SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

**SCHOOL OF COMPUTING**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# UNIT – I – Internet of Things – SCSA5301

# INTRODUCTION TO IOT

Introduction to IoT - Current technological trends and future prospects - Evolution of IoT- Business Scope - Relation with embedded system - Basic Architecture of an IoT - From M2M to IoT - M2M towards IoT - IoT Value Chains - An emerging industrial structure for IoT.

## 1.INTRODUCTION TO IOT

It is a network of physical objects or things sending, receiving, or communicating using the internet or other communication technologies. The Internet of Things (IoT), as this intelligent interconnectivity between the real and the digital world is called, will rapidly transform every aspect of how we work and do business. By connecting apps with their integrated systems, businesses are able to transform their industry significantly. Today almost 90% of all data generated by tablets, smartphones or connected appliances is never acted upon. Imagine you could change that. It seems safe to say that we have never encountered a single technological platform that combines this much complexity, global reach and novelty. Since IoT allows devices to be controlled remotely across the internet, thus it created opportunities to directly connect & integrate the physical world to the computer-based systems using sensors and internet. The interconnection of these multiple embedded devices will be resulting in automation in nearly all fields and enabling advanced applications. This is resulting in improved accuracy, efficiency and economic benefit with reduced human intervention. It encompasses technologies such as smart grids, smart homes, intelligent transportation and smart cities. The major benefits of IoT are:

- **Improved Customer Engagement** – IoT improves customer experience by automating the action. For e.g. any issue in the car will be automatically detected by the sensors. The driver, as well as the manufacturer, will be notified about it. Until the time driver reaches the service station, the manufacturer will make sure that the faulty part is available at the service station.

- **Technical Optimization** – IoT has helped a lot in improving technologies and making them better. The manufacturer can collect data from different car sensors and analyze them to improve their design and make them much more efficient.

  **Reduced Waste** – Our current insights are superficial, but IoT provides real-time information leading to effective decision-making & management of resources. For example, if a manufacturer finds fault in multiple engines, he can track the manufacturing plant of those engines and can rectify the issue with manufacturing belt.

## 1.CURRENT TECHNOLOGICAL TRENDS AND FUTURE PROSPECTS:

Many firms see big opportunity in IoT uses and enterprises start to believe that IoT holds the promise to enhance customer relationships and drive business growth by improving quality, productivity, and reliability on one side, and on the other side reducing costs, risk, and theft.

### IoT and Security Attacks

Forrester thinks that the recent DDoS attack that hit a whopping 1600 websites in the United States was just the tip of the iceberg when it comes to the threat that the connected device poses to the world. That attack confirmed the fear of vulnerability of IoT devices with a massive distributed denial of service attack that crippled the servers of services like Twitter, NetFlix, NYTimes, and PayPal across the U.S. It‘s the result of an immense assault that involved millions of Internet addresses and malicious software. All indications suggest that countless Internet of Things (IoT) devices that power everyday technology like closed-circuit cameras and smart- home devices were hijacked by the malware, and used against the servers.

## IoT and Mobile Elements

IoT is creating new opportunities and providing a competitive advantage for businesses in current and new markets. It touches everything—not just the data, but how, when, where and why you collect it. The technologies that have created the Internet of Things aren‘t changing the internet only, but rather change the things connected to the internet. More mobile moments (the moments in which a person pulls out a mobile device to get what he or she wants, immediately and in context) will appear on the connected device, right from home appliances to cars to smartwatches and virtual assistants. All these connected devices will have the potential of offering a rich stream of data that will then be used by product and service owners to interact with their consumers.

## IoT and artificial Intelligence

In an IoT situation, AI can help companies take the billions of data points they have and boil them down to what‘s really meaningful. The general premise is the same as in the retail applications – review and analyzes the data we‘ve collected to find patterns or similarities that can be learned from so that better decisions can be made.

## IoT and Connectivity

Connecting the different parts of IoT to the sensors can be done by different technologies including Wi-Fi, Bluetooth, Low Power Wi-Fi, Wi-Max, regular Ethernet, Long Term Evolution (LTE) and the recent promising technology of Li-Fi (using light as a medium of communication between the different parts of a typical network including sensors).

## IoT and New Business Models

The bottom line is a big motivation for starting, investing in, and operating any business, without a sound and solid business models for IoT we will have another bubble, this model must satisfy all the requirements for all kinds of e-commerce; vertical markets, horizontal markets, and consumer markets. One key element is to bundle service with the product, for example, devices like Amazon‘s Alexa will be considered just another wireless speaker without the services provided like voice recognition, music streaming, and booking Uber service to mention few. The IoT can find its applications in almost every aspect of our daily life. Below are some of the examples.

1) Prediction of natural disasters: The combination of sensors and their autonomous coordination and simulation will help to predict the occurrence of land-slides or other natural disasters and to take appropriate actions in advance.

2) Industry applications: The IoT can find applications in industry e.g., managing a fleet of cars for an organization. The IoT helps to monitor their environmental performance and process the data to determine and pick the one that need maintenance.

3) Water Scarcity monitoring: The IoT can help to detect the water scarcity at different places. The networks of sensors, tied together with the relevant simulation activities might not only monitor long term water interventions such as catchment area management, but may even be used to alert users of a stream, for instance, if an upstream event, such as the accidental release of sewage into the stream, might have dangerous implications.

4) Design of smart homes: The IoT can help in the design of smart homes e.g., energy consumption management, interaction with appliances, detecting emergencies, home safety and finding things easily, home security etc.

5) Medical applications: The IoT can also find applications in medical sector for saving lives or improving the quality of life e.g., monitoring health parameters, monitoring activities, support for independent living, monitoring medicines intake etc.

6) Agriculture application: A network of different sensors can sense data, perform data processing and inform the farmer through communication infrastructure e.g., mobile phone text message about the portion of land that need particular attention. This may include smart packaging of seeds, fertilizer and pest control mechanisms that respond to specific local conditions and indicate actions. Intelligent farming system will help agronomists to have better understanding of the plant growth models and to have efficient farming practices by having the knowledge of land conditions and climate variability. This will

significantly increase the agricultural productivity by avoiding the inappropriate farming conditions.

7) Intelligent transport system design: The Intelligent transportation system will provide efficient transportation control and management using advanced technology of sensors, information and network. The intelligent transportation can have many interesting features such as non-stop electronic highway toll, mobile emergency command and scheduling, transportation law enforcement, vehicle rules violation monitoring, reducing environmental pollution, anti-theft system, avoiding traffic jams, reporting traffic incidents, smart beaconing, minimizing arrival delays etc.

8) Design of smart cities: The IoT can help to design smart cities e.g., monitoring air quality, discovering emergency routes, efficient lighting up of the city, watering gardens etc.

9) Smart metering and monitoring: The IoT design for smart metering and monitoring will help to get accurate automated meter reading and issuance of invoice to the customers. The IoT can be used to design such scheme for wind turbine maintenance and remote monitoring, gas, water as well as environmental metering and monitoring.

10) Smart Security: The IoT can also find applications in the field of security and surveillance e.g., surveillance of spaces, tracking of people and assets, infrastructure and equipment maintenance, alarming etc.

## 2. EVOLUTION OF IOT AND BUSINESS SCOPE:

The term ―Internet of Things‖ (IoT) was first used in 1999 by British technology pioneer Kevin Ashton to describe a system in which objects in the physical world could be connected to the Internet by sensors.12 Ashton coined the term to illustrate the power of connecting Radio- Frequency Identification (RFID) tags13 used in corporate supply chains to the Internet in order to count and track goods without the need for human intervention.

