using Tinkercad

4. IoT standard tool – Cisco Packet Tracer

Packet Tracer is a cross-platform visual simulation tool designed by Cisco Systems that allows users to create network topologies and imitate modern computer

networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command-line interface. Packet Tracer makes use of a drag and drops user interface, allowing users to add and remove simulated network devices as they see fit.

(Link: http://www.netacad.com/courses/packet-tracer and http://static-course-assets.s3.amazonaws.com/I2PT/en/index.html#6.1.1.1)

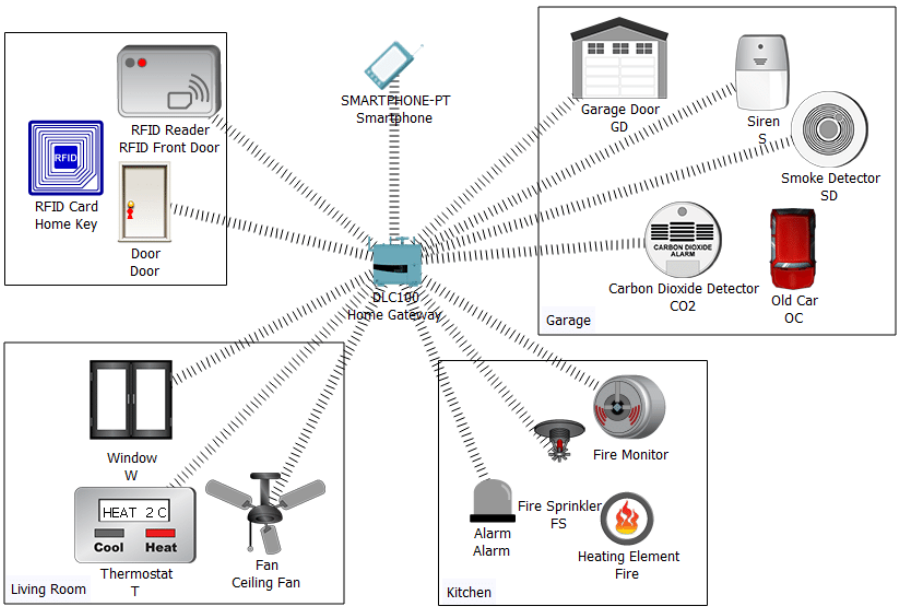

Figure 17: Example about Smart Home using Cisco Packet Tracer

5. IoT standard hardware

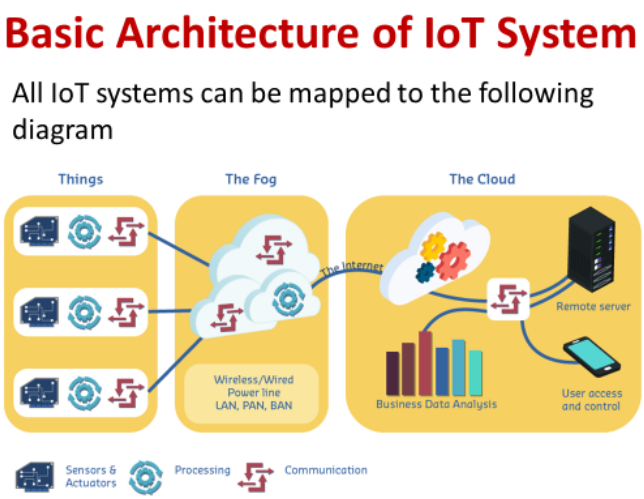

Figure 18: IoT Standard Hardware

The hardware utilized in IoT systems includes devices for a remote dashboard, devices for control, servers, a routing or bridge device, and sensors. These devices

manage key tasks and functions such as system activation, action specifications, security, communication, and detection to support-specific goals and actions.

The most important hardware in IoT might be its sensors. These devices consist of energy modules, power management modules, RF modules, and sensing modules. RF modules manage communications through their signal processing, WiFi, ZigBee, Bluetooth, radio transceiver, duplexer, and BAW.

Figure 19: Sensors

The sensing module manages to sense through assorted active and passive measurement devices. Next, the wearable electronic devices are small devices worn on the head, neck, arms, torso, and feet. Finally, the standard devices like the desktop, tablet, and cellphone remain integral parts of IoT as the command center and remotes. The desktop provides the user with the highest level of control over the system and its settings. The tablet provides access to the key features of the system in a way resembling the desktop, and also acts as a remote. The cellphone allows some essential settings modification and also provides remote functionality. Other key connected devices include standard network devices like routers and switches.

6. IoT standard APIs – Web API & MQTT API

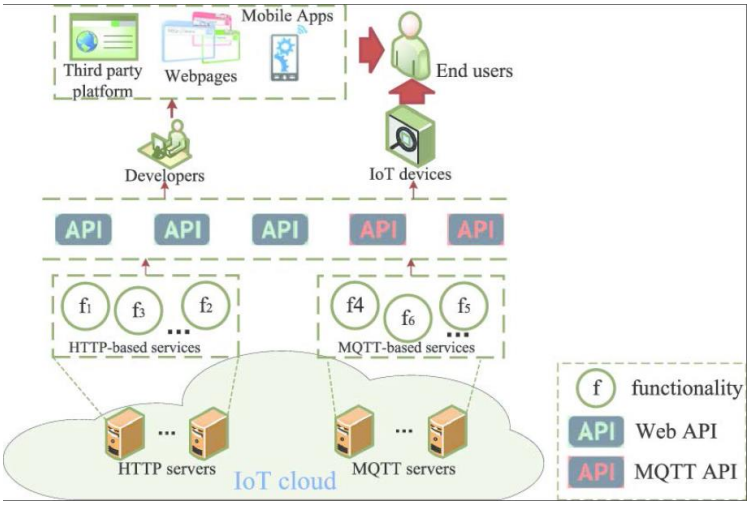

Figure 20: IoT Standard APIs – Web API & MQTT API

The IoT cloud can offer diverse services that help software developers to design and run applications based on the IoT cloud infrastructure or vendors/Original Equipment Manufacturers (OEMs) to produce devices that can access to the IoT cloud. These services are deployed on two kinds of application servers in the IoT cloud, that is, HTTP servers and MQTT servers. Each service contains several different functionalities.

M1 Evaluate the impact of common IoT architecture, frameworks, tools, hardware, and APIs in the software development lifecycle.

With the advent of the Internet-of-Things (IoT), the vision Internet has shifted beyond the World Wide Web (WWW) to a world of connected things, ranging from sensors to actuators, integrated into a variety of daily objects and infrastructures. This reality shift, empowered by emergent network technologies plus cheap and low-powered devices, has opened a State of the Software Development Life-Cycle for the IoT wide range of opportunities both from research and business perspectives, in fields such as industry, city management, and health in addition to the Information and Communication Technologies (ICT) sector.

IoT has already a noticeable impact on our daily lives but several challenges are still open from conception to the maintenance of such systems. In an era of cloud-first design, IoT reality has increased the interest in new architectural approaches as it is observable by the birth of fog and mist/edge computing.

In terms of software engineering body-of-knowledge, approaches such as Visual Programming and Model-driven (Software) Engineering has for long been used to tackle the complexity of software systems, as it is the case of IoT. More recently, work has been pursued on live programming and the use of it as a way to increase

the liveness of software systems development, improving the developer overall experience (e.g. reducing the complexity of detecting and correct bugs).

In IoT scenarios, numerous sensors measure and report several phenomena, and diversified IoT solutions are deployed to collect huge amounts of data. IoT platforms, such as Amazon AWS, IBM Watson, or Microsoft IoT Suite, have been available to aid the development of such services/applications. However, one of the major challenges faced y IoT solutions providers in the supervision and management of a large number of deployed sensors/devices. Presumably, the magnitude and heterogeneity of the IoT systems make it difficult to manage them with conventional IT management tools and techniques. New techniques and tools have to be explored and developed or traditional management solutions have to be adapted to the new challenges.

Internet of Things (IoT) systems and the corresponding network architectures are complex due to distributed services on many IoT devices collaboratively fulfilling common goals of IoT applications. System requirements for different types of IoT application domains are still not well-established. The life cycle view is one of the views used for system architecting, showing different stakeholders' concerns at every stage of the life cycle to derive system requirements. We employ the life cycle view to understand IoT systems in different IoT application domains. It will be the definition of a generic life cycle model for IoT, which is specified by observations on life cycles of existing IoT solutions and generalizations taking into account important IoT functionalities and quality attributes. Such a generic life cycle model for IoT was non-existent. The proposed generic life cycles for IoT devices, IoT services, and IoT applications provide a better understanding of the problems that need to be solved by IoT systems throughout their life cycles.

For example, through the generic life cycle for IoT applications, we understand that IoT applications are not restricted to orchestrating collected data from some database services, as presumed in many IoT discussions. Instead, they can also be configurations in the lower layers of the system, i.e combining sensing and actuating services with application logic on the lower layer devices. The proposed generic life cycles shed light on the problems entailed for realizing these types of IoT applications.

From that, can conclude that the proposed generic life cycles for IoT can help system architects specify and design robust and effective IoT systems. Without taking into account the life cycle needs of an IoT system, desired functionalities and quality attributes for the system are in danger of becoming unrealizable.

The software has a longstanding association with a state of crisis considering its success rate. The explosion of Internet-connected devices, Internet-of-Things will add to the complexity of software systems. The particular characteristics of these systems, such as being large-scale and its heterogeneity, pose increasingly new challenges.

M2 Review specific forms of IoT architecture, frameworks, tools, hardware, and APIs for different problem-solving requirements.

Human beings cannot be happy with any kind of tiredness based work, so they focused on machines to work on behalf of humans. The Internet-based latest technology provides the platforms for human beings to relax and unburden feeling. The Internet of Things (IoT) field efficiently helps human beings with smart decisions through Machine-to-Machine (M2M) communication all over the world.

1. Standard architecture available for use in IoT development

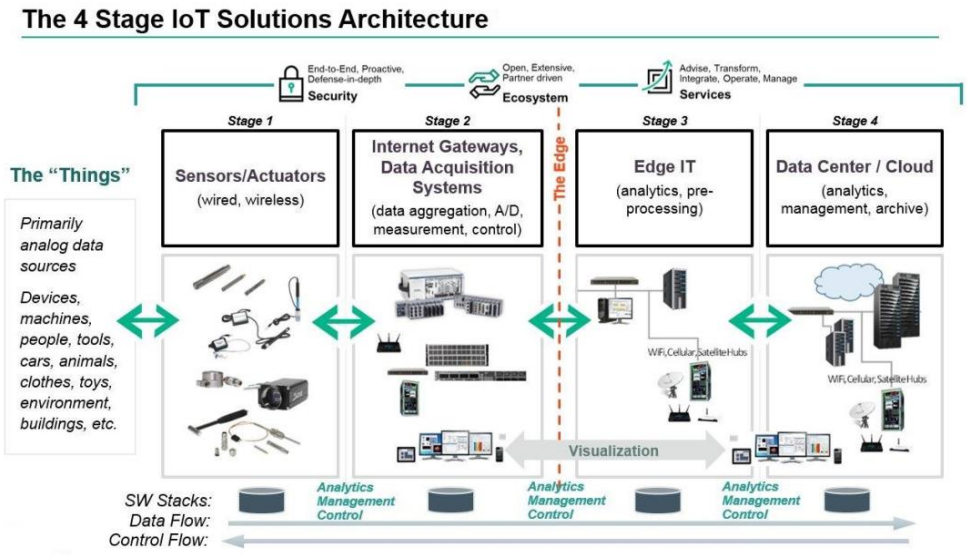

Figure 21: The 4 Stage IoT

Stage 1 of an IoT architecture consists of your networked things, typically wireless sensors and actuators. Stage 2 includes sensor data aggregation systems and analog-to-digital data conversion. In Stage 3, edge IT systems perform preprocessing of the data before it moves on to the data center or cloud. Finally, in Stage 4, the data is analyzed, managed, and stored on traditional back-end data center systems. Clearly, the sensor/actuator state is the province of operations technology (OT) professionals. So is Stage 2, 3, and 4 are typically controlled by IT, although the location of edge IT processing may be at a remote site or nearer to the data center.

