IOT

Unit - 2

Elements of IoT

2.1 Hardware Components – Computing (Arduino, Raspberry Pi)

IoT (Internet of Things) is no longer a buzzword. With several inspiring use cases, emanating daily, multiple firms are now discovering how they could leverage the technology for business growth. It is fast becoming an important feature for new devices to be IoT based, irrespective of the other technologies implemented, and according to Gartner, by 2020, 95% of new devices and systems will use the IoT.

Each is a part of an IoT hardware platform — a combination of hardware, connectivity tools, and software development environment for IoT projects.

Arduino and Pi are not the only and the best IoT platforms worth knowing. In fact, there are dozens of platforms with a diverse choice of hardware, support, security, development infrastructure, and communities. In this article, we'll focus on some popular platforms and try to figure out the perfect matches for different IoT projects.

Arduino hardware is an affordable and easy to set up option for building a basic IoT device that is supposed to perform one action, for example, read humidity sensor data. Arduino community is one of the oldest in this domain, so there won't be a lack of support or resources. On top of that, Arduino's functionality is easily expandable with on-top shields and multiple digital and analog general-purpose input/output pins.

Raspberry Pi is the best choice for data-heavy connected devices like hubs, gateways, datum collectors, or personal cloud servers, however, it will also be a good fit for simpler IoT applications.

Arduino

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards can read inputs – light on a sensor, a finger on a button, or a Twitter message – and turn it into an output – activating a motor, turning on an LED, publishing something online. With the ease of programming and the plug and play nature of Arduino based system, it quickly became loved by many in the hardware space. The early Arduino boards were mostly general-purpose microcontrollers that were connected to the internet using GSM and WiFi modules, but as the IoT began to Open up, boards with special features that support the IoT were developed. Boards like the Arduino 101(developed with Intel), the MKR1000, Arduino WiFi Rev 2, and the MKR Vidor 4000 which is the first Arduino board based on an FPGA Chip.

Raspberry pi

The Raspberry Pi is a Single Board Computer developed by Raspberry Pi Foundation. It is widely popular as a small, inexpensive computing board among experimenters, hobbyists, educators, and technology enthusiasts.

While the Raspberry Pi is naturally a general-purpose device, it will be an injustice to ignore the contribution of the raspberry to the development of some of the IoT products and projects currently in vogue. They are generally too robust and sophisticated to be used in the development of simple connected sensors or actuators, but they find applications serving as data aggregators, hubs, and device gateways in IoT projects. The latest of the raspberry pi boards; the Raspberry pi 3 model B+ features a 1.4GHz Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC, 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, and a Gigabit Ethernet port over USB 2.0 (maximum throughput 300 Mbps). Asides from several other features including 4 USB ports, Audio output, to mention a few, the board comes with a 1GB LPDDR2 SDRAM which makes it quite fast for IoT-based tasks.

Key takeaway

Arduino is an open-source prototyping platform based on easy-to-use hardware and software.

The Raspberry Pi is a Single Board Computer developed by Raspberry Pi Foundation.

It is widely popular as a small, inexpensive computing board among experimenters, hobbyists, educators, and technology enthusiasts.

2.2 Communication

The communication module is the device's final, but most important, component. This is a device that connects devices to storage, either locally or in the cloud. Communication ports including USB, Modbus, and Ethernet/IP, to mention a few, can be found in this module. Wireless communication radio technology, such as Wireless fidelity, might be included as well.

Communication ports such as USB, serial (232/485), and CAN, to name a few, may be included in this module. It could also contain Wi-Fi, LoRA, ZigBee, and other wireless communication technologies.

The communications module might be part of the same device as the other modules, or it can be a distinct device dedicated to communications only. A "gateway architecture" is a term used to describe this approach.

If you have three sensors in a room that need to send data to the Cloud, you could link them all to a single gateway in the same room, which then consolidates the data and delivers it to the Cloud.

WiFi

Wi-Fi networking is typically an obvious choice for many developers, especially with Wi-pervasiveness Fi's within the LAN home environment. It's a wireless local area network that uses IEEE 802.11 specifications in a 5GHz ISM frequency band. Wi-Fi is a short-range technology with a range of around 60 feet between an access point and the user.

Wi-Fi is a wireless technology that was created to take the role of Ethernet. Its purpose was to promote cross-seller interoperability using off-the-shelf, simple-to-install, and simple-to-use short-range wireless communication.

Although ordinary Wi-Fi isn't always the optimal IoT technology, some IoT applications can benefit from it, especially in in-building or campus locations. Building and home automation, as well as in-house energy management, are obvious examples of applications where Wi-Fi can be utilized as the communication channel and devices can be connected to electrical outlets.

Wi-Fi 802.11ah, often known as "HaLow," is intended specifically for IoT and requires particular clients and infrastructure. Wi-Fi technology vendors are always improving and striving to give better technology.

ZigBee

ZigBee is a wireless technology that is comparable to Bluetooth and is primarily utilized in industrial environments. It offers low-power operation, high security, robustness, and high performance in complex systems, and it is well positioned to take advantage of wireless control and sensor networks in IoT applications.

The most recent version of ZigBee is version 3.0, which essentially unifies the multiple ZigBee wireless technologies into a single standard.

LoRAWAN

LoRaWAN is a widely used Internet of Things (IoT) technology that addresses wide-area network (WAN) applications. The LoRaWAN standard was created to give low-power WANs the features they need to allow low-cost mobile secure communication in IoT, smart city, and industrial applications.

Data rates range from 0.3 kbps to 50 kbps, and it is designed to meet low-power requirements and sustain huge networks with millions of devices.

Fig 1: LoRA

Satellite communication

Mobile phones can communicate with the nearest antenna, which is around 10-15 miles away, thanks to satellite communications. Depending on the speed of communication, these are referred to as GSM, GPRS / GSM, 3G, 4G / LTE, 5G, and others. Satellite communication, also known as Machine to Machine communication in the IoT, allows mobile devices to communicate with one another.

The only viable answer to the communication restriction that is the wide-scale interconnection of IoT devices appears to be custom-designed satellite communication. Satellite technology may be able to assist the IoT sector in growing and addressing this wide-ranging connection problem.

For such heavy loads, data transport speed could become an issue. Nonetheless, it is just a matter of time before novel solutions become available.

Satellite providers are already collaborating to develop services and equipment that will allow IoT to reach its full potential. A system to merge fiber, satellite, and wireless networks is already being developed. Satellite systems are the most effective signal transmission on Earth because of their global nature and capacity to simultaneously broadcast to many sites. Satellite transmission is used in conjunction with terrestrial networks to obtain global coverage.

Bluetooth

Since Ericsson's inception in 1994, Bluetooth technology has come a long way. Bluetooth was created as a replacement for ordinary RS connections, which were previously used to link external devices to computers. In the Internet of Things, Bluetooth is used to track devices in the commercial, educational, and health care sectors. Indoor tracking scenarios with minimal power needs benefit greatly from Bluetooth applications. Bluetooth communication, on the other hand, is classified as short-range and does not permit transmission or tracking underwater. Furthermore, Bluetooth connectivity is not recommended for security solutions that involve the transmission of visual or audio data over the network.

Low-Energy Bluetooth, which was introduced in 2009, allowed IoT to use Bluetooth as a communication medium. BLE is a wireless standard for small-scale IoT devices like wearables and beacons, allowing them to deliver modest quantities of data while consuming very little power. Bluetooth's importance and applications in cars and homes will continue to increase and expand. Imagine getting automatic traffic updates or weather reports on your dashboard throughout your daily commute, or utilizing Bluetooth to set up lighting, thermostat, and home theater systems for the right mood or occasion.

Key takeaway

The communication module is the device's final, but most important, component. This is a device that connects devices to storage, either locally or in the cloud.

Wi-Fi networking is typically an obvious choice for many developers, especially with Wi-pervasiveness Fi's within the LAN home environment.

ZigBee is a wireless technology that is comparable to Bluetooth and is primarily utilized in industrial environments.

Satellite communication, also known as Machine to Machine communication in the IoT, allows mobile devices to communicate with one another.

Bluetooth communication, on the other hand, is classified as short-range and does not permit transmission or tracking underwater.

2.3 Sensing

Sensors can be found all over the place. They can be found in our homes and businesses, as well as retail malls and hospitals. They're built into smartphones and play a key role in the Internet of Things (IoT). Sensors have existed for quite some time. Infrared sensors have been around since the late 1940s, while the first thermostat was launched in the late 1880s. The Internet of Things (IoT) and its industrial cousin, the Industrial Internet of Things (IIoT), are taking sensor usage to new heights.

Sensors, in general, are devices that detect and respond to changes in their surroundings. Light, temperature, motion, and pressure are all examples of possible inputs. Sensors produce useful data, which they can exchange with other connected devices and management systems if they are connected to a network.

Sensors are vital to the success of many modern organizations. They can alert you to possible issues before they turn into major issues, allowing firms to do preventative maintenance and avoid costly downtime.

A good sensor should have these three characteristics:

- It needs to be attentive to the phenomenon it's tracking.
- It shouldn't be affected by other bodily factors.
- During the measuring procedure, it should not change the phenomenon being measured.

We can use a variety of sensors to measure practically all of the physical properties around us. Thermometers, pressure sensors, light sensors, accelerometers, gyroscopes, motion sensors, gas sensors, and many other common sensors are widely used in everyday life. Several properties can be used to describe a sensor, the most essential of which is:

- Range - The sensor's range refers to the phenomenon's highest and minimum values.

- Sensitivity - Sensitivity is defined as the smallest change in the measured parameter that results in a noticeable change in the output signal.

- Resolution - The sensor's resolution is the smallest change in the phenomenon it can detect.

Sensor classification

Several criteria can be used to classify sensors:

- Passive or Active - Active sensors, on the other hand, require an external power source to monitor an environment, whereas passive sensors do not.

- Another classification is dependent on how the property was detected and measured (mechanical, chemical, etc.).

- Analog or Digital - Digital sensors produce a discrete signal, whereas analog sensors produce an analog, or continuous, signal.

Types of sensors

There are many different types of IoT sensors, as well as numerous applications and use cases. Here are ten of the most common types of IoT sensors, as well as some of their applications.

1. Temperature sensors

Temperature sensors detect temperature changes and translate them to data by measuring the quantity of heat energy present in a source. Manufacturing machinery frequently necessitates specific environmental and device temperatures. Similarly, soil temperature is an important determinant in crop growth in agriculture.

2. Humidity sensors

These sensors determine how much water vapor is present in the environment of air or other gases. Humidity sensors are often found in both industrial and domestic heating, venting, and air conditioning systems. They can also be found in a variety of different places, such as hospitals and meteorology stations, where they record and forecast weather.

3. Pressure sensors

Changes in gases and liquids are detected by a pressure sensor. The sensor monitors changes in pressure and conveys them to connected systems when they occur. Leak testing, which might occur as a result of degradation, is a common application case. Pressure sensors are also important in the manufacture of water systems since they can easily detect pressure variations or dips.