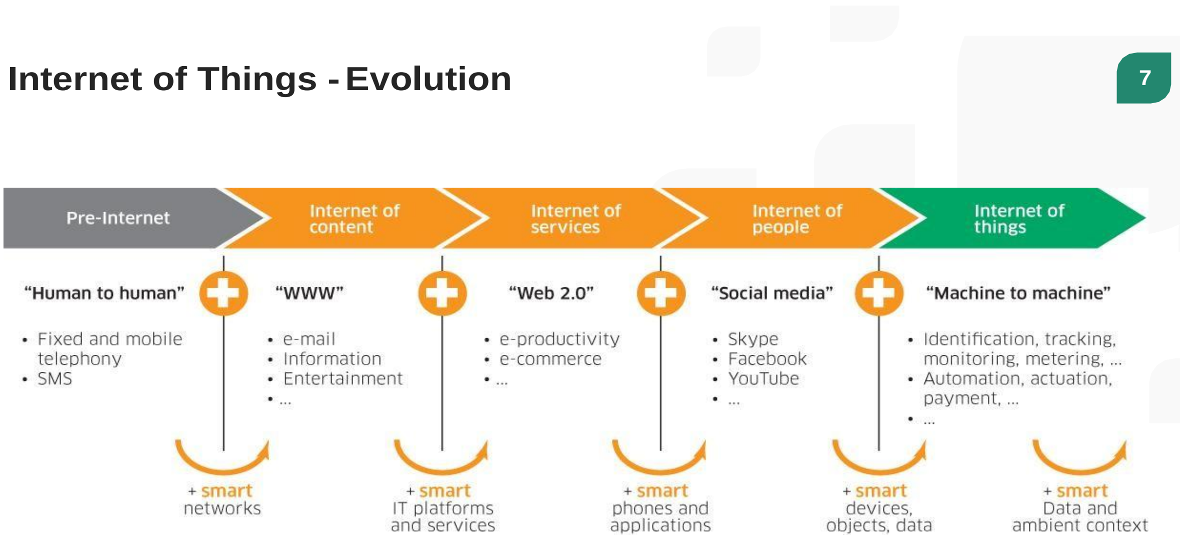


**Fig 1. IOT Evolution Model**

By the late 1970s, for example, systems for remotely monitoring meters on the electrical grid via telephone lines were already in commercial use.14 In the 1990s, advances in wireless technology allowed –machine–to–machine‖ (M2M) enterprise and industrial solutions for equipment monitoring and operation to become widespread. Many of these early M2M solutions, however, were based on closed purpose–built networks and proprietary or industry.

From a broad perspective, the confluence of several technology and market trends20 is making it possible to interconnect more and smaller devices cheaply and easily:

• Ubiquitous Connectivity—Low–cost, high–speed, pervasive network connectivity, especially through licensed and unlicensed wireless services and technology, makes almost everything –connectable‘‘.

• Widespread adoption of IP–based networking— IP has become the dominant global standard for networking, providing a well–defined and widely implemented platform of software and tools that can be incorporated into a broad range of devices easily and inexpensively.

• Miniaturization— Manufacturing advances allow cutting-edge computing and communications technology to be incorporated into very small objects. Coupled with greater computing

economics, this has fueled the advancement of small and inexpensive sensor devices, which drive many IoT applications.

• Advances in Data Analytics— New algorithms and rapid increases in computing power, data storage, and cloud services enable the aggregation, correlation, and analysis of vast quantities of data; these large and dynamic datasets provide new opportunities for extracting information and knowledge.

• Rise of Cloud Computing– Cloud computing, which leverages remote, networked computing resources to process, manage, and store data, allows small and distributed devices to interact with powerful back-end analytic and control capabilities.

From this perspective, the IoT represents the convergence of a variety of computing and connectivity trends that have been evolving for many decades. At present, a wide range of industry sectors – including automotive, healthcare, manufacturing, home and consumer electronics, and well beyond -- are considering the potential for incorporating IoT technology into their products, services, and operations.

## BUSINESS SCOPE

### Increase Business Opportunities

IoT opens the door for new business opportunities and helps companies benefit from new revenue streams developed by advanced business models and services.

IoT-driven innovations build strong business cases, reduce time to market and increase return on investments. IoT has the potential to transform the way consumers and businesses approach the world by leveraging the scope of the IoT beyond connectivity.

**Enhanced Asset Utilization**

IoT will improve tracking of assets (equipment, machinery, tools, etc.) using sensors and connectivity, which helps organizations benefit from real-time insights. Organizations could more easily locate issues in the assets and run preventive maintenance to improve asset utilization.

**Efficient Processes**

Being connected with a maximum number of devices to the internet, IoT allow businesses to be smarter with real-time operational insights while reducing operating costs. The data collected from logistics network, factory floor, and supply chain will help reduce inventory, time to market and downtime due to maintenance.

**Improved Safety and Security**

IoT services integrated with sensors and video cameras help monitor workplace to ensure equipment safety and protect against physical threats. The IoT connectivity coordinates multiple teams to resolve issues promptly.

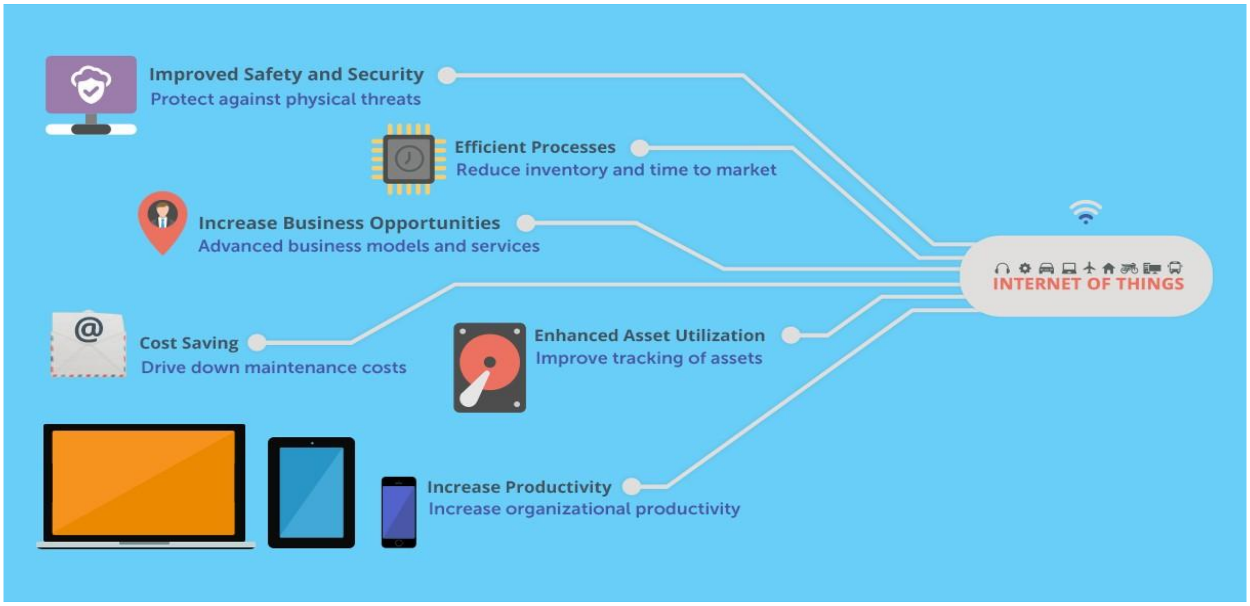


**Fig 2. Business Scope**

## Increase Productivity

Productivity plays a key role in the profitability of any business. IoT offers just-in-time training for employees, improve labor efficiency, and reduce mismatch of skills while increasing organizational productivity.

## Cost Saving

The improved asset utilization, productivity, and process efficiencies can save your expenditures. For example, predictive analytics and real-time diagnostics drive down the maintenance costs.IoT has reached the pinnacle of inflated expectations of emerging technologies. Even though IoT offers great potential value, organizations must overcome some significant challenges like data and information management issues, lack of interoperable technologies, security and privacy concerns, and the skills to manage IoT‘s growing complexity. However, a professional IoT service provider can overcome these challenges and increase your return on investment.

**Logistics**

With IoT sensors, supply chain management and order fulfillment processes improve markedly to meet customer demand. For example, sensors on delivery containers and trucks in transit give managers real-time status updates, allowing them to track their items and ensure they reach the right location at the right time.

Streamlined Industry

IoT also presents automation opportunities for businesses that need to manage and replenish their stock. When data recorded from IoT devices are tied to your enterprise resource planning (ERP) system, you can accurately monitor your inventory, analyze purchase and consumption rates of a particular product, and automatically reorder items when IoT sensors detect that supply is running low. This minimizes out-of-stock incidents and prevents excess stock build-up.

**Fast Payment**

Given how most payments are done electronically via point-of-sale systems or the internet, IoT has the potential to revolutionize the way businesses process transactions. We‘re already seeing a few examples of this today as ApplePay not only allows users to purchase goods and services using smartphone applications, but through wearable technology as well.

