Annamalai University

Faculty of Engineering and Technology

Department of Computer Science and Engineering

M.E(Computer Science and Engineering)

II Semester

CSCSPC22 EMBEDDED CONTROL SYSTEMS AND INTERNET OF THINGS (IoT)

CSCSPC22	EMBEDDED CONTROL SYSTEMS AND INTERNET OF THINGS (IoT)	L	T	P	C
		3	0	0	3

COURSE OBJECTIVES:

- To expose the students about the fundamentals of Embedded System.
- To educate about Firmware design and development.
- To discuss on aspects required in embedded system design techniques.
- To understand the fundamentals of Internet of Things.
- To build a small low cost embedded system using Arduino / Raspberry Pi or equivalent boards and to apply the concept of Internet of Things in the real world scenario.

Embedded System Vs General Computing System - Classification of Embedded System, Purpose of Embedded system, Quality Attributes of Embedded System -Typical Embedded System- Core of Embedded System, Memory, Sensors and Actuators, Communication Interface- Onboard communication interface, External communication interface.

Embedded Firmware Design Approaches- Embedded Firmware Development Languages - Embedded System Development Environment - IDE, Compiler, Linker - Types of File Generated on Cross Compilation-Simulator, Emulator and Debugging- Fundamental issues in Hardware Software Co-design- Integration and Testing of Embedded Hardware and Firmware.

Introduction-Characteristics - Physical design - protocols – Logical design – Enabling technologies – IoT Levels – Domain Specific IoTs – IoT vs. M2M. IoT systems management – IoT Design Methodology – Specifications Integration and Application Development.

Physical device – Raspberry Pi Interfaces – Programming – APIs / Packages – Web services. Intel Galileo Gen2 with Arduino- Interfaces - Arduino IDE – Programming - APIs and Hacks. Various Real time applications of IoT- Connecting IoT to cloud – Cloud Storage for IoT – Data Analytics for IoT – Software & Management Tools for IoT.

IoE – Overview – Architecture-Smart objects and LLNs-Secure mobility. Home automation – Cities: Smart parking – Environment: Weather monitoring – Agriculture: Smart irrigation – Data analytics for IoT – Software & management tools for IoT cloud storage models & Communication APIs – Cloud for IoT – Amazon Web Services for IoT.

UNIT-1

EMBEDDED SYSTEM VS GENERAL COMPUTER SYSTEMS

- The computing revolution began with the general purpose computing requirements. Later it was realized that the general computing requirements are not sufficient for the embedded computing requirements. The embedded computing requirements demand 'something special' if the terms of response to stimuli, meeting the computational deadlines, power efficiency, limited memory availability, etc. Let's take the case of your personal computer, which may be either a desktop PC or a laptop PC or a palmtop PC.
- It is built around a general purpose processor like an Intel@ Centrino or a Duo/Quad core or an AMD Turion^{TM} processor and is designed to support a set of multiple peripherals like multiple USB 2.0 ports, Wi-Fi, Ethernet, video port, IEEEI 394, SD/CF/MMC external interfaces, Bluetooth, etc and with additional interfaces like a CD read/writer, on-board Hard Disk Drive (HDD), gigabytes of RAM, etc.
- You can load any supported operating system (like Windows@ XP (Vista/7, or Red Hat Linux/ Ubuntu Linux, UNIX etc) into the hard disk of your PC. You can write or purchase a multitude of applications for your PC and can use your PC for running a large number of applications (like printing your dear's photo using a printer device connected to the PC and printer software, creating a document using Microsoft @ Office Word tool, etc.) Now let us think about the DVD player you use for playing DVD movies.
- Is it possible for you to change the operating system of your DVD? Is it possible for you to write an application and download it to your DVD player for executing? Is it possible for you to add printer software to your DVD player and connect a printer to your DVD player to take a printout? Is it possible for you to change the functioning of your DVD player to a television by changing the embedded software? The answers to all these questions are 'NO'. Can you see any general purpose interface like Bluetooth or Wi-Fi on your DVD player? Of course 'NO'. The only interface you can find out on the DVD player is the interface for connecting the DVD player with the display screen and one for controlling the DVD player through a remote (May be an IR or any other specific wireless interface).
- Indeed your DVD player is an embedded system designed specifically for decoding digital video and generating a video signal as output to your TV or any other display screen which supports the display inter- face supported by the DVD Player. Let us summaries our findings from the comparison of embedded system and general purpose computing system with the help of a table:

General Purpose Computing System	Embedded System
A system which is a combination of a generic hardware and a General Purpose Operating System for executing a variety of applications	A system which is a combination Of special purpose hardware and embedded OS for executing a specific set of applications
Contains a General Purpose Operating System (GPOS)	may or may not contain the operating system for functioning

CLASSIFICATION OF EMBEDDED SYSTEM

- It is possible to have a multitude of classifications for embedded systems, based Go different criteria. Some of the criteria used in the classification of embedded systems are:
    1. Based on generation
    2. Complexity and performance requirements
    3. Based on deterministic behavior
    4. Based on triggering.
- The classification based on deterministic system behavior is applicable for 'Real Time' systems. The application/task execution behavior for an embedded system can be deterministic or non- deterministic. Based on the execution behavior, Real Time embedded systems are classified into Hard and Soft. We will discuss about hard and soft real time systems in a later chapter. Embedded Systems which are 'Reactive' in nature (Like process control systems in industrial control applications) can be classified based on the trigger. Reactive systems can be either event triggered or time triggered.

Classification Based on Generation

- This classification is based on the Girder in which the embedded processing systems evolved from the First version to where they are today. MAs per this criterion, embedded systems can be classified into:
- **First Generation:** early embedded systems were built around 8bit microprocessors like 8085 and Z80, and 4bit microcontrollers. Simple in hardware circuits with firmware developed in Assembly code. Digital telephone keypads, stepper motor control units etc. are examples of this.

- **Second Generation:** These are embedded systems built around 16bit microprocessors and 8 or 16 bit microcontrollers, following the first generation embedded systems. The instruction set for the second generation processors/controllers were much more complex and powerful than the first generation processors/controllers. Some of the second generation embedded systems contained embedded operating systems for their operation. Data Acquisition Systems, SCADA systems, etc. are examples of second generation embedded systems.
- **Third Generation:** With advances in processor technology, embedded system developers started making use of powerful 32bit processors and 16bit microcontrollers for their design. A new concept of application and domain specific processor controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into the picture. The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved. The processor market was flooded with different types of processors from different vendors. Processors like Intel Pentium, Motorola 68K, etc. gained attention in high performance embedded requirements. Dedicated embedded real time and general purpose operating systems entered into the embedded market. Embedded systems spread its ground to areas like robotics, media, industrial process control, networking, etc.
- **Fourth Generation:** The advent of System on Chips (SOC), reconfigurable processors and multi core processors are bringing high performance, tight integration and miniaturization into the embedded device market. The SOC technique implements a total system on a chip by integrating different functionalities with a processor core on an integrated circuit. We will discuss about SOCs in a later chapter. The fourth generation embedded systems are making use of high performance real time embedded operating systems for their functioning. Smart phone devices, mobile internet devices (MIDs), etc. are examples of fourth generation embedded systems.
- **Classification Based on Complexity and Performance:** This classification is based on the complexity and system performance requirements. According to this classification, embedded systems can be grouped into:
- **Small-Scale Embedded Systems:** Embedded Systems which are simple in application needs and where the performance requirements are not time critical fall under this category. An electronic toy is a typical example of a small-scale embedded system. Small-scale embedded systems are usually built around low performance and low post or 16 bit microprocessors/microcontrollers. A small-scale embedded system may or may not contain an operating system for its functioning.

- **Medium-Scale Embedded Systems:** Embedded systems which are slightly complex in hardware and firmware (software) requirements fall under this category. Medium-scale embedded systems are usually built around medium performance, low cost 16 or 32 bit microprocessors/microcontrollers or digital signal processors, they usually contain an embedded operating system (either general purpose or real time operating system) for functioning.
- **Large-Scale embedded Systems/Complex Systems:** Embedded systems which involve highly complex hardware and firmware requirements fall under this category. They are employed in mission critical applications demanding high performance. Such systems are commonly built around high performance 32 or 64 bit RISC processors/controllers or Reconfigurable System on Chip (RSOC) or multi-core processors and programmable logic devices. They may contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system. Decoding/encoding of media, cryptographic function implementation, etc. are examples for processing requirements which can be implemented using a co-processor/hard- ware accelerator. Complex embedded systems usually contain high performance Real Time Operating System (RTOS) for task scheduling, prioritization and management:

Major Application Areas of Embedded Systems

- We are living in a world where embedded systems play a vital role in our day-to-day life, starting from home to the computer industry, where most of, the people find their job for a livelihood. Embedded technology has acquired a new dimension from its first generation model, the Apollo guidance computer, to the latest radio navigation system combined with in-car entertainment technology and the microprocessor based "Smart" running shoes launched by Adidas in April 2005. The application areas and the products in the embedded domain are countless. A few of the important domains and products are listed below:

    1. ***Consumer electronics:*** Camcorders, cameras, etc.
    2. ***Household appliances:*** Television, DVD players, washing machine, fridge, microwave oven, etc.
    3. ***Home automation and security systems:*** Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, fire alarms, etc.
    4. ***Automotive industry:*** Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.