4. Proximity sensors

Proximity sensors are used to detect objects that are close to the sensor without having to touch them. Electromagnetic fields or beams of radiation, such as infrared, are frequently emitted by these sensors. Proximity sensors have a variety of applications. A proximity sensor in retail can detect motion between a customer and a product that piques his or her attention. Any discounts or special offers on products near the sensor might be notified to the user. Mall parking lots, stadium parking lots, and airport parking lots all use proximity sensors to signal parking availability. They can also be employed in production lines in the chemical, food, and a variety of other industries.

5. Level sensors

The level of substances such as liquids, powders, and granular materials is detected using level sensors. Level sensors are used in a variety of industries, including oil production, water treatment, and beverage and food manufacturing. Level sensors can measure the amount of rubbish in a garbage can or dumpster, which is a frequent use case for waste management systems.

6. Gyroscope sensors

Gyroscope sensors measure the angular rate or velocity, which is commonly defined as the speed and rotation around an axis. Automobiles, such as automobile navigation and electronic stability control (anti-skid) systems, are examples of use cases. Motion sensing for video games and camera shake detection systems are two other applications.

7. Gas sensors

These sensors track and detect changes in air quality, such as the presence of toxic, flammable, or dangerous gases. Mining, oil and gas, chemical research, and manufacturing are all industries that use gas sensors. Carbon dioxide detectors, which are found in many homes, are a common consumer use case.

8. Infrared sensors

By producing or detecting infrared radiation, these sensors detect features in their surroundings. They can also detect the heat that objects emit. Infrared sensors are employed in a range of IoT projects, including healthcare, because they make blood flow and blood pressure monitoring easier. Infrared sensors are used by televisions to decipher the signals supplied by a remote control. Art historians utilizing infrared sensors to observe hidden layers in paintings to assist establish whether a work of art is real or fake, or has been altered by a restoration technique, is another intriguing application.

9. Optical sensors

Optical sensors are devices that transform light beams into electrical impulses. Optical sensors have a wide range of applications and use cases. Vehicles in the automotive industry utilize optical sensors to detect signs, obstructions, and other objects that a driver might see while driving or parking. Optical sensors are crucial in the development of self-driving cars. Optical sensors are widely used in mobile phones.

Key takeaway

Sensors can be found all over the place. They can be found in our homes and businesses, as well as retail malls and hospitals.

Sensors are vital to the success of many modern organizations.

Temperature sensors detect temperature changes and translate them to data by measuring the quantity of heat energy present in a source.

Humidity sensors are often found in both industrial and domestic heating, venting, and air conditioning systems.

Pressure sensors are also important in the manufacture of water systems since they can easily detect pressure variations or dips.

Proximity sensors are used to detect objects that are close to the sensor without having to touch them.

Optical sensors are devices that transform light beams into electrical impulses.

2.4 Actuation

A physical object (“thing”) + controller (“brain”) + sensors + actuators + networks make up an IoT device (Internet). A machine component or system that moves or regulates a mechanism or system is known as an actuator. The device's sensors detect the surroundings, and control signals are sent to the actuators based on the activities required.

An actuator is something like a servo motor. They can move to a defined angular or linear location and are either linear or rotatory actuators. For IoT applications, we can use servo motors and rotate them to 90 degrees, 180 degrees, and other angles as needed.

The controller instructs the actuator to conduct the task depending on the sensor data, as shown in the diagram below.

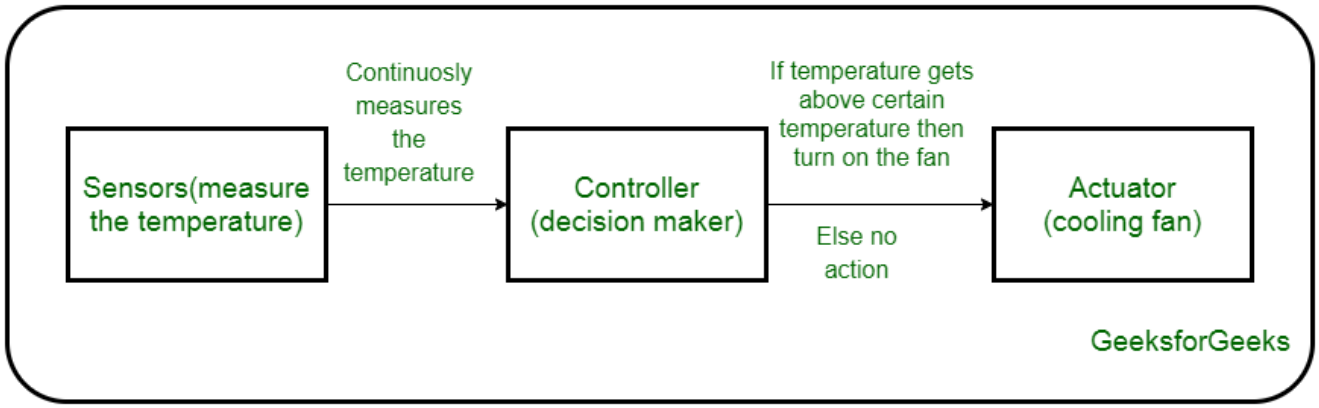


Fig 2: Use of actuations

Through the actuator, the control system affects the environment. It necessitates an energy supply as well as a control signal. It turns the source of energy into a mechanical operation when it gets a control signal. On this premise, on the form of energy it uses, it has different types given below.

Types of Actuators

1. Hydraulic actuator

A hydraulic actuator is a mechanical device that employs hydraulic power to complete a task. A cylinder or a fluid motor drives them. According to the needs of the IoT device, mechanical motion is translated to rotary, linear, or oscillatory motion. Hydraulic actuators are used in construction equipment because they can create a considerable amount of force.

Advantages

- Hydraulic actuators have the ability to generate significant amounts of force at a high rate.
- Used in welding, clamping, and other applications.
- In car transport carriers, it's used to lower or raise the vehicles.

Disadvantages

- Leaks in hydraulic fluid can reduce performance and complicate cleanup.

- Noise reduction equipment, heat exchangers, and high-maintenance systems are all required.
- It is expensive.

2. Pneumatic Actuators

A pneumatic actuator converts energy created by vacuum or high-pressure compressed air into linear or rotary motion. For example, sensors that act like human fingers and are powered by compressed air are used in robotics.

Advantages

- They are a low-cost solution that is employed in extreme temperatures where employing air rather than chemicals is a safer option.
- They require little maintenance, are long-lasting, and have a lengthy service life.
- It is quite quick to initiate and stop the action.

Disadvantages

- It can become less efficient if there is a loss of pressure.
- The air compressor should be turned on all the time.
- It is possible for air to be polluted, and it must be maintained.

3. Electrical actuators

An electric actuator works by converting electrical energy into mechanical torque and is usually powered by a motor. A solenoid-based electric bell is an example of an electric actuator.

Advantages

- It can automate industrial valves, which makes it useful in a variety of sectors.

- It makes less noise and is completely safe to use because there are no fluid leaks.
- It has the ability to be reprogrammed and delivers the highest level of control and precision positioning.

Disadvantages

- It's not cheap.
- It is highly dependent on the surrounding environment.

Key takeaway

A physical object (“thing”) + controller (“brain”) + sensors + actuators + networks make up an IoT device (Internet).

A machine component or system that moves or regulates a mechanism or system is known as an actuator.

A hydraulic actuator is a mechanical device that employs hydraulic power to complete a task.

A pneumatic actuator converts energy created by vacuum or high-pressure compressed air into linear or rotary motion.

An electric actuator works by converting electrical energy into mechanical torque and is usually powered by a motor.

2.5 I/O interfaces

The Input Output Interface (IOI) is a technique for exchanging data between internal and external storage and I/O devices.

The Input-Output Interface (I/O) is a means for moving data between internal storage devices, such as memory, and external peripheral devices. A peripheral device, often known as an input-output device, is a device that provides input and output for a computer. Consider the following scenario: Input devices, such as a keyboard and

mouse, offer input to the computer, and output devices, such as a monitor and printer, provide output to the computer. In addition to external hard drives, several peripheral devices that can provide both input and output are also available.

For peripherals connected to a computer to communicate with the central processing unit, special communication cables are required.

The communication link's aim is to reconcile the disparities between the central computer and each peripheral.

The following are the main differences:

1. Electronic devices are CPU and memory, whereas electromechanical and electromagnetic devices are peripherals. As a result, signal values may need to be converted.
2. Because peripheral data transfer rates are typically slower than CPU data transfer rates, a synchronization mechanism may be required.
3. The word format in the CPU and RAM differs from the data codes and formats in the peripherals.
4. Peripheral operating modes differ from one another and must be managed so as not to interfere with the operation of other peripherals attached to the CPU.

Computer systems include additional hardware components between the CPU and peripherals to supervise and synchronize all input and out transfers in order to resolve these mismatches.

- Because they connect the CPU bus to peripheral devices, these components are known as Interface Units.

Functions

- It's utilized to keep the CPU's operating speed in sync with the input-output devices.

- It chooses the input-output device that is best suited for the input-output device's interpretation.

- It can send out signals such as control and timing signals.

- Data buffering is possible through the data bus in this case.

- There are several types of error detectors.
- Serial data is converted to parallel data and vice versa.
- It can also convert digital data to analog signals and the other way around.

2.6 Software Components- Programming API's (using Python / Node.js /Arduino) for Communication

The Arduino Rest API is a way for Arduino and other external systems to communicate data. It is possible to operate Arduino from afar using the Arduino Rest API framework. The confluence of APIs and IoT creates new integration possibilities. The creation of an API ecosystem is a fascinating topic, and the way we use APIs to access IoT services exposed by remote IoT boards is a difficult component.

In greater detail, a client application uses the Arduino Rest API to read or transmit data to the Arduino board. An external system or application that retrieves sensor values is a common use case for HTTP Rest API.

When different systems and boards are connected and share information, the Arduino Rest API framework can be used in IoT projects. The Arduino Rest API is used by IoT cloud companies as well. This type of method is used when an external application (client) submits a request to Arduino, and Arduino responds with data. Because the Arduino Rest API uses the HTTP protocol, these queries are synchronous. Other protocols, like as MQTT, can be utilized in IoT applications. When Arduino is acting as a server in a client-server scenario, the Arduino API over HTTP plays a critical role. MQTT, for example, employs a structure known as publish-subscribe.

Arduino rest framework

There is an intriguing library called aRest that may be used to construct a Rest API architecture. This library is a framework for Restful services that includes a number of useful features. This library works with a variety of development boards, including Arduino, Raspberry Pi, and ES8266. More information is available on the aRest website.

This library is easy to use and may be acquired from the Arduino library directly through the Arduino IDE.

We can implement the API using this library because it supports aRest:

- Reading pin values in rest style
- Writing pin values in rest style
- Remote sketch function call

Arduino implementations

Now that we've covered the fundamentals of Arduino and how to utilize it to connect it to an external system, we'll go through how to put it into reality. We'll use Rest API requests to control an LED strip in this example. Because we need to concentrate on the Arduino Rest API, the sketch is straightforward. The LED strip is a Neopixels RGB Stick Board, and it is able to select a single RGB led color using the Adafruit library.