Soon enough, IoT devices might even allow restaurants and retailers to register or charge their customers the moment they walk through the door.

**Market Insight**

Businesses that can somehow make sense of IoT-collected data will gain a competitive edge. Marketers, for example, can gather valuable insight into how their products are used and which demographic is utilizing them the most. This information can then inform future marketing efforts and give businesses more direction on how to improve their products and services for their customers. Although businesses will certainly face many challenges in implementing the Internet of Things, those who manage to overcome them will reap all the benefits of this burgeoning technology.

## 3. RELATIONSHIP WITH EMBEDDED SYSTEMS

Embedded systems are part and parcel of every modern electronic component. These are low power consumption units that are used to run specific tasks for example remote controls, washing machines, microwave ovens, RFID tags, sensors, actuators and thermostats used in various applications, networking hardware such as switches, routers, modems, mobile phones, PDAs, etc. Usually embedded devices are a part of a larger device where they perform specific task of the device. For example, embedded systems are used as networked thermostats in Heating, Ventilation and Air Conditioning (HVAC) systems, in Home Automation embedded systems are used as wired or wireless networking to automate and control lights, security, audio/visual systems, sense climate change, monitoring, etc. Embedded microcontrollers can be found in practically all machines, ranging from DVD players and power tools to automobiles and computed tomography scanners. They differ from PCs in their size and processing power. Embedded systems typically have a microprocessor, a memory, and interfaces with the external world, but they are considerably smaller than their PC counterparts. Frequently, the bulk of the electronic circuitry can be found in a single chip.

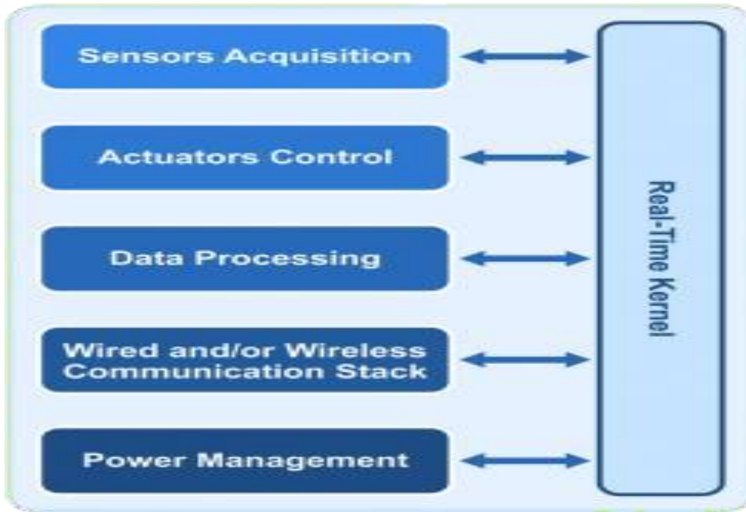


**Fig 3. Embedded Processing**

A sensor detects (senses) changes in the ambient conditions or in the state of another device or a system, and forwards or processes this information in a certain manner.

- **Analog Sensors** produce a continuous output signal or voltage which is generally proportional to the quantity being measured.
- Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc. are all analog quantities as they tend to be continuous in nature.
- **Digital Sensors** produce discrete digital output signals or voltages that are a digital representation of the quantity being measured.
- Digital sensors produce a binary output signal in the form of a logic –1‖ or a logic –0‖, (–ON‖ or –OFF‖).

**An actuator is a component of a machine or system that moves or controls the mechanism** or the system. An actuator is the mechanism by which a control system acts upon an environment

An actuator requires a control signal and a source of energy.

**Power Conservation**

Until recently, a common strategy to save power in an embedded system was to execute as quickly as possible, and then go into sleep mode immediately. But there are now processor core architectures that consume almost no power, although with reduced performance. This is an attractive option for a WSN edge node design.

The programming languages used in deeply embedded systems include C, C++ and sometimes Java. It is important to note that Java always runs on top of an operating system. So, your choice is not between C/C++ or Java; it is whether you will use C/C++ and Java. Java is attractive for IoT devices because the number of Java developers worldwide brings tremendous growth potential to the industry. Oracle‘s Java ME Embedded is designed for small devices

When cost is not an issue, we can select a single powerful processor to run all the tasks required of your device. However, a common engineering compromise is to use two processors in the sensor/actuator device. One low-cost processor (8 or 16 bit) is used for the physical-world interface, and a second 32-bit processor runs the network interface. This second processor is often placed in a separate module, one that has already been certified for the protocol and FCC compliance.

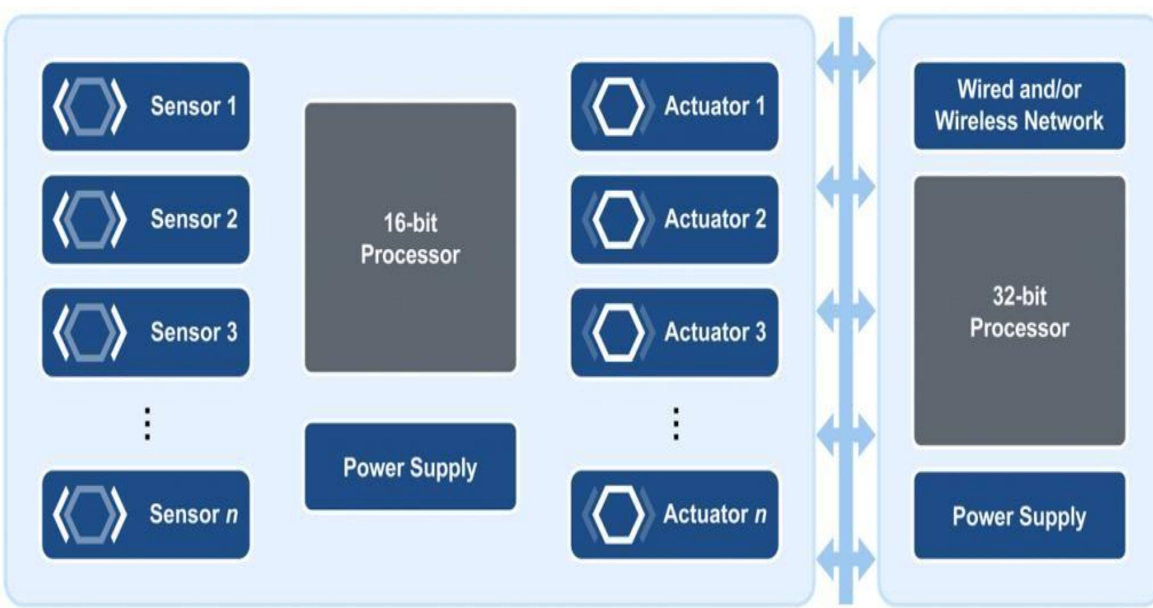


**Fig 4. IoT Devices with Two Processors**

**Gateway Design**

A gateway connects two dissimilar networks so that data can flow between them. Usually this is a connection between a proprietary network and the Internet.

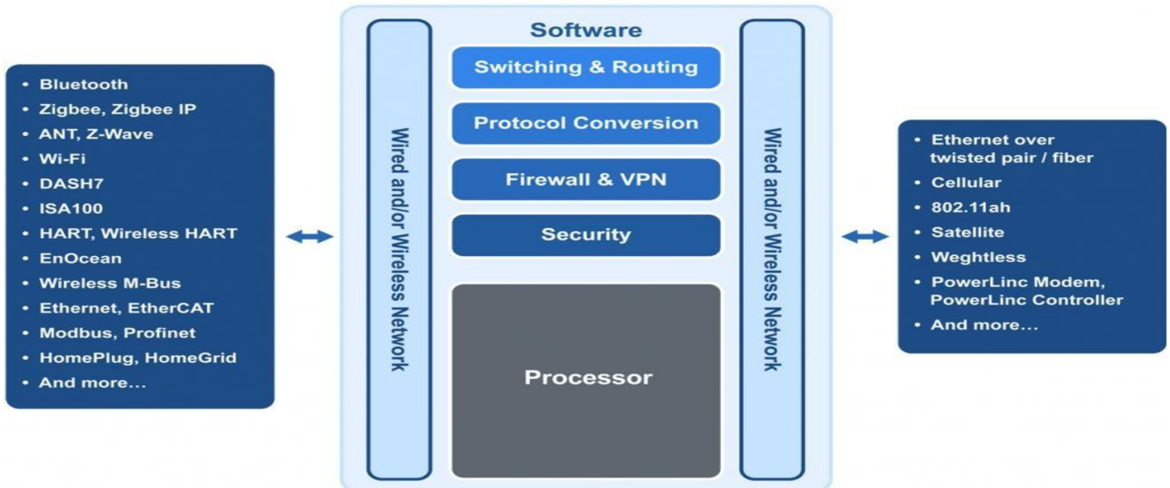


**Fig 5 Embedded Devices with Gateway**

**Bluetooth** is a wireless technology standard for exchanging data over short distances from fixed and mobile devices, and building personal area networks.