The dashed vertical line labeled "the edge" is the traditional demarcation between OT and IT responsibilities, although this is blurring.

Stage 1: Sensors/actuators

Sensors collect data from the environment or object under measurement and turn it into useful data. Think of the specialized structures in your cell phone that detect the directional pull of gravity and the phone's relative position to the “thing” we call the earth, then convert it into data that your phone can use to orient the device. Actuators can also intervene to change the physical conditions that generate the data. For example, an actuator might shut off a power supply, adjust an airflow valve, or move a robotic gripper in an assembly process.

The sensing/actuating stage covers everything from legacy industrial devices to robotic camera systems, water-level detectors, air quality sensors, accelerometers, and heart rate monitors. And the scope of the IoT is expanding rapidly, thanks in part to low-power wireless sensor network technologies and Power over Ethernet, which enable devices on a wired LAN to operate without the need for an A/C power source.

In an IoT architecture, some data processing can occur in each of the four stages. However, while you can process data at the sensor, what you can do is limited by the processing power available on each IoT device. Data is at the heart of an IoT architecture, and you need to choose between immediacy and depth of insight when processing that data. The more immediate the need for information, the closer to the end devices your processing needs to be.

Stage 2: The Internet gateway

The data from the sensors start in analog form. That data needs to be aggregated and converted into digital streams for further processing downstream. Data acquisition systems (DAS) perform these data aggregation and conversion functions. The DAS connects to the sensor network, aggregates outputs, and performs the analog-to-digital conversion. The Internet gateway receives the aggregated and digitized data and routes it over Wi-Fi, wired LANs, or the Internet, to Stage 3 systems for further processing.

The reason needs to preprocess the data because the analog data streams that come from sensors create large volumes of data quickly. The measurable qualities of the physical world in which your business may be interested like motion, voltage, vibration, and more can create voluminous amounts of constantly changing data. Think about how much sensor data a complex machine like an aircraft engine might

generate in one day, and there's no theoretical limit to the number of sensors that could be feeding data into an IoT system. What's more, an IoT system is always on, providing continuous connectivity and data feeds. IoT data flows can be immense (I've seen as much as 40 TB/second in one case. That's a lot of data to transport into the data center. It's best to preprocess it).

Intelligent gateways can build on additional, basic gateway functionality by adding such capabilities as analytics, malware protection, and data management services. These systems enable the analysis of data streams in real-time. Although delivering business insights from the data is a little less immediate at the gateway than it would be when sent directly from the sensor/actuator zone, the gateway has the compute power to render the information in a form that is more understandable to business stakeholders.

Stage 3. Edge IT

Once IoT data has been digitized and aggregated, it's ready to cross into the realm of IT. However, the data may require further processing before it enters the data center. This is where edge IT systems, which perform more analysis, come into play. Edge IT processing systems may be located in remote offices or other edge locations, but generally, these sit in the facility or location where the sensors reside closer to the sensors, such as in a wiring closet. Because IoT data can easily eat up network bandwidth and swamp your data center resources, it's best to have systems at the edge capable of performing analytics as a way to lessen the burden on core IT infrastructure. If you just had one large data pipe going to the data center, you'd need enormous capacity. You'd also face security concerns, storage issues, and delays processing the data. With a staged approach, you can preprocess the data, generate meaningful results, and pass only those on.

Stage 4. The data center and cloud

Data that needs more in-depth processing, and where feedback doesn't have to be immediate, get forwarded to the physical data center or cloud-based systems, where more powerful IT systems can analyze, manage, and securely store the data. It takes longer to get results when you wait until data reaches Stage 4, but you can execute a more in-depth analysis, as well as combine your sensor data with data from other sources for deeper insights. Stage 4 processing may take place on-premises, in the cloud, or in a hybrid cloud system, but the type of processing executed in this stage remains the same, regardless of the platform.

2. Frameworks available for use in IoT development

Figure 22: Frameworks

An IoT framework is a middleware layer beneath one or more IoT applications that presents a network-facing application interface through which peer framework nodes interact. Frameworks often support multiple communication technologies and message passing techniques. IoT frameworks also expose security capabilities including hardware-roots-of-trust to applications and peer framework nodes. Like you know, the IoT isn't a single element. It's an ecosystem – an infrastructure of connected devices communicating with each other over the internet. When massive data generation and transmission across diverse devices, there has to be one place where everything is collated. This collation is essential to make sense of the data generated. This is where IoT frameworks come in. In simpler words, IoT frameworks are the components that make data transmission seamless. The fundamental components of IoT frameworks include Hardware Devices (sensors, controllers, micro-controllers, and other hardware devices), Software Applications (involves written applications to configure controllers and operate them from remote and do more), Cloud and Communication Platforms (an inevitable part of IoT over which all communications happen), Cloud Applications (written applications that bind local hardware devices and cloud-based devices). Bellow is Top 10 Open Source IoT Frameworks.

KAA

Is one of the most efficient and rich open-source IoT cloud platform where anyone has a free way to materialize their smart product concepts. On this platform, you can manage an unlimited number of connected devices with cross-device interoperability. You can achieve real-time device monitoring with the possibility of remote device provisioning and configuration. It is one of the most flexible IoT platforms for your business which is fast, scalable, and modern.

MACCHINA.io

Macchina.io IoT platforms provide a web-enabled, modular, and extensible JavaScript and C runtime environment for developing IoT gateway applications. It also supports a wide variety of sensors and connection technologies including Tinkerforge, bricklets, Xbee, and many others including accelerometers. This platform able to develop and deploy device software for automotive telematics and V2X, building and home automation, industrial edge computing and IoT gateways, smart sensors, or energy management systems.

ZETTA

Are a server-oriented platform that has been built around NodeJS, REST, and a flow based reactive programming development philosophy linked with the Siren hypermedia APIs. They are connected with cloud services after being abstracted as REST APIs. These cloud services include visualization tools and support for machine analytics tools like splunk. It creates a zero-distributed network by connecting endpoints such as Linux and Arduino hacker boards with platforms such as Heroku.

GE PREDIX

As a service software for industrial IoT is based on the concept of cloud foundry. It adds asset management, device security, and real-time, predictive analytics that also supports heterogeneous data acquisition, access, and storage. GE Predix was developed by GE for its own operations and consequently has become one of the most successful of the enterprise IoT platforms and with the recent partnering of GE and HPE, the future looks even better.

ThingSpeak

Is another IoT platform that lets you analyze and visualize the data in MATLAB and eliminates the need to buy a license for the same. It helps you to collect and store sensor data in private channels while giving you the freedom to share them in public channels. It works with Arduino, particle photon and electron, and many more applications. It is used mostly for sensor logging, location tracking and alerts, and analysis. It also has a worldwide community which is quite helpful in itself.

DeviceHive

Is yet another feature-rich open-source IoT platform that is currently distributed under the Apache 2.0 license and is free to use and change. It provides Docker and Kubernetes deployment options and can be downloaded and use it with both public and private cloud. It allows you to run batch analytics and machine learning on top of

your device data and more. Various libraries, including Android and iOS libraries, are supported in DeviceHive.

Distributed Services Architecture

Is an open-source IoT that unifies the separate devices,
services, and applications in the structured and real-time data model and facilitates decentralized device inter-communication, logic, and applications. Distributed service links is a community library that allows protocol translation and data integration to and from 3rd part data sources. All these modules are lightweight making it more flexible in use. It implements DSA query DSL and has inbuilt hardware integration support.

Eclipse

Is built around the java/OSGi-based Kura API container and aggregation platform for M2M applications running on service gateways. Kura is a framework based on Eurotech's everywhere cloud IoT framework and is often integrated with the Apache Camel. Some of its major sub-projects include the PAho messaging protocol framework and the Eclipse SmartHome framework.

Open Connectivity Foundation

Is an amalgamation of the intel and Samsung backed open
interconnect consortium organization and the UPnP forum which is working hard to become the leading open-source standard group for IoT and its OCF IoTivity depends on RESTful, JSON, and CoAp. OIC was created in July 2014. The first version of OCF 1.0 was released in September 2015 for the core framework, smart home device, resource type, security, and remote access capabilities.

OpenHAB

Is capable of running on any device that is capable of running a JVM. All the IoT technologies are abstracted by the modular stack into "items", and offer rules, scripts, and supports for persistence (the ability to retain device states for a period of time). It offers a variety of web-based UIs and is supported by major Linux hacker boards. It is deployed on-premise and connects to devices and services from different vendors.

3. Tools available for use in IoT development

Tessel 2

This is a hardware provider that can be used to build basic IoT solutions and prototypes. Tessel 2 lends a helping hand through its numerous sensors and modules. This is a board that can hold up to a dozen modules including the RFID, camera, GPS, and accelerometer. All those Java developers who are proficient with Node.JS can use this device as Tessel can be programmed using Node.JS. This way, Tessel can be used to churn out a host of server and hardware firmware IoT solutions. Tagged as a robust IoT platform, you can leverage all the libraries of Node.JS to unveil a host of devices; within a matter of minutes. It comes with two processors, the Tessel hardware makes use of a 580MHz Mediatek MT7620n and 48MHz Atmel SAMD21 coprocessor. While one can be used to run your firmware applications at a rapid speed, the other finds its use in exercising better input/output control and the efficient management of power.

Eclipse IoT

If you as an IoT developer are ordained to build IoT devices, Cloud platforms, and Gateways, then Eclipse IoT can be your top bet. Recognized as a collaboration of various companies and individuals who are striving towards the development and establishment of IoT open technologies, Eclipse IoT can make all your IoT dreams come true. Allowing you to develop, promote, and adopt open source IoT technologies, Eclipse IoT is an instrument that can help you gain technical expertise. Simply look into the vast assembly of services and projects delivered by the Eclipse team and you are all covered.

Arduino

If you are looking to build a computer that can sense and exercise better control over the physical world when compared to your normal stand-alone computer, then Arduino can be your intelligent choice. Offering a perfect blend of IoT hardware and software, Arduino is an open-source prototyping and simple-to-use IoT platform. Arduino operates through a set of hardware specifications that can be applied to interactive electronics. The software of Arduino comes in the form of the Arduino programming language and Integrated Development Environment (IDE).