The following sketch demonstrates how to connect it to an Arduino UNO.

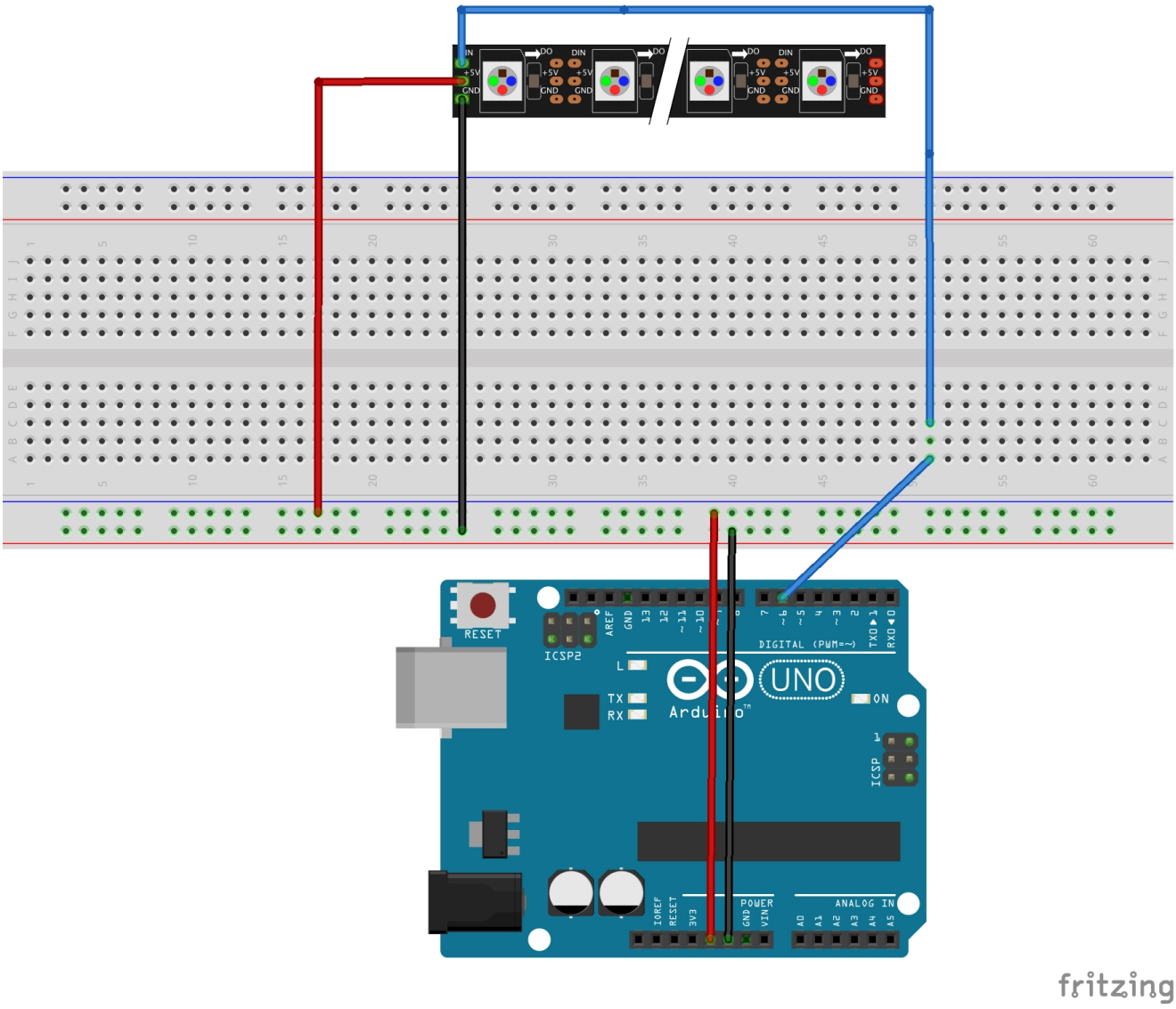


Fig 3: Arduino UNO

The Neopixel components in this image are different, but the connections are the same.

We want to change the color of the led strip using a Rest JSON API call. The color is supplied as a HEX parameter to the sketch function. This sample exemplifies the library's versatility. The Arduino code is straightforward:

// Create aREST instance

AREST rest = aREST();

// NeoPixel Init

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

```
Void setup() {
Serial.begin(115200);

// Register RGB function
Rest.function("rgb", setPixelColor);
Serial.println("Try DHCP...");
If (Ethernet.begin(macAdd) == 0) {
Serial.println("DHCP FAIL...Static IP");
Ethernet.begin(macAdd , ip, myDns, myGateway) ;
}

Server.begin();
Serial.print("server IP: ");
Serial.println(Ethernet.localIP());

Pixels.begin();
Serial.println("Setup complete.\n");
}

Void loop() {
// listen for incoming clients
EthernetClient client = server.available();
Rest.handle(client);
Wdt_reset();
```

```
}

Int setPixelColor(String hexColor) {

HexColor="0x" + hexColor;

Serial.println("Hex color " + hexColor);

Long n = strtol( &hexColor[0], NULL, 16);

Serial.println("N :" + String(n));

Long r = n << 16;

Long g = n << 8 && 0xFF;

Long b = n && 0xFF;


// set single pixel color

Return 1;

}
```

SetColor is the Arduino function that we'd like to make available via an Arduino HTTP Rest API. As a result, the sketch registers it as rgb at line 36.

2.7 Protocols-MQTT

MQTT (Message Queuing Telemetry Transport) is an IBM-developed lightweight messaging protocol that was initially introduced in 1999. It interprets communications between devices, servers, and applications using the pub/sub pattern.

The MQTT protocol was established with the goal of connecting sensors on oil pipelines to communications satellites with the least amount of battery loss and bandwidth use possible.

MQTT has been evolving since its creation, with version 5.0 being released in May 2018. Version 3.1.1 was accepted as an ISO standard after being submitted to the OASIS consortium in 2013.

Advantages

The following are some of the advantages of MQTT:

- Lightweight code footprint - The MQTT protocol requires only a few lines of code to get up and running on devices.
- Minimized data packets - MQTT is a low-energy protocol. If a device is battery-powered or has a low CPU, this is ideal.
- Speed - Outside of QoS, MQTT functions in real time with no delays.
- Ease of Implementation - MQTT includes libraries in a number of programming languages, including Elixir and Python.
- Last will and testament - If a client disconnects abruptly, you can send a message to all subscribers with information on how to fix the problem.
- Retained messages - When a client subscribes to a topic, it will automatically get one retained message for that subject (like a pinned post on social media).

MQTT architecture

In the MQTT protocol, connected devices are known as "clients," and they communicate with a server known as the "broker." Data communication between clients is handled by the broker.

When a customer (known as a "publisher") wants to disseminate information, it publishes to a specific topic, which the broker then distributes to any clients who have subscribed to that topic (known as "subscribers").

The publisher does not require information on the number of subscribers or their locations. As a result, subscribers do not require any information about the publisher. Any client can be both a publisher and a subscriber. The clients are usually unaware of each other and are only aware of the broker who acts as an intermediary. The “pub/sub model” is a popular term for this structure.

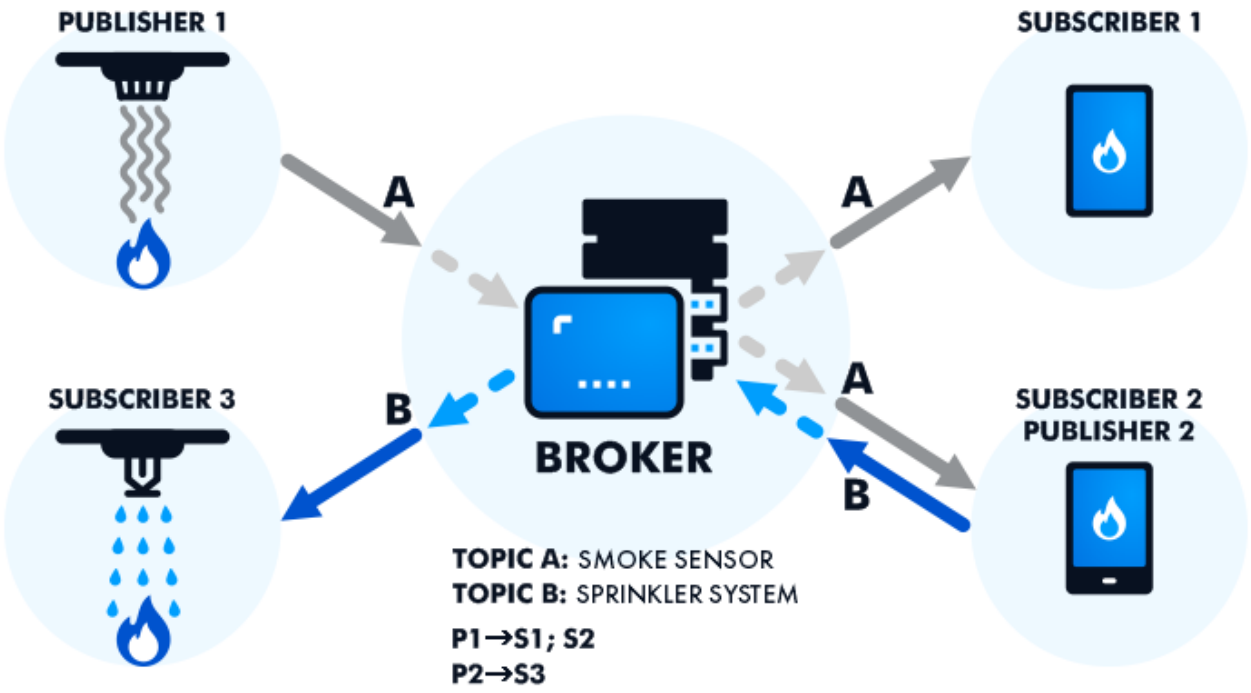


Fig 4: MQTT workflow

MQTT messages

A “publish” occurs when a customer wishes to submit data to the broker. A client will "subscribe" to a subject or topics if they desire to get data from the broker. When a client subscribes to a topic, it will get all future messages that are published on that subject.

The publisher additionally sends a QoS (Quality of Service) level along with the message. This level specifies the message's delivery guarantee. The following are the levels of QoS:

- At most once - The broker will only get the message "at most once" after it is published. This level should not be utilized for mission-critical information since it increases the danger of the message not being received by the intended recipients.

- At least once - The publisher will continue to resend the message until it obtains a response from the broker on the specific message. In other words, ensuring that the message is received is more critical than ensuring that it is only received once. This is most likely the most used QoS level.

- Exactly once - The publisher and the broker collaborate to ensure that the broker receives and acts on messages only once. This necessitates some more effort in the form of a four-part handshake. This is the safest QoS level, but it's also the slowest, therefore it's only used when absolutely required.

Key takeaway

MQTT (Message Queuing Telemetry Transport) is an IBM-developed lightweight messaging protocol that was initially introduced in 1999.

