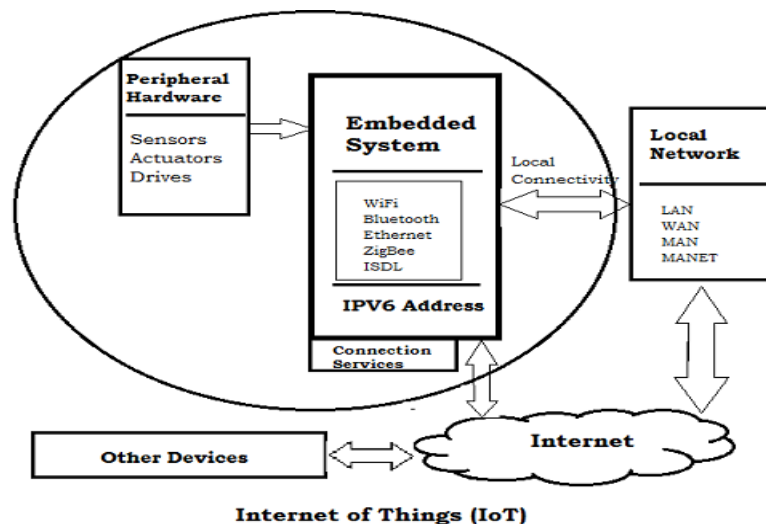


## Embedded Computing Logic

It is essential to know about the embedded devices while learning the IoT or building the projects on IoT. The embedded devices are the objects that build the unique computing system. These systems may or may not connect to the Internet.

An embedded device system generally runs as a single application. However, these devices can connect through the internet connection, and able communicate through other network devices.



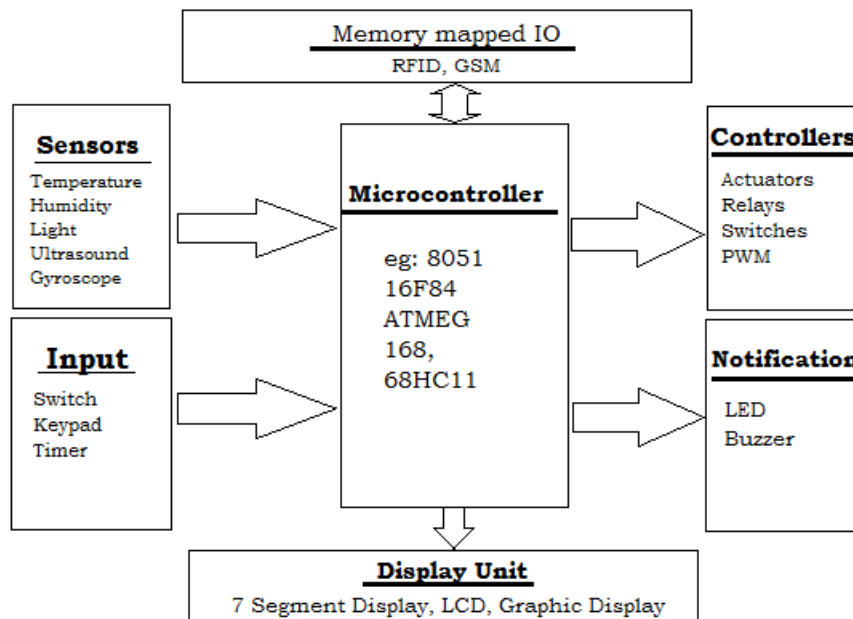
First developed in the 1960s for aerospace and the military, embedded computing systems continue to support new applications through numerous feature enhancements and cost-to-performance improvements of microcontrollers and programmable logic devices. Today, embedded computing systems control everyday devices which we don't generally think of as "computers": digital cameras, automobiles, smart watches, home appliances, and even smart garments. These embedded computing systems are commonly found in consumer, industrial, automotive, medical, commercial, and military applications.

Unlike general-purpose computers, embedded control systems are typically designed to perform specific tasks. The embedded computing system designer's task is to identify the set of components that will implement the system's functional, performance, usability, and reliability requirements, typically within tight cost and development timeline constraints. Accordingly, the selection of a microcontroller and its characteristics, including data processing capabilities, speed, peripherals, and power consumption, is one of the earliest and most critical aspects of system design.

Part of the designer's responsibility involves being aware of trends in their particular industry and taking advantage of relevant components and techniques. Let's look for examples among the top industries for microcontroller applications, the Internet of Things.

## Embedded System Hardware

The embedded system can be of type microcontroller or type microprocessor. Both of these types contain an integrated circuit (IC). The essential component of the embedded system is a RISC family microcontroller like Motorola 68HC11, PIC 16F84, Atmel 8051 and many more. The most important factor that differentiates these microcontrollers with the microprocessor like 8085 is their internal read and writable memory. The essential embedded device components and system architecture are specified below.



## Embedded System Software

The embedded system that uses the devices for the operating system is based on the language platform, mainly where the real-time operation would be performed. Manufacturers build embedded software in electronics, e.g., cars, telephones, modems, appliances, etc. The embedded system software can be as simple as lighting controls running using an 8-bit microcontroller. It can also be complicated software for missiles, process control systems, airplanes etc.

## Microcontrollers for Embedded Computing with IoT Devices

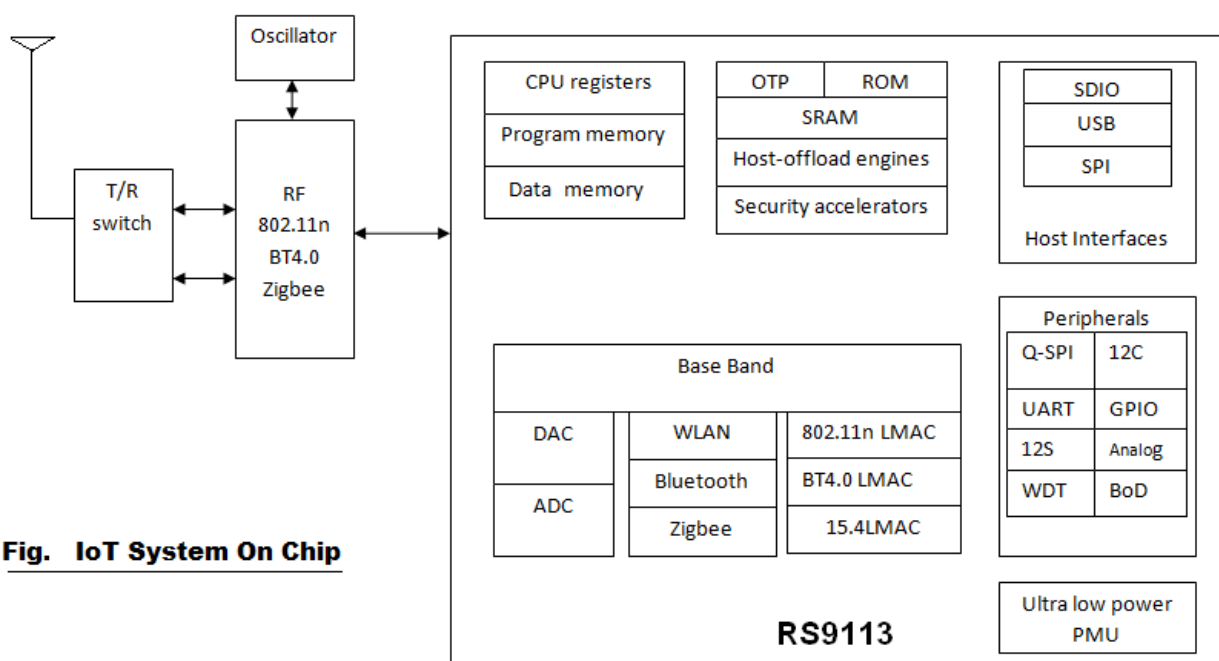
IoT devices are meant to be inexpensive, therefore the microcontroller needs to be chosen so that its capabilities are not underutilized by the application. The microcontroller specifications that determine the best part for your application are:

- **Bit depth:** The register and data path width impacts the speed and accuracy with which microcontrollers can perform non-trivial computations.
- **Memory:** The amount of RAM and Flash in a microcontroller determines the code size and complexity the component can support at full speed. Large memories have larger die area and component cost.

- **GPIO:** These are the microcontroller pins used to connect to sensors and actuators in the system. These often share their functionality with other microcontroller peripherals, such as serial communication, A/D, and D/A converters.
- **Power consumption:** Power consumption is critically important for battery-operated devices and it typically increases with microcontroller speed and memory size.

## System on Chips

System on Chip in IoT designed by Redpine Signals is discussed below. This IoT SoC supports WLAN, bluetooth and Zigbee systems on a single chip. It also supports 2.4 and 5GHz radio frequencies.



**Fig. IoT System On Chip**

As we know IoT is the technology which will provide communication between things, between things and people using internet and IP enabled protocols. As we have seen in IoT tutorial any IoT compliant system will have two major parts viz. front end and back end. Front end provides connectivity with physical world and consists of sensors while backend consists of processing and network connectivity interfaces.

Typical **IoT system on chip** support more than one RATs (Radio Access Technologies). It will have following modules.