5. ***Telecom:*** Cellular telephones, telephone Switches, handset multimedia applications, etc.
6. ***Computer peripherals:*** Printers, scanners, fax machines, etc.
7. ***Computer networking systems:*** Network routers, switches, hubs, firewall etc.
8. ***Healthcare:*** Different kinds of scanners, EEG, ECG •machines etc.
9. ***Measurement & Instrumentation:*** Digital multi meters, digital Os, logic analyzers PLC systems, etc.
10. ***Banking & Retail:*** Automatic teller machines (ATM) and currency counters, point of sales (POS)
11. ***Card Readers:*** Barcode, smart card readers, hand held devices etc.

PURPOSE OF EMBEDDED SYSTEMS

- As mentioned in the previous section, embedded systems are used in various domains like consumer Electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking the domain itself, according to the application usage context, they may have different functionalities. Each embedded system is designed to serve the purpose of any one or a combination following tasks:
    1. Data collection/Storage/Representation
    2. Data communication
    3. Data (signal) processing.
    4. Monitoring
    5. Control
    6. Applications specific user interface

1. Data Collection/Storage/Representation

- Embedded systems designed for the purpose of data collection perform acquisition of data from the external world. Data collection is usually done for storage, analysis, manipulation and transmission. The term "data" refers all kinds of information, viz. text, voice, image, video, electrical signals and any other measurable quantities. Data can be either analog (continuous) or digital (discrete). Embedded systems with analog data capturing techniques collect data directly in the form of analog signals whereas embedded systems with digital data collection mechanism converts the analog signal to

corresponding digital signal using analog to digital (A/D) conveners and then collects the binary equivalent of the analog data. If the data is digital, it can be directly captured without any additional interface by digital embedded systems.

- The collected data may be stored directly in the system or may be transmitted to some other systems or it may be processed by the system or it may be deleted instantly after giving a meaningful representation. These actions are purely dependent on the purpose for which the embedded system is designed. Embedded systems designed for pure measurement applications without storage, used in control and instrumentation domain collects data and gives a meaningful representation of the collected data by means of graphical representation or quantity value and deletes the collected data when new data arrives at the data collection terminal. Analog and digital CROs without storage memory are typical examples of this. Any measuring equipment used in the medical domain for monitoring without storage functionality also comes under this category. Some embedded systems store the collected data for processing and analysis. Such systems incorporate a built-plug-in storage memory for storing the captured data. Some of them give the user a meaningful representation of the collected data by visual (graphical/quantitative) or audible means using display units [Liquid Crystal Display (LCD), Light Emitting Diode (LED), etc.] buzzers, alarms, etc. Examples are: measuring instruments with storage memory and monitoring instruments with storage memory used in medical applications. Certain embedded systems store the data and will not give a representation of the same to the user, whereas the data is used for internal processing. A digital camera is a typical example of embedded system with data collection / storage representation of data. Images are captured and digital camera for image capturing/ the captured image may be stored within the memory of the camera. The captured image can also be presented to the user through a graphic LCD unit.

2. Data Communication

- Embedded data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems. As mentioned earlier in this chapter, the data collected by an embedded terminal may require transferring of the same to some other system located remotely. The transmission is achieved either by a wire-line medium or by a wire- less medium. Wire-line medium was the most common choice in all olden days embedded systems. As technology is changing, wireless medium is becoming the de-facto standard for data communication in embedded systems. A wireless medium offers cheaper connectivity solutions and make the communication link free from the hassle of wire bundles. Data can either if be transmitted by analog means or by digital means. Modern industry trends are settling towards digital communication.

- The data collecting embedded terminal itself can incorporate data communication units like wireless storage/display modules (Bluetooth, ZigBee, Wi-Fi, EDGE, GPRS, etc.) or wire-line modules (RS-232C, USB, TCP/IP, PS2, etc.). Certain embedded systems act as a dedicated transmission unit between the sending and receiving terminals, offering sophisticated functionalities like data packetizing, encrypting and decrypting. Network hubs, routers, switches, etc. are typical examples of dedicated data transmission embedded systems. They act as mediators in data communication and provide various features like data security, monitoring etc.

3. Data (Signal) Processing

- As mentioned earlier, the data (voice, image, video, electrical signals and other measurable quantities) collected by embedded systems may be used for various kinds of data processing. Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc. A digital hearing aid is a typical example of an embedded system employing data processing. Digital hearing aid improves the hearing capacity of hearing impaired persons. A digital hearing aid employ

4. Monitoring

- Embedded systems falling under the category are specifically designed for monitoring purpose. Almost all embedded products coming under the medical domain are with monitoring functions only. They are used for determining the state of some variables using input sensors. They cannot impose control over variables. A very good example is the electro cardiogram (ECG) machine for monitoring the heartbeat of a patient. The machine is intended to do the monitoring of the heartbeat. It cannot impose control over the heartbeat. The sensors used in ECG are the different electrodes connected to the patient's body. Some other examples of embedded systems with monitoring function are measuring instruments like digital CRO, digital millimeters, logic analyzers, etc. used in Control & Instrumentation applications. They are used for knowing (monitoring) the status of some variables like current, voltage, etc. They cannot control the variables in turn.

5. Control

- Embedded systems with control functionalities impose control over some variables according to the changes in input variables. A system with control functionality contains both sensors and actuators. Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable. The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range.

- Air conditioner system used in our home to control the room temperature to a specified limit is a typical example for embedded system for control purpose. An air conditioner contains a room temperature- sensing element (sensor) which may be a thermostat and a handheld unit for setting up (feeding) the desired temperature. The handheld unit may be connected to the central embedded unit residing inside the air conditioner through a wireless link or through a wired link. The air compressor unit acts as the actuator. The compressor is controlled according to the current room temperature and the desired temperature set by the end user. Here the input variable is the current room temperature and the controlled variable is also the room temperature. The controlling variable is cool air flow e compressor unit. If the controlled variable and input variable are not at the same value, controlling variable tries to equalize them through t taking actions on the cool air flow.

6. Application Specific User Interface

- These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc. Mobile phone is an example for this. In mobile phone the user interface is provided through the keypad, graphic LCD module, system speaker vibration alert, etc.

TYPICAL EMBEDDED SYSTEM

- A typical embedded system (Fig. 2.1) contains a single chip controller, which acts as the master brain of the system. The controller can be a Microprocessor' (e.g. Intel 8085) or a microcontroller (e.g. Atmel AT89C51) or a Field Programmable Gate Array (FPGA) device (e.g. Xilinx Spartan) or a Digital Signal Processor (DSP) (e.g. Black fin @ Processors from Analog Devices) or an Application Specific Integrated

- Circuit (ASIC)/ Application specific Standard Product (ASSP) (e.g. ADE7760 Single Phase Energy Metering IC from) Analog Devices for energy metering applications). Embedded hardware/software systems are basically designed to regulate a physical variable or to manipulate the state of some devices by sending some control signals to the Actuators or devices connected to the O/p ports of the system, in response to the input signals provided by the end users or Sensors which are connected to the input ports. Hence an embedded system can be viewed as a reactive system.

- The control is achieved by processing the information coming from the sensors and user interfaces, and controlling some actuators that regulate the physical variable. Key boards, push button switches, etc. are examples for common user interface input devices where- as LEDs, liquid crystal displays, piezoelectric buzzers, etc. are examples

for common user interface output devices for a typical embedded system. It should be noted that it is not necessary that all embedded systems should incorporate these I/O user interfaces. It solely depends on the type of the application for which the embedded system is designed. For example, if the embedded system is designed for any handheld application, such as a mobile handset application, then the system should contain user inter- faces like a keyboard for performing input operations and display unit for providing users the status of various activities in progress. Some embedded systems do not require any manual intervention for their operation.

- They automatically sense the variations in the input parameters in accordance with the changes in the real world, to which they are interacting through the sensors which are connected to the input port of the system. The sensor information is passed to the processor after signal conditioning and digitization. Upon receiving the sensor data the processor or brain of the embedded system performs some pre-defined operations with the help of the firmware embedded in the system and sends some actuating signals to the actuator connected to the output port of the embedded system, which in turn acts on the controlling variable to bring the controlled variable to the desired level to make the embedded system work in the desired manner. The Memory of the system is responsible for holding the Control algorithm and other important con-figuration details.
- For most of embedded systems, the memory for storing the algorithm or configuration data is of fixed type, which is a kind of Read Only Memory (ROM) and it is not available for the end user for modifications, which means the memory is protected from unwanted user interaction by imply menting some kind of memory protection mechanism. The most common types of memories used in embedded systems for control algorithm storage are OTP, PROM, UVEPROM, EEPR () M and FLASH. Depending on the control application, the memory size may vary or a few bytes to megabytes. We will discuss them in detail in the coming sections. Sometimes the "stem requires temporary memory for performing arithmetic operations or control algorithm execution and this type of memory is known as "working memory". Random Access Memory (RAM) is used in most of the systems as the working .memory. Various types of RAM like SRAM, DRAM and NVRAM are used for this purpose.
- The size of the RAM also varies from a few bytes to kilobytes or megabytes depending on the application. The details given under the section Memory will give you a more detailed description of the working memory. An embedded system without a control algorithm implemented memory is just like a new born baby. It is having all the peripherals but is not capable of making any decision depending on the situational as well as real world changes. He only difference is that the memory of a new born baby is

self-adaptive, meaning that the baby will two to learn from the surroundings and from the mistakes committed. For embedded systems it is the responsibility of the designer to impart intelligence to the system. In a controller-based embedded system, the controller may contain internal memory for storing the control algorithm and it maybe an EEPROM or FLASH memory varying from a few kilobytes to mega- bytes. Such controllers are called controllers with on-chip ROM, e.g. Atmel AT89C51. Some controllers may not contain on-chip memory and they require an external (off-chip) memory for holding the control Algorithm, e.g. INTEL 803 IAH.