The MQTT protocol was established with the goal of connecting sensors on oil pipelines to communications satellites with the least amount of battery loss and bandwidth use possible.

2.8 ZigBee

Zigbee is a standards-based wireless technology that enables low-cost, low-power wireless M2M and internet of things (IoT) networks.

Zigbee is an open standard for low-data-rate, low-power applications. This allows for the mixing of implementations from different manufacturers in theory, but in practice, vendors have expanded and personalized Zigbee devices, resulting in interoperability concerns. Unlike Wi-Fi networks, which employ a mesh networking protocol to avoid hub devices and establish a self-healing design, Zigbee networks support significantly lower data rates and utilize a mesh networking protocol to avoid hub devices and build a self-healing architecture.

Furthermore, these mesh networks self-configure when devices are added or withdrawn, and their interconnection allows for a greater coverage range than competitors like Bluetooth Low Energy.

Despite this, Zigbee remains a low-range communication standard, with a coverage range of up to 100 meters. As a result, it's an excellent solution for personal spaces such as smart homes or in-building operations.

A coordinator, a router, and a device make up a Zigbee network. The coordinator serves as the network's hub, connecting all of its components and managing data transfer and storage.

The routers in the network support Zigbee coordinators. These are data-transfer devices that send and receive data from a variety of devices, including televisions, doors, cameras, and thermostats.

Zigbee is known for its secure connections, which is why it's often the chosen choice for personal area networks, particularly in corporate networks with sensitive data.

Because of its low power consumption, Zigbee is a better choice for IoT connectivity than Wifi. It's utilized for a variety of tasks, such as smart building automation, temperature management, and more.

Zigbee is an ideal partner for any business where little amounts of data are exchanged rarely, such as remote monitoring or in-building applications.

Finding ZigBee’s place in the industrial IoT

The properties of ZigBee set it unique from other conceivable IoT protocols, allowing it to carve out its own niche in the market. Because of its mesh topography, it can handle longer distances than Bluetooth Low Energy, and its extremely low energy consumption makes it more IoT-friendly than Wi-Fi.

According to Gigaom, after the chips are extensively utilized in houses, the next stage will be to include a ZigBee chip in cellphones. Because the homeowner's primary method of communicating with sensors in the home is through their smartphone or tablet, having such a chip built inside would put it in the hands of millions of smartphone owners who could turn it on at the touch of a button.

ZigBee radios are currently installed in homes via a hub, router, or set-top box, making their adoption by homeowners reliant on service providers and early adopters of products such as the Almond Router, SmartThings hub, or Revolv hub.

Challenges and Shortcoming

In the battle to become an internationally acknowledged IIoT standard, Zigbee has its work cut out for it. The space to become the solution to future IoT solutions is increasingly competitive, with established competitors such as Wi-Fi and Bluetooth, as well as newcomers Thread.

	Z-Wave	ZigBee	WeMo	Thread
Operating range	100 feet	35 feet	100 feet	100 feet (theoretical)
Max no. devices	232	65,000	Router-dependent	250-300
Data rate	9.6-100 kbps	40-250 kbps	Router-dependent	250 kbps
Frequency	908/916 MHz (U.S.)	915 MHz/2.4 GHz	2.4 GHz	2.4 GHz
Network type	Mesh	Mesh	Star	Mesh
Needs hub?	Yes	Yes	No	Yes

However, ZigBee already has a considerable presence in the market. In fact, if you have a Nest thermostat, Comcast's new router, or a Hue lighting, you already have ZigBee chips.

Unfortunately, the specification has had several difficulties in recent years, with interoperability issues highlighting much of its content. The issue is that the standard is more than simply a wireless transport method; it also includes a layer of software that can build profiles that conflict with other ZigBee profiles. So, unlike Wi-Fi, it's possible that two ZigBee-enabled devices won't be able to communicate with one another.

Another obstacle that ZigBee must overcome is Z-Wave, its closest competitor. According to Gigaom, nine out of ten sensors use the proprietary Z-wave protocol instead of ZigBee, and more startups are releasing Bluetooth Low Energy devices that can talk with smartphones.

Key takeaway

Zigbee is a standards-based wireless technology that enables low-cost, low-power wireless M2M and internet of things (IoT) networks.

Zigbee is an open standard for low-data-rate, low-power applications.

2.9 Bluetooth

Bluetooth was created by Ericsson in 1994 to provide wireless headsets. Bluetooth has since evolved into a wide range of products, including Bluetooth headsets, speakers, printers, game controllers, and much more.

Bluetooth is particularly vital for the Internet of Things, which includes smart homes and industrial applications, which is quickly developing. It's a low-power, low-range, high-bandwidth solution for connectivity. When Bluetooth devices (for example, your phone and your wireless speaker) connect, the parent-child paradigm is used, with one device acting as the parent and the others as the children. The parent communicates with the child, and the youngster listens for the parent to communicate with him or her.

Because a Bluetooth parent can have up to seven children, your computer can connect to several devices through Bluetooth at the same time. A “piconet” is a Bluetooth connection between two or more devices.

A device can be a parent in one piconet and a kid in another at the same time, and the parent-child relationship can also change. When you put your Bluetooth device in pairing mode to connect it, it becomes the parent for a short time in order to establish a connection before connecting as the child.

Bluetooth, unlike WiFi, which we discussed in the previous chapter, was designed for portable devices and related applications, so it excels when you need to connect two devices with minimal configuration. Furthermore, because Bluetooth uses weak signals, there is less interference, and devices may interact even in noisy surroundings.

Machines sending short bursts of data in excessively loud surroundings are common in the Industrial Internet of Things. WiFi is too difficult to set up when there are hundreds of sensors and devices sending data.

The lesser bandwidth of Bluetooth is a disadvantage, although for many industrial applications, this isn't an issue.

Bluetooth can also be used in a smart home. Many smart home gadgets don't require high bandwidth connections, and setting up Bluetooth is significantly easier.

Furthermore, recent Bluetooth versions may build a self-healing mesh network, which implies that individual devices can still interact even if one is removed or loses power. If your door locks, HVAC system, washer, dryer, fridge, and lights are all connected, you don't want them to all fail at the same time.

Bluetooth Strengths and Weaknesses

Bluetooth's main strength is communication between Bluetooth-enabled devices such as smartphones and tablets and specialized Bluetooth-enabled equipment. Bluetooth requires a gateway for access since, unlike cellular or satellite communications, it does not connect directly to the Internet. Bluetooth now has a range of roughly 328 feet (100 meters), making it ideal for use in the home and small business. Multiple gateways can be set up throughout a facility to receive Bluetooth communications for larger spaces.

Bluetooth 5 was announced by the Bluetooth SIG in March 2016, and it will double the existing range of the technology, perhaps making Bluetooth more appealing for IoT applications than Wi-Fi because it uses less power. Despite the improved range, Bluetooth IoT requires device-to-device communications to take place within a specific radius, and after the devices have communicated, their data must still reach a gateway for Internet access. When compared to satellite or wireless cellular networks, which can monitor object data regardless of location and interact with the internet instantaneously, satellite or wireless cellular networks are superior. Cellular connectivity will continue to be a better choice in remote locations, such as when used to track oil and construction equipment.

Bluetooth Connecting IoT in the Future

The roles and applications of Bluetooth in autos and households will continue to increase and expand. It's not out of the realm of possibility to receive automatic traffic updates or weather reports on your dashboard throughout your daily commute, or to script home automations utilizing Bluetooth to configure lighting, thermostats, and home entertainment systems for the appropriate mood or occasion. Industry will find Bluetooth a more commonplace, cost-effective alternative for IoT/M2M communications as gateway development gets easier and less expensive

(PCWorld.com claims that a gateway can be created on a $35 Raspberry Pi computer). Bluetooth IoT could lead to smarter waste management in cities, more accurate health monitoring, and swift communication across a range of digital devices.

Key takeaway

In addition to the features discussed above, Bluetooth may be used to track interior assets by deploying several Bluetooth beacons and triangulating position based on their respective signal strengths.

GPS is fantastic for outside applications, but it has inherent accuracy limitations and fails indoors when sensors/devices are unable to receive the GPS satellite signal.

Bluetooth is a viable alternative for many indoor Internet of Things applications because of its advantages in loud conditions and ease of setup.

2.10 CoAP

In the Internet of Things, the Confined Application Protocol (CoAP) is a customized web transfer protocol for usage with constrained nodes and constrained networks. CoAP is a protocol that allows basic, restricted devices to connect to the Internet of Things, even across constrained networks with poor bandwidth and availability. Machine-to-machine (M2M) applications such as smart energy and building automation are common uses.

The main features of CoAP protocols are:

- Web protocol used in M2M with constrained requirements
- Asynchronous message exchange
- Low overhead and very simple to parse
- URI and content-type support
- Proxy and caching capabilities

Architecture

CoAP is a document transport protocol, similar to HTTP. Unlike HTTP, CoAP is created with restricted devices in mind.

HTTP TCP transfers are substantially larger than CoAP packets. To save space, bitfields and mappings from strings to integers are frequently utilized. Packets are simple to create and can be parsed in place on restricted devices without spending additional RAM.

CoAP uses UDP rather than TCP. Connectionless datagrams are used to communicate between clients and servers. In the application stack, retries and reordering are implemented. By eliminating the need for TCP, compact microcontrollers may be able to provide complete IP networking. CoAP enables the usage of UDP broadcast and multicast for addressing.

The client/server model is used in CoAP. Clients send requests to servers, and servers respond. Clients have access to resources that they can GET, PUT, POST, and DELETE.

Through simple proxies, CoAP is meant to work with HTTP and the RESTful web in general.

CoAP can be used on top of SMS and other packet-based communications protocols because it is datagram-based.