- Transmit and receive switch.

- RF part mainly consists of Transmitter, receiver, oscillator and amplifiers.
- Memories i.e. Program memory, data memory to store the code and data
- Physical layer (baseband processing) either on FPGA or on processor based on complexity and latency requirement.
- MAC layer and upper protocol stacks TCP/IP etc. running on processor
- ADC and DAC to provide interface between digital baseband and analog RF portions.
- Various interfaces such as SDIO, USB, SPI etc to provide interface with the host.
- Other peripherals such as UART, I<sup>2</sup>C, GPIO, WDT etc. to use the IoT SoC for various connections.

As IoT system on chip supports multiple wireless protocols and RF hardware to support multiple frequency bands, following factors need to be carefully analyzed and to be optimized.

- Power-consumption
- Data-throughput
- Device-size
- Performance in terms of latency and other factors

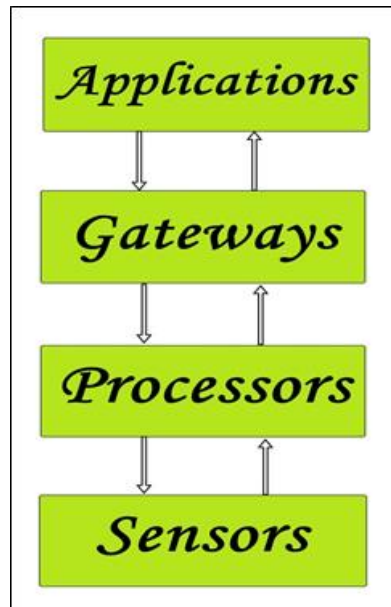
Figure depicts one such IoT System on Chip model no. RS9113, which has been designed and developed by Redpine Signals recently. It supports WLAN (802.11n), Bluetooth version 4.0 and Zigbee (802.15.4-2006) in the same chip. Hence the IoT device can be connected with any of the said wireless technology based networks.

This IoT SoC (system on chip in IoT) can be used for numerous applications as mentioned below:

- Mobile
- M2M-Communication
- Real time location finding tags
- Thermostats
- Smart meters
- Wireless sensor devices
- Serial to WiFi converter
- Voice Over WiFi compliant phones
- Home automation
- Health care devices and equipments

## Building Blocks Of IoT

Four things form basic building blocks of the IoT system –sensors, processors, gateways, applications. Each of these nodes has to have its own characteristics in order to form an useful IoT system.



**Figure:** Simplified block diagram of the basic building blocks of the IoT

### Sensors:

- These form the front end of the IoT devices. These are the so-called “Things” of the system. Their main purpose is to collect data from its surroundings (sensors) or give out data to its surrounding (actuators).
- These have to be uniquely identifiable devices with a unique IP address so that they can be easily identifiable over a large network.
- These have to be active in nature which means that they should be able to collect real-time data. These can either work on their own (autonomous in nature) or can be made to work by the user depending on their needs (user-controlled).
- Examples of sensors are gas sensor, water quality sensor, moisture sensor, etc.

### Processors:

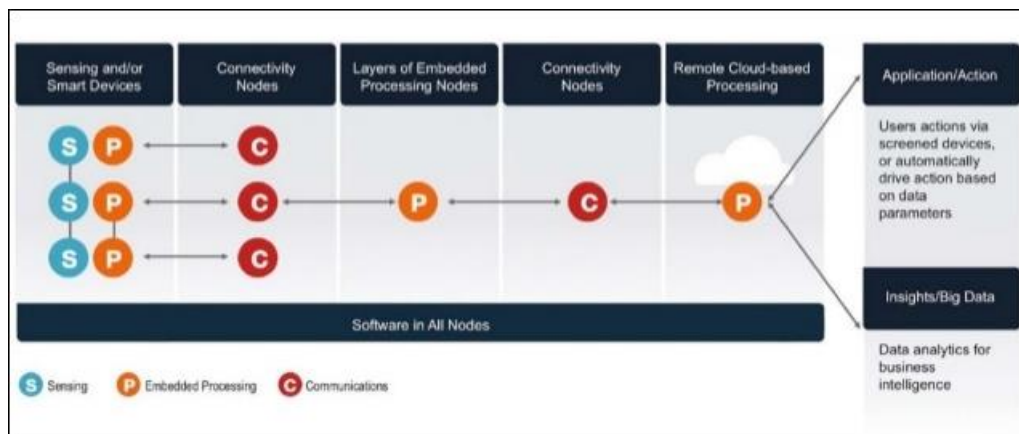
- Processors are the brain of the IoT system. Their main function is to process the data captured by the sensors and process them so as to extract the valuable data from the enormous amount of raw data collected. In a word, we can say that it gives intelligence to the data.
- Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontroller, etc are the ones that process the data because they have processors attached to it.

## Gateways:

- Gateways are responsible for routing the processed data and send it to proper locations for its (data) proper utilization.
- In other words, we can say that gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate.
- LAN, WAN, PAN, etc are examples of network gateways.

## Applications:

- Applications form another end of an IoT system. Applications are essential for proper utilization of all the data collected.
- These cloud-based applications which are responsible for rendering the effective meaning to the data collected. Applications are controlled by users and are a delivery point of particular services.
- Examples of applications are home automation apps, security systems, industrial control hub, etc.



**Figure :** Basic building blocks of IoT

In a nutshell, from the figure we can determine that the information gathered by the sensing node (end node) is processed first then via connectivity it reaches the embedded processing nodes that can be any embedded hardware devices and are processed there as well. It then passes through the connectivity nodes sent again and reaches the remote cloud-based processing that can be any software and is sent to the application node for the proper applied usage of the data collected and also for data analysis via big data.

## HOW IoT WORKS

How the IoT works is quite simple.

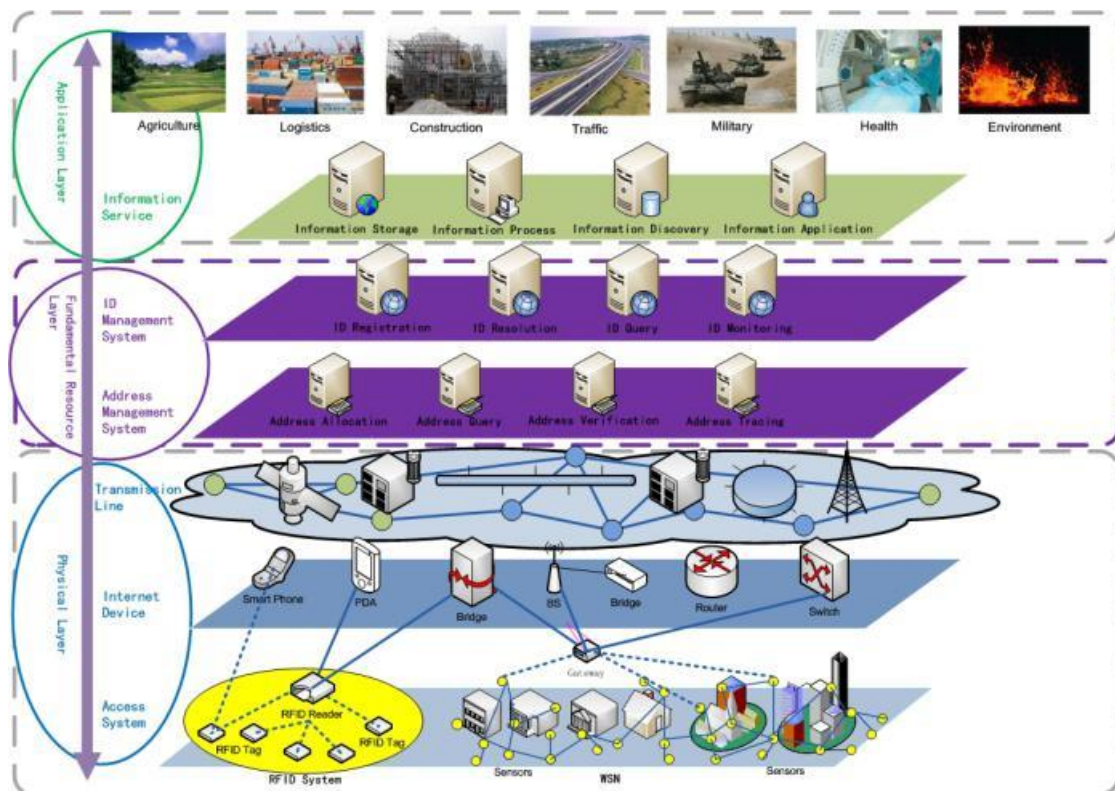
First, it acquires information with respect to basic resources (names, addresses and so on) and related attributes of objects by means of automatic identification and perception technologies such as RFID, wireless sensor and satellite positioning, in other words, the

sensors, RFID tags, and all other uniquely identifiable objects or "things" acquire real-time information (data) with the virtue of a central hub like smartphones.

Second, by virtue of many kinds of communications technologies, it integrates object-related information into the information network and realizes the intelligent indexing and integration of the information related to masses of objects by resorting to fundamental resource services (similar to the resolution, addressing and discovery of the internet).