CORE OF THE EMBEDDED SYSTEM

- Embedded systems are domain and application specific and are built around a central core. The core of the embedded system falls into any one of the following categories:
  1. General Purpose and Domain Specific Processors
     a. Microprocessors
     b. Microcontrollers
     c. Digital Signal Processors
  2. Application Specific Integrated Circuits (ASICs)
  3. Programmable Logic Devices (PLDs)
  4. Commercial off-the-shelf Components (COTS)
- If you examine any embedded system you will find that it is built around any of the core units mentioned above.

1. General Purpose and Domain Specific Processors

- Almost 80% of the embedded systems are processor/controller based. The processor may be a micro-processor or a microcontroller or a digital signal processor, depending on the domain and application. Most of the embedded systems in the industrial control and monitoring applications make use of the commonly available microprocessors or microcontrollers whereas domains which require signal processing such as speech coding, speech recognition, etc. make use of special kind of digital signal processors supplied by manufacturers like, Analog Devices, Texas Instruments, etc.

a). Microprocessors:

- A Microprocessor is a silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-de-fined set of instructions, which is specific to the manufacturer. In general the CPU

contains the Arithmetic and Logic Unit (ALU), control unit and working registers. A microprocessor is a dependent unit and it requires the combination of other hardware like memory, timer unit, and interrupts controller, etc. for proper functioning. Intel claims the credit for developing the first microprocessor unit Intel 4004, a 4bit processor which was released in November 1971. It featured 1k data memory, a 12bit program counter and 4K program memory, sixteen 4bit general purpose registers and 46 instructions. It ran at a clock speed of 740 kHz. It was designed for olden day's calculators. In 1972, 14 more instructions were added to the 4004 instruction set and the program space is upgraded to 8K. Also interrupt capabilities were added to it and it is renamed as Intel 4040. It was quickly replaced in April 1972 by Intel 8008 which was similar to Intel 4040, the only difference was that its program counter was 14 bits wide and the 8008 served as a terminal controller. In April 1974 Intel launched the first 8 bit processor, the Intel 8080, with 16bit address bus and program counter and seven 8bit registers (A-E,H,L: BC, DE, and HL pairs formed the 16bit register for this processor). Intel 8080 was the most commonly used processors for industrial control and other embedded applications in the 1975s. Since the processor required other hardware components as mentioned earlier for its proper functioning, the systems made out of it were bulky and were lacking compactness.

- Immediately after the release of Intel 8080, Motorola also entered the market with their processor, Motorola 6800 with a different architecture and instruction set compared to 8080.
- In 1976 Intel came up with the upgraded version of 8080 — Intel 8085, with two newly added instructions, three interrupt pins and serial I/O. Clock generator and bus controller circuits were built-in and the power supply part was modified to a single +5 V supply.
- In July 1976 Zilog entered the microprocessor market with its Z80 processor as competitor to Intel. Actually it was designed by an ex-Intel designer, Frederico Faggin and it was an improved version of Intel' 8080 processor, maintaining the original 8080 architecture and instruction set with an 8bit data bus and a 16bit address bus and yes capable of executing all instructions of 8080..It included 80 more new instructions and it brought out the concept of register banking by doubling the register set. Z80 also included two sets of index registers for flexible design.
- Technical advances in the field of semiconductor industry brought a new dimension to the micro- processor market and twentieth century witnessed a fast growth in, processor technology. 16, 32 and 64 bit processors came into the place of conventional 8bit processors. The initial 2 MHz clock is now an old story. Today processors with clock speeds up to 2.4 GHz are available in the market. More and more competitors entered

into the processor market offering high speed, high performance and low cost processors for customer design needs.

- Intel, AMD, Freescale, IBM, TI, Cyrix, Hitachi, NEC, LSI Logic, 'etc. are the key players in the processor market. Intel still leads the market with cutting edge technologies in the processor industry.
- Different instruction set and system architecture are available for the design of a microprocessor. Harvard and Von-Neumann are the two common system architectures for processor design. Processors based on Harvard architecture contains separate buses for program memory and data memory, whereas processors based on Von-Neumann architecture shares a single system bus for program and data memo- ry. We will discuss more about these architectures later, under a separate topic. Reduced Instruction Set Computing (RISC) and Complex Instruction Set Computing (CISC) are the two common Instruction Set Architectures (ISA) available for processor design. We will discuss the same under a separate topic in this section...

General Purpose Processor (GPP) vs. Application-Specific Instruction Set Processor (ASIP)

- A General Purpose Processor or GPP is a processor designed for general computational tasks. The processor running inside your laptop or desktop (Pentium 4/AMD 4thlon, etc.) is a typical ex- ample for general purpose processor. They are produced in large Volumes and targeting the general market. Due to the high volume production, per unit cost for is low compared to ASIC or other specific ICs. A typical general purpose processor contains am Arithmetic and Logic Unit (ALU) and Control Unit (CU). On the other hand, Application, Specific Instruction Set Processors (ASIPs) are processors with architecture and instruction set optimized to specific-domain/application requirements like network processing, automotive, telecom media applications, digital signal processing, control applications, etc. ASIPs fill the architectural spectrum between general purpose processors and Application Specific Integrated Circuits (ASICs) the need for an ASIP arises when the traditional general purpose processor are unable to meet the increasing application needs. Most of the embedded systems are built around application specific instruction set processors. Some microcontrollers (like automotive AVR, USB AVR from Atmel), System on chips, digital signal processors, etc. are examples for application specific instruction processors (ASIPs). ASIPs incorporate a processor and on-chip peripherals, demanded by the application requirement, program and data memory.

b). Microcontroller

- Microcontroller is a highly integrated chip that contains a CPU, scratch pad RAM, special and general purpose register arrays, on chip ROWFLASH memory for program storage,

timer and interrupt control units and dedicated I/O ports. Microcontrollers can be considered as a super set of microprocessors. Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors. Apart from this, they are cheap, cost effective and are readily available in the market. Texas Instrument's TMS 1000 is considered as the world's first microcontroller. We cannot say it as a fully functional microcontroller when we compare it with modern microcontrollers. TI followed Intel's 4004/4040, 4 bit processor design and added some amount of RAM, program storage memory (ROM) and I/O support on a single chip, there by eliminated the requirement of multiple hardware chips for self-functioning. Provision to add custom instructions to the CPU was another innovative feature of TMS 1000. TMS 1000 was released in 1974. In 1977 Intel entered the microcontroller market with a family of controllers coming under one umbrella named MCS-48TM family. The processors came under this family were, 8038HL, 8039HL, 8040AHL, 8048H, 8049H and 8050AH. Intel 8048 is recognized as Intel's first microcontroller and it was the most prominent member in the MCS-48TMt family. It was used in the original IBM PC key- board. The inspiration behind 8048 was Fairchild's F8 microprocessor and Intel's goal o' developing a low cost and small size processor. The design of 8048 adopted a true Harvard architecture where pro- gram and data memory shared the same address bus and is differentiated by the related control signals.

- Eventually Intel came out with its most fruitful design in the 8bit microcontroller domain—the 8051family and its derivatives. It is the most popular and powerful 8bit microcontroller ever built. It was developed in the 1980s and was put under the family MCS-51. Almost 75% of the microcontrollers used in the embedded domain were 8051 family based controllers during the 1980—90s. 8051 processor cores are used in more than 100 devices by more than 20 independent manufacturers like Maxim, Philips, Atmel, etc. under the license from Intel. Due to the low cost, wide availability, memory efficient instruction set, mature development tools and Boolean processing (bit manipulation operation) capability, 8051 family derivative microcontrollers are much used in high-volume consumer electronic devices, entertainment industry and other gadgets where cost-cutting is essential.
- Another important family of microcontrollers used in industrial control and embedded applications is the PIC family micro controllers from Microchip Technologies (It will be discussed in detail in a later section of this book). It is a high performance RISC microcontroller complementing the CISC (complex instruction set computing) features of 8051. The terms RISC and CISCS will are explained in detail in a separate heading.