PlatformIO

Next in the list of top IoT development tools and platforms is PlatformIO which is a cross-platform IoT development environment. This platform comes with a build system, supported by a library manager and IDE. You have a choice to port the IDE on top of the

Atom editor or you can go ahead and install it as a plug-in. The best part of PlatformIO is that it is compatible with more than 200 boards. Coming with a wonderful debugging integration, PlatformIO is conspicuous of a serial port monitor. All those who are employing PlatformIO unanimously express one feeling and that is – "PlatformIO hastens up the development process of an IoT application, allowing us to deliver IoT solutions in record time".

IBM Watson

Last in the list of top IoT development tools, but definitely not the least is IBM Watson, an API that allows you to attach a host of cognitive computing features to your IoT applications. This is an innovative tool that can also be used to predict the future. Simplifying the tasks of IoT developers, IBM Watson through its numerous services helps to unveil chatbots that can understand the natural language. These chatbots can then be deployed on to messaging platforms and websites which can be viewed on various devices. It is through IBM Watson that IoT developers can successfully and swiftly build cognitive search and content analytics engines.

Raspbian

This IoT IDE is created for Raspberry Pi board by IoT tech enthusiasts. With over 35,000 packages and numerous examples along rapid installation with the use of pre-compiled software makes it an essential IoT development tool. Often, it is regarded as the best tool for Raspberry Pi for IoT app development. Another best part about this tool is that it is under constant development and has widened the reach for computing so that users can gain maximum benefits.

OpenSCADA

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

Node-RED

Is a simple visual tool that is built on Node.js, a server-side JavaScript

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

Kinomo Create

It is a device that can connect two devices without extensive knowledge of JavaScript. Kinoma Create consists of everything which is required for developing small IoT applications like connecting light, temperature, or movement sensors for a specific purpose with mobile notifications in case of any alterations. You can also find numerous

tutorials on its practical applications at Kinoma's website. It has been widely used for developing a synthesizer, a camera trap that takes a photo of an animal or any other objects disturbs the laser beam and an automatic alarm bell for alerting individuals in case of any situations.

Device Hive

Is a free open-source machine to machine (M2M) communication framework which was launched in 2012. It is a Data Art's AllJoyn based device and is considered one of the most preferred IoT app development platforms. It is generally a cloud-based API that you can control remotely without the need for network configuration. The same implies to the libraries, portals, and management protocols. Mostly, it is used for security, automation, smart home tech, and sensors. Additionally, it also has a dynamic community and myriads of online resources available to help you out.

Home Assistant

It is an open-source tool that is primarily aimed at home automation and functions with a Python-based coding system. The IoT system developed with this tool can be easily controlled with mobile or desktop browsers. Furthermore, its setup is easy and is trusted for operations, security, and privacy. The software supports any systems which are running on Python 3 and all the systems get regular updates within 2 weeks. Despite the lack of cloud components, its ability to protect the data during this internet age gives it an edge over others.

4. Hardware available for use in IoT development

IoT Hardware includes a wide range of devices such as devices for routing, bridges, sensors, etc. These IoT devices manage key tasks and functions such as system

activation, security, action specifications, communication, and detection of support-specific goals and actions. IoT Hardware components can vary from low-power boards; single-board processors like the Arduino Uno which are basically smaller boards that are plugged into mainboards to improve and increase its functionality by bringing out specific functions or features (such as GPS, light and heat sensors, or interactive displays). A programmer specifies a board's input and output, then creates a circuit design to illustrate the interaction of these inputs and outputs.

Figure 23: Hardware Arduino Uno

Another well-known IoT platform is Raspberry Pi 2, which is a very affordable and tiny computer that can incorporate an entire web server. Often called "RasPi," it has enough processing power and memory to run Windows 10 on it as well as IoT Core. RasPi exhibits great processing capabilities, especially when using the Python programming language.

Figure 24: Hardware Raspberry Pi 2

BeagleBoard is a single-board computer with a Linux-based OS that uses an ARM processor, capable of more powerful processing than RasPi. Tech giant Intel's Galileo and Edison boards are other options, both great for larger-scale production, and Qualcomm has manufactured an array of enterprise-level IoT technology for cars and cameras to healthcare.

Figure 25: Hardware BeagleBoard

5. APIs available for use in IoT development

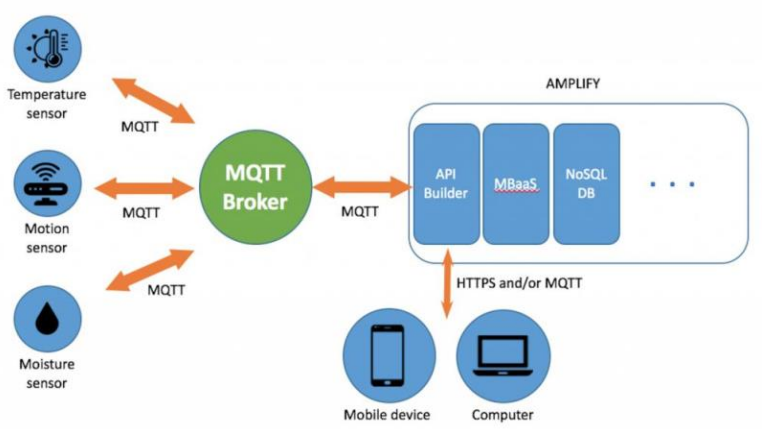

Figure 26: How to MQTT Work

An Application Programming Interface (API) is an interface used by programs to access an application. It allows a program to send commands to other programs and receive responses back from that application. Although HTTP based APIs are common on the web. However, MQTT is becoming the main IoT protocol, and we will likely see a large growth of MQTT based APIs. All of the main IoT providers like Amazon, Azure, and IBM provide both HTTP and MQTT APIs to access their services. MQTT offers some advantages, as it was designed to work in environments that want to send small portions of data frequently like temperature, humidity, and others but also allowing to run on systems with less computer or memory. On the other hand, the protocol has some constraints in terms of security which need to be taken into account too. IoT devices leveraging MQTT over Websockets can be considered similar to new and modern browsers like Chrome supporting Websockets. This allows exchanging data in a quick and unconstrained way compared to Webhooks or API polls.

LO2 Outline a plan for an appropriate IoT application using common architecture, frameworks, tools, hardware and APIs

P3 Investigate architecture, frameworks, tools, hardware and API techniques available to develop IoT applications

1. IoT standard architecture – Application layer – Application protocols

The application protocols such as HTTP and XMPP traditionally found in the Application layer. However, these are not suitable options for communication in an IoT constrained environment with a large number of connected devices. Therefore,

more lightweight protocols are required by the IoT industry to address this problem. Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT) are the two most popular IoT application protocols.

Application Type	Application-layer protocol	Transport Protocol
Electronic mail	Send: Simple Mail Transfer Protocol SMTP [RFC 821]	TCP 25
	Receive: Post Office Protocol v3 POP3 [RCF 1939]	TCP 110
Remote terminal access	Telnet [RFC 854]	TCP 23
World Wide Web (WWW)	HyperText Transfer Protocol 1.1 HTTP 1.1 [RFC 2068]	TCP 80
File Transfer	File Transfer Protocol FTP [RFC 959]	TCP 21
	Trivial File Transfer Protocol TFTP [RFC 1350]	UDP 69
Remote file server	NFS [McKusik 1996]	UDP or TCP
Streaming multimedia	Proprietary (e.g., Real Networks)	UDP or TCP
Internet telephony	Proprietary (e.g., Vocaltec)	Usually UDP

Figure 27: Application Layer Protocols

2. IoT standard architecture – Application layer – Constrained Application protocols – CoAP

Constrained Application Protocols (CoAP) is based on HTTP protocols and were designed by the IETF Constrained RESTful Environment (CoRE) working group. The communication between HTTP client and server is performed by using the Representational State Transfer (REST) standard. REST is a resource-demanding standard, the lightweight RESTful interface was designed in CoAP so that resource-constrained devices in IoT network can use RESTful services. Resources are retrieved from the server using URLs/URLs. CoAP message exchange is done using UDP (User Datagram Protocol) with a strong security measure, through the utilization of the Datagram Transport Layer Security (DTLS). Both IPv4 and IPv6 are supported by CoAP. However, IPv6 is used in IEEE 802.15.4 networks for constrained devices.

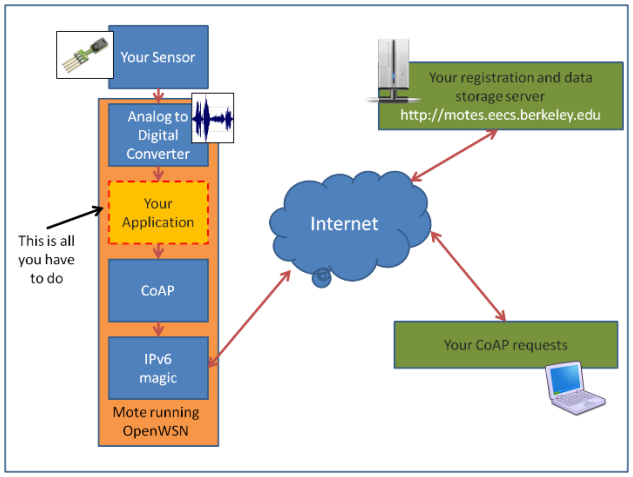

Figure 28: Constrained Application Protocols

3. IoT standard architecture – Application layer – Application protocols – MQTT

In 1999, Message Queuing Telemetry Transport (MQTT) was introduced by IBM, and by 2013 it was standardized by OASIS. MQTT is a lightweight protocol specifically suitable for constrained environments. For example, a harsh environment with low bandwidth connections, such as in the oil and gas industries. MQTT protocol is the publish-subscribe communication framework between the nodes. The three components of the MQTT network are a publisher (client), a subscriber (client), and a message broker (server).

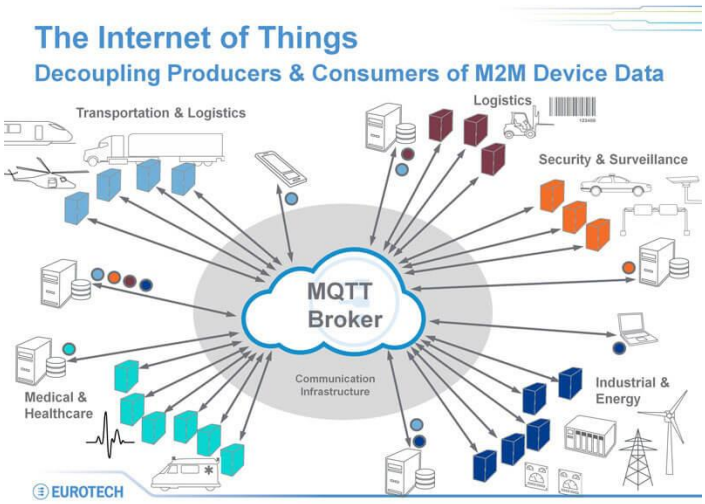

Figure 29: Message Queuing Telemetry Transport (MQTT)

4. IoT framework – IOT Design Framework Layers and Design Framework example (Quad-copter)

The product Infrastructure layer includes the Hardware & Software of the product. The sensor layer has sensors & the networks that support them. Connectivity Layers

is a communication protocol necessary to send information from the product to the cloud. The analytics layer will be translating the sensor data into meaningful information. Finally, the smart apps layer will integrate all of the other layers together to supports different business decisions.