Finally, utilizing intelligent computing technologies such as cloud computing, fuzzy recognition, data mining, and semantic analysis, it analyzes and processes the information related to masses of objects so as to eventually realize intelligent decision and control in the physical world.

Let's have a look at the following diagram.



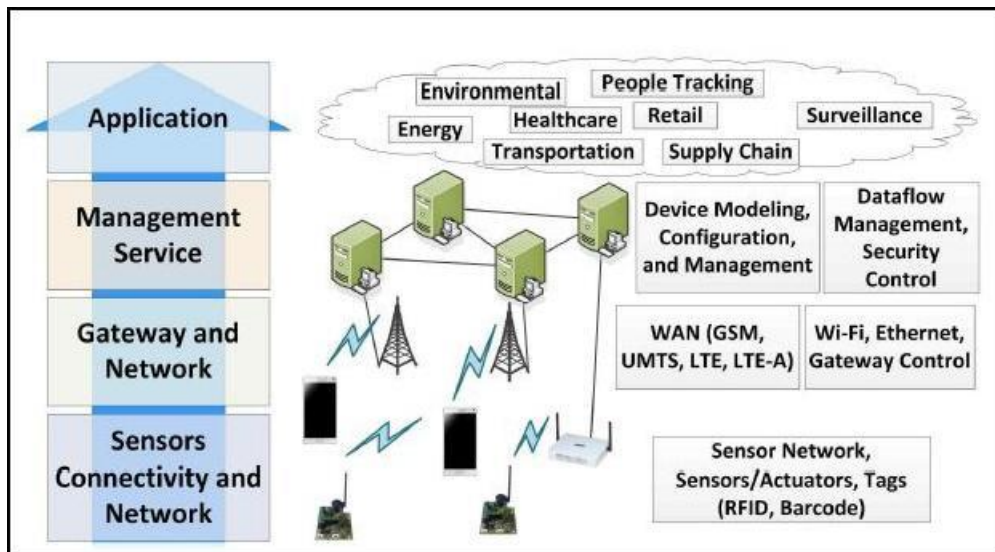
**Figure : Layers of the IoT**

In the Physical layer, all the data collected by the access system (uniquely identifiable "things") collect data and go to the internet devices (like smartphones). Then via transmission lines (like fiber-optic cable) it goes to the management layer where all the data is managed separately (stream analytics and data analytics) from the raw data. Then all the managed information is released to the application layer for proper utilization of the data collected.



## IoT Architecture Layers

There are four major layers.



At the very bottom of IoT architecture, we start with the Sensors and Connectivity network which collects information. Then we have the Gateway and Network Layer. Above which we have the Management Service layer and then at the end, we have the application layer where the data collected are processed according to the needs of various applications.

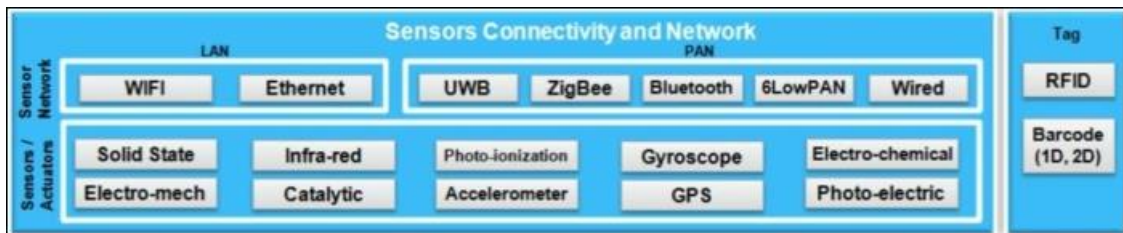
Let's discuss the features of each of these architectural layers separately.

### Sensor, Connectivity and Network Layer

- This layer consists of RFID tags, sensors (which are an essential part of an IoT system and are responsible for collecting raw data). These form the essential “things” of an IoT system.
- Sensors, RFID tags are wireless devices and form the Wireless Sensor Networks (WSN).
- Sensors are active in nature which means that real-time information is to be collected and processed.
- This layer also has the network connectivity (like WAN, PAN, etc.) which is responsible for communicating the raw data to the next layer which is the Gateway and Network Layer.
- The devices which are comprised of WSN have finite storage capacity, restricted communication bandwidth and have small processing speed.
- We have different sensors for different applications – temperature sensor for collecting temperature data, water quality for examining water quality, moisture sensor for measuring moisture content of the atmosphere or soil, etc.

As per the figure below, at the bottom of this layer, we have the tags which are the RFID tags or barcode reader, above which we have the sensors/actuators and then the communication networks.



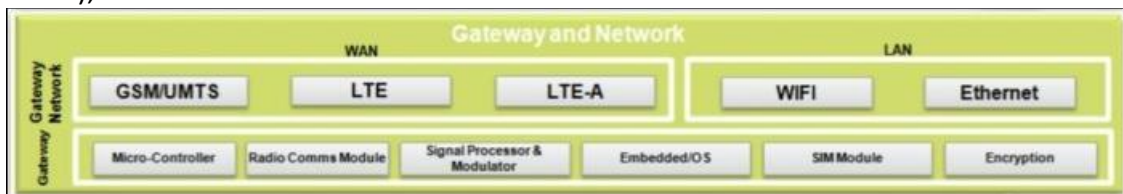


**Figure : Sensor, Connectivity and Network Layer**

### Gateway and Network Layer

- Gateways are responsible for routing the data coming from the **Sensor, Connectivity and Network layer** and pass it to the next layer which is the **Management Service Layer**.
- This layer requires having a large storage capacity for storing the enormous amount of data collected by the sensors, RFID tags, etc. Also, this layer needs to have a consistently trusted performance in terms of public, private and hybrid networks.
- Different IoT device works on different kinds of network protocols. All these protocols are required to be assimilated into a single layer. This layer is responsible for integrating various network protocols.

From the figure below, at the bottom, we have the gateway which is comprised of the embedded OS, Signal Processors, and Modulators, Micro-Controllers etc. Above the gateway we have the Gateway Networks which are LAN(Local Area Network), WAN(Wide Area Network), etc.



**Figure : Gateway and Network Layer**

### Management Service Layer

- This layer is used for managing IoT services. The management Service layer is responsible for Securing Analysis of IoT devices, Analysis of Information (Stream Analytics, Data Analytics), Device Management.
- Data management is required to extract the necessary information from the enormous amount of raw data collected by the sensor devices to yield a valuable result of all the data collected. This action is performed in this layer.
- Also, a certain situation requires an immediate response to the situation. This layer helps in doing that by abstracting data, extracting information and managing the data flow.
- This layer is also responsible for data mining, text mining, service analytics, etc.

From the figure below, we can see that, management service layer has Operational Support Service (OSS) which includes Device Modeling, Device Configuration and Management and many more. Also, we have the Billing Support System (BSS) which supports billing and reporting.

Also, from the figure, we can see that there are IoT/M2M Application Services which includes Analytics Platform; Data – which is the most important part; Security which includes Access Controls, Encryption, Identity Access Management, etc. ; and then we have the Business Rule Management (BRM) and Business Process Management (BPM).



**Figure : Management Service Layer**

## Application Layer

- Application layer forms the topmost layer of IoT architecture which is responsible for effective utilization of the data collected.
- Various IoT applications include Home Automation, E-health, E-Government, etc.
- From the figure below, we can see that there are two types of applications which are Horizontal Market which includes Fleet Management, Supply Chain, etc. and on the Sector-wise application of IoT we have energy, healthcare, transportation, etc.



**Figure : Application Layer**

## IoT Platform

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. It basically connects your hardware, however diverse, to the cloud by using flexible connectivity options, enterprise-grade security mechanisms, and broad data processing powers. For developers, an IoT platform provides a set of ready-to-use features that greatly speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility.