- Some embedded system applications require only 8bit controller whereas some embedded applications requiring superior performance and computational need demand 16/32bit microcontrollers. Infineon, Free scale, Philips, Atmel, Maxim, Microchip etc. key suppliers of 16bit microcontrollers. Philips tried to extend the 8051 family microcontrollers to use for 16bit applications by developing the Philips XA (extended Architecture) microcontroller series.
- 8bit microcontrollers are commonly used in embedded systems where the processing power is not a big constraint. As mentioned earlier, more than 20 companies are producing different flavors of the 8051 family microcontroller. They try to add more and more functionalities like built in SPI, 12C serial buses, USB controller, ADC, Networking capability, etc. So the competitive market is driving towards a one-stop solution chip in microcontroller domain. High processing speed microcontroller families like ARM11 series are also available m the market, which provides solution to applications requiring hardware acceleration and high processing capability. Free scale, NEC, Zilog, Hitachi, Mitsubishi, Infineon, ST Micro Electronics, National, Texas Instruments, Toshiba, Philips, Microchip, Analog Devices, Daewoo, Intel, Maxim, Sharp, Silicon Laboratories, TDK, Triscend Win bond, Atmel, etc. are the key players in the microcontroller market. Of these at me] has got special significance. They are the manufacturers of a variety of Flash memory based microcontrollers. They also provide In-System Programmability (which will be discussed in detail in a later section of this book) for the controller. The Flash memory technique helps in fast reprogramming of the chip and thereby reduces the product development time. Atmel also provides another special family of microcontroller called AVR (it will be discussed in detail in a later chapter), an 8bit RISC Flash microcontroller, and fast enough to execute powerful instructions in a single clock cycle and provide the latitude you need to optimize power consumption. The instruction set architecture of a microcontroller can be either RISC or CISC. Microcontrollers are designed for either general purpose application requirement (general purpose controller) or domain- specific application requirement (application specific instruction set processor). The Intel 8051 micro- controller is a typical example for a general purpose microcontroller, whereas the automotive AVR microcontroller family from Atmel Corporation is a typical example for ASIP specifically designed for the automotive domain.

Microprocessor vs. Microcontroller The following table summarizes the differences between a microcontroller and microprocessor.

MICROPROCESSOR	MICROCONTROLLER
A silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operation according to a pre defined set of instructions	A micro controller is a highly integrated chip that contains a CPU, scratch pad ram, special and general purpose register arrays on chip ROM/flash memory for program storage, timer and interrupted and control units and dedicated I/O ports.
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips, interrupt controls, etc. for functioning	It is a self contain unit And it does not require external interrupt controller, timer, UART, etc. for its functioning.
Most of the time general purpose in design and operation	Mostly application oriented or domain specific
Does not contain a built in I/o port functionally needs to be implemented with the help external programmable peripheral inter phase chips like 8255.	Most of the processor contains multiple build in I/o ports which can be operated as a single 8 or 16 or 32 bit port or as individual port pins.
Targeted for high end market where performance is important.	Targeted for embedded market where performance is not so critical (at present this demarcation is invalid)
Limited power saving options compare to micro controllers	Includes lot of power savings features

c). Digital Signal Processors

- Digital Signal Processors (DSPs) are powerful special purpose 8/16/32 bit microprocessors designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications applications. Digital signal processors are 2 to 3 times faster than the general purpose microprocessors in signal processing applications. This is because of the architectural difference between the two. DSPs implement algorithms in hardware which speeds up the execution

whereas general purpose processors implement the algorithm in firm- ware and the speed of execution depends primarily on the clock for the processors. In general, DSP can be viewed as a microchip designed for performing high speed computational operations for 'addition', 'subtraction', 'multiplication' and 'division'. A typical digital signal processor incorporates the following key units:

- **Program Memory**: for storing the program required by DSP to process the data
- **Data Memory:** Working memory for storing temporary variables and data/signal to be processed.
- **Computational Engine:** performs the signal processing in accordance with the stored program memory. Computational Engine incorporates many specialized arithmetic units and each of them operates simultaneously to increase the execution speed. It also incorporates multiple hardware shifters for shifting operands and thereby saves execution time.
- **I/O Unit:** Acts as an interface between the outside world and DSP. It is responsible for capturing signals to be processed and delivering the processed signals.

2. Application Specific Integrated Circuits. (ASIC)

- ASICs are a microchip design to perform specific and unique applications.
- Because of using single chip for integrates several functions there by reduces the system development cost.
- Most of the ASICs are proprietary (which having some trade name) products, it is referred as Application Specific Standard Products (ASSP).
- As a single chip ASIC consumes a very small area in the total system. Thereby helps in the design of smaller system with high capabilities or functionalities.
- The developers of such chips may not be interested in revealing the internal detail of it.

3. Programmable logic devices (PLD's)

- A PLD is an electronic component. It used to build digital circuits which are reconfigurable.
- A logic gate has a fixed function but a PLD does not have a defined function at the time of manufacture.
- PLDs offer customers a wide range of logic capacity, features, speed, voltage characteristics.
- PLDs can be reconfigured to perform any number of functions at any time.

- A variety of tools are available for the designers of PLDs which are inexpensive and help to develop, simulate and test the designs.
- PLDs having following two major types.
    - CPLD (Complex Programmable Logic Device): CPLDs offer much smaller amount of logic up to 1000 gates.
    - FPGAs (Field Programmable Gate Arrays): It offers highest amount of performance as well as highest logic density, the most features.

Advantages of PLDs:-

- PLDs offer customer much more flexibility during the design cycle.
- PLDs do not require long lead times for prototypes or production parts because PLDs are already on a distributor's shelf and ready for shipment.
- PLDs can be reprogrammed even after a piece of equipment is shipped to a customer

4. Commercial off-the-shelf components (COTs)

- A Commercial off the Shelf product is one which is used 'as-is'.
- The COTS components itself may be develop around a general purpose or domain specific processor or a ASICs or a PLDs.
- The major advantage of using COTS is that they are readily available in the market, are chip and a developer can cut down his/her development time to a great extent
- The major drawback of using COTS components in embedded design is that the manufacturer of the COTS component may withdraw the product or discontinue the production of the COTS at any time if rapid change in technology occurs.

Advantages of COTS:

- Ready to use
- Easy to integrate
- Reduces development time

Disadvantages of COTS:

- No operational or manufacturing standard (all proprietary)
- Vendor or manufacturer may discontinue production of a particular COTS product

MEMORY

- Memory is an important part of a processor/controller based embedded systems. Some of the processors/controllers contain built in memory and this memory is referred as on-chip memory. Others do not contain any memory inside the chip and require external memory to be connected with the controller/processor to store the control algorithm. It is called off-chip memory. Also some working memory is required for holding data temporarily during certain operations. This section deals with the different types of memory used in embedded system applications.

Program Storage Memory (ROM)

- The program memory or code storage memory of an embedded system stores the program instructions and it can be classified into different types as per the block diagram representation given in Fig. 2.8.
- Classification of Program Memory (ROM): The code memory retains its contents even after the power to it is turned off. It is generally known as non-volatile storage memory. Depending on fabricate, erasing and programming techniques they are classified into the following types.

Masked ROM (MROM):

- Masked ROM is a one-time programmable device. Masked ROM makes use of the hardwired technology for storing data. The device is factory programmed by masking and metallization process at the time of production itself, according to the data provided by the end user. The primary •advantage of this is low, cost for high volume production. They are the least expensive type of solid state memory. Different mechanisms are used for the masking process of the ROM, like

    1. Creation of an enhance mentor depletion mode transistor through channel implant.
    2. By creating the memory all either uses a standard transistor or a high threshold transistor. In the high threshold mode, the supply voltage required to turn ON the transistor is above the normal ROM IC operating Voltage. This ensures that the transistor is always off and the memory cell stores always logic O.

- Masked ROM is a good candidate for storing the embedded firmware for low cost embedded devices. Once the design is proven and the firmware requirements are tested and frozen, the binary data (The firmware cross compiled/assembled to target processor specific machine code) corresponding to it can be given to the MROM fabricator. The limitation with MROM based firmware storage is the inability to modify

the device firmware against firmware upgrades. Since the MROM is permanent in bit storage, it is not possible to alter the bit information.

Programmable Read Only Memory (PROM) / (OTP)

- Unlike Masked ROM Memory, One Time Programmable Memory (OTP) or PROM is not pre-programmed by the manufacturer. The end user is responsible for programming these devices. This memory has nichrome or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses. It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored. Fuses which are not blown/burned represents a logic "I" whereas fuses which are blown/burned represents a logic 0 . The default state is logic "I". OTP is widely used for commercial production of embedded systems whose proto-typed versions are proven and the code is finalized. It is a low cost solution for commercial production. OTPs cannot be reprogrammed.

Erasable Programmable Read Only Memory (EPROM)

- OTPs are not useful and worth for development purpose. During the development phase the code is subject to continuous changes and using an OTP each time to load the code is not economical. Erasable Programmable Read Only Memory (EPROM) gives the flexibility to re-program the same chip. EPROM stores the bit information by charging the floating gate of an FET. Bit information is stored by using an EPROM programmer, which applies high voltage to charge the floating gate. EPROM contains a quartz crystal window for erasing the stored information. If the window is exposed to ultraviolet rays for a fixed duration, the entire memory will be erased. Even though the EPROM chip is flexible in terms of re-programmability, it needs to be taken out of the circuit board and put in a UV eraser device for 20 to 30 minutes. So it is a tedious and time-consuming process.