Quad-copter includes 4 motors, a circuit board that forms the structure, a video camera, and an antenna for transmitting sensor data. The product infrastructure layer includes hardware components is propellers in the frame and software components built into the quadcopter such as the control algorithms that enable the flight to sensor components. The sensors layer includes a gyroscope that provides orientation information to the control algorithms and a camera to provide visual data. The connectivity layer will control the quad-copter remotely by use radio communication protocol connect to a gateway computer to stream sensor data. The analytics layer will convert the orientation data from the gyroscope into motor speeds to adjust for optimal flight. Finally, the smart apps layer includes two different apps are one app is used to display video and another to control sensor data.

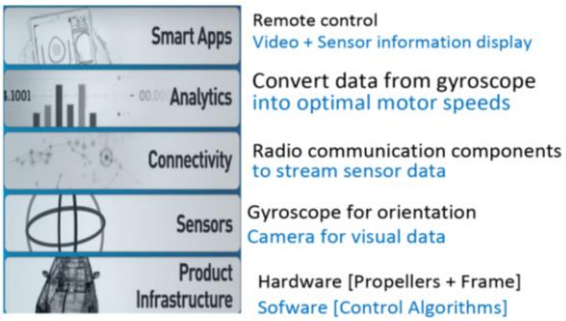

Figure 30: Design Framework for Quad-copter

5. IoT tools – Tinkercad: example

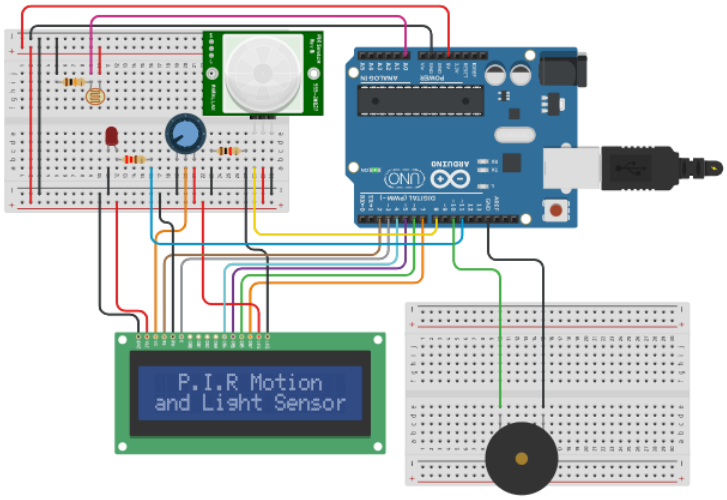

Figure 31: Example for Tinkercad

6. IoT framework – Cisco packet tracer: example

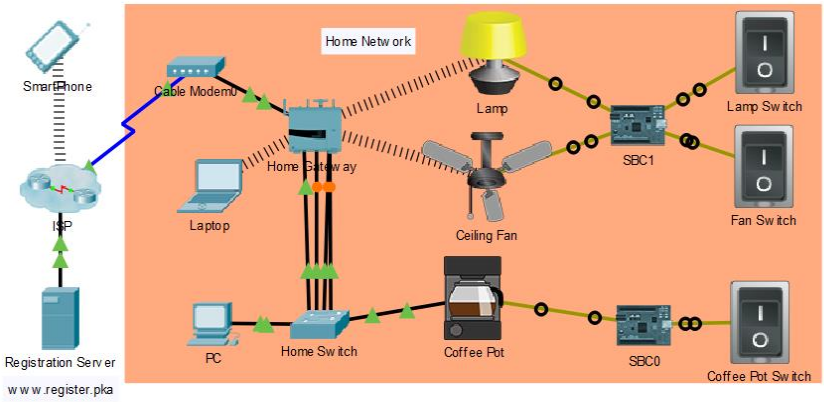

Figure 32: Example for Cisco Packet Tracer

7. IoT standard hardware – Sensors & Actuators: example

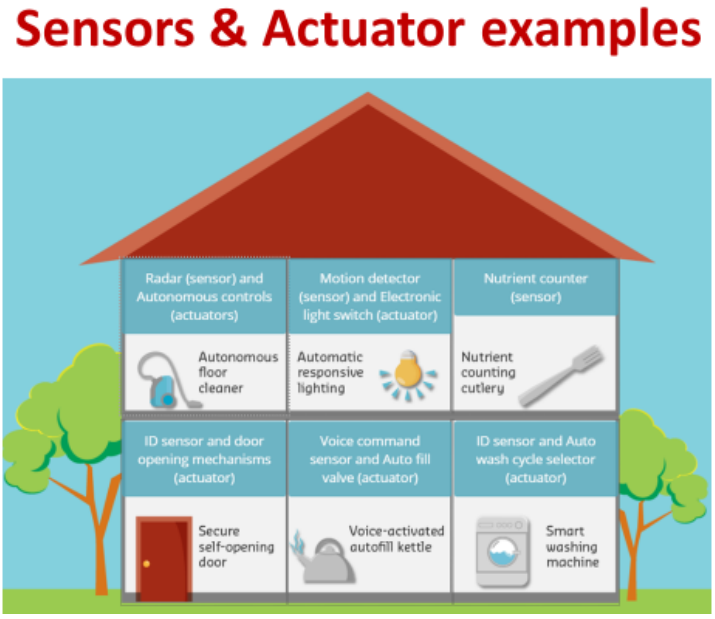

Figure 33: Example for Sensors & Actuators

8. IoT standard hardware – The Fog

Fog computing is known as edge computing, provides a way to gather and process data at local computing devices instead of in the cloud or at a remote data center. Under this model, sensors and other connected devices send data to a nearby edge computing device. This cloud be a gateway device, such as a switcch or router, that processors and analyzes this data.

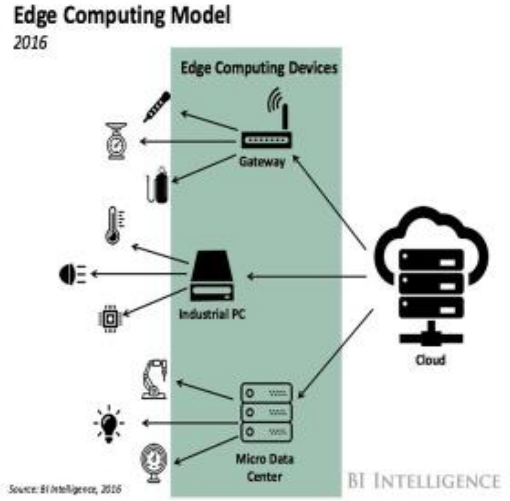

Figure 34: Edge Computing Model

9. IoT standard hardware – The Cloud

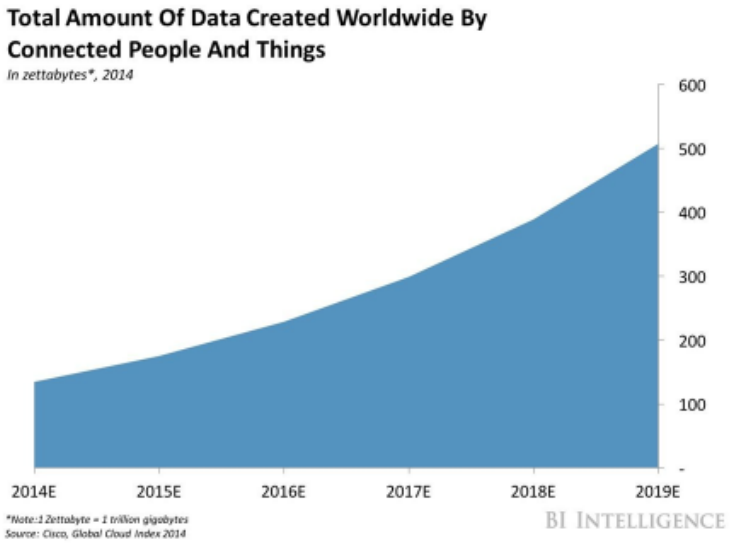

Figure 35: Total Amount of Data Created Worldwide by Connected People and Things

Cloud computing will be a major part of big data, especially by making all of the connected devices work together.

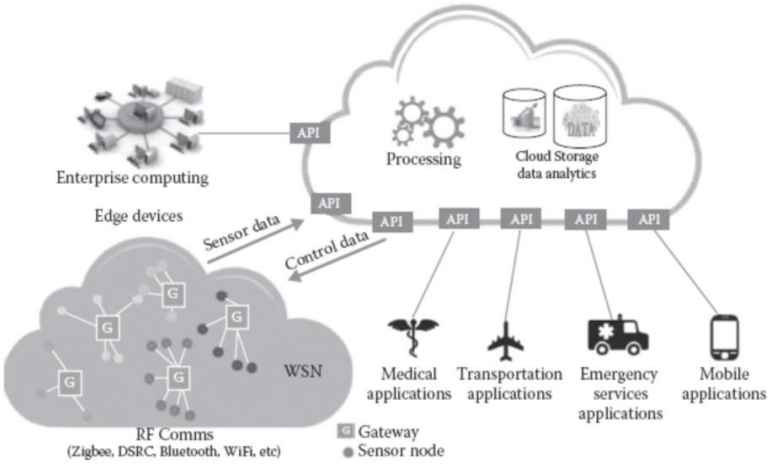

Figure 36: Cloud Storage

Cloud computing, often called simply "the cloud," involves delivering data, applications, photos, videos, and more over the Internet to data centers. IBM has helpfully broken down cloud computing into six different categories are Software as a service (SaaS), Platform as a service (PaaS), Infrastructure as a service (IaaS), Public Cloud, Private Cloud and Hybrid Cloud. Cloud computing and the IoT both serve to increase efficiency in our everyday tasks, and the two have a complimentary relationship. The IoT generates massive amounts of data, and cloud computing provides a pathway for that data to travel to its destination.

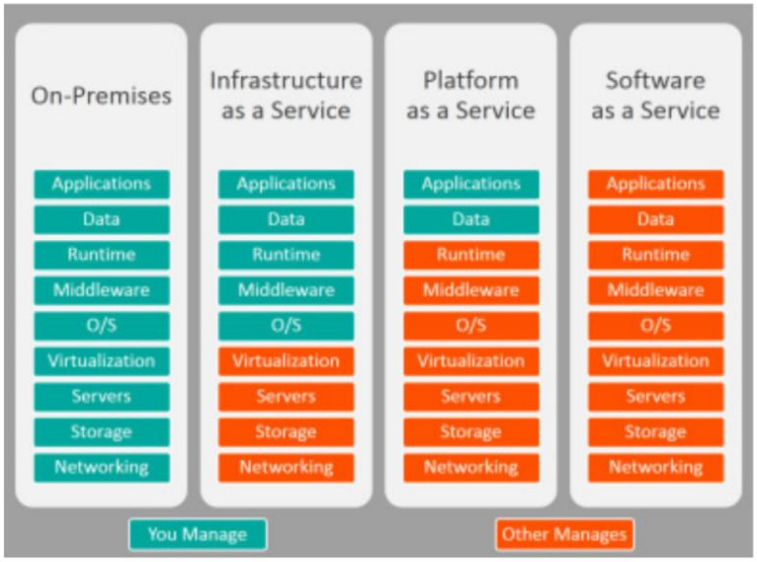

Figure 37: Cloud

10. IoT standard APIs – CoAP vs MQTT APIs

CoAP versus MQTT

FACTOR	CoAP	MQTT
Main transport protocol	**UDP**	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	Many-to-many
Strengths	**Lightweight** and **fast**, with **low overhead**, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages.	**TCP** and **multiple QoS options** provide robust communications; simple management and scalability using a broker architecture.
Weaknesses	**Not as reliable** as TCP-based MQTT, so the application must ensure reliability.	**Higher overhead** for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support.