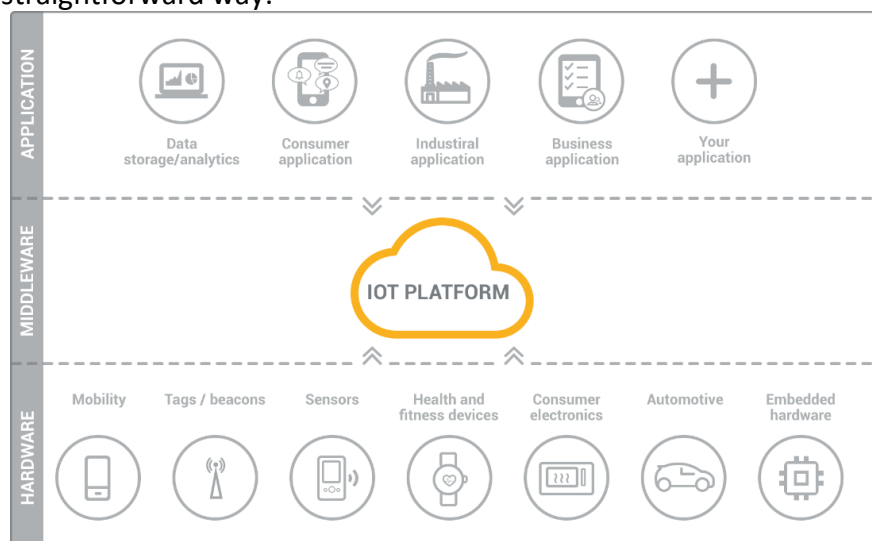
Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices

to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

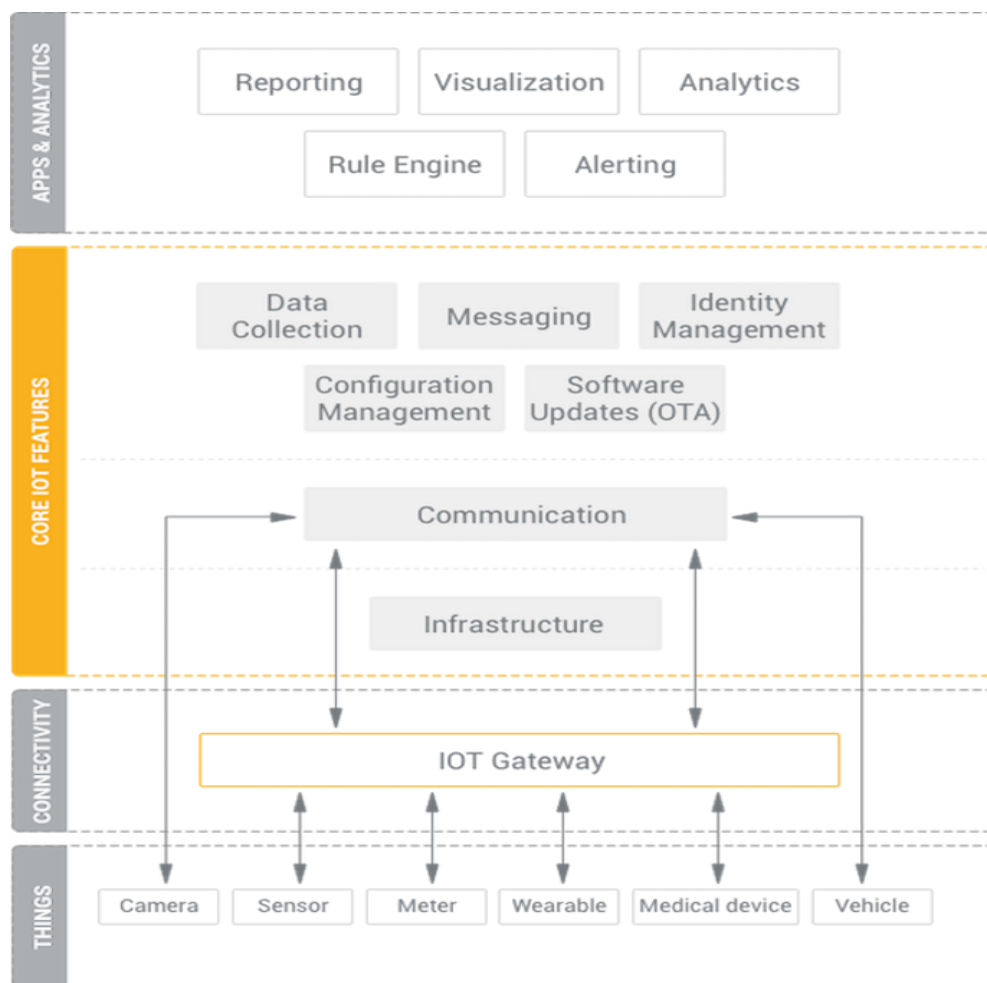
## IoT platform as the middleware

IoT platforms originated in the form of IoT middleware, which purpose was to function as a mediator between the hardware and application layers. Its primary tasks included data collection from the devices over different protocols and network topologies, remote device configuration and control, device management, and over-the-air firmware updates.

To be used in real-life heterogeneous IoT ecosystems, IoT middleware is expected to support integration with almost any connected device and blend in with third-party applications used by the device. This independence from underlying hardware and overhanging software allows a single IoT platform to manage any kind of connected device in the same straightforward way.



Modern IoT platforms go further and introduce a variety of valuable features into the hardware and application layers as well. They provide components for frontend and analytics, on-device data processing, and cloud-based deployment. Some of them can handle end-to-end IoT solution implementation from the ground up.



## IoT platform technology stack

In the four typical layers of the IoT stack, which are things, connectivity, core IoT features, and applications & analytics, a top-of-the-range IoT platform should provide you with the majority of IoT functionality needed for developing your connected devices and smart things.

Your devices connect to the platform, which sits in the cloud or in your on-premises data center, either directly or by using an IoT gateway. A gateway comes useful whenever your endpoints aren't capable of direct cloud communication or, for example, you need some computing power on edge. You can also use an IoT gateway to convert protocols, for example, when your endpoints are in LoRaWan network but you need them to communicate with the cloud over MQTT.

An IoT platform itself can be decomposed into several layers. At the bottom there is the infrastructure level, which is something that enables the functioning of the platform. You

can find here components for container management, internal platform messaging, orchestration of IoT solution clusters, and others.

The communication layer enables messaging for the devices; in other words, this is where devices connect to the cloud to perform different operations.

The following layer represents core IoT features provided by the platform. Among the essential ones are data collection, device management, configuration management, messaging, and OTA software updates.

Sitting on top of core IoT features, there is another layer, which is less related to data exchange between devices but rather to processing of this data in the platform. There is reporting, which allows you to generate custom reports. There is visualization for data representation in user applications. Then, there are a rule engine, analytics, and alerting for notifying you about any anomalies detected in your IoT solution.

Importantly, the best IoT platforms allow you to add your own industry-specific components and third-party applications. Without such flexibility adapting an IoT platform for a particular business scenario could bear significant extra cost and delay the solution delivery indefinitely.

#### Advanced IoT platforms

There are some other important criteria that differentiate IoT platforms between each other, such as scalability, customizability, ease of use, code control, integration with 3rd party software, deployment options, and the data security level.

- **Scalable (cloud native)** – advanced IoT platforms ensure elastic scalability across any number of endpoints that the client may require. This capability is taken for granted for public cloud deployments but it should be specifically put to the test in case of an on-premises deployment, including the platform's load balancing capabilities for maximized performance of the server cluster.
- **Customizable** – a crucial factor for the speed of delivery. It closely relates to flexibility of integration APIs, loose coupling of the platform's components, and source code transparency. For small-scale, undemanding IoT solutions good APIs may be enough to fly, while feature-rich, rapidly evolving IoT ecosystems usually require developers to have a greater degree of control over the entire system, its source code, integration interfaces, deployment options, data schemas, connectivity and security mechanisms, etc.
- **Secure** – data security involves encryption, comprehensive identity management, and flexible deployment. End-to-end data flow encryption, including data at rest, device authentication, user access rights management, and private cloud infrastructure for sensitive data – this is the basics of how to avoid potentially compromising breaches in your IoT solution.

Cutting across these aspects, there are two different paradigms of IoT solution cluster deployment offered by IoT platform providers: a public cloud IoT PaaS and a self-hosted private IoT cloud.

## IoT cloud enablement

An IoT cloud is a pinnacle of the IoT platforms evolution. Sometimes these two terms are used interchangeably, in which case the system at hand is typically an IoT platform-as-a-service (PaaS). This type of solution allows you to rent cloud infrastructure and an IoT platform all from a single technology provider. Also, there might be ready-to-use IoT solutions (IoT cloud services) offered by the provider, built and hosted on its infrastructure. However, one important capability of a modern IoT platform consists in a private IoT cloud enablement. As opposed to public PaaS solutions located at a provider's cloud, a private IoT cloud can be hosted on any cloud infrastructure, including a private data center. This type of deployment offers much greater control over the new features development, customization, and third-party integrations. It is also advocated for stringent data security and performance requirements.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are –

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

## Board Types

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface

(hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

**Arduino boards based on ATMEGA328 microcontroller**

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATmega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATmega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible



							Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI- Compatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI- Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI- Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI- Compatible Header

#### Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

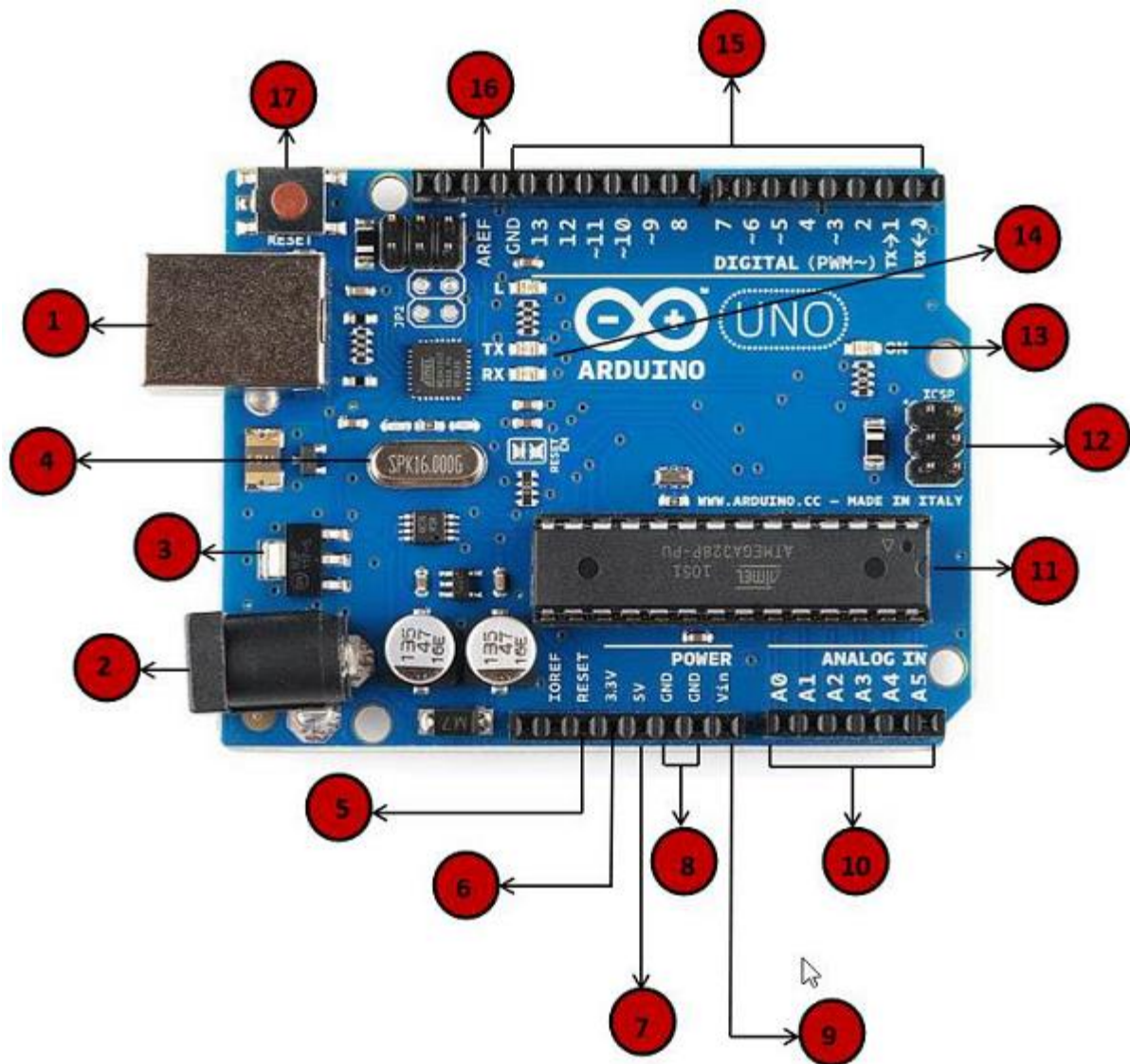
### Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATmega16U2B
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header






### Arduino boards based on AT91SAM3X8E microcontroller






Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

In this chapter, we will learn about the different components on the Arduino board. We will study the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.



<p>1</p>	<p><b>Power USB</b></p> <p>Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).</p>
<p>2</p>	<p><b>Power (Barrel Jack)</b></p> <p>Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).</p>
<p>3</p>	<p><b>Voltage Regulator</b></p> <p>The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.</p>

	<p><b>Crystal Oscillator</b></p> <p>The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.</p>
	<p><b>Arduino Reset</b></p> <p>You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).</p>
	<p><b>Pins (3.3, 5, GND, Vin)</b></p> <ul style="list-style-type: none"> <li>• 3.3V (6) – Supply 3.3 output volt</li> <li>• 5V (7) – Supply 5 output volt</li> <li>• Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.</li> <li>• GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.</li> <li>• Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.</li> </ul>
	<p><b>Analog pins</b></p> <p>The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>
	<p><b>Main microcontroller</b></p> <p>Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.</p>

	<p><b>ICSP pin</b></p> <p>Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.</p>
	<p><b>Power LED indicator</b></p> <p>This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
	<p><b>TX and RX LEDs</b></p> <p>On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
	<p><b>Digital I/O</b></p> <p>The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.</p>
	<p><b>AREF</b></p> <p>AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

**Step 1** – First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega

2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

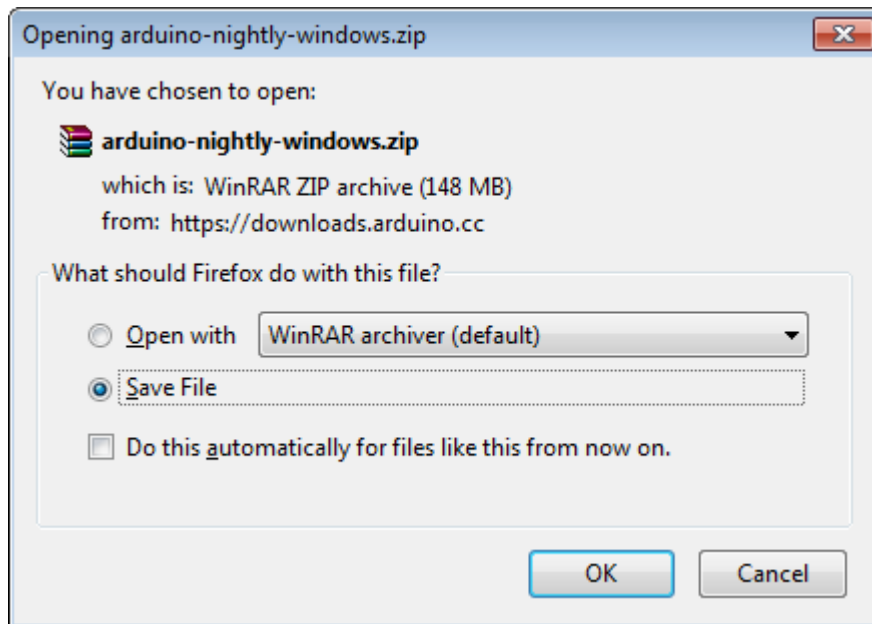


In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.



## **Step 2 – Download Arduino IDE Software.**

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



### **Step 3 – Power up your board.**

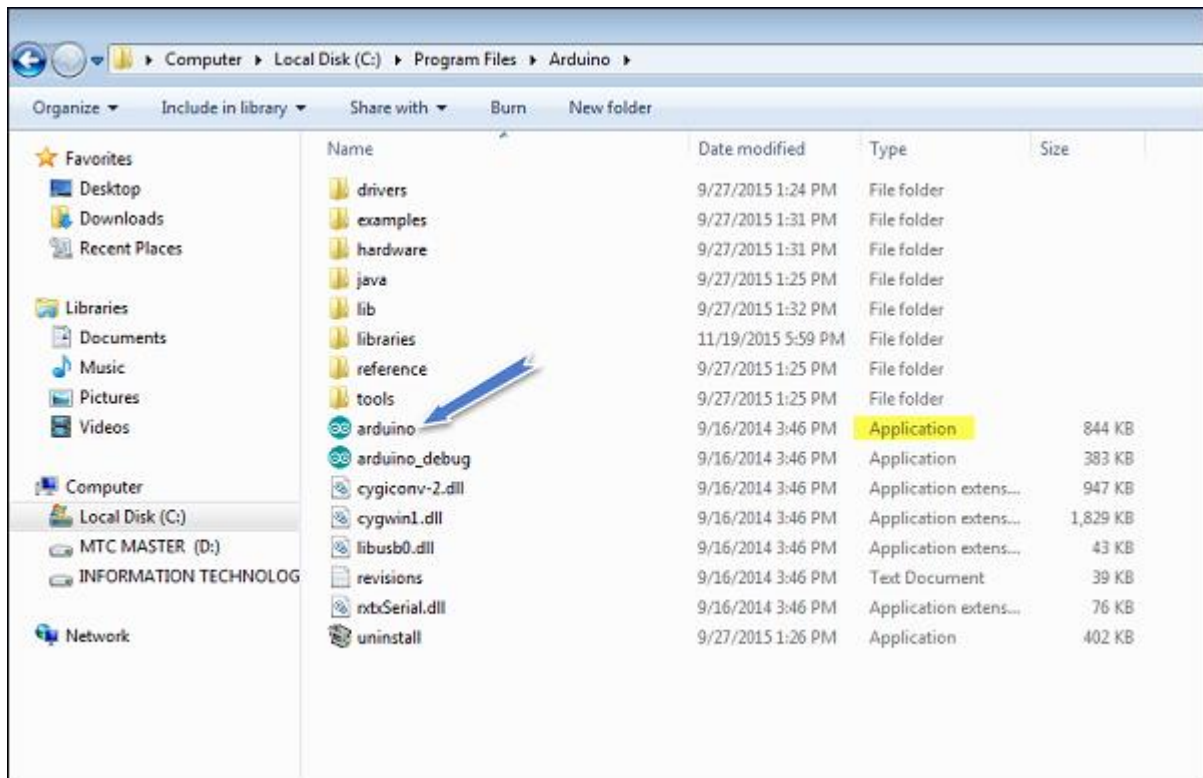
The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