**Zigbee** wireless technology is specially designed for sensors and control devices that employ low cost connectivity and widely used for several applications.

**Z-Wave** is a **wireless communications protocol** used primarily for home automation. It is a mesh network using **low-energy radio waves to communicate from appliance to appliance**, allowing for wireless control of residential appliances and other devices, such as lighting control, security systems, thermostats, windows.

**Wi-Fi** is the name of a popular wireless networking technology that uses **radio waves to provide wireless high-speed Internet** and network connections. A common misconception is that the term **Wi-Fi** is short for **"wireless fidelity**.

**ISA100.11a** is a wireless networking technology standard developed by the International Society of Automation (ISA). The official description is "Wireless

Systems for Industrial Automation: Process Control and Related Applications.

The **EnOcean** technology is an energy harvesting **wireless technology used primarily in building automation systems,** and is also applied to other applications in industry, transportation, logistics and smart homes. Modules based on EnOcean technology **combine micro energy converters with ultra low power electronics, and enable wireless communications between batteryless wireless sensors, switches, controllers and gateways.**

In home automation, different utilities companies may install a wide variety of IoT devices in your house, each with their own gateway. These can include electricity or gas, water, phone, Internet, cable/satellite, alarm system, medical devices, and so on. Some of these gateways may require additional functions, such as local storage, or a user interface.

## 4. INTRODUCTION TO AUDUINO AND RASPBERRYPI

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. Accepts analog and digital signals as input and gives desired output.

BOARD DETAILS:

| **ARDUIN0 UN0** | |
|---|---|
| **Feature** | **Value** |
| **OperatingVoltage** | **5V** |
| **ClockSpeed** | **16MHz** |
| **Digital I/O** | **14** |
| **AnalogInput** | **6** |
| **PWM** | **6** |
| **UART** | **1** |
| **Interface** | **USB via ATMega16U2** |

- **Power Supply:**
- **USB or power barrel jack**
- **Voltage Regulator**
- **LED Power Indicator**
- **Tx-Rx LED Indicator**
- **Output power,**
- **Ground**
- **Analog Input Pins**
- **Digital I/O Pin**

**SET UP:**

- Power the board by connecting it to a PC via USB cable
- Launch the Arduino IDE
- Set the board type and the port for the board
- TOOLS -> BOARD -> select your board
- TOOLS -> PORT -> select your port

**TYPES:**

1. Arduino Uno (R3)
2. LilyPad Arduino
3. RedBoard
4. Arduino Mega (R3)
5. Arduino Leonardo

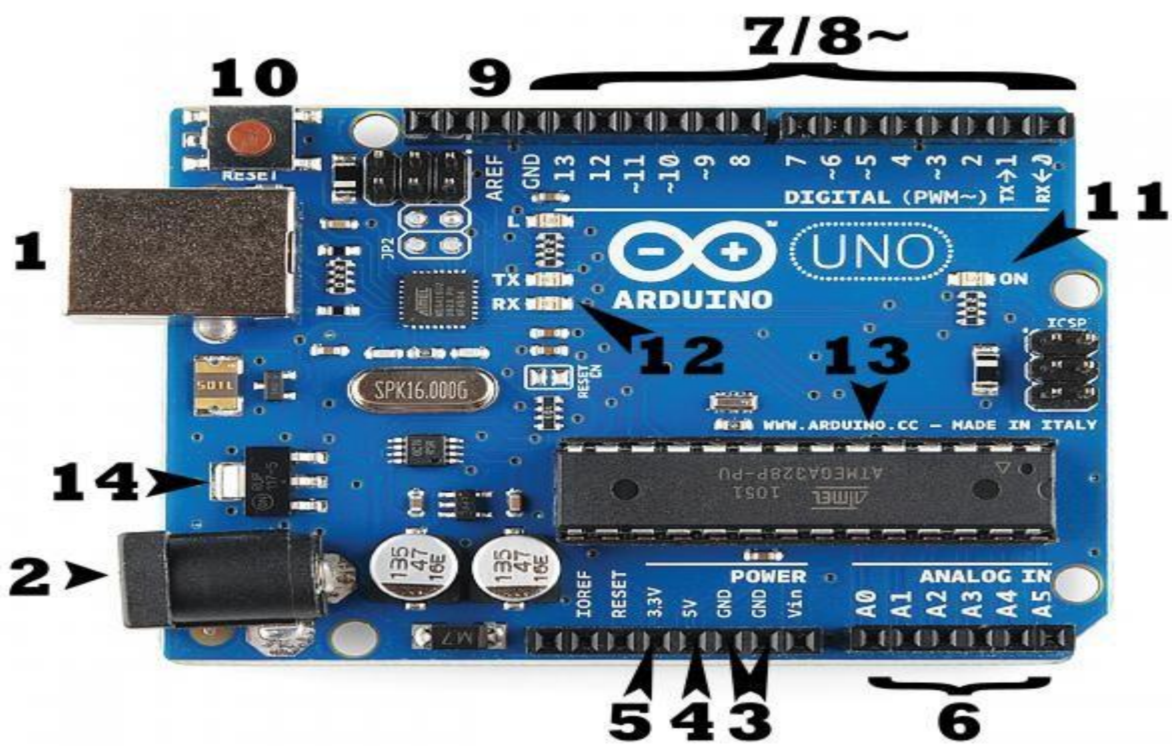


**Fig 6. Arduino Board**

**Power (USB / Barrel Jack):**

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).The USB connection is also how you will load code onto your Arduino board.

**NOTE:** Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

**Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):**

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic ‗headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

**GND (3)**: Short for ‗Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

**5V (4) & 3.3V (5)**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

**Analog (6)**: The area of pins under the ‗Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

**Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the

UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

**PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

**AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

### Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn‘t repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn‘t usually fix any problems.

### Power LED Indicator

Just beneath and to the right of the word –UNO‖ on your circuit board, there‘s a tiny LED next to the word ‗ON‘ **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn‘t turn on, there‘s a good chance something is wrong. Time to re-check your circuit!

### TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we‘re loading a new program onto the board).

**Main IC**

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board

type, but is usually from the ATmega line of IC‘s from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC‘s, reading the datasheets is often a good idea.p

**Voltage Regulator**

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it‘s for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don‘t hook up your Arduino to anything greater than 20 volts.

**ARDINO IDE OVERVIEW**:

Program coded in Arduino IDE is called a SKETCH

1. To create a new sketchFile -> New

To open an existing sketch File -> open ->

There are some basic ready-to-use sketches available in the EXAMPLES section File -> Examples -> select any program

2. Verify: Checks the code for compilation errors
3. Upload: Uploads the final code to the controller board
4. New: Creates a new blank sketch with basic structure
5. Open: Opens an existing sketch
6. Save: Saves the current sketch


**Fig 7.Compilation and Execution**

Serial Monitor: Opens the serial console

- All the data printed to the console are displayed here

**SKETCH STRUCTURE**


**Fig 8.Structure of SKETCH**

A sketch can be divided into two parts:

- Setup ()
- Loop()

The function setup() is the point where the code starts, just like the main() function in C and C++ I/O Variables, pin modes are initialized in the Setup() function  Loop() function, as the name suggests, iterates the specified task in the program

**DATA TYPES:**

Void ,Long, Int ,Char ,Boolean, Unsigned char ,Byte, Unsigned int, Word ,Unsigned long ,Float, Double, Array ,String-char array, String-object, Short

**Arduino Function libraries**

Input/Output Functions:

The arduino pins can be configured to act as input or output pins using the pinMode() function Void setup ()

```
{
pinMode (pin , mode);
}
```

Pin- pin number on the Arduino board Mode- INPUT/OUTPUT

**digitalWrite()** : Writes a HIGH or LOW value to a digital pin

**analogRead()** : Reads from the analog input pin i.e., voltage applied across the pin

Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false)

Delay() function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds)

**EXAMPLE BLINKING LED:**

Requirement:

-  Arduino controller board, USB connector, Bread board, LED, 1.4Kohm resistor, connecting wires, Arduino IDE
-  Connect the LED to the Arduino using the Bread board and the connecting wires
-  Connect the Arduino board to the PC using the USB connector
-  Select the board type and port  Write the sketch in the editor, verify andupload

Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino Board

```
void setup()
{
pinMode(12, OUTPUT); // set the pin mode
} void loop()
{
digitalWrite(12, HIGH); // Turn on the LED
delay(1000); digitalWrite(12, LOW); //Turn of
the LED delay(1000);
}
```

Set the pin mode as output which is connected to the led, pin 12 in this case. Use digitalWrite() function to set the output as HIGH and LOW

Delay() function is used to specify the delay between HIGH-LOW transition of the output

**Connect he board to the PC**

-  Set the port and board type
-  Verify the code and upload,

notice the TX – RX led in the board starts flashing as the code is uploaded.