Figure 38: CoAP vs MQTT APIs

MQTT and CoAP are both useful as IoT protocols, but have fundamental differences. MQTT is a many-to-many communication protocol for passing messages between multiple clients through a central broker. It decouples producer and consumer by letting clients publish and having the broker decide where to route and copy messages. While MQTT has some support for persistence, it does best as a communications bus for live data. CoAP is, primarily, a one-to-one protocol for transferring state information between client and server. While it has support for

observing resources, CoAP is best suited to a state transfer model, not purely event based. MQTT clients make a long-lived outgoing TCP connection to a broker. This usually presents no problem for devices behind NAT. CoAP clients and servers both send and receive UDP packets. In NAT environments, tunnelling or port forwarding can be used to allow CoAP, or devices may first initiate a connection to the head-end as in LWM2M. MQTT provides no support for labelling messages with types or other metadata to help clients understand it. MQTT messages can be used for any purpose, but all clients must know the message formats up-front to allow communication. CoAP, conversely, provides inbuilt support for content negotiation and discovery allowing devices to probe each other to find ways of exchanging data. Both protocols have pros and cons, choosing the right one depends on your application.

P4 Determine a specific problem to solve using IoT

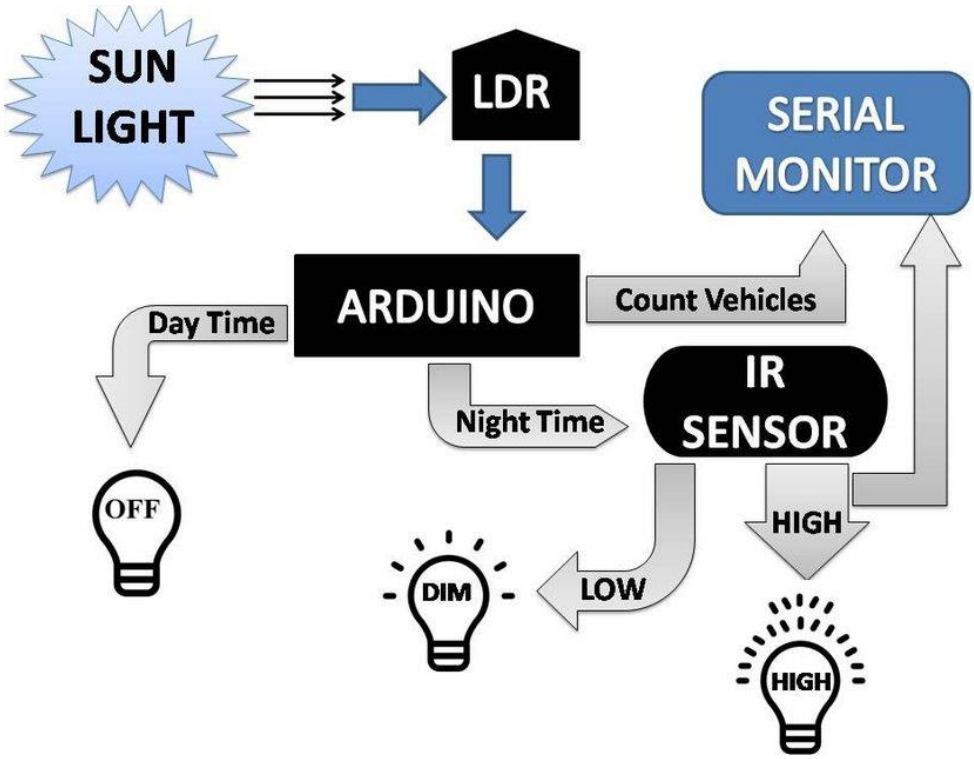

Figure 39: Design Thinking

Now, we want to develop an application for IoT. This app is designed to help reduce the electrical energy wastage caused by people forgetting or being lazy to turn off the lights. In addition, it can also help people with mobility restrictions such as the elderly, people with disabilities, etc. From that, we can come up with the following question table.

Table 1: Questions for Defining a Problem

What is the problem?	It is usually the case at aither public place or at home that people turn on the light for their usuage. But after that, they will sometimes forget to turn off the light. This will leave the light keep running and waste a lot of energy. On the other, it is usefull for people that do not want to bother turning off the light themselves so it can be quite convenient.
Who has the problem?	Everyone who uses light bulb regularly.
How do they interact with their environment?	The light will turn on/off depend on how dark the room is and when someone enter a room or a place.
What cause the problem?	It is not quite convenient to turn off all light when someone is in a rush. Or sometimes they forget to turn off the light.
When does the problem occur?	When the light is used.
Can you turn this problem into a question?	How to turn on/off the light automatically?

After defining the problem, we also conduct the following table in research for available solutions.

Table 2: Research on Available Solutions

	Smart Light	**Sensor Light**
Pros	The light can connect to WI-Fi Bluetooth, remotely be controlled, consumes less energy, and the higher expectancy	Automatically detects motion to turn on/off, can identify day/night and only illuminate at night, requires no electrician for installation, turn off light when no motion is detected after some fixed amount of time
Cons	Has to be remotely controlled through phone and cannot turn on/off automatically	Doesn't have a switch for manually turn on/off when needed and can only illuminate at night
How is it relevant, innovative, and/or	It is relevant and innovative to the problem defined as it is also about the light that tries to eliminate the means to use	It is a suitable solution for the problem as it uses a motion sensor to do automatic lighting

disruptive?	switches	
What are the limits of this solution?	As said, it limits only to be able to remotely control the light through a phone. It isn't able to detect motion in a room to provide automatic lighting.	It doesn't have a switch to manually turn on/off light. It should have a way to set a timeframe where the automatic process is turned off.

We want to employ the second solution but with a bit of modification and the ability to remotely control from the phone like the first solution. But we lack sufficient modules to do so (namely ESP8266).

The solution for this problem is to use a motion sensor and a photoresistor. First, the motion sensor will be used to see if someone has walked in the room. Next, the photoresistor is used to see the current brightness of the room. If the room is already fairly bright, there is no need to turn on the light. But if it is, then the light is turned on. When the light is turned on, after some fixed amount of time, it will tue off if there is no one in the room. And will also check the time, if it is with a certain timeframe, the light will turn off and the automatic light process will stop. At this point, everything needs to be done manually by the users. This will ensure the light not automatically turned on when they are sleeping. Summary, the requirements for this application is: First, turn the light on when there is a person in the room and the current brightness of the room is low (not turn it on when the brightness is high). Next, after some fixed amount of time, if no one is in the room, then it is turned off. Next, when the automatic process is stop, the light is turned will off within certain timeframe. Finally, there should be a switch for the light bulb so that the users can turn on/off manually.

M3 Select the most appropriate IoT architecture, frameworks, tools, hardware, and API techniques to include in an application to solve this problem.

1. Investigate architecture

Regardless of the use case, nearly every IoT solution involves the same four components are devices, connectivity, platform, and application. Some use cases may involve additional layers, but these four components represent the foundation of every IoT solution.

IoT DEVICES

IoT devices make up the physical hardware component of your solution. In an Industrial Equipment Monitoring use case, these are things like engines and engine controllers. For Smart Environment use cases, these could be motion sensors or badge readers. For Asset

Tracking use cases, these are GPS trackers. In many industrial IoT and smart building use cases, customers prefer to use off-the-shelf devices that can be added to an existing environment or piece of equipment. One of the challenges when choosing off-the-shelf hardware can be gaining access to the data. Many vendors provide siloed solutions, which may work well to solve very specific problems, but don't work well when you'd like to utilize that data as part of a broader IoT application. When investigating hardware solutions, make sure you can gain access to the data through local protocols like Modbus, Serial, or OPC UA. Some vendors may also have a cloud service where you can access the data through an API.

IoT CONNECTIVITY

In most IoT solutions, devices are sending state data to and receiving commands from a centralized IoT Platform. There are a lot of options for how that device-to-platform connection is made and it depends heavily on the environment and constraints of the device itself. If the device is outside and moving around, like in asset tracking use cases, cellular connectivity is a good choice. If the device is indoors in a home or building environment, you may have Ethernet or WiFi available. If the device is battery powered, you may need to investigate low-power options like Bluetooth Low Energy or LPWAN. Below is a table from our guide, Business Innovation, and Expansion with Industrial IoT, that can help you determine the available connectivity options based on your environment and device constraints.

CONNECTIVITY CHART

	ETHERNET	CELLULAR	WiFI	LPWAN
LOW POWER REQUIREMENT (LONG-TERM BATTERY)				X
EASY TO MOVE AND TRANSFER DEVICES		X		
HIGH-POWER EQUIPMENT (PLUGGED IN OR CHARGED)	X		X	
INDOOR	X		X	X
OUTDOOR		X		X
HIGH-DATA DEMAND	X		X	
LOW-DATA DEMAND				X

Figure 40: Connectivity chart

IoT GATEWAYS AND EDGE COMPUTE

Some devices can't connect directly to the central platform and require the use of an IoT gateway to bridge the gap between your local environment and your platform. This is common in industrial environments where you may be interfacing with existing equipment over local protocols like Modbus, OPC UA, or Serial. Gateways are also required when using wireless technologies like BLE and LPWAN since those don't provide any direct connection to your network or the Cloud. In these situations, the device is connected to the gateway. The gateway reads the required information and then sends the data to the platform using a "backhaul" connection, like cellular or WiFi, which can access your network or the Cloud. Gateways also allow you to introduce Edge Compute into your architecture. Edge Compute shifts processing and control from the Cloud and places it at or near your equipment. The Cloud is an important component of your architecture, but it comes with limitations outside of your control, like internet connection reliability and communication latency. If decisions need to be made in real-time, or your devices generate too much data to send to the Cloud, introducing Edge Compute into your IoT architecture could be a good solution.

IoT PLATFORM

Your IoT Platform is the central data warehouse and orchestration engine for your solution. Building a secure and scalable platform is not an easy task, so we recommend choosing a partner to provide this component of the IoT architecture for you. Choosing the right platform can be a complicated process. Below is the platform evaluation checklist that can be found in our guide, IoT Implementation From Concept to Production: Industry knowledge, IoT knowledge, Quality and level of support available, Limits on devices, payloads, or data, Stability, Functionality meets the use case needs, Security, High availability/disaster recovery, Continuing education, and Documentation.