### **Step 4 – Launch Arduino IDE.**

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



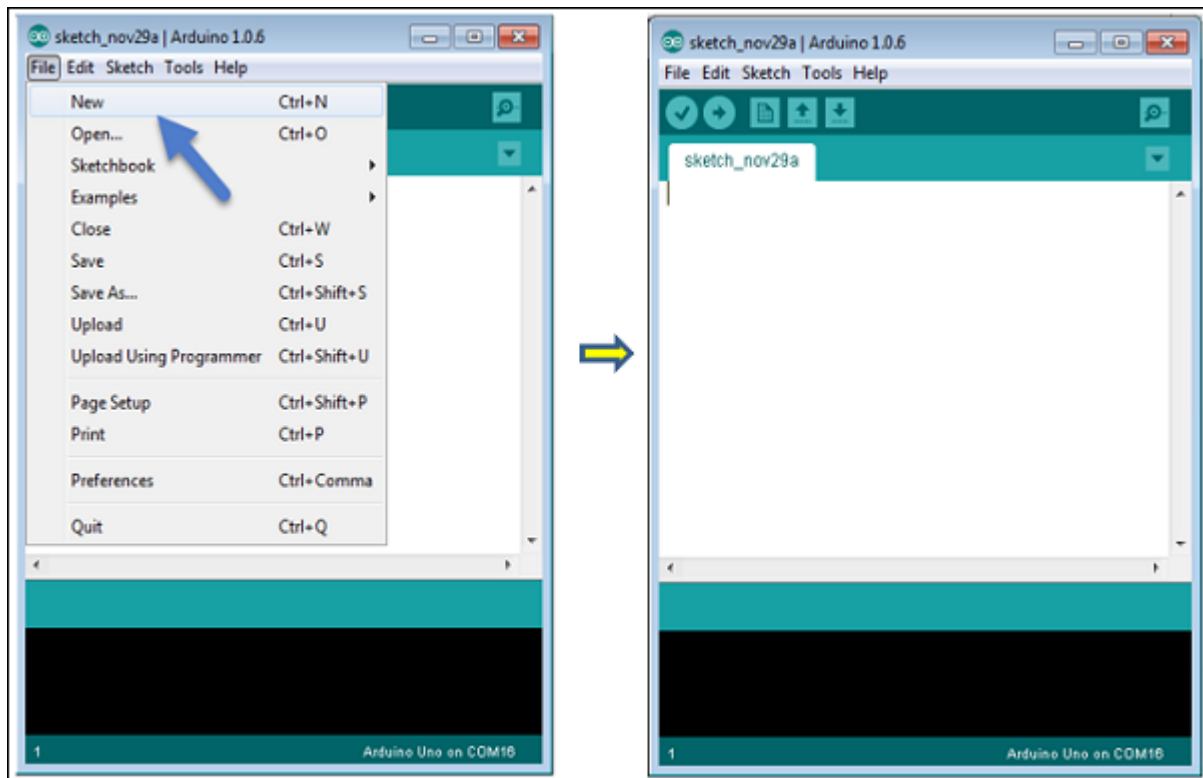


### Step 5 – Open your first project.

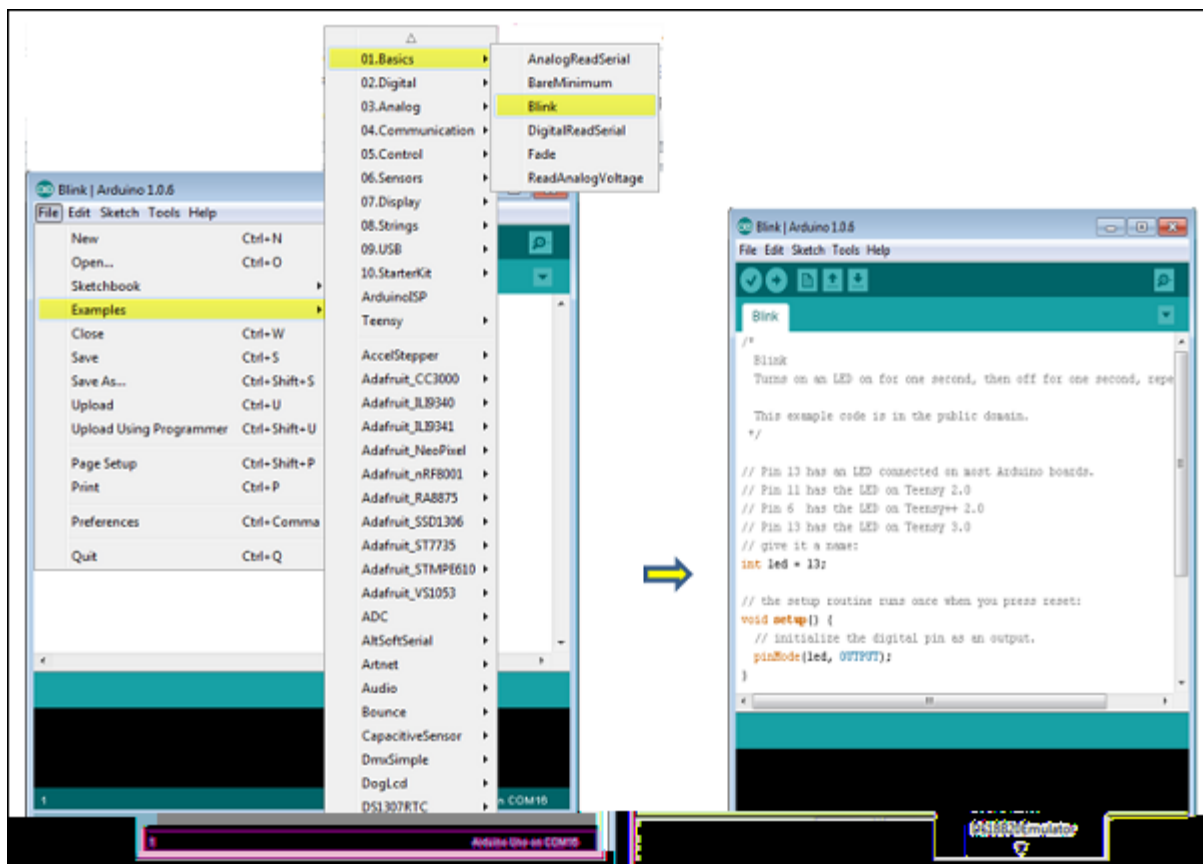
Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.