**RASPBERRY PI:**

Raspberry Pi is a credit card sized micro processor available in different models with different processing speed starting from 700 MHz. Whether you have a model B or model B+, or the very old version, the installation process remains the same. People who have checked out the official Raspberry Pi website, But using the Pi is very easy and from being a beginner, one will turn pro in no time. So, it's better to go with the more powerful and more efficient OS, the Raspbian. The main reason why Raspbian is extremely popular is that it has thousands of pre built libraries to perform many tasks and optimize the OS. This forms a huge advantage while building applications.

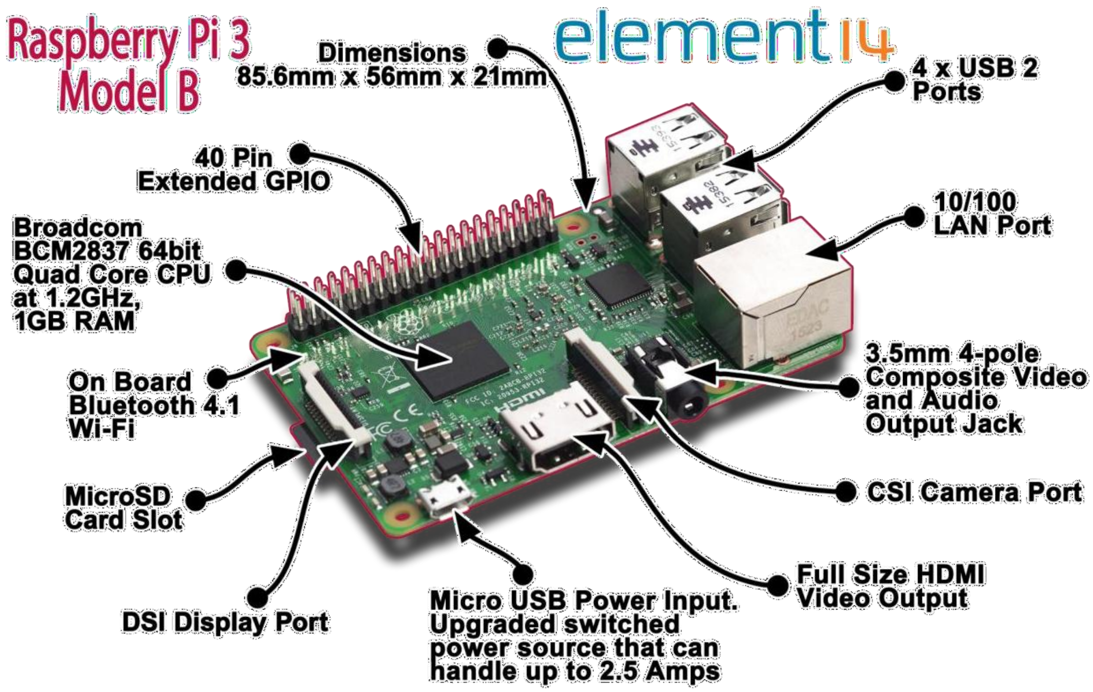


**Fig 9. Raspberry Pi  Element**

**Specifications and performance**

As for the specifications, the Raspberry Pi is a credit card-sized computer powered by the Broadcom BCM2835 system-on-a-chip (SoC). This SoC includes a 32-bit ARM1176JZFS processor, clocked at 700MHz, and a Videocore IV GPU.

It also has 256MB of RAM in a POP package above the SoC. The Raspberry Pi is powered by a 5V micro USB AC charger or at least 4 AA batteries (with a bit of hacking).

While the ARM CPU delivers real-world performance similar to that of a 300MHz Pentium 2, the Broadcom GPU is a very capable graphics core capable of hardware decoding several high definition video formats. The Raspberry Pi model available for purchase at the time of writing — the Model B — features HDMI and composite video outputs, two USB 2.0 ports, a 10/100 Ethernet port, SD card slot, GPIO (General Purpose I/O Expansion Board) connector, and analog audio output (3.5mm headphone jack). The less expensive Model A strips out the Ethernet port and one of the USB ports but otherwise has the same hardware.

| | Raspberry pi 3 model B | Raspberry pi 2 model B | Raspberry Pi zero |
|---|---|---|---|
| RAM | 1GB SDRAM | 1GB SDRAM | 512 MB SDRAM |
| CPU | Quad cortex A53@1.2GHz | Quad cortex A53@900MHz | ARM 11@ 1GHz |
| GPU | 400 MHz video core IV | 250 MHz video core IV | 250 MHz video core IV |
| Ethernet | 10/100 | 10/100 | None |
| Wireless | 802.11/Bluetooth 4.0 | None | None |
| Video output | HDMI/Composite | HDMI/Composite | HDMI/Composite |
| GPIO | 40 | 40 | 40 |

**Fig 10. Configuration**

Raspberry Pi Basics: installing Raspbian and getting it up and running

1 **Downloading Raspbian and Image writer.**

You will be needing an image writer to write the downloaded OS into the SD card (micro SD card in case of Raspberry Pi B+ model). So download the "win32 disk imager" from the website.

2 **Writing the image**

Insert the SD card into the laptop/pc and run the image writer. Once open, browse and select the downloaded Raspbian image file. Select the correct device, that is the drive representing the SD card. If the drive (or device) selected is different from the SD card then the other selected drive will become corrupted. SO be careful.

After that, click on the "Write" button in the bottom. As an example, see the image below, where the SD card (or micro SD) drive is represented by the letter "G:\"

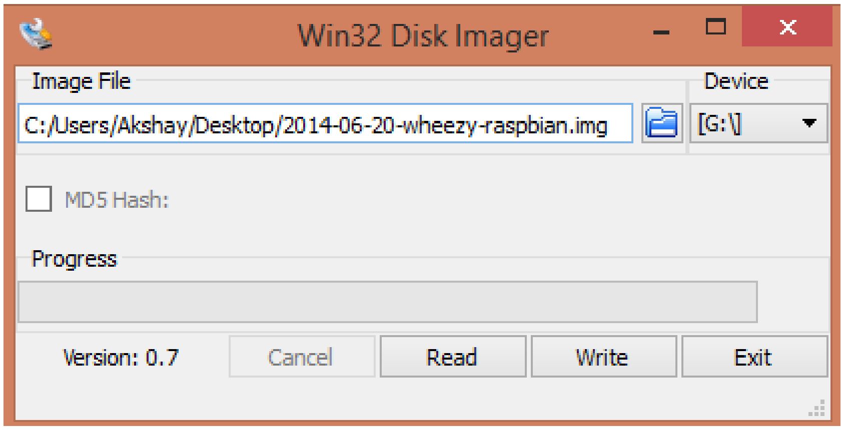


**Fig 1. OS Installation**

Once the write is complete, eject the SD card and insert it into the Raspberry Pi and turn it on. It should start booting up.

## 3 Setting up the Pi

Please remember that after booting the Pi, there might be situations when the user credentials like the "username" and password will be asked. Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are:

**login: pi**

**password: raspberry**

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the image below.