CoAP can be represented as follows from the abstraction protocol layer:

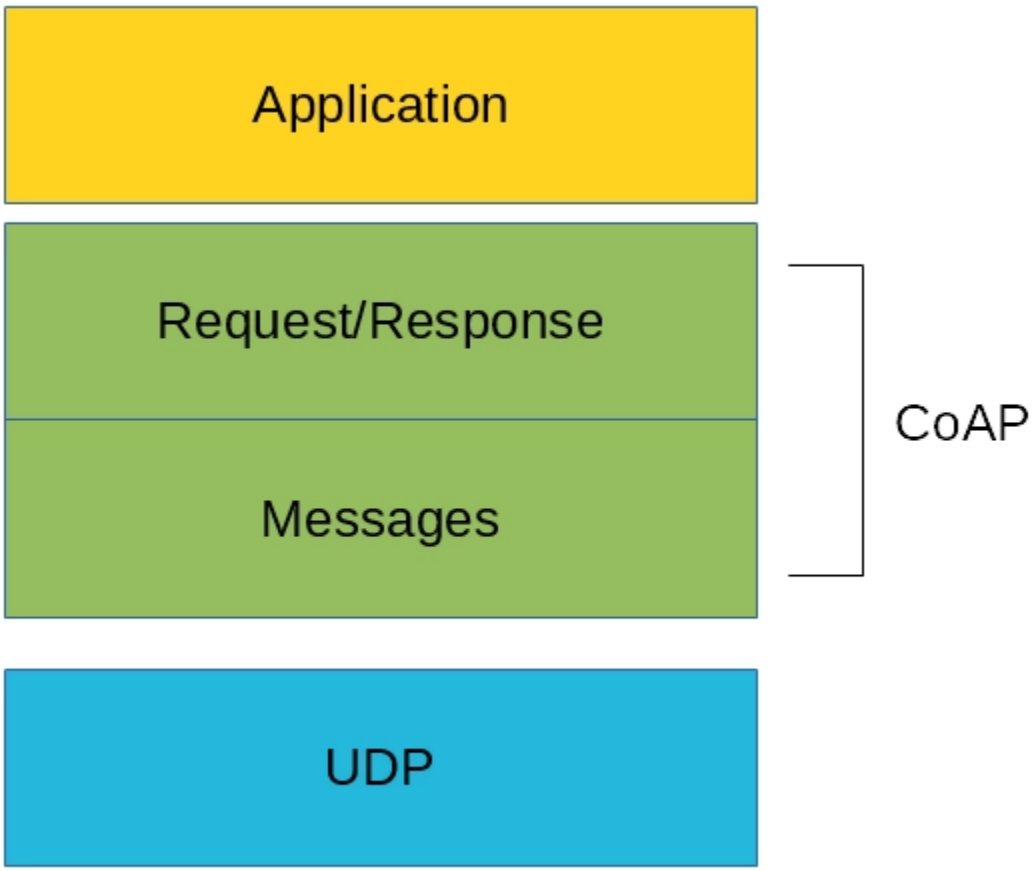


Fig 5: CoAP

As you can see, the CoAp protocol is made up of two layers: messages and request/response. The Messages layer is responsible for UDP and asynchronous messages. Based on request/response messages, the Request/Response layer manages request/response interaction.

Four different message types are supported by CoAP:

- Confirmable
- Non-confirmable
- Acknowledgment
- Reset

The CoAp protocol, it's helpful to define a few terminologies.

- Endpoint - An object that takes part in the CoAP protocol. An Endpoint is usually associated with a host.
- Sender - The person or thing who transmits a message.
- Recipients - A message's final destination.
- Client - The sender of a request and the recipient of the response.
- Server - The entity that accepts a client's request and responds with a response to the client.

Key takeaway

In the Internet of Things, the Confined Application Protocol (CoAP) is a customized web transfer protocol for usage with constrained nodes and constrained networks.

CoAP is a protocol that allows basic, restricted devices to connect to the Internet of Things, even across constrained networks with poor bandwidth and availability.

2.11 UDP

User Datagram Protocol (UDP) is less widespread in IoT (and data transmission in general) than TCP. However, IoT manufacturers favor UDP since it requires fewer network resources to transmit and does not require a continual connection between the two endpoints. To put it another way, it uses less data and utilizes less energy.

Resource constrained devices

Low-power, lossy networks are frequently used by IoT devices (LLNs). Because LLNs are optimized for power efficiency, they have a limited number of resources. The Constrained Application Protocol (CoAP) was created expressly to assist these devices in communicating, and it works on UDP-capable devices.

Low downlink transmissions

UDP does not acknowledge that a communication has been received. Although the sender has no way of knowing whether the data packets have reached, the exchange requires less downlink budget. UDP can be a useful communication protocol for communications with limited downlink allowances.

Low power applications

Because IoT devices send data on a regular basis, losing a single datapoint is rarely a problem. Every time a datapoint is lost or contains an error, attempting to resend the data uses an additional battery. Because UDP does not provide an acknowledgement, the device can shut down faster after sending or receiving data. This makes UDP appealing to developers who want to get the most out of their resources.

UDP Security

While UDP is simple to set up and has a lower overhead, it makes your devices more exposed to cyber threats.

It's more difficult to deliver a packet to a specific application using TCP because it establishes a direct connection and provides a sequence number. Intercepting, altering, and replicating data packets to attack a device, application, or network is easy with UDP.

Botnets (large networks of compromised computers, including IoT devices) can also deliver false packets to a computer using UDP. Bots frequently transmit spam to non-existent ports, forcing the computer to respond by stating that the port is inaccessible. This swiftly diverts network resources away from legitimate data transmissions, potentially causing the network to go down.

You must be vigilant about IoT security with UDP and be certain that your hardware and network-level security can secure your customers' data and keep your devices from becoming liabilities.

UDP datagrams

UDP traffic is carried in datagram packets, each of which is made up of a single message unit. The first eight bytes hold the header information, but the remaining bytes hold the actual message. The header of a UDP datagram is broken into four

pieces, each of which is two bytes long. These are the components:

- Source port - This 16-bit (2-byte) information is used to identify the data-sending sender port. The range of permitted UDP port numbers is 0 to 65535.

- Destination port - This 16-bit data is used to identify the port on which the data will be received by the receiver. The range of permitted UDP port numbers is 0 to 65535. This parameter is mandatory and identifies the receiver's port.

- Length - The length field gives the UDP packet's total length (UDP header and UDP data). This specific field is a 16-bit field. In the absence of UDP data, the Length field must be at least 8 bytes long.

- Checksum - The checksum value computed by the sender before delivering the data to the recipient is stored in this field. UDP checksums keep message data from being tampered with. The checksum value is an encoding of the datagram data that is calculated by the sender first and then by the receiver. In UDP, checksums are optional, whereas in TCP, checksums are required.

UDP Features

The User Datagram Protocol offers properties that make it useful for applications that can suffer data loss. Consider the following scenario:

- It permits packets to be lost and received in a different order than when they were sent, making it ideal for real-time applications where latency is an issue.

- It's suitable for transaction-based protocols like DNS and Network Time Protocol (NTP).

- Gaming, audio or video conferencing, and streaming media are examples of applications where a large number of customers are connected and real-time mistake correction isn't required.

Advantages of UDP

The UDP/IP stack has the following benefits over the TCP/IP stack:

1) It is superior to TCP for applications that require continual data flow, large amounts of data, and a higher level of speed than reliability.

2) Because it provides point-to-multipoint transmission, UDP is the best choice for multicast and broadcast applications. In contrast to TCP/IP, where the sender is responsible for each packet, the sender does not need to keep track of data retransmission for many receivers.

3) UDP has a tiny packet header overhead (only 8 bytes), but TCP has a header of 20 bytes.

Key takeaway

User Datagram Protocol (UDP) is less widespread in IoT (and data transmission in general) than TCP.

UDP traffic is carried in datagram packets, each of which is made up of a single message unit.

The User Datagram Protocol offers properties that make it useful for applications that can suffer data loss.

2.12 TCP

TCP is used in conjunction with the Internet Protocol (IP), which specifies how computers exchange data packets. TCP and IP are the fundamental rules that govern the Internet. The Internet Engineering Task Force (IETF) defines TCP in RFC 793, a Request for Comment (RFC) standards document. TCP is a connection-oriented protocol, which means it establishes and maintains a connection until both ends' application programs have done exchanging messages.

TCP (Transmission Control Protocol) is a connection-oriented transport-layer protocol that provides end-to-end reliable and ordered data transfer between applications operating on Internet hosts. It was created four decades ago. TCP has remained the dominant transport-layer protocol on the Internet ever since, with many key applications (such as the WWW, e-mail, file transfer, instant messaging, and so on) relying on it. However, as the Internet has grown beyond its core qualities, TCP has faced and overcome enormous hurdles.

TCP has been effectively employed on mobile networks, despite the fundamental premise of TCP congestion control that the Internet is a wired network and the challenges that arise in wireless environments. TCP's extensive presence in mobile

phones today is due to optimization approaches.

The Internet of Things is a new challenge for TCP (IoT). Tens of billions of affordable devices (e.g., sensors, actuators, etc.) attached to everyday things will be connected to the Internet to enable smart scenarios in this big networking trend. IoT devices, on the other hand, are frequently constrained (in terms of memory, computation, and energy), employ low-speed and error-prone connectivity, and have multi-hop networks.

TCP has been heavily chastised as a transport-layer protocol for the Internet of Things due to these challenging networking settings. As a result, many early IP-based IoT implementations had to rely on UDP for application-layer dependability. The Constrained Application Protocol (CoAP), a lightweight RESTful application-layer protocol designed at the IETF for the Internet of Things, took the same approach. Similarly, the IPv6 over Low Power Wireless Personal Area Networks project produced enhancements for UDP header compression while neglecting TCP.

TCP in the Internet of Things

End-to-end communication between IoT devices and other computers on the same network is enabled through Connecting Things to the Internet. Cloud backend systems can communicate with IoT devices in this manner (e.g., for sensor data centralization, actuator triggering, and device management). The primary protocol and architectural alternatives for end-to-end connectivity with IoT devices using TCP are described in this section. We concentrate on scenarios involving HTTP, CoAP, MQTT, and AMQP.

HTTP

HTTP has several advantages as an IoT protocol: it is a free, open standard, and HTTP development tools are plentiful because it is the most widely used application-layer protocol on the Internet. Furthermore, it is the protocol that has the best chance of passing security middle boxes. In an IoT scenario, HTTP/2 is more appropriate than HTTP/1.1. A binary, compact header is included in HTTP/2, and pseudo-header fields can be compressed using the HPACK format. An IETF specification for leveraging HTTP/2 in IoT settings is currently being developed. As a result, HTTP (and consequently TCP) is a viable option for IoT device connection.

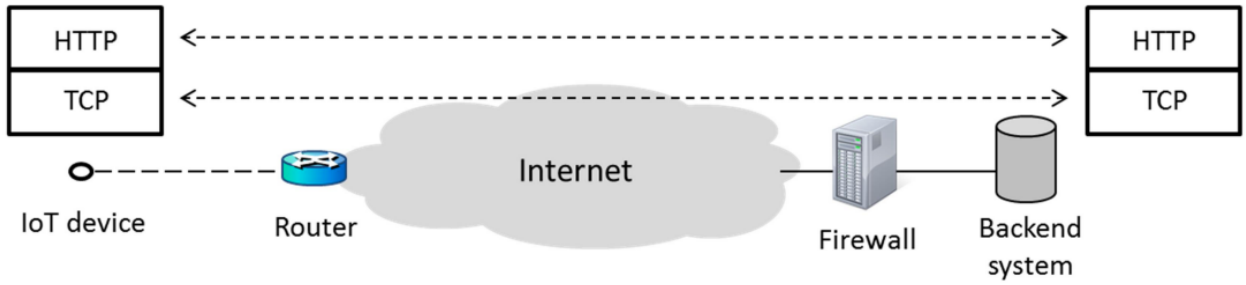


Fig 6: HTTP

CoAP

CoAP was created as a lightweight alternative to HTTP/1.1, preserving the fundamental principles of HTTP, such as the REST architecture, but with a lot less complexity. CoAP provides interoperability with HTTP using protocol translation proxies, allowing it to fully realize its promise. CoAP was created with UDP in mind, with stop-and-wait dependability as an option. However, implementation experience has shown that in order to circumvent connectivity limits imposed by corporate firewalls, CoAP via TCP or WebSocket's must be enabled.