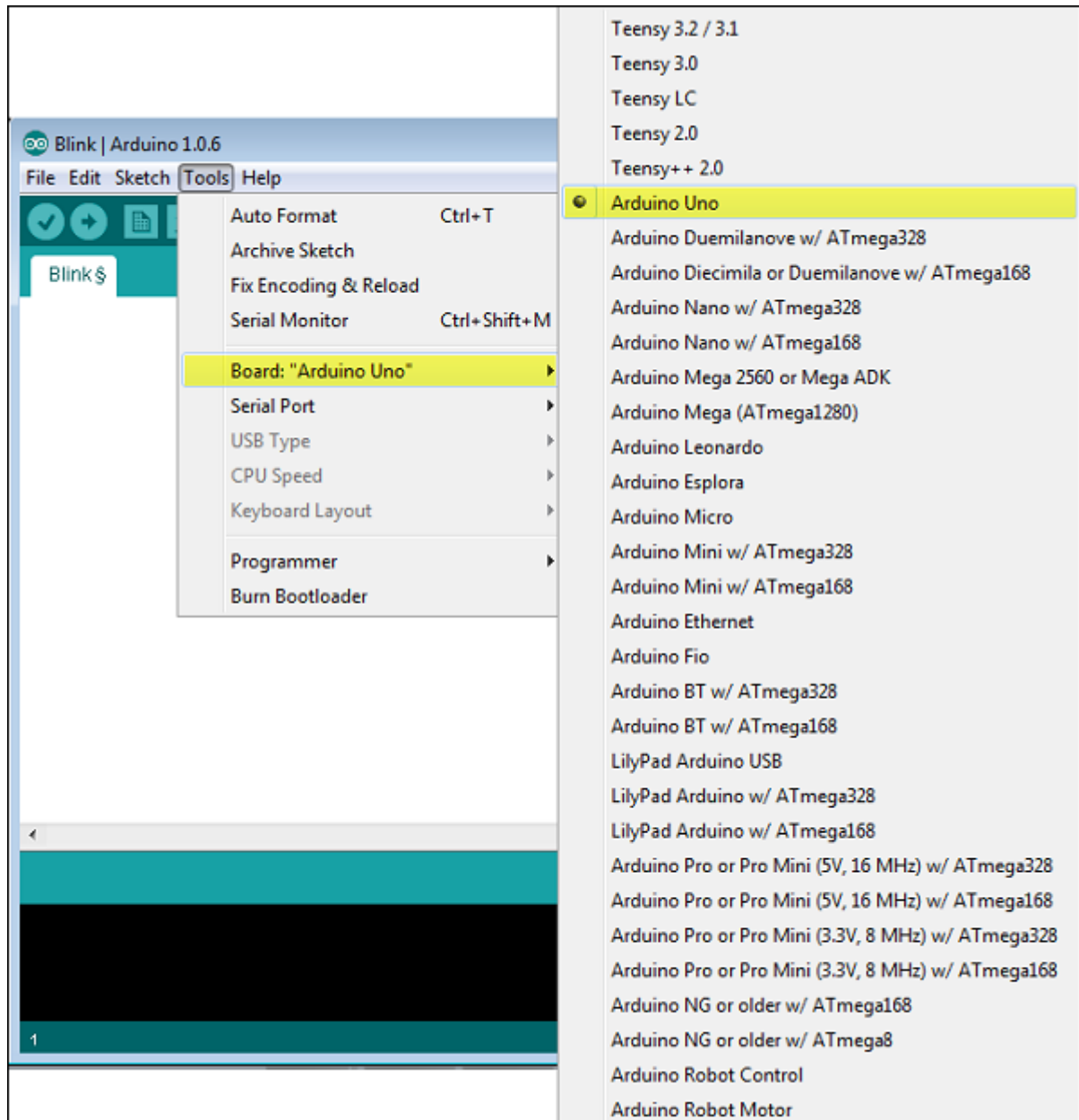


Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

## Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

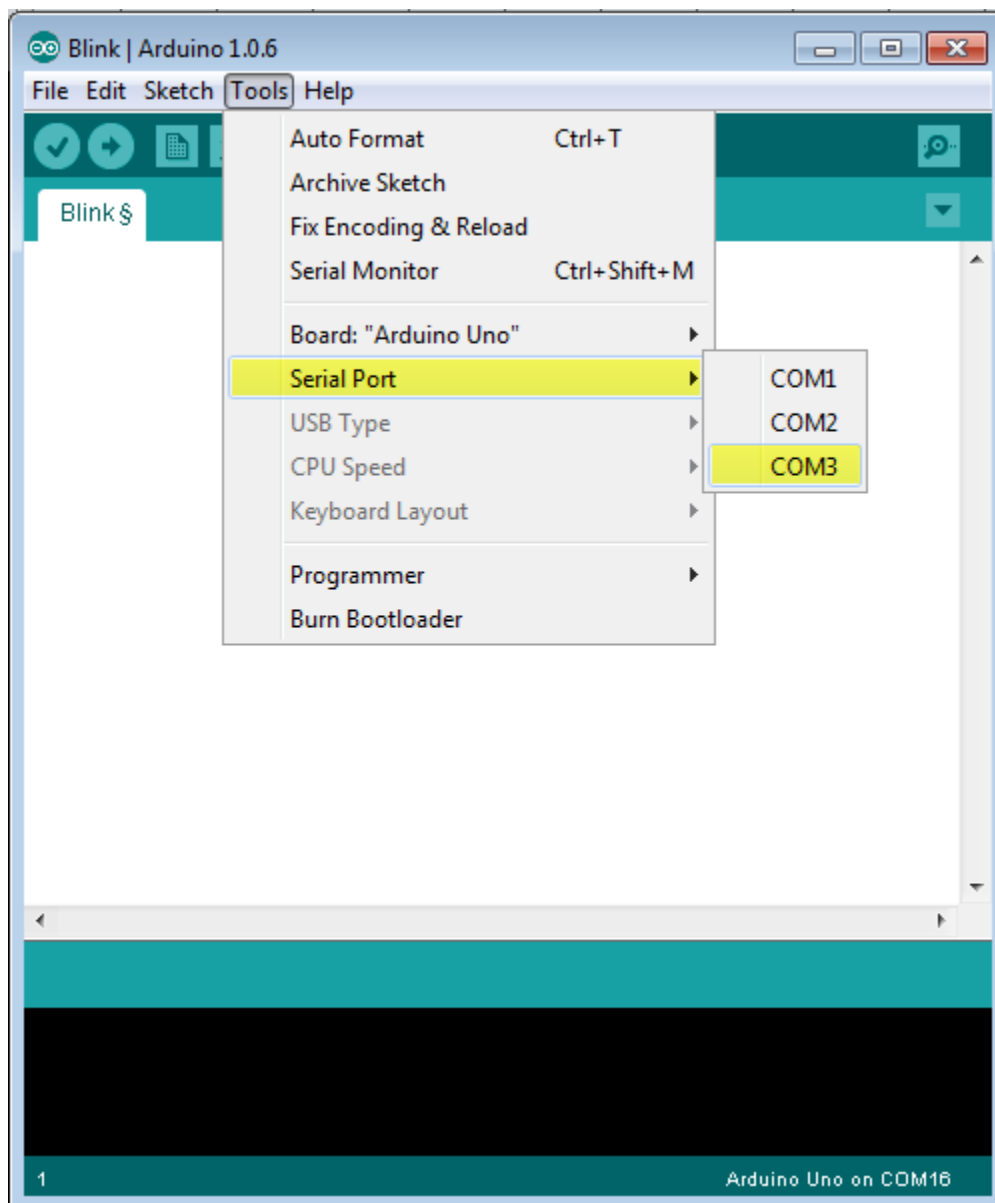
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

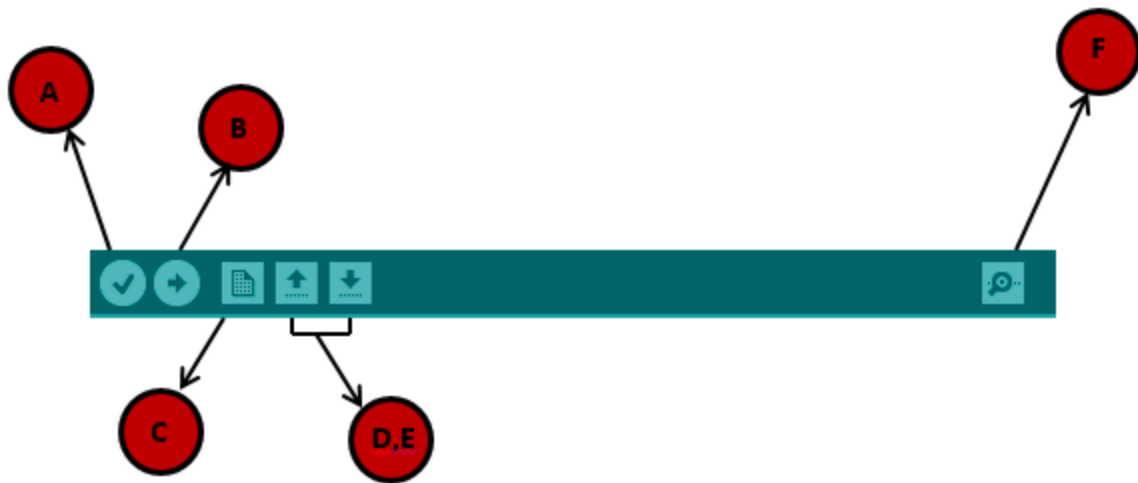
## Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



### Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**A** – Used to check if there is any compilation error.

**B** – Used to upload a program to the Arduino board.

**C** – Shortcut used to create a new sketch.

**D** – Used to directly open one of the example sketch.

**E** – Used to save your sketch.

**F** – Serial monitor used to receive serial data from the board and send the serial data to the board.

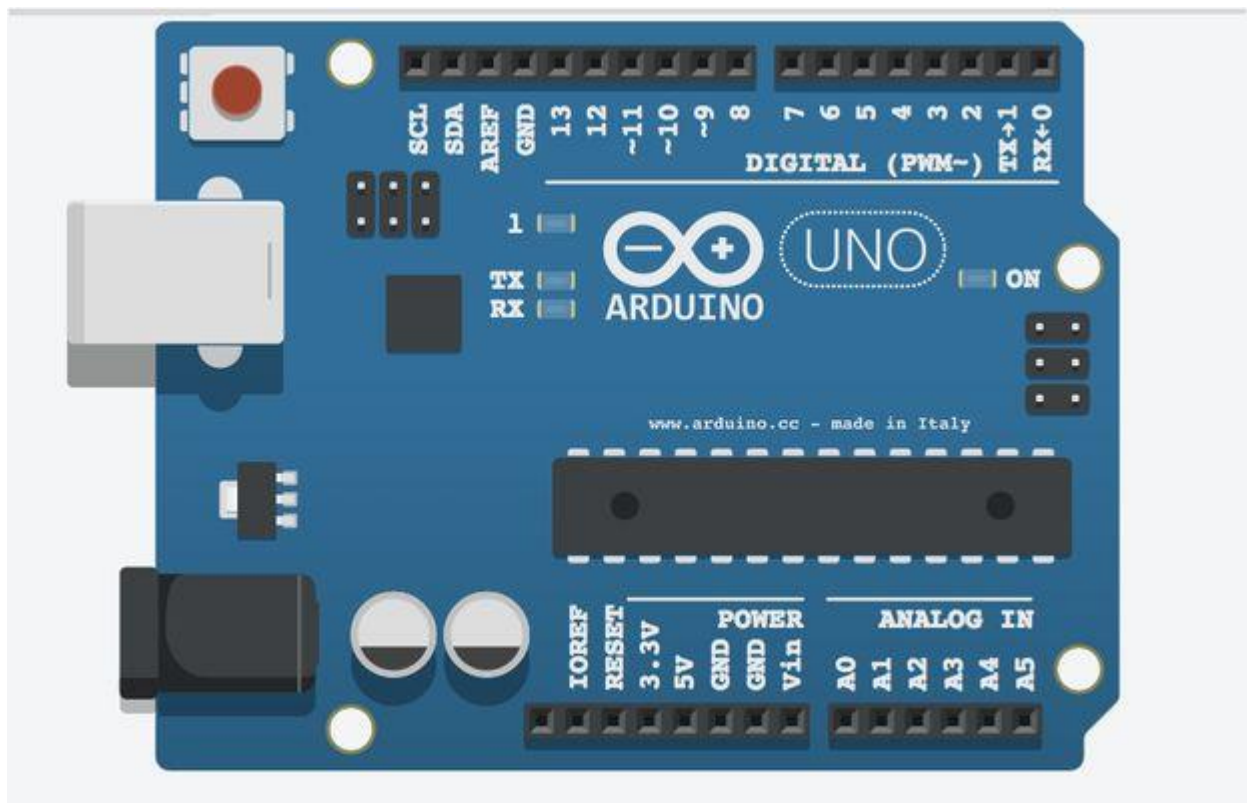
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