**Fig 12.Raspberry Configuration**

If you have missed the "Setup Options" screen, its not a problem, you can always get it by typing the following command in the terminal.

**sudoraspi-config**

Once you execute this command the "Setup Options" screen will come up as shown in the image above.

Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, please do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

- The first thing to do:

  select the first option in the list of the **setup options window**, that is select the "**Expand Filesystem**" option and hit the enter key. We do this to make use of all the space present on the SD card as a full partition. All this does is, expand the OS to fit the whole space on the SD card which can then be used as the storage memory for the Pi

- The second thing to do:

Select the third option in the list of the setup options window, that is select the "**Enable BootTo Desktop/Scratch" option** and hit the enter key. It will take you to another window called the "**choose boot option**" window that looks like the image below.


**Fig 13.Boot Options**

In the "choose **boot option window**", select the second option, that is, "**Desktop Log in as user 'pi' at the graphical desktop**" and hit the enter button. Once done you will be taken back to the "**Setup Options**" page, if not select the "OK" button at the bottom of this window and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step then the Raspberry Pi boots into a terminal each time with no GUI options.Once, both the steps are done, select the **"finish"** button at the bottom of the page and it should reboot automatically. If it doesn't, then use the following command in the terminal to reboot.

**sudo reboot**

Updating the firmware

After the reboot from the previous step, if everything went right, then you will end up on the desktop which looks like the image below.

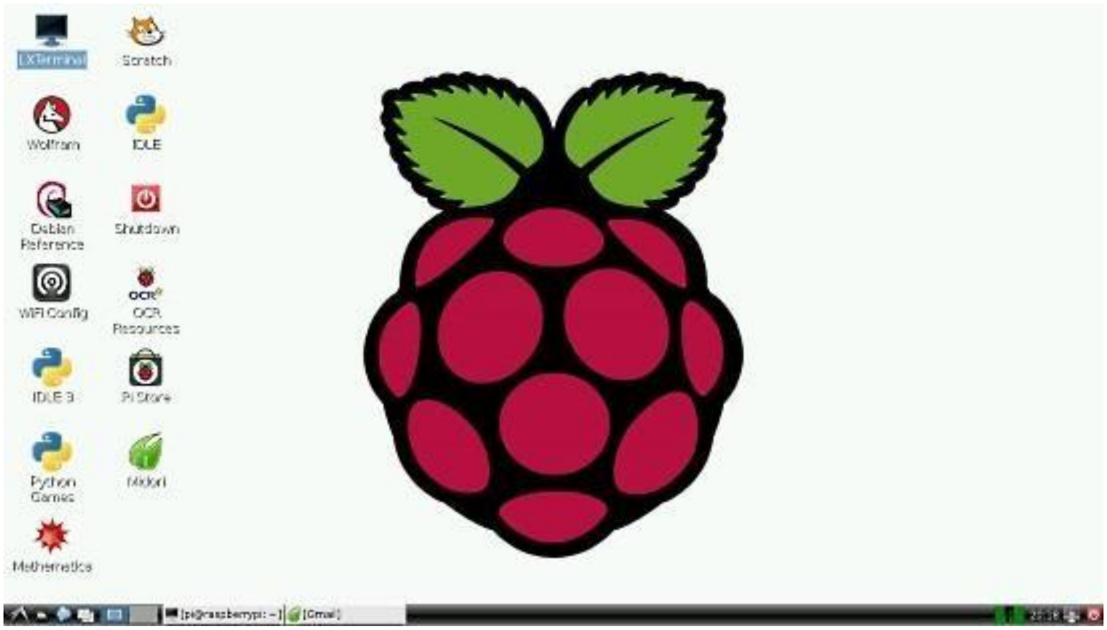


**Fig 14.Raspberry Desktop**

Once you are on the desktop, open a terminal and enter the following command to update the firmware of the Pi.

**sudorpi-update**

Updating the firmware is necessary because certain models of the Pi might not have all the required dependencies to run smoothly or it may have some bug. The latest firmware might have the fix to those bugs, thus its very important to update it in the beginning itself.

5 Conclusion

So, we have covered the steps to get the Pi up and running. This method works on all the different models of Raspberry Pi (model A, B, B+ and also RPi 2) as Raspbain was made to be supported on all models. However, while installing other software or libraries, the procedure might change a bit while installing depending on the model of the Pi or the version of Raspbian itself. The concept of Raspberry is to keep trying till you get the result or build that you want. This might involve a lot of trial and error but spending the time will be worth it. The actual usage doesn't end here. This is just the beginning. It is up to you to go ahead to build something amazing out of it.

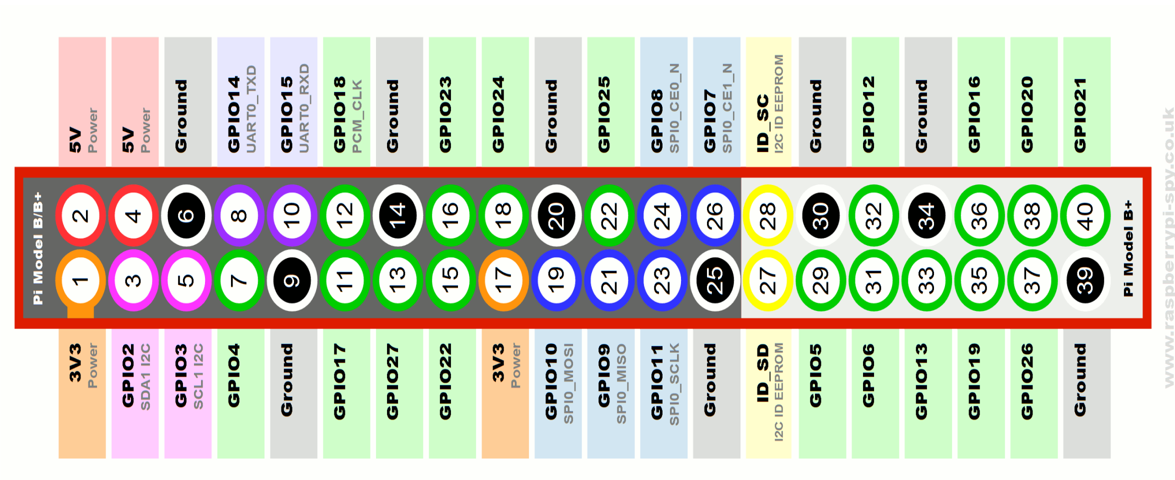


**Fig 15.GPIO Pins**

GPIO:

Act as both digital output and digital input.

**Output**: turn a GPIO pin high or low.

**Input**: detect a GPIO pin high or low

**Installing GPIO library:**

Open terminal

Enter the command –sudoapt**-get install python-dev”** to install python

development Enter the command **“sudoapt-get install python-**

**rpi.gpio”** to install GPIO library. **Basic python coding:**

Open terminal enter the command

**sudonanofilename.py**

This will open the nano editor where you can write your code Ctrl+O : Writes the code to the file

Ctrl+X : Exits the editor

**Blinking LED Code:**

```
import RPi.GPIO as GPIO #GPIO library import time
GPIO.setmode(GPIO.BOARD) # Set the type of board for pin
numbering GPIO.setup(11, GPIO.OUT) # Set GPIO pin 11as output
pin
for i in range (0,5):
     GPIO.output(11,True) # Turn on GPIO
    pin 11 time.sleep(1)
    GPIO.output(11,False)
T   ime.sleep(2)
    GPIO.output(11,Tr
    ue)
GPIO.cleanup()
```

**Power Pins**

The header provides 5V on Pin 2 and 3.3V on Pin 1. The 3.3V supply is limited to 50mA. The 5V supply draws current directly from your microUSB supply so can use whatever is left over after the board has taken its share. A 1A power supply could supply up to 300mA once the Board has drawn 700mA.

**Basic GPIO**

The header provides 17 Pins that can be configured as inputs and outputs. By default they are all configured as inputs except GPIO 14 & 15.

In order to use these pins you must tell the system whether they are inputs or outputs. This can be achieved a number of ways and it depends on how you intend to control them. I intend on using Python.