ARCHITECTURE OF AN IoT PLATFORM

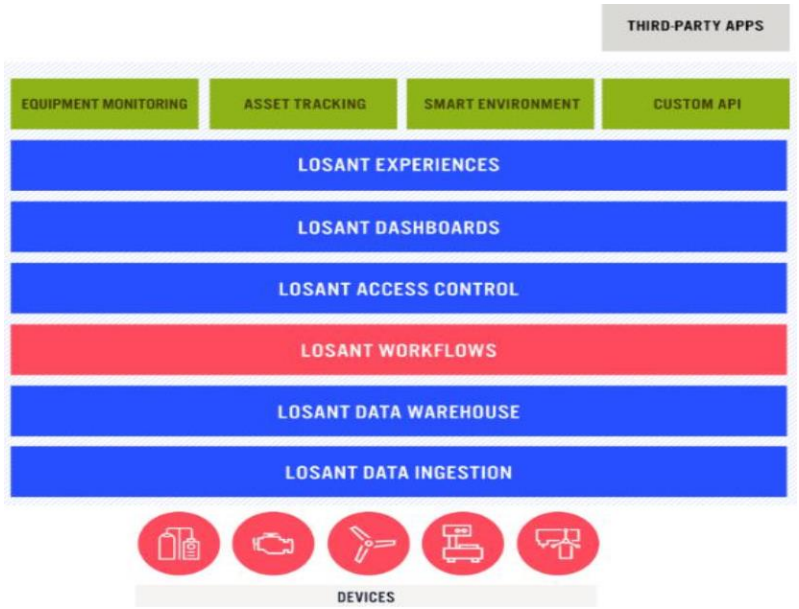

Figure 41: Third-party apps

IoT platforms are complicated enough that they have their own architecture requirements in order to fulfill the needs of your IoT solution. Understanding a well-designed platform can help you when choosing a partner. The above diagram shows the architecture for the Losant Enterprise IoT Platform. This diagram shows the primary components you should look for when investigating a platform. These include Edge Compute, Data Ingestion Services, Data Warehousing, Workflows or Rules Engines, Dashboards, and End-User Experiences. The green boxes at the top represent your specific IoT applications. A well-designed platform can enable the creation of many different applications that your organization may be pursuing.

IoT APPLICATION

The IoT application delivers the end-user experience, or how you or your customers engage with the data collected from your IoT devices. This could be a mobile app, a website, a desktop application, or even a passive experience that no one interacts with directly. Building these can be challenging and time-consuming. When you're investigating IoT platforms, look specifically for application enablement platforms, since they provide tools that can greatly accelerate that effort. The IoT application is an important component of your IoT architecture and is where the actual value of your solution is realized. Deploying hardware and collecting sensor data is meaningless unless you're presenting it in useful ways and solving real-world problems for your customers.

2. Investigate Framework

IoT frameworks have four primary design goals include reduce development time and bring IoT solutions to market sooner; reduce apparent complexity of deploying and operating an IoT network; improve application portability and interoperability; and improve serviceability, reliability, and maintainability. Given the vast range of existing and emerging communication technology choices, it is untenable for applications to manage the combinations of possible ways to connect. Frameworks hide connectivity complexity beneath a higher-level message passing abstraction like REST and publish-subscribe. Standards organizations help achieve these goals through standardization of the framework layer interconnect, message passing interface definition, and data definitions leveraged by applications. Standards groups also document IoT system design principles, architecture, and interconnect options. Standards organizations and industry consortia may assist developers by supplying and certifying reference implementations that include source code. Reference code helps streamline development by providing implementations that pass compliance tests and correctly interpret standards specifications.

Reference codebases are easier to maintain benefiting from a large diverse community of open source developers who cooperate by actively developing code and improving the codebase. Frameworks simplify IoT networks by creating an abstraction of the IoT device networks that hides much of the underlying complexity while exposing data, interfaces, and functions that facilitate interoperation. All it should take to develop an IoT application is to create an application in a high-level language such as Node.js that utilizes framework APIs. The framework provides a semantically rich description of IoT nodes, objects, and interactions that allow IoT network designers to focus only on node interaction semantics rather than on the details of connectivity. Frameworks facilitate improved application portability. This can be achieved at different levels. The bottom layer of the framework is operating system specific. The top layer of the framework is IoT use case-specific in that it exposes a data model abstraction that reinforces an IoT usage context.

Some examples include lighting control, home automation, health monitoring, entertainment, process automation, industrial control, and autonomous control. IoT applications can be developed once given the framework abstraction and can execute on any OS the framework is ported to. The details of dissimilar OSs and platforms can be hidden where porting of framework code to another OS (source code–level compatibility) can happen independently of application development.

Binary compatible platforms can migrate compiled framework code across platforms using the same binary. Platforms that are not binary compatible may rely on virtualization to host framework images or may rely on device management services that hide the complexity associated with paring and installing the right framework with the correct platform. Frameworks enable interoperable devices in heterogeneous environments. Consider a hypothetical scenario where devices are running different OSs and HW platforms. These devices could be built by different platform vendors using silicon from multiple vendors running different OSs such as Windows IoT Embedded and VxWorks running different middleware stacks. This is a perfect storm scenario for an IoT network deployment where there are too many possible combinations of connectivity and message exchange options to expect speedy deployments.

IoT frameworks come to the rescue by building the connectivity intelligence into the framework – hidden from application view and simplified from the device and network management view. Frameworks also facilitate seamless manageability and serviceability by leveraging the framework's infrastructure to expose platform status information through the framework layer in accordance with the framework's data model abstraction. For example, a firmware update availability notification may be easily propagated across an IoT network. If the framework supports applying a firmware update, either push or pull, the firmware update images may be distributed over the air using the connectivity solution worked out by the framework.

3. Investigate tool Arduino IoT Cloud

Log, graph and analyze sensor data, trigger events, and automate your home or business. Here's why you might like it:

Easy for beginners, fast for professionals

- Based on the Arduino environment familiar to millions of users
- Quickly build-remote sensor monitoring using widgets
- Connect to a spreadsheet, database, or automate alerts using webhooks
- Create your own app by using Arduino IoT API

Serious technology (made simple)

- Devices secured using X.509 certificate-based authentication
- Developers can create custom apps using Arduino IoT Cloud APIs (contact create sales at "arduino.cc" for more info)

- Based on open hardware and open IoT standards

Practical steps to success

- Get an Arduino MKR board (WiFi now, soon NB-IoT, LTE Cat-M, Ethernet, or LoRa)
- Sign up free for Arduino IoT Cloud
- The best way to learn about IoT is to do it

Arduino IoT Cloud Components

Figure 42: Overview of The Arduino IoT Cloud Components

Based on what the user wants to achieve, an IoT application will require a few basic components include Devices to collect data or control something; Software to define the behavior of the hardware (e.g., Arduino Sketch); and Cloud application to store data, or remotely control the equipment.

Devices & Things in the Arduino IoT Cloud

Devices are physical objects like a hardware board that can be contained inside a product (e.g., MKR WiFi 1010). They're the hardware that runs the software, reads sensors, controls actuators, and communicates with the Arduino IoT Cloud.

Things are the logical representation of a connected object

Things represent the inherent properties of the object, with as little reference to the actual hardware used to implement them. Each thing is represented by a collection of properties (e.g., temperature, light).

Arduino IoT Cloud Properties

Properties are the qualities defining the characteristics of a system. A property can be something like a 'read-only' (RO) setting to indicate the Arduino IoT Cloud can read the data, but cannot change the value of the property. A property might be designed as 'read and write' (RW) if the Arduino IoT Cloud can also remotely change the property's value and send an event notification to the device. For example, a device might have a sensor that will provide the room temperature. That would be read-only. It might also include a thermostat which will be able to change the room's temperature.

Arduino IoT Cloud Events

The Arduino IoT Cloud becomes aware of events when it receives application messages that indicate something has happened. For example, it might be informed by a face-recognition application that someone is at a door, or it has received a request from another app that light has to be turned on.

Software for Arduino IoT Cloud

Arduino boards usually require you to program them, to enter some code by way of an Arduino Sketch. The Arduino IoT Cloud will quickly and automatically generate a Sketch when setting up a new thing: this is one of its convenient features. Arduino IoT Cloud allows other methods of interaction, including HTTP REST API, MQTT, Command-Line Tools, Javascript, and Websockets. It's a very versatile system. For more details, check out the API documentation.

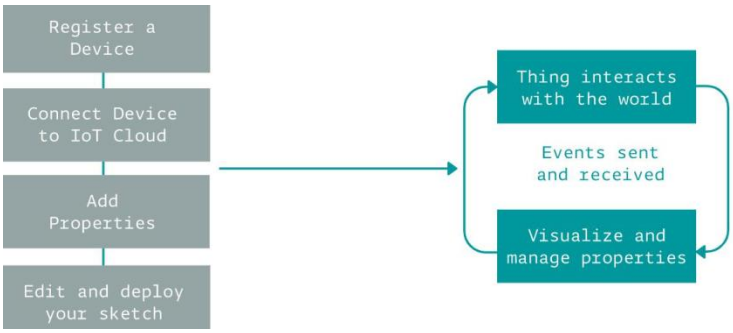

Figure 43: IoT project flow

To understand better how this might work, suppose we want to build an IoT greenhouse (i.e., a small enclosure constructed primarily of glass, used for growing plants). The goal is to control remotely this greenhouse: to be able to turn off and on the lights, start the irrigation system, read the temperature inside the garden, etc. The basic components needed to do this include a device to control the irrigation system. We'll use an Arduino MKR WiFi 1010, attached to a couple of sensors for measuring the temperature, light, etc. Also attached will be actuators such as an irrigation pump, as well as light and fan switches. The software (i.e., an Arduino Sketch) that will be uploaded to the MKR board, will automatically control the properties of the actuators. For instance, it will make changes to activate the ventilation fans when there is too much humidity or it's too hot in the greenhouse. The properties will be stored in the Cloud and maybe remotely changed from there. There are several properties that will define the greenhouse such as Pressure, Temperature, Humidity, UVA/B Ray Intensity, Pump Status, Light Status, and Fan

Status Once this is all in place, the system will wait for events and react to them. Events can be switching the pump on, switching the fan off, turning on the lights, etc.

4. Investigate hardware Arduino Uno

Figure 44: Arduino Uno

Arduino Uno is a microcontroller board based on the Atmega328P chip. Uno has 14 digital I / O pins (including 6 PWM output pins), 6 analog input pins, 1 16MHz quartz, 1 USB port, 1 DC power jack, 1 reset button. Uno fully supports everything we need to start working.