## IoT Platforms Overview: Arduino, Raspberry Pi

The IoT concepts imply a creation of network of various devices interacting with each other and with their environment. Interoperability and connectivity wouldn't be possible without hardware platforms that help developers solve issues such as building autonomous interactive objects or completing common infrastructure related tasks.

Let's go through the most popular IoT platforms and see how they work and benefit IoT software developers.

## Arduino



The Arduino platform was created back in 2005 by the Arduino company and allows for open source prototyping and flexible software development and back-end deployment while providing significant ease of use to developers, even those with very little experience building IoT solutions.

Arduino is sensible to literally every environment by receiving source data from different external sensors and is capable to interact with other control elements over various devices, engines and drives. Arduino has a built-in micro controller that operates on the Arduino software.

Projects based on this platform can be both standalone and collaborative, i.e. realized with use of external tools and plugins. The integrated development environment (IDE) is composed of the open source code and works equally good with Mac, Linux and Windows OS. Based on a *processing* programming language, the Arduino platform seems to be created for new users and for experiments. The processing language is dedicated to visualizing and building interactive apps using animation and Java Virtual Machine (JVM) platform.

Let's note that this programming language was developed for the purpose of learning basic computer programming in a visual context. It is an absolutely free project available to every interested person. Normally, all the apps are programmed in C/C++, and are wrapped with *avr-gcc* (WinAVR in OS Windows).

Arduino offers analogue-to-digital input with a possibility of connecting light, temperature or sound sensor modules. Such sensors as *SPI* or *I2C* may also be used to cover up to 99% of these apps' market.

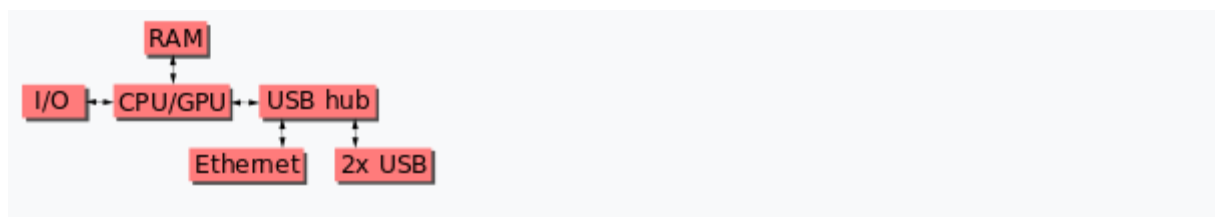
Arduino is a microcontroller (generally it is the 8-bit ATmega microcontroller), but not a mini-computer, which makes Arduino somehow limited in its features for advanced users. Arduino provides an excellent interactivity with external devices and offers a wide range of user manuals, project samples as well as a large community of users to learn from / share knowledge with.

## Raspberry Pi

**Raspberry Pi** (/paɪ/) is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries. Later, the original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It is now widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design.

After the release of the second board type, the Raspberry Pi Foundation set up a new entity, named Raspberry Pi Trading, and installed Eben Upton as CEO, with the responsibility of developing technology. The Foundation was rededicated as an educational charity for promoting the teaching of basic computer science in schools and developing countries. The Raspberry Pi is one of the best-selling British computers.

The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.

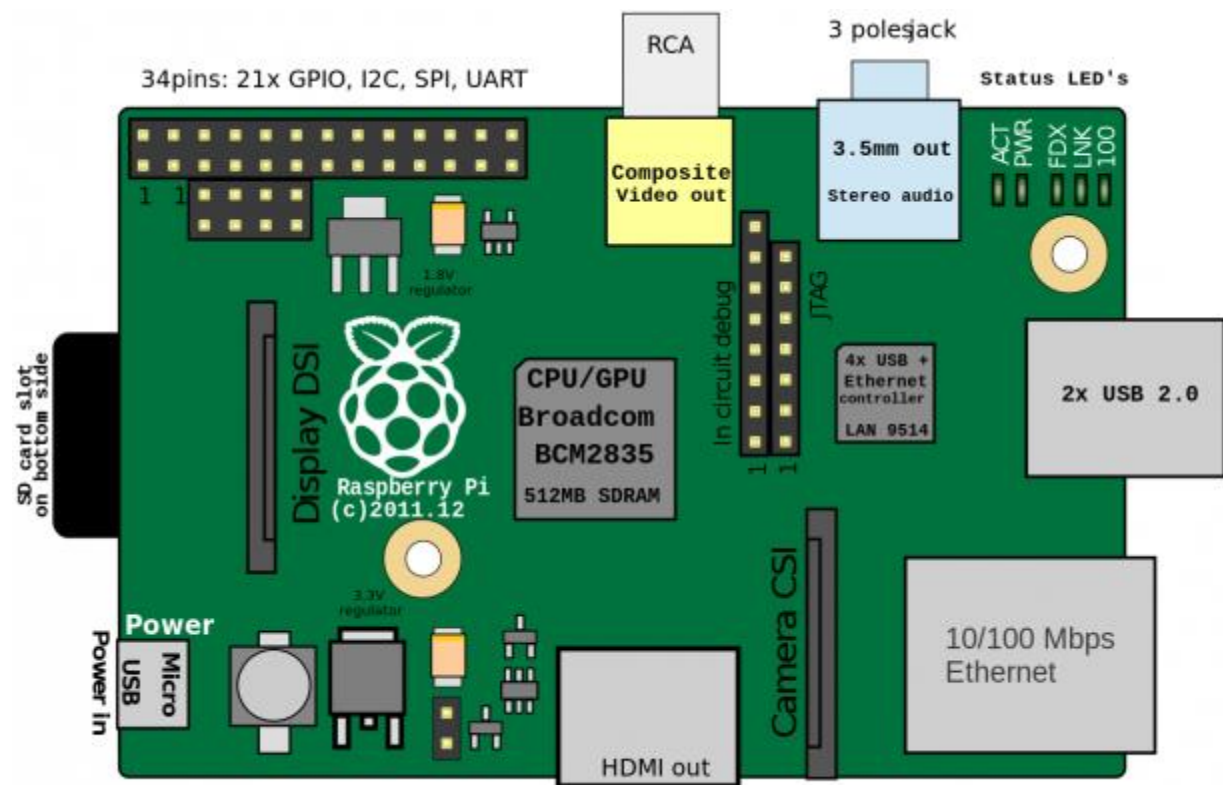


This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port. Unlike all other Pi models, the 40 pin GPIO connector is omitted on the Pi Zero, with solderable through-holes only in the pin locations. The Pi Zero WH remedies this.

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MiB to 1 GiB random-access memory (RAM), with up to 8 GiB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards



have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I<sup>2</sup>C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.



Raspberry Pi is a mono-board computing platform that's as tiny as a credit card. Initially it was developed for computer science education with later on progress to wider functions.

Since the inception of Raspberry, the company sold out more than 8 million items. Raspberry Pi 3 is the latest version and it is the first 64-bit computing board that also comes with built-in Wi-Fi and Bluetooth functions. According to Raspberry Pi Foundation CEO Eben Upton, *"it's been a year in the making"*. The Pi3 version is replaced with a quad-core 64-bit 1.2 GHz ARM Cortex A53 chip, 1GB of RAM, VideoCore IV graphics, Bluetooth 4.1 and 802.11n Wi-Fi. The developers claim the new architecture delivers an average 50% performance improvement over the Pi 2.

Another peculiarity of Raspberry Pi is the *GPIO* (General Purpose Input-Output), which is a low-level interface of self-operated control by input-output ports. Raspberry has it as a 40-pin connector.

Raspberry Pi uses Linux as its default operating system (OS). It's also fully Android compatible. Using the system on Windows OS is enabled through any virtualization system like *XenDesktop*. If you want to develop an application for Raspberry Pi on your computer, it

is necessary to download a specific toolset comprised of ARM-compiler and some libraries complied down to ARM-target platform like *glibc*.