Electrically Erasable Programmable Read Only Memory (EEPROM)

- As the name indicates, the information contained in the EEPROM memory can be altered by using electrical signals at the register/Byte level. They can be erased and reprogrammed in-circuit: These chips include a chip erase mode and in this mode they can be erased in a few milliseconds at provide greater flexibility for system design. The only limitation is their capacity is limit when compared with the standard ROM (A few kilobytes).

FLASH

- FLASH is the Latest ROM technology and is the most popular ROM technology used in today's embedded designs. FLASH memory is @variation of EEPROM technology. It

combines the re-programmability of EEPROM and the high capacity of standard ROMs. FLASH memory is organized as sectors (blocks) or pages. FLASH memory stores information in an array of floating gate MOS- FET transistors. The erasing of memory can be done at sector level or page level without affecting the other sectors or pages. Each sector/page should be erased before re-programming. The typical erasable capacity of FLASH is 1000 cycle 7C512 from WINBOND is an example of 64KB FLASH memory.

NVRAM

- Non-voltaic KAM is a random access memory with battery backup. It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply The memory and battery are packed together in a single package. The life span of NVRAM is expected to be around 10 years. DS1644 from Maxim/Dallas is an example of 32KB NVRAM.

Read-Write Memory/Random Access Memory (RAM)

- RAM is the data memory or working memory of the controller/processor. Controller/processor can read fpm it and write to it. RAM is volatile, meaning when the power is turned off, all the contents are destroyed. RAM is a direct access memory, meaning we can access the desired memory location directly without the need for traversing through the entire memory locations to reach the desired memory position (i.e. random access of memory location). This is in contrast to the Sequential Access Memory (SAM), where the desired memory location is accessed by either traversing through the entire memory or through a 'seek' method. Magnetic tapes, CD ROMs, etc. are examples of sequential access memories. RAM generally falls into three categories: Static RAM (SRAM), dynamic RAM (DRAM) and non-volatile RAM (NVRAM) (Fig. 2.9).

Static RAM (SRAM)

- Static RAM stores data in the form of voltage. They are made up of flip-flops. Static RAM is the fastest form of RAM available. In typical implementation, an SRAM cell (bit) is realized using six transistors (or 6 MOSFETs). Four of the transistors are used for building the
- Classification of Working Memory (RIM):
- Latch (flip-flop) part of the memory cell and two for controlling the, access. SRAM is fast in operation due to its resistive networking and switching capabilities. To its simplest representation an SRAM cell can be visualized as shown in Fig. 2.10:

SRAM cell implementation:

- This implementation in its simpler form can be visualized as two-cross coupled inverters with read/ write control through transistors. The four transistors in the middle form the cross-coupled inverters. This can be visualized as shown in Fig. 2.11 From the SRAM implementation diagram, it is clear that access to the memory cell is controlled by the line Word Line, which controls the access transistors (MOSFETs) Q5 and Q6. The access transistors control the connection to bit lines B & B\. In order to write a value to the memory cell, apply the desired value to the bit control lines (For writing I, make B — I and B\ =O; for writing O, make B = O and B\ —I) and assert the Word Line (Make Word line high). This operation latches the bit written in the flip-flop. For reading the content of the memory cell, assert both B and bit lines to me and set the Word line to me. The major limitations of SRAM are low capacity and high cost. Since a minimum of six transistors are required to build a single memory cell, imagine how many memory cells we can fabricate on a silicon wafer.

Dynamic RAM (DRAM)

- Dynamic RAM stores data in the form of charge. They are made up of MOS transistor gates. The advantages of DRAM are its high density and low cost compared to SRAM. The disadvantage is that since the information is stored as charge it gets leaked off with time and to prevent this they need to be refreshed periodically. Special circuits called DRAM controllers are used for the refreshing operation. The refresh operation is done periodically in millisecond's interval. Figure 2.12 illustrates the typical implementation of a DRAM cell.
- The MOSFET acts as the gate for the incoming and outgoing data whereas the capacitor acts as the bit storage unite Table given below summarizes the relative merits and demerits of SRAM and DRAM technology.

NVRAM

- On-volatile RAM is a random access memory with battery backup. It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply. The memory and battery are packed together in a single package. NVRAM is used for the non-volatile storage of results of operations or for setting up of flags, etc. The life span Of NV RAM is expected to be around 10 years. DS1744 from Maxim/DaIIas is an example for 32KB NVRAM.

Memory According to the Type of Interface

- The interface (connection) of memory with the processor/controller can be of various types. It may be a parallel interface [The parallel data lines (DO-D7) for an 8 bit

processor/controller will be connected to DO-D7 of the memory] or the interface may be a serial interface like 12C (Pronounced as I Square C. It is a 2 line serial interface) or it may be an SPI (Serial peripheral interface, 2+n line interface where n stands for the total number of SPI bus devices in the system). It can also be of a single wire interconnection (like Dallas I-Wire interface). Serial interface is commonly used for data storage memory like EEPROM. The memory density of a serial memory is usually expressed in terms of kilobits, whereas that of a parallel interface memory' is expressed in terms of kilobytes. Atmel Corporations AT24C512 is an example for serial memory with capacity 512 kilobits and 2-wire interface. Please refer to the section 'Communication Interface' for more details on 12C, SPI and I-Wire Bus.

Memory Shadowing

- Generally the execution of a program or a configuration from Read Only Memory (ROM) is very slow (120 to 200 ns) compared to the execution from a random access memory (40 to70 ns). From the timing parameters it is obvious that RAM access is about three times as fast as ROM access. Shadowing of me only is a technique adopted to solve the execution speed problem in processor-based systems. In computer systems and video systems there will be a configuration holding ROM called Basic Input Output Configuration ROM or simply BIOS. In personal computer systems BIOS stores the hardware configuration information like the address assigned for various serial ports and other non-plug 'n' play devices, etc. Usually it is read and the system is configured according to it during system boot up and it is time consuming. Now the manufactures included a RAM behind the logical layer of BIOS at its same address as a shadow to the BIOS and the first step that happens during the boot up is copying the BIOS to the shadowed RAM and write protecting the RAM then disabling the BIOS reading. You may be thinking that what a stupid idea it is and why both RAM and ROM are needed for holding the same data. The answer is: RAM is volatile and it cannot hold the configuration data which is copied from the BIOS when the power supply is switched off. Only a ROM can hold it permanently. But for high system performance it should be accessed from RAM instead of accessing from a ROM.

Memory Selection for Embedded Systems

- Embedded systems require a program memory for holding the control algorithm (For a super-loop based design) or embedded OS and the applications designed to run on top of it (for OS based designs), data memory for holding variables and temporary data during task execution, and memory for holding non- volatile data (like configuration data, look up table etc) which are modifiable by the application (Unlike program memory which is non-volatile as well unalterable by the end user). The memory requirement for an embedded system in terms of RAM and ROM

(EEPROM/FLASH/NVRAM) is solely dependent on the type of the embedded system and the applications for which it is designed. There is no hard and fast rule for calculating the memory requirements. Lot of factors need to be considered when selecting the type and size of memory for embedded system. For example, if the embedded system is designed using SOC or a microcontroller with on-chip RAM and ROM (FLASH/EEPROM), depending on the application need the on-chip memory may be sufficient for designing the total system. As a rule of thumb, identify your system requirement and based on the type of processor (SOC or microcontroller with on- chip memory) used for the design, take a decision on whether the on-chip memory is sufficient or external memory is required. Let's consider a simple electronic toy design as an example. As the complexity of requirements are less and data memory requirement are minimal, we can think of a microcontroller with a few bytes Of internal RAM, a few bytes or kilobytes (depending on the number of tasks and the complexity of tasks) of FLASH memory and a few bytes of EEPROM (if required) for designing the system. Hence there is no need for external memory at all. A PIC microcontroller device which satisfies the I/O and memory requirements can be used in this case. If the embedded design is based on an RTOS, the RTOS requires certain amount of RAM for its execution and ROM for storing the RTOS image (Image is the common name give) for the binary data generated by the compilation of all RTOS source files). Normally the binary code for RTOS kernel containing all the services is stored in a non-volatile memory (Like FLASH) as either compressed or non-compressed data. During boot-up of the device, the RTOS files are copied from the program storage memory, decompressed if required and then loaded to the RAM for execution. The supplier of the RTOS usually gives a rough estimate on the run time RAM requirements and program memory requirements for the RTOS. On top of this add the RAM requirements for executing user tasks and ROM for storing user applications. On a safer side, always add a buffer value to the total estimated RAM and ROM size requirements. A smart phone device with Windows mobile operating system is a typical example for embedded device with ()S. Say 64MB RAM and 128MB ROM are the minimum requirements for running the Windows mobile device, indeed you need extra RAM and ROM for running user applications. So while building the system, count the memory for that also and arrive at a value which is always at the safer side, so that you won't end up in a situation where you don't have sufficient memory to install and run user applications. There are two parameters for representing a memory. The first one is the size of the memory ship (Memory density expressed in terms of number of memory bytes per chip). There is no option to get a memory chip with the exact required number of bytes. Memory chips come in standard sizes like 512bytes, 1024bYtes (1 kilobyte), 2048bytes (2 kilobytes), 4Kb,t 8Kb, 16Kb, 32Kb, 256Kb, 512Kb, 1024Kb (I megabytes), etc. Suppose your embedded application requires only 750 bytes