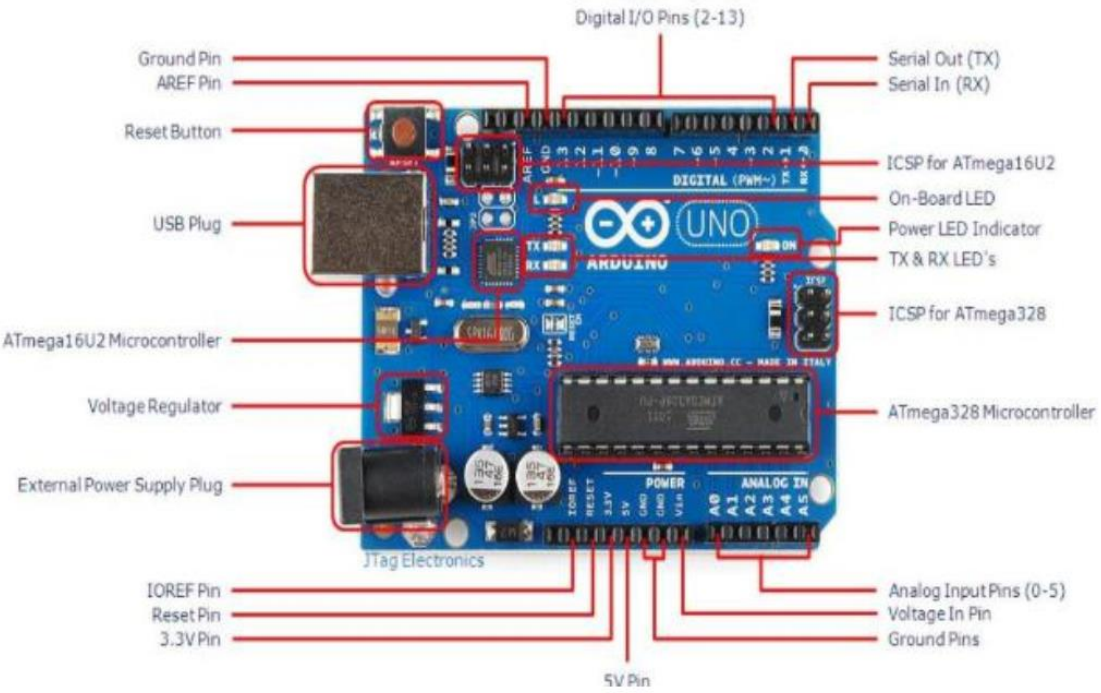

Figure 45: Detailed diagram of Uno R3

Table 3: Specifications - Uno R3

Microcontrollers	Atmega328P	Flash memory	32kB (Atmega32P) – inside 0.5 kB use for bootloader
Operating voltage	5V	SRAM	2 kB (Atmega328P)
Supply voltage (works well)	7 – 12 V	EEPROM	1 kB (Atmega328P)
Supply voltage (limited)	6 – 12 V	Clock speed	16 MHz
Digital I / O	pin 14 (There are 6 PWM pulse output pins)	Size	68.6 x 53.4 mm
Analog input pins	6 (A0 – A5)	Weight	25 g
Current per pin I / O	20 mA		
Arduino Uno R3	50 mA		

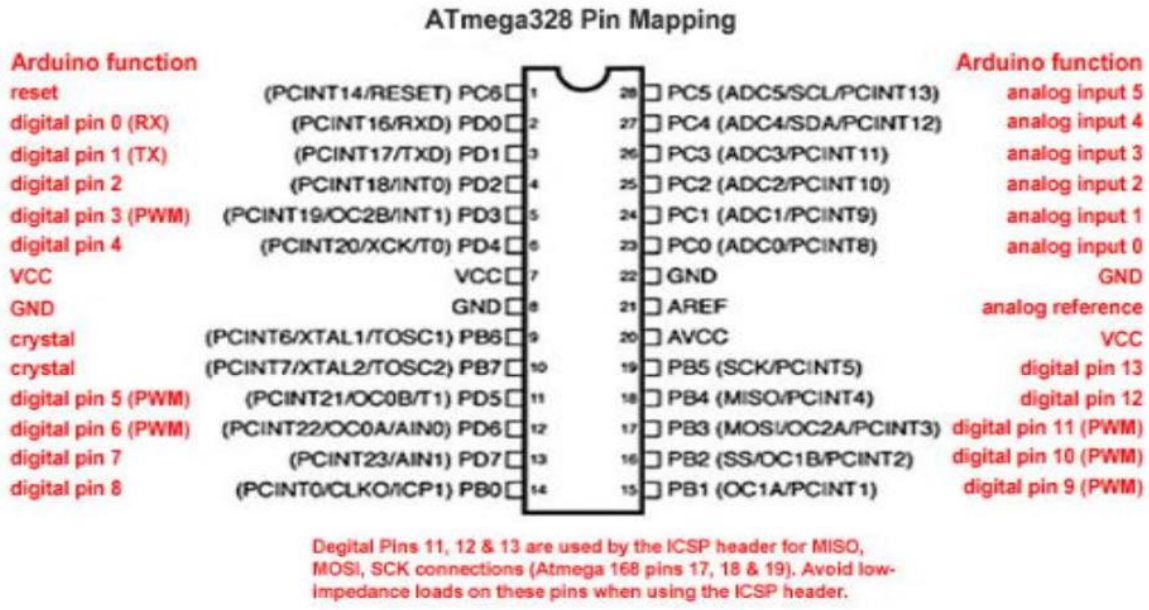

Figure 46: I/O Pins

DIGITAL

The digital I / O pins (pin 2 - 13) are used as input pins, output digital signals through the main functions: pinMode (), digitalWrite (), digitalRead (). The operating voltage is 5V, the current through these pins in normal mode is 20mA, the current exceeding 40mA will damage the microcontroller.

ANALOG

Uno has 6 analog Input pins (A0 - A5), the resolution of each pin is 10 bits (0 - 1023). These pins are used to read the 0 - 5V voltage (default) signal corresponding to 1024 values, using the analogRead () function.

PWM

Pins are numbered 3, 5, 6, 9, 10, 11; has PWM (8 bit) pulse function via analogWrite () function.

UART

Atmega328P allows data transfer via two pins 0 (RX) and pin 1 (TX).

SOURCE

There are two main ways to supply power to an Uno board: the USB port and the DC jack. The voltage limit for Uno is 6 - 20V. However, the recommended voltage range is 7 - 12 V (preferably 9V). The reason is that if the power supply is below 7V, the voltage at '5V feet' may be lower than 5V and the circuit may not work properly; If the power supply is greater than 12V, it may overheat the board or damage it. Source pins on Uno include Vin - can power Uno through this pin (However, this way of supplying power is rarely used); 5V - this pin can supply 5V from the Uno board (However, powering this pin or the 3.3 V pin can damage the board); 3.3V - this pin for source 3.3 V and the maximum current is 50mA; and GND - barefoot.

M4 Apply your selected techniques to create an IoT application development plan.

When it comes to security for business, IoT can certainly save the day. IoT sim cards can be applied to different devices that can be connected to sensors like biometric sensors, facial recognition, and more to increase home security, offices, and buildings. Just imagine the Door app can be unlocked via face recognition, the lights can be activated by motion sensors or footprints, the camera is activated and the live feed is triggered. Activated on your smartphone or the police can automatically be notified if an intruder broke the lock at your home.

RFID RC522 NFC helps to automatically manage the number of home openings, used to read and write data for 13.56mhz NFC tags, this module is the first choice for RFID card reading and recording applications.

Figure 47: RFID RC522 NFC 13.56mhz

- **Explore various forms of IoT functionality (Application)**

Smart Home

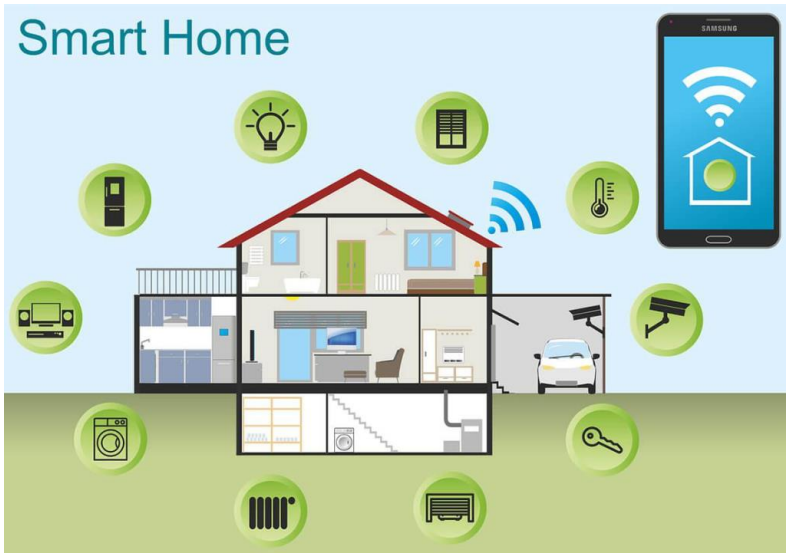

Figure 48: Smart home

Whenever we think of IoT systems, the most important and efficient application that stands out is the smart home, ranking the highest IoT application on all channels. The number of people searching for smart homes increases every month by about 60,000 people. Another interesting thing is that the database of smart homes for IoT analytics includes 256 companies and startups. More companies are now actively involved in smart homes, as well as similar applications in the field. The estimated amount of funding for smart home startups exceeds $2.5 billion and growing at a

rapid rate. The list of startups includes prominent startup company names, such as AlertMe or Nest, as well as a number of multinational corporations, like Philips, Haier, or Belkin.

Wearables

Figure 49: Wearables

Just like smart homes, wearables remain a hot topic among potential IoT. Every year, consumers all across the globe await the release of the latest Apple smartwatch. Apart from this, there are plenty of other wearable devices that make our life easy, such as the Sony Smart B Trainer, LookSee bracelet, or the Myo gesture control.

Smart City

Figure 50: Smart city

Smart cities, like its name suggests, is a big innovation and spans a wide variety of use cases, from water distribution and traffic management to waste management and environmental monitoring. The reason why it is so popular is that it tries to remove the discomfort and problems of people who live in cities. IoT solutions offered in the smart city sector solve various city-related problems, comprising of traffic, reducing air and noise pollution, and helping to make cities safer.

Smart Grids

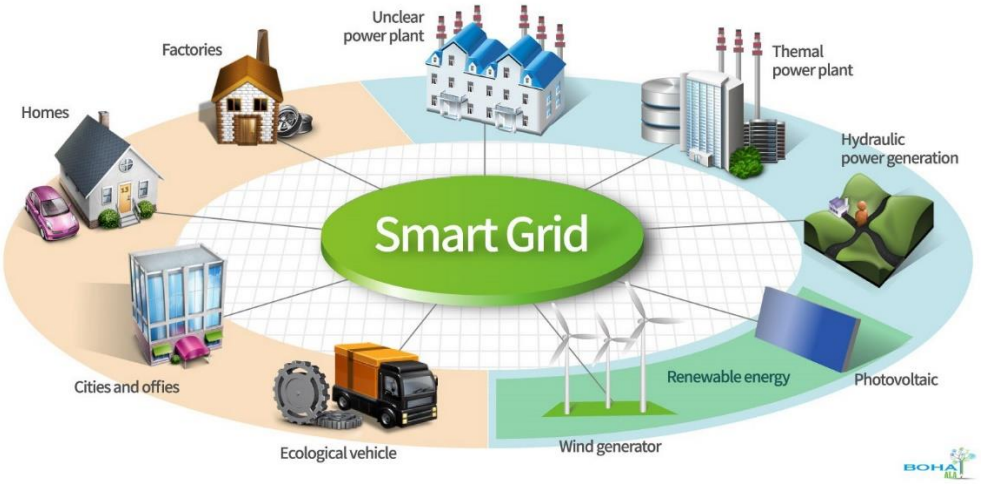

Figure 51: Smart Grids

Smart grids are another area of IoT technology that stands out. A smart grid basically promises to extract information on the behaviors of consumers and electricity suppliers in an automated fashion to improve the efficiency, economics, and reliability of electricity distribution. 41,000 monthly Google searches is a testament to this concept's popularity.