**SDA & SCL:** The 'DA' in SDA stands for data, the 'CL' in SCL stands for clock; the S stands for serial. You can do more reading about the significance of the clock line for various types of computer bus, You will probably find $I^2C$ devices that come with their own userspace drivers and the linux kernel includes some as well. Most computers have an $I^2C$ bus, presumably for some of the purposes listed by wikipedia, such as interfacing with the RTC (real time clock) and configuring memory. However, it is not exposed, meaning you can't attach anything else to it, and there are a lot of interesting things that could be attached -- pretty much any kind of common sensor (barometers, accelerometers, gyroscopes, luminometers, etc.) as well as output devices and displays. You can buy a USB to $I^2C$ adapter for a normal computer, but they cost a few hundred dollars. You can attach multiple devices to the exposed bus on the pi.

**UART, TXD & RXD:** This is a traditional serial line; for decades most computers have had a port for this and a port for parallel.[1] Some pi oriented OS distros such as Raspbian by default boot with this serial line active as a console, and you can plug the other end into another computer and use some appropriate software to communicate with it. Note this interface does not have a clock line; the two pins may be used for full duplex communication (simultaneous transmit and receive).

**PCM, CLK/DIN/DOUT/FS:** PCM is is how uncompressed digital audio is encoded. The data stream is serial, but interpreting this correctly is best done with a separate clock line (more lowest level stuff).

**SPI, MOSI/MISO/CE0/CE1:** SPI is a serial bus protocol serving many of the same purposes as $I^2C$, but because there are more wires, it can operate in full duplex which makes it faster and more flexible.

**Raspberry Pi Terminal Commands**

[sudo apt-get update] - Update Package Lists

[sudo apt-get upgrade] - Download and Install Updated

Packages [sudoraspi-config] - The Raspberry Pi

Configuration Tool

[sudo apt-get clean] - Clean Old
Package Files [sudo reboot] - Restart
your Raspberry Pi [sudo halt] - Shut
Down your Raspberry Pi

## 6. KEY ELEMENTS OF IOT

### 1. Sensing

The first step in IoT workflow is gathering information at a –point of activity.‖ This can be information captured by an appliance, a wearable device, a wall mounted control or any number of commonly found devices. The sensing can be biometric, biological, environmental, visual or audible (or all the above). The unique thing in the context of IoT is that the device doing the sensing is not one that typically gathered information in this way. Sensing technology specific to this purpose is required.

### 2. Communication

This is where things start to get interesting. Many of the new IoT devices we are seeing today are not designed for optimal communication with cloud services**. IoT devices require a means for transmitting the information sensed at the device level to a Cloud-based service for subsequent processing.** This is where the great value inherent in IoT is created. This requires either **WiFi** (wireless LAN based communications) or **WAN** (wide area network… i.e. cellular) communications. In addition, depending on the need short range communication, other capabilities may also be needed. These could include **Bluetooth, ZigBee, Near-field or a range of other short range communication methods. For positioning, GPS** is often required as well.

### 3. Cloud Based Capture & Consolidation

Gathered data is transmitted to a cloud based service where the information coming in from the IoT device is aggregated with other cloud based data to provide useful information for the end user. The data being consolidated can be information from other internet sources as well as from others subscribing with similar IoT devices. Most often, there will be some data processing required to provide useful information that is not necessarily obvious in the raw data.

**4. Delivery of Information**

The last step is delivery of useful information to the end user. That may be a consumer, a commercial or an industrial user. It may also be another device in the M2M workflow. The goal in a consumer use case is to provide the information in as simple and transparent a method as possible. It requires execution of a well thought out, designed and executed user interface that provides an optimized experience across multiple device platforms – tablets, smartphones, desktop – across multiple operating systems – iOS, Android, Windows, etc.

## 7.REFERENCE ARCHITECTURE OF IOT:

The reference architecture consists of a set of components. Layers can be realized by means of specific technologies, and we will discuss options for realizing each component.There are also some cross-cutting/vertical layers such as access/identity management.

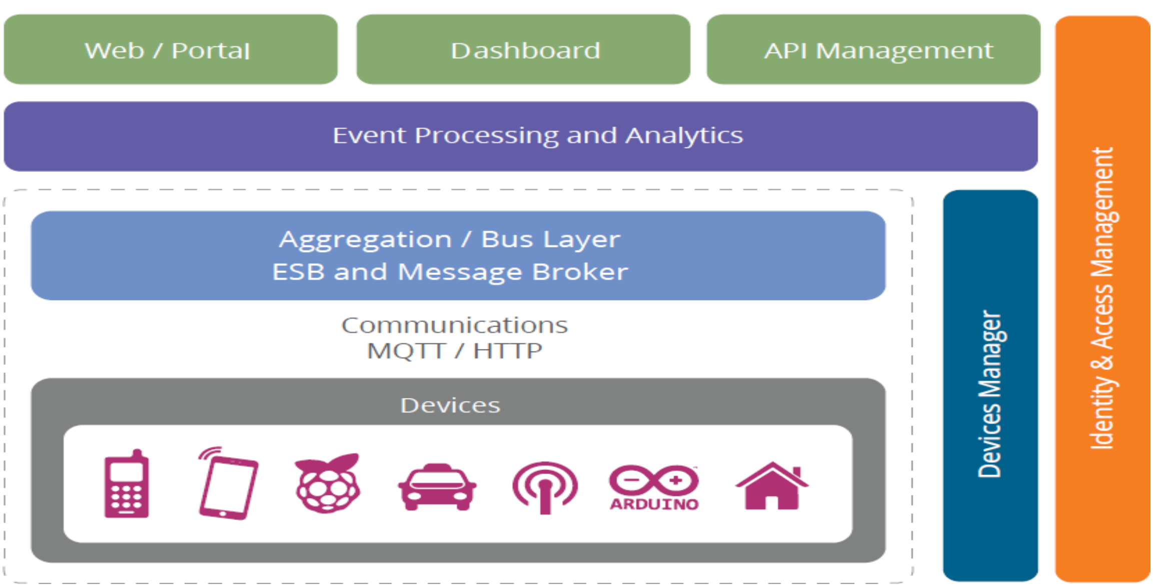

**Fig 16.Reference architecture for IoT**

The layers are

• Client/external communications - Web/Portal, Dashboard, APIs

• Event processing and analytics (including data storage)

• Aggregation/bus layer – ESB and message broker

• Relevant transports - MQTT/HTTP/XMPP/CoAP/AMQP, etc.

• Devices

The cross-cutting layers are

• Device manager

• Identity and access managements

**THE DEVICE LAYER**

The bottom layer of the architecture is the device layer. Devices can be of various types, but in order to be considered as IoT devices, they must have some communications that either indirectly or directly attaches to the Internet. Examples of direct connections are

• Arduino with Arduino Ethernet connection

• Arduino Yun with a Wi-Fi connection

• Raspberry Pi connected via Ethernet or Wi-Fi

• Intel Galileo connected via Ethernet or

Wi-Fi Examples of indirectly connected device include

• ZigBee devices connected via a ZigBee gateway

• Bluetooth or Bluetooth Low Energy devices connecting via a mobile phone

• Devices communicating via low power radios to a

Raspberry Pi There are many more such examples of each type.

Each device typically needs an identity. The identity may be one of the following:

• A unique identifier (UUID) burnt into the device (typically part of the System-on-

Chip, or provided by a secondary chip)

• A UUID provided by the radio subsystem (e.g. Bluetooth identifier, Wi-Fi MAC address)

• An OAuth2 Refresh/Bearer Token (this may be in addition to one of the above)

• An identifier stored in nonvolatile memory such as EEPROM

For the reference architecture we recommend that every device has a UUID (preferably an unchangeable ID provided by the core hardware) as well as an OAuth2 Refresh and Bearer token stored in EEPROM.

The specification is based on HTTP; however, (as we will discuss in the communications section) the reference architecture also supports these flows over MQTT.

**COMMUNICATIONS LAYER**

The communication layer supports the connectivity of the devices. There are multiple potential protocols for communication between the devices and the cloud. The most wellknown three potential protocols are

• HTTP/HTTPS (and RESTful approaches on those)

• MQTT 3.1/3.1.1(**Message Queuing Telemetry Transport)**

• Constrained application protocol (CoAP)

Let‘s take a quick look at each of these protocols in turn.