of RAM, you don't have the option of getting a memory chip with size 750 bytes the only option left with is to choose the memory chip with a size closer to the size needed. Here 1024 bytes is the least possible option. We cannot go for 512 bytes, because the minimum requirement 750 bytes. While you select a memory size, always keep in mind the address range supported by your processor. For example, for a processor/ controller with 16 bit address bus, the maximum number of memory locations that can be addressed is 216 = 65536 bytes = 64Kb. Hence it is meaningless to select a 128Kb memory chip for a processor with 16bit wide address bus. Also, the entire memory range supported by the processor/controller may not be available to the memory chip alone. It may be shared between I/O, other ICs and memory. Suppose the address bus is 16bit wide and only the lower 32Kb address range is assigned to the memory chip, the memory size maximum required is 32Kb 6nIy. It is not worth to use a memory chip with size 64Kb in such a situation. The second parameter that needs to be considered in selecting a memory is the word size of the memory. The word Size refers to the number of memory bits that can be read/write together at a time. 4, 8, 12, 16, 24, 32, etc. are the word sizes supported by memory chips. Ensure that the word size supported by the memory chip matches with the data bus width of the processor/controller. FLASH memory is the popular choice for ROM (program storage memory) in embedded applications. It is a powerful and cost-effective solid-state storage technology for mobile electronics devices and other consumer applications. FLASH memory comes in two major variants, namely, NAND and NOR FLASH. NAND FLASH is a high-density low cost non-volatile storage memory: On the other hand, NOR FLASH is less dense and slightly expensive. But it supports the Execute in Place (XIP) technique for program execution. The XIP technology allows the execution of code memory from ROM itself without the need for copying it to the RAM as in the case of conventional execution method. It is a good practice to use a combination of NOR and NAND memory for storage memory requirements, where NAND can be used for storing the program code and or data like the data captured in a camera device. NAND FLASH doesn't support XIP and if NAND FLASH is used for storing program code, a DRAM can be used for copying and executing the program code. NOR FLASH supports XIP and it can be used as the memory for boot loader or for even storing the complete program code. The EEPROM data storage memory is available as either serial interface or parallel interface chip. If the processor/controller of the device supports serial interface and the amount of data to write and read to and from the device is less, it is better to have a serial EEPROM chip. The serial EEPROM saves the address space of the total system. The memory capacity of the serial EEPROM is usually expressed in bits or kilobits. 512 bits, 1Kbits, 2Kbits, 4Kbits, etc. are examples for serial EEPROM memory representation. For embedded systems with low power requirements like portable devices, choose low power memory devices.

Certain embedded devices may be targeted for operating at extreme environmental conditions like high temperature, high humid area, etc. Select an industrial grade memory chip in place of the commercial grade chip for such devices.

SENSORS AND ACTUATORS

- At the very beginning of this chapter it is already mentioned that an embedded system is in constant interaction with the Real world and the controlling/monitoring functions executed by the embedded system is achieved in accordance with the changes happening to the Real world. The changes in sys- tem environment or variables are detected by the sensors connected to the input port of the embedded system. If the embedded system is designed for any controlling purpose the system will produce some changes in the controlling variable to bring the controlled variable to the desired value. It is achieved through an actuator connected to the output port of the embedded system. If the embedded system is designed for monitoring purpose only, then there is no need for including an actuator in the system. For example, take the case of an ECG machine. It is design to monitor the heart beat status of a patient and it cannot impose a control over the patient's heart beat and its order. The sensors used here are the different electrode sets connected to the body of the patient. The variations are captured and presented to the user (may be a doctor) through a visual display or some printed chart.

SENSORS

- Sensors are also called as detectors.
- The changes in the system environment or variables are detected by the sensors connected to the input port of the embedded system.
- It is a transducer that converts energy from one type to another type for any particular purpose.
- Example- ECG machine it is designed to monitor the heartbeat status of a patient and it cannot impose a control over the patient’s heart beat and its order. The sensors are used here are the different electrode sets connected to the body of the patient.
- The variations are captured and presented to the user through a visual display or some printed chart.

ACTUATORS

- Actuator is a form of transducer device which converts signals to corresponding physical action.

- Actuator acts as an output device.
- If the embedded system is designed for any controlling purpose the system will produce some changes in the controlling variable to bring the controlled variable to the desired value. This is achieved through an actuator connected to the output port of the embedded system.
- If the E.S is designed for monitoring purpose only then there is no need for including an actuator in the system.
- Types of Actuators
    - Multi-Turn Actuator
    - Part-Turn Actuator Linear Actuator
    - Multi-turn Actuator-
- It is an actuator which transmits to the valve torque for at least one full revolution. It is capable of withstanding thrust.
- It is required for the automation of multi-turn valves.
- One of the main types of this is the gate valve

Part-turn actuators –

- It is an actuator which transmits a torque to the valve for less than one full revolution. It is not capable of withstanding thrust.
- The major representatives of this type are butterfly valves and ball valves.

Linear Actuator –

- The major representative of this type is the control valves.
- Just like the plug in the bathtub is pressed into the drain the plug is pressed into the plug seat by a stroke.

The I/O Subsystem

- The I/O subsystem of the embedded system facilitates the interaction of the embedded system with the external world. As mentioned earlier the interaction happens through the sensors and actuators connected to the input and output ports respectively of the embedded system. The sensors may not be directly interfaced to the input ports; instead they may be interfaced through signal conditioning and translating systems like ADC, up to couplers, etc. This section illustrates some of the sensors and actuators used in embedded systems and the I/O systems to facilitate the interaction of embedded systems with external world.

COMMUNICATION INTERFACES

These are the devices through with the E.S can interact with various subsystems and the external world. For embedded product communication interface can be viewed in two different perspectives:

1. Device/board level communication interface (Onboard communication Interface)
2. Product level communication interface (External communication interface)

ONBOARD COMMUNICATION INTERFACE

- The communication channel which interconnects the various components with in an embedded product is referred as device/broad level communication interface.
- Examples – Serial interfaces like I2C, I-Wire, and parallel bus interface.

Inter Integrated Circuit Bus (I2C Bus) –

- It is a synchronous bi-directional half duplex two–wire serial bus which provides communication link between integrated circuits.
- It was designed by Philips Semiconductors in1980s.
- It was developed to provide an easy way of connection between a microprocessor /microcontroller system and peripheral chips in television sets.
- It comprises of two bus lines i.e. Serial Clock-SCL and Serial Data-SDA.
- SCL line is responsible for generating synchronization clock pulses.
- SDA is responsible for transmitting the serial data across devices.
- I2C bus is a shared bus system to which many number ofI2C devices can be connected.
- Devices connected to the I2C runs can act as either “Master” device or “Slave” device.
- The Master device is responsible for controlling the communication by initiating or terminating data transfer, sending data and generating necessary synchronization clock pulses.
- The Slave devices wait for the commands from the Master and respond upon receiving the commands.
- Master and Slave devices can act as either transmitter or receiver.
- Regardless whether a master is acting as transmitter or receiver the synchronization clock signal is generated by Master device only.
- I2C supports multi masters on the same bus.

EXTERNAL COMMUNICATION INTERFACE

- These are the E.S which may be a part of large distributed system and they require interaction and data transfer between various devices and sub modules.
- The product level communication interface is responsible for data transfer between the E.S and other devices or modules.
- The external communication interface can be either a wired media or a wireless media and it can be a serial or a parallel interface.
- Examples – Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves etc.

Infrared –

- Infrared is a serial, half duplex, line of sight based wireless technology for data communication between devices. The remote control of TV, AC works on the infrared data communication principle.IR uses infrared waves of the electromagnetic spectrum for transmitting the data. It supports point-point and point-to-multipoint communication. The typical communication range for IR lies in the range of 10 cm to1m.The range can be increased by increasing the transmitting power of the IR device.IR supports data rates ranging from 9600bits/sec to 16Mbps.

UNIT-2

EMBEDDED FIRMWARE

INTRODUCTION:

- The control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system.
- It is an un-avoidable part of an embedded system.
- The embedded firmware can be developed in various methods like
    - Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (The IDE will contain an editor, compiler, linker, debugger, simulator etc. IDEs are different for different family of processors/controllers.
    - Write the program in Assembly Language using the Instructions Supported by your application's target processor/controller

EMBEDDED FIRMWARE DESIGN & DEVELOPMENT:

- The embedded firmware is responsible for controlling the various peripherals of the embedded hardware and generating response in accordance with the functional requirements of the product.
- The embedded firmware is the master brain of the embedded system.
- The embedded firmware imparts intelligence to an embedded system.
- It is a onetime process and it can happen at any stage.
- The product starts functioning properly once the intelligence imparted to the product by embedding the firmware in the hardware.
- The product will continue serving the assigned task till hardware breakdown occurs or a corruption in embedded firmware.
- In case of hardware breakdown, the damaged component may need to be replaced and for firmware corruptions the firmware should be re-loaded, to bring back the embedded product to the normal functioning.
- The embedded firmware is usually stored in a permanent memory (ROM) and it is non alterable by end users.