Industrial Internet

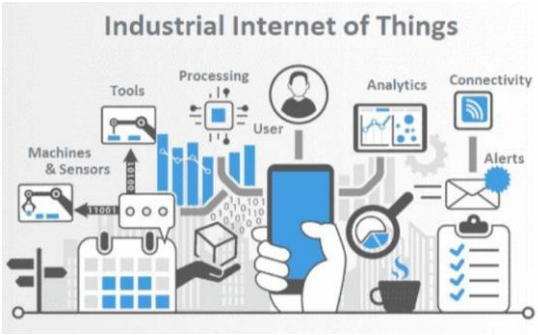

Figure 52: Industrial Internet

One way to think of the Industrial Internet is by looking at connected machines and devices in industries such as power generation, oil, gas, and healthcare. It also makes use of situations where unplanned downtime and system failures can result in life-threatening situations. A system embedded with the IoT tends to include devices such as fitness bands for heart monitoring or smart home appliances. These systems are functional and can provide ease of use but are not reliable because they do not typically create emergency situations if the downtime was to occur.

Connected Car

Figure 53: Connected Car

Connected car technology is a vast and extensive network of multiple sensors, antennas, embedded software, and technologies that assist in communication to navigate in our complex world. It has the responsibility of making decisions with consistency, accuracy, and speed. It also has to be reliable. These requirements will become even more critical when humans give up control of the steering wheel and brakes to the autonomous vehicles that are being tested on our highways right now.

Smart retail

Figure 54: Smart retail

Retailers have started adopting IoT solutions and using IoT embedded systems across a number of applications that improve store operations, increasing purchases, reducing theft, enabling inventory management, and enhancing the consumer's shopping experience. Through IoT physical retailers can compete against online challengers more strongly. They can regain their lost market share and attract consumers into the store, thus making it easier for them to buy more while saving money.

Smart Farming

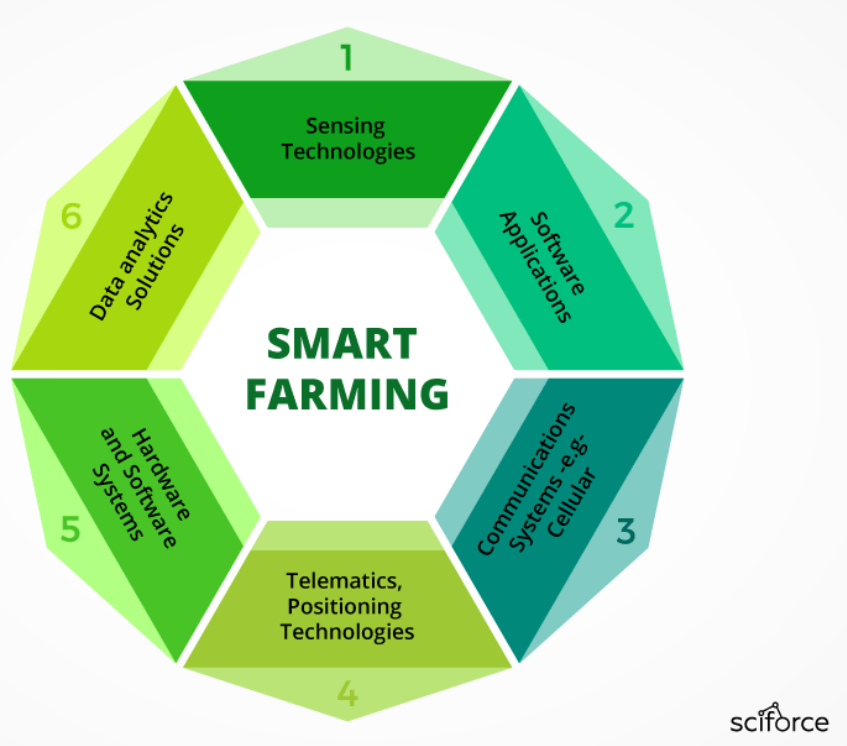

Figure 55: Smart Farming

Smart farming is often overlooked in IoT applications. However, because the number of farming operations is usually remote and the large number of livestock that farmers work on, all of this can be monitored by the IoT and can revolutionize the way farmers operate day today. But, this idea is yet to reach large-scale attention. Nevertheless, it still remains one of the IoT applications that should not be underestimated. Smart farming has the potential to become an important application field, specifically in the agricultural-product exporting countries.

Reference

[1] Aayush, 2017. *Internet Of Things(Iot): Introduction, Applications And Future Scope*. [online] GKMIT. Available at: <https://www.gkmit.co/blog/internet-of-things/internet-of-things-iot-introduction-applications-and-future-scope> [Accessed 18 October 2020].

[2] HIWANI CHOUDHARY, 2018. *All You Need To Know About Internet Of Underwater Things | Dspcomm*. [online] Underwater Wireless Modem and Communication devices. Available at: <https://www.dspcommgen2.com/all-you-need-to-know-about-internet-of-underwater-things/> [Accessed 18 October 2020].

[3] Tareq Y. Al-Naffouri, Nasir Saeed and Mohamed-Slim Alouin, 2019. Towards the Internet of Underground Things: A Systematic Survey. *IEEE*, [online] (arXiv:1902.03844v2 [eess.SP]), p.1. Available at: <https://arxiv.org/pdf/1902.03844.pdf> [Accessed 18 October 2020].

[4] Alexander Kott, Ananthram Swami and Bruce J. West, 2016. *The Internet Of Battle Things*. [ebook] IEEE. Available at: <https://arxiv.org/ftp/arxiv/papers/1712/1712.08980.pdf> [Accessed 18 October 2020].

[5] Admin, 2015. *The Internet Of Space Things (Cubesats) | Broadband Wireless Networking Lab*. [online] Bwn.ece.gatech.edu. Available at: <http://bwn.ece.gatech.edu/projects/iost/index.html#:~:text=BWN%20Projects-,The%20Internet%20of%20Space%20Things%20(CubeSats),sensors%20to%20unmanned%20aerial%20vehicles.> [Accessed 18 October 2020].

[6] Mainor Alberto, Patricia Bazán and Cruz Alvarado, 2018. *Understanding The Internet Of Nano Things: Overview, Trends, And Challenges*. [ebook] Available at: <https://dialnet.unirioja.es/descarga/articulo/7026211.pdf> [Accessed 18 October 2020].

[7] I. F. Akyildiz, M. Pierobon, S. Balasubramaniam and Y. Koucheryavy, 2015. "The internet of Bio-Nano things," in IEEE Communications Magazine, vol. 53, no. 3, pp. 32-40. [ebook] Available at:< https://ieeexplore.ieee.org/document/7060516/citations#citations> [Accessed 18 October 2020].

[8] Admin, 2020. *What Is Iot Architecture?*. [online] Avsystem.com. Available at: <https://www.avsystem.com/blog/what-is-iot-architecture/#:~:text=The%20IoT%20architecture%20for%20the,medical%20intakes%2C%20and%20physical%20activities.> [Accessed 18 October 2020].

[9] En.wikipedia.org. 2020. *Packet Tracer*. [online] Available at: <https://en.wikipedia.org/wiki/Packet_Tracer> [Accessed 18 October 2020].

[10] Admin, 2020. *Internet Of Things - Hardware - Tutorialspoint*. [online] Tutorialspoint.com. Available at:

<https://www.tutorialspoint.com/internet_of_things/internet_of_things_hardware.htm> [Accessed 18 October 2020].

[11] Kan Zheng, 2016. *Design And Implementation Of Application Programming Interface For Internet Of Things Cloud*. [ebook] Wiley Online Library. Available at: <https://www.researchgate.net/figure/Types-of-services-and-design-of-APIs-in-IoT-cloud-API-application-programming_fig1_304536960> [Accessed 19 October 2020].

[12] William G. Wong, 2015. *Engineering Essentials: Iot Standards And Frameworks*. [online] Electronic Design. Available at: <https://www.electronicdesign.com/technologies/iot/article/21800824/engineering-essentials-iot-standards-and-frameworks> [Accessed 19 October 2020].

[13] Joncas, R., 2020. *MQTT And Coap, Iot Protocols | The Eclipse Foundation*. [online] Eclipse.org. Available at: <https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php#:~:text=MQTT%20and%20CoAP%20are%20both,clients%20through%20a%20central%20broker.&text=CoAP%20is%2C%20primarily%2C%20a%20one,information%20between%20client%20and%20server.> [Accessed 20 October 2020].

[14] FPT Greenwich University, 2020. *Iot-Internet Of Things*. [online] Google Drive. Available at: <https://drive.google.com/drive/u/0/folders/1i_puCHwH-Oip5HJLjhzMq4HMtinpFQbP> [Accessed 20 October 2020].

[15] Biliyaminu Umar, Hamdan Hejazi, László Lengyel, Károly Farkas, 2018. *Evaluation Of Iot Device Management Tools*. [ebook] Available at: <http://www.thinkmind.org/download.php?articleid=accse_2018_2_10_90016> [Accessed 22 October 2020].

[16] João Pedro Dias and Hugo Sereno Ferreira, 2018. *State Of The Software Development Life-Cycle For The Internet-Of-Things*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/328899486_State_of_the_Software_Development_Life-Cycle_for_the_Internet-of-Things> [Accessed 22 October 2020].

[17] Leila Fatmasari Rahmana, Tanir Ozcelebia and Johan Lukkien, 2018. *Understanding Iot Systems: A Life Cycle Approach*. [ebook] Available at: <https://www.sciencedirect.com/science/article/pii/S1877050918305106> [Accessed 22 October 2020].

[18] Zhejiang University Hangzhou, 2017. *A Review Of Internet Of Things Architecture, Technologies And Analysis Smartphone-Based Attacks Against 3D Printers*. [ebook] Available at: <https://arxiv.org/ftp/arxiv/papers/1708/1708.04560.pdf> [Accessed 22 October 2020].

[19] Admin, 2020. *What Is Edge Computing?*. [online] CloudFlare. Available at: <https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/> [Accessed 22 October 2020].

[20] Admin, 2020. *Technical Overview Of Internet Of Things | Solutions | Google Cloud*. [online] Google Cloud. Available at: <https://cloud.google.com/solutions/iot-overview> [Accessed 22 October 2020].

[21] Parikshit Joshi, 2018. *The 4 Computing Types For The Internet Of Things*. [online] IoT For All. Available at: <https://www.iotforall.com/4-computing-types-for-iot> [Accessed 22 October 2020].

[22] JR Fuller, 2016. *How To Design An Iot-Ready Infrastructure: The 4-Stage Architecture*. [online] TechBeacon. Available at: <https://techbeacon.com/enterprise-it/4-stages-iot-architecture> [Accessed 22 October 2020].

[23] Nisarg Mehta, 2018. *10 Most Popular Open Source Iot Frameworks*. [online] Techtic Solutions. Available at: <https://www.techtic.com/blog/top-10-open-source-iot-frameworks/> [Accessed 22 October 2020].

[24] Admin, 2019. *Top 10 Popular Iot Development Tools*. [online] Eduonix Blog. Available at: <https://blog.eduonix.com/internet-of-things/top-10-popular-iot-development-tools/> [Accessed 22 October 2020].

[25] Brandon Cannaday, 2019. *The Fundamental Iot Architecture*. [online] Losant.com. Available at: <https://www.losant.com/blog/the-fundamental-iot-architecture> [Accessed 22 October 2020].

[26] Cheruvu, S., Kumar, A., Smith, N. and Wheeler, D., 2020. *Demystifying Internet Of Things Security*. Berkeley, CA: Apress. [Accessed 22 October 2020].