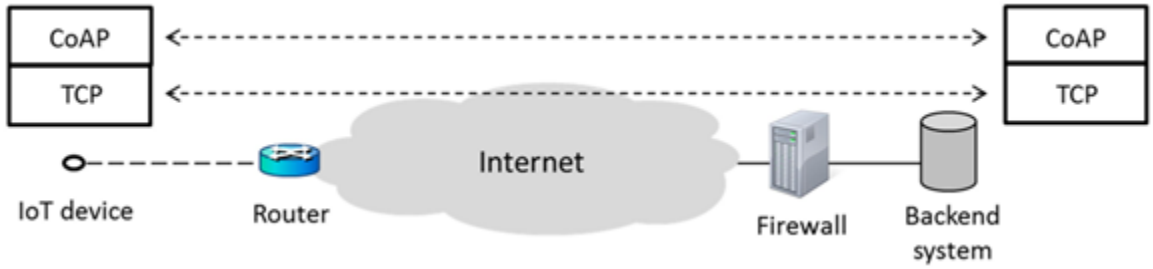


Fig 7: CoAP over TCP

MQTT

MQTT is a communications protocol developed by ISO/IEC for monitoring applications. It is built on the publish-subscribe paradigm, in which publishers (such as sensors) send data messages to a broker, who then distributes the messages to subscribers (e.g., backend systems). The broker becomes more complex as a result of this flexible approach. MQTT also specifies a lightweight header format and necessitates a minimal code footprint. A TCP connection is created between the broker and the publisher or subscriber in MQTT.

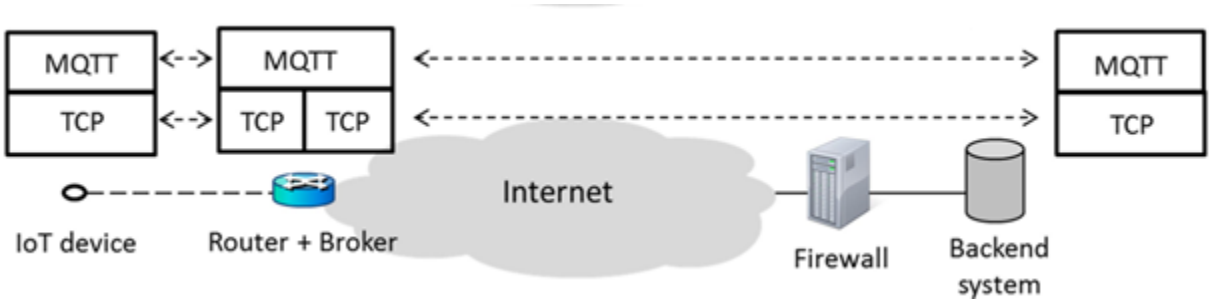


Fig 8: MQTT

AMQP

AMQP is another ISO/IEC messaging protocol that was created with the financial industry in mind. It is compatible with a number of broker-based architectures, including publish subscribe. AMQP has more complicated mechanisms than MQTT (for example, fine-grained control, queue management, and error handling), but at the cost of more implementation complexity and bigger message headers. TCP is also used in AMQP.

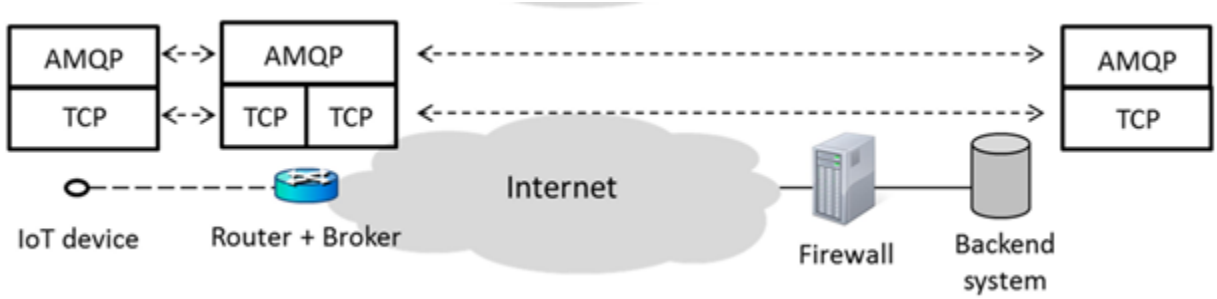


Fig 9: AMQP

Key takeaway

TCP is used in conjunction with the Internet Protocol (IP), which specifies how computers exchange data packets.

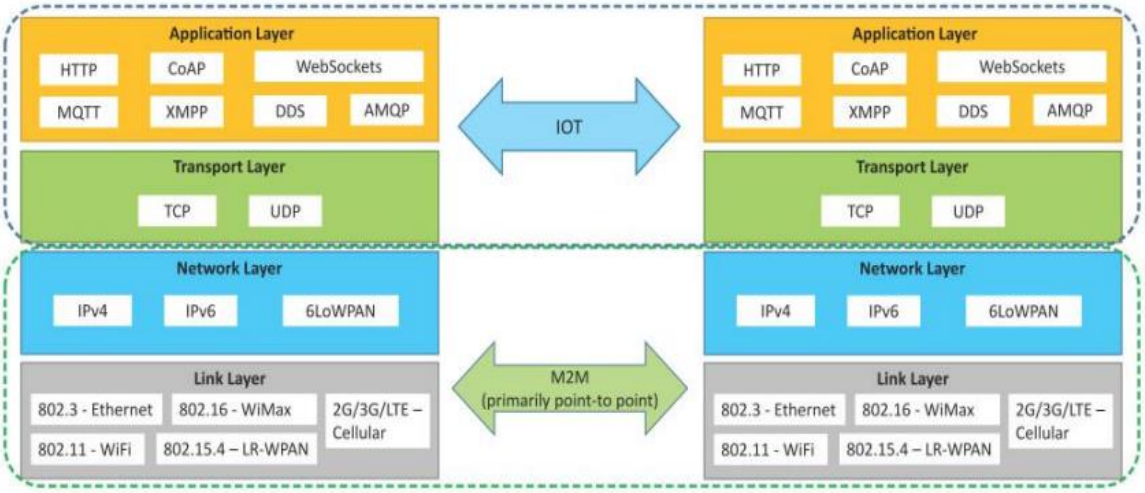
TCP and IP are the fundamental rules that govern the Internet.

TCP (Transmission Control Protocol) is a connection-oriented transport-layer protocol that provides end-to-end reliable and ordered data transfer between applications operating on Internet hosts.

References:

1. Olivier Hersent, David Boswarthick, Omar Elloumi “The Internet of Things key applications and protocols”, willey
2. Jeeva Jose, Internet of Things, Khanna Publications
3. Michael Miller “The Internet of Things” by Pearson
4. Raj Kamal “INTERNET OF THINGS”, McGraw-Hill, 1ST Edition, 2016
5. ArshdeepBahga, Vijay Madisetti “Internet of Things (A hands-on approach)” 1ST edition, VPI publications,2014
6. Adrian McEwen, Hakin Cassimally “Designing the Internet of Things” Wiley India

Communication protocols used for M2M local area networks



- Communication protocols: M2m and IoT can differ in how the communication between the machines are device happens. M2M uses other proprietary or not IP based communication protocol for communication with in the M2M area networks.
- Commonly uses M2m protocol include zigbee, Bluetooth, ModBus, wireless M-Bus, power line communication(PLC),6LOWPAN,IEEE 802.15.4, Z-WAVE etc.
- The focus of communication in M2M is usually on the protocols below the network layer. Focus of communication in IoT is usually a protocol in network layer such as http web sockets, MQTT, XMPP, DDS, AMQP.

Differences between Machines in M2M and Things in IOT

- **Machines in M2M vs Things in IoT:** The " things " IoT refers to Physical objects that have unique identifier and can sense and communicate with the external environment or their internal physical states.
- The unique identifiers the things in IoT are the IP addresses. Things have software component for accessing processing and storing sensor information on controlling actuator connected.
- IoT systems can have heterogeneous things(e.g: a home automation IoT system can include IoT devices of various types such as fire alarms , door alarms, lighting control devices.
- M2M systems, in contrast to IoT typically have homogeneous machine types with in an M2M area network.
- **Hardware versus software emphasis**: while the emphasis of M2M is more on hardware with embedded modules, the emphasis modules, the emphasis of IoT is more on software .

IoT devices run specialist software sensor Data Collection, data analysis and interfacing with cloud through IP based communication.

Data collection and analysis:

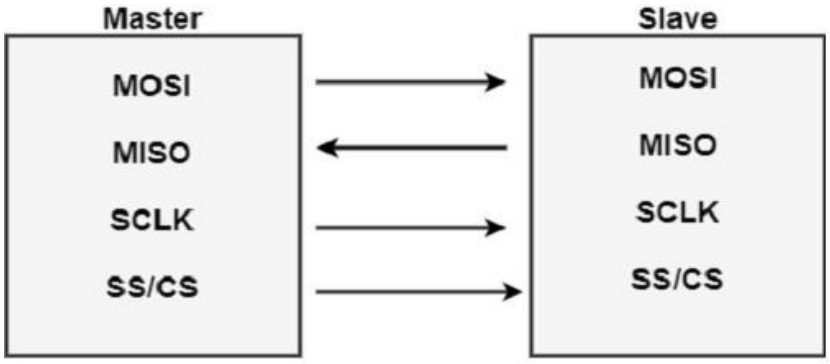
- M2M data is collected in point solutions and often in on premises storage infrastructure. In contrast to M2M,the data in IoT is connected in the cloud.
- The analytical component analysis the data and stores the result in the cloud database. Data and analysis results are visualized with the cloud based applications.
- The centralized controller is aware of the status of all the nodes and send Control Commands to the nodes

Applications:

- M2m data is collected in point solutions and can be accessed by on premises application diagnosis applications, service management applications , and on-premises enterprise

Use of SPI and I2C interfaces? Illustrate how to interface a switch

- SPI stands for the Serial Peripheral Interface. SPI is a general-objective synchronous serial interface. During an SPI transfer, send and receive data is simultaneously shifted out and in serially.
- The SPI is used to authorize a microcontroller to communicate with peripheral devices such as E^2PROMs.
- SPI devices transmitted using a master-slave relationship. Because of its lack of built-in device addressing, SPI needs more effort and more hardware resources than I^2C when more than one slave is involved. But SPI tends to be easier and more efficient than I^2C.



MOSI – MOSI represents Master Output Slave Input. It can send data from the master to the slave.

MISO – MISO represents Master Input Slave Output. It can send data from the slave to the master.

SCK or SCLK (Serial Clock) – It is frequently used to signal the clock.

SS/CS (Slave Select / Chip Select) – The master uses it to send data by selecting a slave.