- Designing Embedded firmware requires understanding of the particular embedded product hardware, like various component interfacing, memory map details, I/O port details, configuration and register details of various hardware chips used and some programming language (either low level Assembly Language or High level language like C/C++ or a combination of the two)
- The embedded firmware development process starts with the conversion of the firmware requirements into a program model using various modeling tools.
- The firmware design approaches for embedded product is purely dependent on the complexity of the functions to be performed and speed of operation required.
- There exist two basic approaches for the design and implementation of embedded firmware, namely;
    - The Super loop based approach
    - The Embedded Operating System based approach
- The decision on which approach needs to be adopted for firmware development is purely dependent on the complexity and system requirements

A). Embedded firmware Design Approaches – The Super loop:

- The Super loop based firmware development approach is Suitable for applications that are not time critical and where the response time is not so important (Embedded systems where missing deadlines are acceptable).
- It is very similar to a conventional procedural programming where the code is executed task by task
- The tasks are executed in a never ending loop.
- The task listed on top on the program code is executed first and the tasks just below the top are executed after completing the first task
- A typical super loop implementation will look like:
    - Configure the common parameters and perform initialization for various hardware components memory, registers etc.
    - Start the first task and execute it
    - Execute the second task
    - Execute the next task
    - Execute the last defined task

- Jump back to the first task and follow the same flow.

- The 'C' program code for the super loop is given below

```
void main ()
 {
Configurations ();
Initializations ();
while (1)
 {
Task 1 ();
Task 2 ();
:
:
Task n ();
 }
}
```

Pros:

- Doesn't require an Operating System for task scheduling and monitoring and free from OS related overheads
- Simple and straight forward design
- Reduced memory footprint

Cons:

- Non Real time in execution behavior (As the number of tasks increases the frequency at which a task gets CPU time for execution also increases)
- Any issues in any task execution may affect the functioning of the product (This can be effectively tackled by using Watch Dog Timers for task execution monitoring)

Enhancements:

- Combine Super loop based technique with interrupts
- Execute the tasks (like keyboard handling) which require Real time attention as Interrupt Service routines.

***B).* Embedded firmware Design Approaches – Embedded OS based Approach:**

The embedded device contains an Embedded Operating System which can be one of:

- A Real Time Operating System (RTOS)
- A Customized General Purpose Operating System (GPOS)
- The Embedded OS is responsible for scheduling the execution of user tasks and the allocation of system resources among multiple tasks
- It Involves lot of OS related overheads apart from managing and executing user defined tasks
- Microsoft® Windows XP Embedded is an example of GPOS for embedded devices
- Point of Sale (PoS) terminals, Gaming Stations, Tablet PCs etc are examples of embedded devices running on embedded GPOSs
- 'Windows CE', 'Windows Mobile', 'QNX', 'VxWorks', 'ThreadX', 'MicroC/OS-II', 'Embedded Linux', 'Symbian' etc are examples of RTOSs employed in Embedded Product development
- Mobile Phones, PDAs, Flight Control Systems etc are examples of embedded devices that runs on RTOSs

EMBEDDED FIRMWARE DEVELOPMENT LANGUAGES/OPTIONS

- Assembly Language
- High Level Language
    - Subset of C (Embedded C)
    - Subset of C++ (Embedded C++)
    - Any other high level language with supported Cross-compiler
- Mix of Assembly & High level Language
    - Mixing High Level Language (Like C) with Assembly Code
    - Mixing Assembly code with High Level Language (Like C)
    - Inline Assembly

(A). Assembly Language

- 'Assembly Language' is the human readable notation of 'machine language'
- 'Machine language' is a processor understandable language

- Machine language is a binary representation and it consists of 1s and 0s
- Assembly language and machine languages are processor/controller dependent
- An Assembly language program written for one processor/controller family will not work with others
- Assembly language programming is the process of writing processor specific machine code in mnemonic form, converting the mnemonics into actual processor instructions (machine language) and associated data using an assembler
- The general format of an assembly language instruction is an Opcode followed by Operands
- The Opcode tells the processor/controller what to do and the Operands provide the data and information required to perform the action specified by the opcode
- It is not necessary that all opcode should have Operands following them. Some of the Opcode implicitly contains the operand and in such situation no operand is required. The operand may be a single operand, dual operand or more

The 8051 Assembly Instruction

MOV A, #30

Moves decimal value 30 to the 8051 Accumulator register. Here MOV A is the Opcode and 30 is the operand (single operand). The same instruction when written in machine language will look like

01110100 00011110

The first 8 bit binary value 01110100 represents the opcode MOV A and the second 8 bit binary value 00011110 represents the operand 30.

- Assembly language instructions are written one per line
- A machine code program consists of a sequence of assembly language instructions, where each statement contains a mnemonic (Opcode + Operand)
- Each line of an assembly language program is split into four fields as:

LABEL OPCODE OPERAND COMMENTS

- LABEL is an optional field. A 'LABEL' is an identifier used extensively in programs to reduce the reliance on programmers for remembering where data or code is located. LABEL is commonly used for representing
    - A memory location, address of a program, sub-routine, code portion etc.

- The maximum length of a label differs between assemblers.ϖ Assemblers insist strict formats for labeling. Labels are always suffixed by a colon and begin with a valid character. Labels can contain number from 0 to 9 and special character _ (underscore).

```
#################################################################

; SUBROUTINE FOR GENERATING DELAY

; DELAY PARAMETR PASSED THROUGH REGISTER R1

; RETURN VALUE NONE, REGISTERS USED: R0, R1

################################################################ #####

DELAY:          MOV R0, #255          ; Load Register R0 with 255

DJNZ R1, DELAY          ; Decrement R1 and loop till          R1= 0

RET                     ; Return to calling program
```

- The symbol; represents the start of a comment. Assembler ignores the♣ text in a line after the ; symbol while assembling the program
- DELAY is a label for representing the start address of the memory♣ location where the piece of code is located in code memory
- The above piece of code can be executed by giving the label DELAY as♣ part of the instruction. E.g. LCALL DELAY; LMP DELAY

Assembly Language – Source File to Hex File Translation:

- The Assembly language program written in assembly code is saved as .asm (Assembly file) file or a .src (source) file or a format supported by the assembler
- Similar to 'C' and other high level language programming, it is possible to have multiple source files called modules in assembly language programming. Each module is represented by a '.asm' or '.src' file or the assembler supported file format similar to the '.c' files in C programming
- The software utility called 'Assembler' performs the translation of assembly code to machine code
- The assemblers for different family of target machines are different. A51 Macro Assembler from Keil software is a popular assembler for the 8051 family micro controller

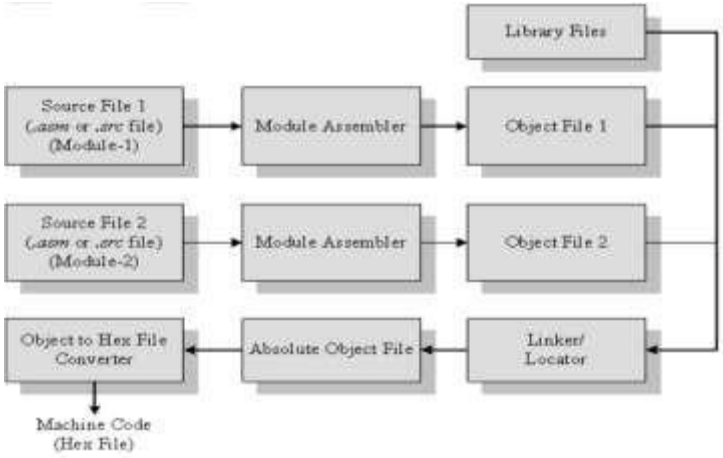


Figure 5: Assembly Language to machine language conversion process

- Each source file can be assembled separately to examine the syntax errors and incorrect assembly instructions
- Assembling of each source file generates a corresponding object file. The object file does not contain the absolute address of where the generated code needs to be placed (a re-locatable code) on the program memory
- The software program called linker/locater is responsible for assigning absolute address to object files during the linking process
- The Absolute object file created from the object files corresponding to different source code modules contain information about the address where each instruction needs to be placed in code memory
- A software utility called 'Object to Hex file converter' translates the absolute object file to corresponding hex file (binary file)

Advantages:

1. ***Efficient Code Memory & Data Memory Usage (Memory Optimization):***
   - The developer is well aware of the target processor architecture and¬ memory organization, so optimized code can be written for performing operations.
   - This leads to less utilization of code memory and efficient utilization¬ of data memory.
2. ***High Performance:***
   - Optimized code not only improves the code memory usage but also¬ improves the total system performance.