HTTP is well known, and there are many libraries that support it. Because it is a simple text based protocol, many small devices such as 8-bit controllers can only partially support the protocol – for example enough code to POST or GET a resource. The larger 32-bit based devices can utilize full HTTP client libraries that properly implement the whole protocol. There are several protocols optimized for IoT use. The two best known are MQTT**6** and CoAP**7**. **MQTT was invented in 1999 to solve issues in embedded systems and SCADA.** It has been through some iterations and the current version (3.1.1) is undergoing standardization in the OASIS MQTT Technical Committee**8**. **MQTT is a publish-subscribe messaging system based on a broker model.** The protocol has a very small overhead (as little as 2 bytes per message), and was designed to support lossy and intermittently connected networks. **MQTT was designed to flow over TCP.** In addition there is an associated specification designed for ZigBee-style networks called MQTT-SN (Sensor Networks). **CoAP is a protocol from the IETF that is designed to provide a RESTful application protocol modeled on HTTP semantics**, but with a much smaller footprint and a binary rather than a text- based approach. CoAP is a more traditional client-server approach rather than a brokered approach**. CoAP is designed to be used over UDP.**

For the reference architecture we have opted to select MQTT as the preferred device communication protocol, with HTTP as an alternative option.

The reasons to select MQTT and not CoAP at this stage are

• Better adoption and wider library support for MQTT;

• Simplified bridging into existing event collection and event processing systems; and

• Simpler connectivity over firewalls and NAT networks

However, both protocols have specific strengths (and weaknesses) and so there will be some situations where CoAP may be preferable and could be swapped in. In order to support MQTT we need to have an MQTT broker in the architecture as well as device libraries. We will discuss this with regard to security and scalability later.

One important aspect with IoT devices is not just for the device to send data to the cloud/ server, but also the reverse. This is one of the benefits of the MQTT specification: because it is a brokered model, clients connect an outbound connection to the broker, whether or not the device is acting as a publisher or subscriber. This usually avoids firewall problems because this approach works even behind firewalls or via NAT. In the case where the main communication is based on HTTP, the traditional approach for sending data to the device would be to use HTTP Polling. This is very inefficient and costly, both in terms of network traffic as well as power requirements. The modern replacement for this is the WebSocket protocol**9** that allows an HTTP connection to be upgraded into a full two-way connection. This then acts as a socket channel (similar to a pure TCP channel) between the server and client. Once that has been established, it is up to the system to choose an ongoing protocol to tunnel over the connection. For **the reference architecture we once again recommend using MQTT as a protocol with WebSockets**. In some cases, MQTT over WebSockets will be the only protocol. This is because it is even more firewall-friendly than the base MQTT specification as well as supporting pure browser/JavaScript clients using the same protocol. Note that while there is some support for WebSockets on small controllers, such as Arduino, the combination of network code, HTTP and WebSockets would utilize most of the available code space on a typical Arduino 8-bit device. **Therefore, we only recommend the use of WebSockets on the larger 32-bit devices.**

## AGGREGATION/BUS LAYER

An important layer of the architecture is the layer that aggregates and brokers communications. This is an important layer for three reasons:

1. The ability to support an HTTP server and/or an MQTT broker to talk to the devices
2. The ability to aggregate and combine communications from different devices and to route communications to a specific device (possibly via a gateway)
3. The ability to bridge and transform between different protocols, e.g. to offer HTTP based APIs that are mediated into an MQTT message going to the device.

The aggregation/bus layer provides these capabilities as well as adapting into legacy protocols. The bus layer may also provide some simple correlation and mapping from different correlation models (e.g. mapping a device ID into an owner's ID or vice-versa). Finally the aggregation/bus layer needs to perform two key security roles. It must be able to act as an OAuth2 Resource Server (validating Bearer Tokens and associated resource access scopes). It must also be able to act as a policy enforcement point (PEP) for policy-based access. In this model, the bus makes requests to the identity and access management layer to validate access requests. The identity and access management layer acts as a policy decision point (PDP) in this process. The bus layer then implements the results of these calls to the PDP to either allow or disallow resource access.

## EVENT PROCESSING AND ANALYTICS LAYER

This layer takes the events from the bus and provides the ability to process and act upon these events. A core capability here is the requirement to store the data into a database. This may happen in three forms. The traditional model here would be to write a server side application, e.g. this could be a JAX-RS application backed by a database. However, there are many approaches where we can support more agile approaches. The first of these is to use a big data analytics platform. This is a cloud-scalable platform that supports technologies such as Apache Hadoop to provide highly scalable map reduce analytics on the data coming from the devices. The second approach is to support complex event processing to initiate near real-time activities and actions based on data from the devices and from the rest of the system. Our recommended approach in this space is to use the following approaches:

- Highly scalable, column-based data storage for storing events
- Map-reduce for long-running batch-oriented processing of data
- Complex event processing for fast in-memory processing and near real-time reaction and autonomic actions based on the data and activity of devices and other systems
- In addition, this layer may support traditional application processing platforms, such as Java Beans, JAX-RS logic, message-driven beans, or alternatives, such as node.js, PHP, Ruby or Python.

## CLIENT/EXTERNAL COMMUNICATIONS LAYER

The reference architecture needs to provide a way for these devices to communicate outside of the device-oriented system. This includes three main approaches. Firstly, we need the ability to create web-based front-ends and portals that interact with devices and with the event-processing layer. Secondly, we need the ability to create dashboards that offer views into analytics and event processing. Finally, we need to be able to interact with systems outside this network using machine-to-machine communications (APIs). These APIs need to be managed and controlled and this happens in an API management system. The recommended approach to building the web front end is to utilize a modular front-end architecture, such as a portal, which allows simple fast composition of useful UIs. Of course the architecture also supports existing Web server-side technology, such as Java Servlets/ JSP, PHP, Python, Ruby, etc. Our recommended approach is based on the Java framework and the most popular Java-based web server, Apache Tomcat. The dashboard is a re-usable system focused on creating graphs and other visualizations of data coming from the devices and the event processing layer.

The API management layer provides three main functions:

• The first is that it provides a developer-focused portal (as opposed to the user focused portal previously mentioned), where developers can find, explore, and subscribe to APIs from the system. There is also support for publishers to create, version, and manage the available and published APIs;

• The second is a gateway that manages access to the APIs, performing access control checks (for external requests) as well as throttling usage based on policies. It also performs routing and load- balancing;

• The final aspect is that the gateway publishes data into the analytics layer where it is stored as well as processed to provide insights into how the APIs are used.

## DEVICE MANAGEMENT

Device management (DM) is handled by two components. A server-side system (the device manager) communicates with devices via various protocols and provides both individualand bulk control of devices. It also remotely manages software and applications deployed on the device. It can lock and/or wipe the device if necessary. The device manager works in conjunction with the device management agents. There are multiple different agents for different platforms and device types. The device manager also needs to maintain the list of device identities and map these into owners. It must also work with the identity and access management layer to manage access controls over devices (e.g. who else can manage the device apart from the owner, how much control does the owner have vs. the administrator, etc.)

There are three levels of device: non-managed, semi-managed and fully managed (NM, SM, FM).

Fully managed devices are those that run a full DM agent. A full DM agent supports:

• Managing the software on the device

• Enabling/disabling features of the device (e.g. camera, hardware, etc.)

• Management of security controls and identifiers

• Monitoring the availability of the device

• Maintaining a record of the device‘s location if available

• Locking or wiping the device remotely if the device is compromised, etc.

Non-managed devices can communicate with the rest of the network, but have no agent involved. These may include 8-bit devices where the constraints are too small to support the agent. The device manager may still maintain information on the availability and location of the device if this is available.

Semi-managed devices are those that implement some parts of the DM (e.g. feature control, but not software management).

## IDENTITY AND ACCESS MANAGEMENT

The final layer is the identity and access management layer. This layer needs to provide the following services:

• OAuth2 token issuing and validation

• Other identity services including SAML2 SSO and OpenID Connect support for identifying inbound requests from the Web layer

• XACML PDP

• Directory of users (e.g. LDAP)

• Policy management for access control (policy control point)

The identity layer may of course have other requirements specific to the other identity and access management for a given instantiation of the reference architecture. In this section we have outlined the major components of the reference architecture as well as specific decisions we have taken around technologies. These decisions are motivated by the specific requirements of IoT architectures as well as best practices for building agile, evolvable, scalable Internet architectures. Of course there are other options, but this reference architecture utilizes proven approaches that are known to be successful in real-life IoT projects we have worked on.