SPI Uses

There are various uses of SPI, which are as follows –

- **It can full-duplex communication.**
- **It is used for the arbitrary choice of message sizes, contents and purpose.**
- **It is used for low power requirements.**
- **It has separate MISO and MOSI lines so that the data can be sent and received simultaneously.**
- **It can do simple hardware interfacing.**
- **There is no requirement for a unique address of the slave in this protocol.**

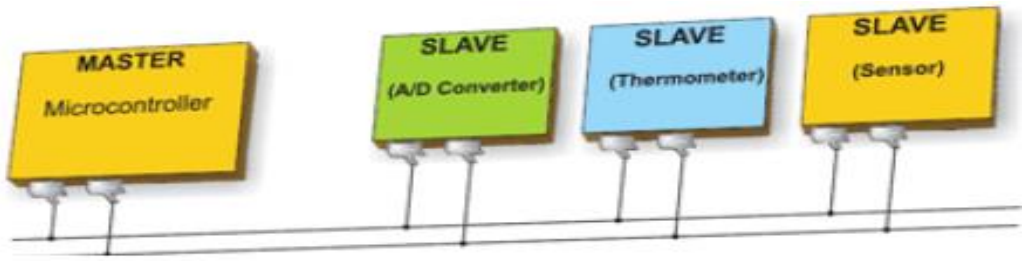
I2C (Inter Integrated circuit)

➢Inter-integrated circuit (I2C) is a system for serial data exchange between the microcontrollers and specialized integrated circuits of a new generation.

➢ It is used when the distance between them is short (receiver and transmitter are usually on the same printed board). Connection is established via two conductors.

➢ One is used for data transfer and the other is used for synchronization (clock signal).

- As seen in the following figure, one device is always a master. It performs addressing of one slave chip before the communication starts. In this way, one microcontroller can communicate with 112 different devices. Baud rate is usually 100 Kb/sec (standard mode) or 10 Kb/sec (slow baud rate mode). Systems with the baud rate of 3.4 Mb/sec have recently appeared. The distance between devices, which communicate over an I2C bus is limited to several meters.



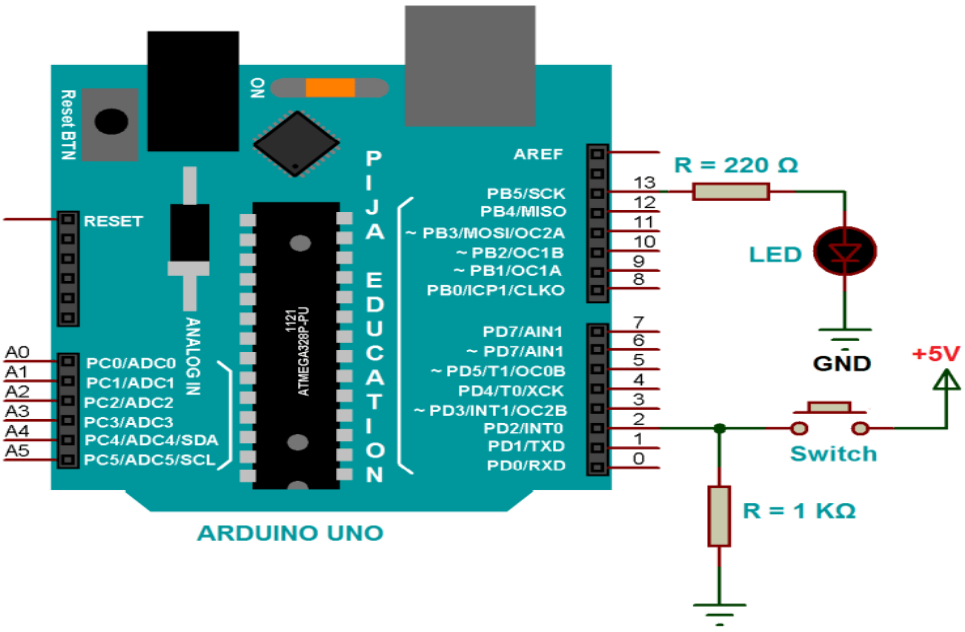
I2C uses

1. Reading certain memory ICs
2. Accessing DACs and ADCs
3. Transmitting and controlling user-directed actions
4. Reading hardware sensors
5. Communicating with multiple micro-controller

How to interface a switch

- Required hardware or components for Interfacing of Switch with Arduino Uno
- CIRCUIT DIAGRAM : INTERFACING OF SWITCH AND ARDUINO
- CONNECTION TABLE
- ARDUINO CODE : INTERFACING OF SWITCH AND ARDUINO
- ARDUINO PROGRAMMING CODE EXPLANATION

➢ We all know switch is an input device and used; to make or break an electrical connection. Now let's learn Interfacing of Switch and Arduino, Here we will use switch to take input signal to the microcontroller so that we will be able to interface more input devices to the microcontroller, like sensors output as input to microcontroller.



CONNECTION TABLE

S.N.	Arduino	LED
1.	13	Anode (+)
2.	GND	Cathode (-)

S.N.	Arduino	Switch	Pull Down Resistor
1.	GND		One Terminal
2.	2 (common to both)	One T erminal	Other terminal
3.	+5V	Other terminal	

ARDUINO CODE : INTERFACING OF SWITCH AND ARDUINO

```
int buttonState = 0;

void setup() {
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  buttonState = digitalRead(2);
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
  } else if (buttonState == LOW) {
    digitalWrite(13, LOW);
  }
}
```

ARDUINO PROGRAMMING CODE EXPLANATION

Define a global variable buttonState, we will use this to read **state** of pin number 2. State of digital pin can be either LOW=0 or HIGH=1 set default to LOW(0)

```
int buttonState = 0;
```

Here we assigned pin number 2 as input and 13 as ouput of arduino using pinMode(pinNumber, mode) function.

```
void setup() {
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}
```

- Here we are using a new inbuilt function that is **digitalRead(pinNumber)**, it is just opposite to digitalWrite where we set two states (HIGH or LOW) of pin.

- The digitalRead( ) function will read the state of the arduino pin, Here we read the state of pin number 2 of the arduino. After Reading the state of Digital pin 2, we will store the value to variable buttonState, So now buttonState value will be Either LOW=0 or HIGH=1

- If the button is pressed down, that means current from the 5V power source can flow to pin number 2 of arduino which makes pin 2 state HIGH. Now at this time if we read the state of pin number 2 using digitalRead( ) function

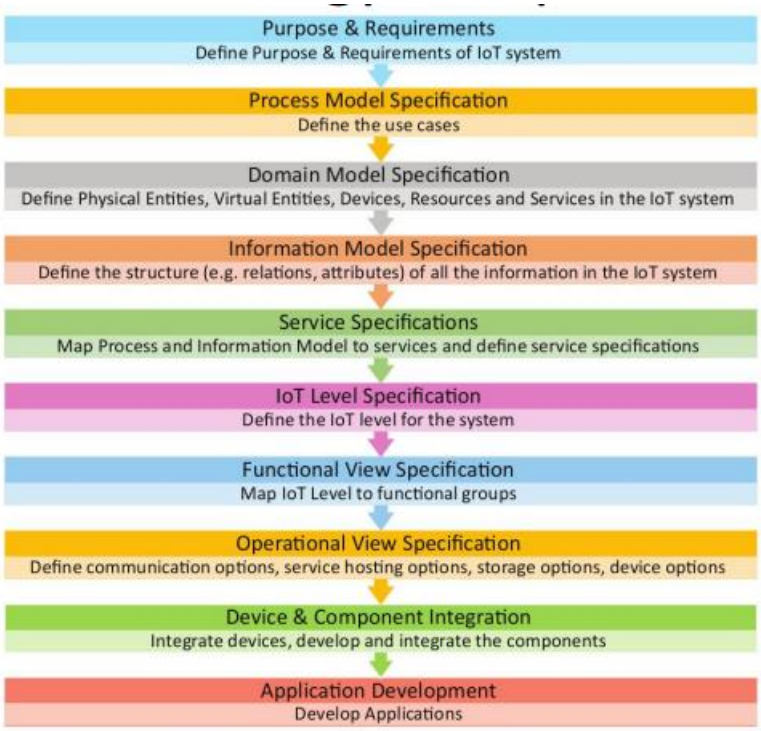
then the value of button State is HIGH (1). Now we will check, using if function does button State is HIGH, if yes, then make Pin number 13 also HIGH this will turn ON the Led connected on pin 13.

```
void loop() {
  buttonState = digitalRead(2);
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
  }
```

As you press the button you find the LED to glow and as you release you find it OFF. Else If button is not pressed, that means buttonState read from pin number 2 is LOW, if LOW then make Pin number 13 also LOW this will turn OFF the Led connected on pin 13.

```
 else if (buttonState == LOW) {
    digitalWrite(13, LOW);
  }
}
```

various service types used in service specifications step of IoT system design methodology.



Step 1: Purpose & Requirements Specification

• The first step in IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behavior and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements, ...) are captured.

Step 2: Process Specification

• The second step in the IoT design methodology is to define the process specification. In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications

Step 3: Domain Model Specification

- The third step in the IoT design methodology is to define the Domain Model. The domain model describes the main concepts, entities and objects in the domain of IoT system to be designed. Domain model defines the attributes of the objects and relationships between objects.
- Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform. With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed.

Step 4: Information Model Specification

• The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored.

- To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations.

Step5:ServiceSpecifications

• The fifth step in the IoT design methodology is to define the service specifications. Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects.

Step 6: IoT Level Specification

• The sixth step in the IoT design methodology is to definetheIoTlevelfor the system. In Chapter-1, we defined five IoT development levels

Step 7: Functional View Specification

• The seventh step in the IoT design methodology is to define the Functional View. The Functional View (FV) defines the functions of the IoT systems grouped into various Functional Groups (FGs). Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.

Step 8: Operational View Specification

• The eighth step in the IoT design methodology is to define the Operational View Specifications. In this step, various options pertaining to the IoT system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options

Step 9: Device & Component Integration

• The ninth step in the IoT design methodology is the integration of the devices and components

Step 10: Application Development

• The final step in the IoT design methodology is to develop the IoT application.

Difference between Web of Things versus Internet of Things

IoT	WoT
• IoT is a network of Things, which are anything that can be connected in some form to the Internet	• WoT is web network created for proper handling and using the potential of IoT platforms to provide better future
• IoT is a hardware layer to connect everything to the Internet	• WoT is a software layer to connect everything to the web
• IoT deals with sensors, actuators, computation and communication interfaces. From a box of oranges with an RFID tag, to a smart city and to everyThing in between, all these digitally augmented objects make up the IoT	• WoT deals with protocols and web servers. All those applications for IoT devices make up the WoT
• There is a different protocol for each and every IoT devices	• WoT makes it easy by using single protocol for multiple IoT devices
• IoT platforms are hard to program due to multiple protocols	• Due to common API's to handle the protocol WoT programming is easier
• IoT standards and prototypes are not public. They are privately funded and are not publicly accessible Insecure data transmission	• WoT is free for everyone and can be accessed anywhere, anytime
• IoT is tightly coupled between the applications and networks	• whereas WoT in application layer is loosely coupled

Define an API? How will you implement the API in an application or service

Implementing the APIs in an Application or Service

An application or service consists of mashed APIs. Following are the steps for implementing the APIs:

1. Make an implementation table for sequences of API actions. A session of message exchanges between two ends of API may consist of number of actions.
   - Column 1 of each row has the actions in sequences which occur one by one.
   - Column 2 may specify for each action, the authentication code or method (device platform ID, such as MAC address) for secured communication to and from other end (server or application). Column 2 has authentication method for use that may involve user/password communication. User in case of IoT device platform is device platform ID, such as MAC address. Password can be some code internally generated at device platform using some algorithm using a secret key as input (Example 9.8).
   - Column 3 specifies the API inputs for initiating the action on event.
   - Column 4 specifies the API outputs for the inputs. The outputs communicate to other end and initiate the execution of methods, callback functions, generate requests or send responses.
2. Use the secure communication protocols, such as DTLS/TLS.
3. Use the standard formats for object or message exchanges: JSON, TLV or XML or REST style of URIs or URLs for message format (Section 3.2).
4. Use for access to remote methods standard protocols, such as SOAP (for XML-RPC) or JSON-RPC. RPC stands for remote procedure call. Procedure also means *function* in C/C++ or method in Java (Sections 3.3 and 3.4).
5. Use the standard client-server protocols, such as MQTT when using publish/subscribe or such as XMPP models of message or object exchanges (Section 3.2).
6. Use the standard client and server protocol for client-server http, or CoAP or ws protocol using client-response model of message or object exchanges (ws:// used in place of http:// for bi-directional message exchanges) (Section 3.4).
7. Use the standard methods for web object or message exchanges. For example, REST request-response model methods, such as HTTP GET/POST methods or WebSocket methods for bi-direction message exchanges (Section 3.4).

8. Use the scripts, for example, JavaScript or Node.js framework for creating set of codes in event model or codes for running at distributed computing systems (device platforms or services at web/cloud). Event model calls a `callback()` `[onEventAction()]` functions on inputs. The function may initiate sending a client data or sending a server response. Function runs once on each input-event.
9. Use the language or scripting language framework which provides wider support of libraries, community and test tools.

Example 9.15 explains the table for easy implementation of API.

Example 9.15

Problem

Drawing an implementation table for weather web APIs:

How does a table make it easy writing codes for weather web APIs? Use the sequences shown in Figure 9.4.

Solution

Table 9.4 gives the actions, inputs and outputs for weather APIs.

Table 9.4 Implementation table for weather web APIs

Sequence Number and Action	Authentication	Inputs	Outputs
1: New connection to weather service API	MAC address or deviceID/ ApplicationID/ API_ID	Location (City Name)	Message to weather web client to enable client connect to send request for weather messages
2: New connection to weather Service	MAC address or weather service API ID	Messages (city name, ClientID, time stamp)	Subscribe to the weather service, if a new city name (messaged from the location server)
3: New connection to weather service	MAC address or weather service API ID	Messages of weather web-service API on new city name	Subscription message(s) on new city to the weather service and messages (clientID, weather information with with time stamp from messaged data)
3: New response of weather service	MAC address or weather service ID	Weather messages and warnings for this and predictions of next two days for the city from repository	Response message(s)

Different wired and wireless connectivity we can used in IoT explain with example.

Communications or Internet connectivity allows the device to communicate with each other. The Internet connection can be wired, through Ethernet cables, or wireless, such as a Wi-Fi or cellular network. Communications technology is advanced to the extent that it makes data sharing and data access simple and seamless.

Wired Solutions:

- IoT technology is deployed in many ways so no single network solution is right. It depends on the situation and where the devices are located. Some of the factors affecting the selection of the type of network are network range, network bandwidth, power usage, interoperability, intermittent connectivity and security.

- A wired network uses Ethernet cable to connect to the network. The Ethernet cable is in turn connected to a DSL or cable to the network gateway. The wired networks are mature technology and it is easy to get plugged into if you already have phone lines, power lines, and coaxial cable lines.

- Even in the case of wireless network, those networks are usually connected to a wired network at some point; hence the most commonly used network is a hybrid of both wired and wireless network connectivity.

Wireless IoT Implementation:

- As most wired networks tend to be bulky and expensive, Wireless IoT implementations are the common solution. Setting up a wireless network is a simple process that involves configuring it to get it up and running in no time.

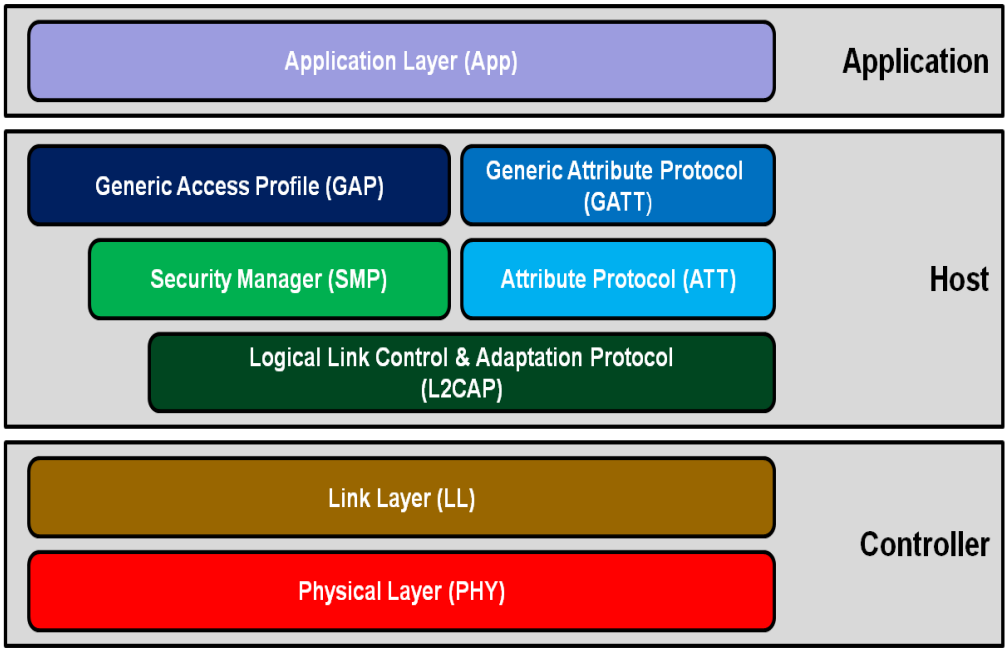
- With the evolution of network technologies, we see a wide range of solutions like RFID, Bluetooth, WiFi as well as the less familiar ones like - ZigBee, Z-Wave or UWB (Wltra Wide-Band).

There are 4 common communication models used by IoT 1. Device-to-Device, 2.Device-to-Cloud, 3.Device-to-Gateway, and 4.Back-End Data-Sharing.

The type of wireless implemented will depend on the communication model.

1. Device to Device uses Bluetooth, Z-Wave or Zigbee as it involves transmitting small amounts of data.
2. Device to Cloud uses WiFi or Cellular technology. Cloud connections allow users to obtain access to the device remotely.
3. Device to Gateway uses the network of your smart device like a smart phone or a smart watch. Examples of this are fitness trackers that upload data into your mobile app.
4. Backend Data Sharing extends the single device to cloud communications to authorized third parties. This can use any network connectivity like WiFi, Cellular or even satellite. It all comes down to the use case of your business.

BLE protocol stack with neat block diagram



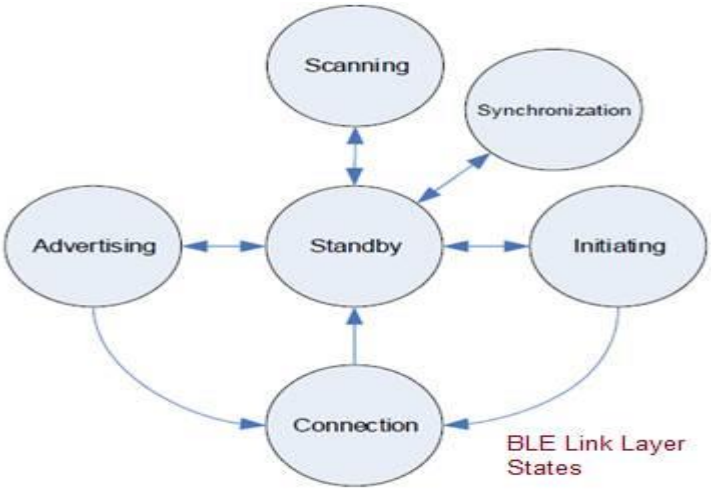
Physical Layer :

- The transmitter uses GFSK modulation and operates at unlicensed 2.4 GHz frequency band.
- Using this PHY layer, BLE offers data rates of 1 Mbps (Bluetooth v4.2)/2 Mbps (Bluetooth v5.0).
- It uses frequency hopping transceiver.
- Two PHY layer variants are specified viz. uncoded and coded.

Link Layer :

➢This layer sits above the Physical layer. It is responsible for advertising, scanning, and creating/maintaining connections.

➢The role of BLE devices changes in peer to peer (i.e. Unicast) or broadcast modes. The common roles are Advertiser/Scanner (Initiator), Slave/Master or Broadcaster/Observer. Link layer states are defined in the figure below.



HCI : It provides communication between controller and host through standard interface types. This HCI layer can be implemented either using API or by interfaces such as UART/SPI/USB. Standard HCI commands and events are defined in the bluetooth specifications.

L2CAP : This layer offers data encapsulation services to upper layers. This allows logical end to end data communication.

SMP : This security Manager layer provides methods for device pairing and key distributions. It offers services to other protocol stack layers in order to securely connect and exchange data between BLE devices.

GAP : This layer directly interfaces with application layer and/or profiles on it. It handles device discovery and connection related services for BLE device. It also takes care of initiation of security features.

GATT : This layer is service framework which specifies sub-procedures to use ATT. Data communications between two BLE devices are handled through these sub-procedures. The applications and/or profiles will use GATT directly.

ATT : This layer allows BLE device to expose certain pieces of data or attributes.

List out the essential features associated with PSOC4 BLE architecture

Application Layer :

• The BLE protocol stack layers interact with applications and profiles as desired. Application interoperability in the Bluetooth system is accomplished by Bluetooth profiles.

• The profile defines the vertical interactions between the layers as well as the peer-to-peer interactions of specific layers between devices.

• A profile composed of one or more services to address particular use case. A service consists of characteristics or references to other services.

• Any profiles/applications run on top of GAP/GATT layers of BLE protocol stack. It handles device discovery and connection related services for the BLE device.

List out the essential features associated with PSOC4 BLE architecture.

Features

The PSoC 4000 family has these major components:

■ 32-bit Cortex-M0 CPU with single-cycle multiply, delivering up to 14 DMIPS at 16 MHz

■ Up to 16 KB flash and 2 KB SRAM

■ A center-aligned pulse-width modulator (PWM) with complementary, dead-band programmable outputs

■ I2C communication block with slave, master, and multi- master operating modes

■ CapSense

■ Low-power operating modes: Sleep and Deep-Sleep

■ Programming and debugging system through serial wire debug (SWD)

■ Two current sourcing/sinking DACs (IDACs)

■ Comparator with 1.2 V reference

■ Fully supported by PSoC Creator™ IDE too