- Through effective assembly coding, optimum performance can be¬ achieved for target processor.

3. ***Low level Hardware Access:***

   - Most of the code for low level programming like accessing external¬ device specific registers from OS kernel ,device drivers, and low level interrupt routines, etc are making use of direct assembly coding.

4. ***Code Reverse Engineering:***

   - It is the process of understanding the technology behind a product by¬ extracting the information from the finished product.
   - It can easily be converted into assembly code using a dis-assembler¬ program for the target machine.

Drawbacks:

1. ***High Development time:***

   - The developer takes lot of time to study about architecture, memory¬ organization, addressing modes and instruction set of target processor/controller.
   - More lines of assembly code are required for performing a simple¬ action.

2. ***Developer dependency:***

   - There is no common written rule for developing assembly language¬ based applications.

3. ***Non portable:***

   - Target applications written in assembly instructions are valid only for¬ that particular family of processors and cannot be re-used for other target processors/controllers.
   - If the target processor/controller changes, a complete re-writing of the¬ application using assembly language for new target processor/controller is required.

(B). High Level Language

- The embedded firmware is written in any high level language like C, C++
- A software utility called 'cross-compiler' converts the high level language to target processor specific machine code

- The cross-compilation of each module generates a corresponding object file. The object file does not contain the absolute address of where the generated code needs to be placed (a re-locatable code) on the program memory
- The software program called linker/locater is responsible for assigning absolute address to object files during the linking process
- The Absolute object file created from the object files corresponding to different source code modules contain information about the address where each instruction needs to be placed in code memory
- A software utility called 'Object to Hex file converter' translates the absolute object file to corresponding hex file (binary file)

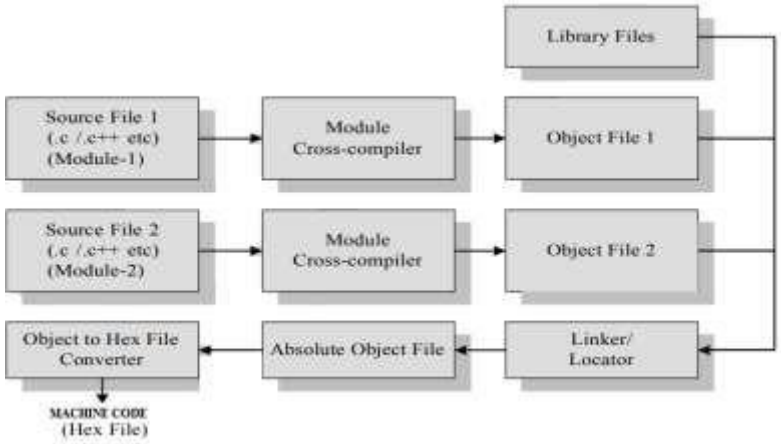


Figure 6: High level language to machine language conversion process

Advantages:

- ***Reduced Development time:*** Developer requires less or little knowledge on internal hardware details and architecture of the target processor/Controller.
- ***Developer independency:*** The syntax used by most of the high level languages are universal and a program written high level can easily understand by a second person knowing the syntax of the language
- ***Portability:*** An Application written in high level language for particular target processor /controller can be easily be converted to another target processor/controller specific application with little or less effort

Drawbacks:

- The cross compilers may not be efficient in generating the optimized target processor specific instructions.

- Target images created by such compilers may be messy and no optimized in terms of performance as well as code size.
- The investment required for high level language based development tools (IDE) is high compared to Assembly Language based firmware development tools.

EMBEDDED SYSTEM DEVELOPMENT ENVIRONMENT

- The most important characteristic of E.S is the cross-platform development technique.
- The primary components in the development environment are the host system, the target system and many connectivity solutions between the host and the target E.S.
- The development tools offered by the host system are the cross complier, linker and source-level debugger.
- The target embedded system offers a dynamic loader, link loader, a monitor and a debug agent.
- Set of connections are required between the source computer and the target system.
- These connections can be used for transmitting debugger information between the host debugger and the target debug agent.

IDE:

- In E.S IDE stands for an integrated environment for developing and debugging the target processor specific embedded firmware.
- An IDE is also known as integrated design environment or integrated debugging environment.
- IDE is a software package which bundles a “Text Editor”, “Cross-compiler”, “Linker” and a “Debugger”.
- IDE is a software application that provides facilities to computer programmers for software development.
- IDEs can either command line based or GUI based.

IDE consists of:

1. Text Editor or Source code editor
2. A compiler and an interpreter
3. Build automation tools
4. Debugger

5. Simulators
6. Emulators and logic analyzer

- The example of IDE is Turbo C/C++ which provides platform on windows for development of application programs with command line interface.
- The other category of IDE is known as Visual IDE which provides the platform for visual development environment, ex- Microsoft Visual C++.
- IDEs used in embedded firmware are slightly different from the generic IDE used for high level language based development for desktop applications.
- In Embedded applications the IDE is either supplied by the target processor/controller manufacturer or by third party vendors or as Open source.
- An Integrated Development Environment (IDE) is software that assists programmers in developing software
- IDEs normally consist of a source code editor, a compiler, a linker/locater and usually a debugger.
- Sometimes, an IDE is devoted to one specific programming language or one (family of) specific processor or hardware
- But more often the IDEs support multiple languages, processors, etc. Some commonly used IDEs for embedded systems are the GNU compiler collection (GCC), Eclipse, Delphi,

EDITOR:

- A source code editor is a text editor program designed specifically for editing source code to control embedded systems. It may be a standalone application or it may be built into an integrated development environment (e.g. IDE).Source code editors may have features specifically designed to simplify and speed up input of source code, such as syntax highlighting and auto complete functionality. These features ease the development of code

COMPILER:

- A compiler is a computer program that translates the source code into computer language (object code).Commonly the output has a form suitable for processing by other programs (e.g., a linker), but it may be a human readable text file. A compiler translates source code from a high level language to a lower level language (e.g., assembly language or machine language). The most common reason for wanting to translate source code is to create a program that can be executed on a computer or on an embedded system. The compiler is called a cross compiler if the source code is

compiled to run on a platform other than the one on which the cross compiler is run. For embedded systems the compiler always runs on another platform, so a cross compiler is needed.

LINKER:

- A linker or link editor is a program that takes one or more objects generated by compilers and assembles them into a single executable program or a library that can later be linked to in it. All of the object files resulting from compiling must be combined in a special way before the program locator will produce an output file that contains a binary image that can be loaded into the target ROM. A commonly used linker/ locater for embedded systems ISLD (GNU).

TYPES OF FILES GENERATED ON CROSS COMPILATION

• The various files generated during cross compilation process are:

1. List File
2. Hex File (.hex)
3. Pre-processor Output file
4. Map File (File extension linker dependent)
5. Object File (.obj)

List Files

- At the time of cross compilation the .lst file is generated by the system which contains the information code generated from the source file.

Hex file

- The Hex file is an ASCII text file with lines of text that follow the Intel Hex file format.
- Intel Hex files are often used to transfer the program and data that would be stored in a Rom or EPROM.

Map Files

- These files are used to keep the information of linking and locating process.
- Map files use extensions .H,.HH,.HM
- Object files
- It is the lowest level file format for any platform.
- Cross compiling each source module converts the various embedded instructions and other directives present in the module to an object (.OBJ) file.

- The object file is specially formatted file with data records for symbolic information, object code, debugging information etc.

Disassembler

- Disassembler is a utility program which converts machine codes into target processor specific Assembly instructions.
- The process of converting machine codes into Assembly code is known as "Disassembling".
- Disassembling is complementary to assembling or cross assembling.
- The output of the disassembler is often formatted for human-read ability rather than suitability for input to an assembler.
- Assemble language source code generally permits the use of constant and programmer comments.
- These are usually removed from the assembled machine code by the assembler.
- A disassembler operating on the machine code would produce disammembly lacking these constants and comments; the disassembled output becomes more difficult for a human to interpret than the original annotated sources code.
- The interactive disassembler allows the human user to make up mnemonic symbols for values of code in an interactive session.
- A disassembler may be stand-alone or interactive.
- A stand-alone disassembler when executed generates an assembly language file which can be examined.
- Interactive shows the effect of any change the user makes immediately.

Decompiler

- Decompiler is the utility program for translating machine codes into corresponding high level language instructions.
- A decompiler is the name given to a computer program that performs the reverse operation to that of a compiler.
- The tool that accomplishes this task is called a decompiler.
- The decompiler does not reconstruct the original source code and its output is far less intelligible to a human than original source code

- Decompilation was first used in 1960s to facilitate the migration of a program form one platform to another.
- Decompile means to convert executable program code into some form of higher-level programming language so that it can be read by a human.
- Decompilation is a type of reverse engineering that does the opposite of what a compiler does.
- The are many reasons for decompilation such as understanding a program, recovering the source code for purposes of achieving or updating, finding viruses, debugging programs and translating obsolete code.

SIMULATOR

- Simulator and emulators are two important tools used in embedded system development.
- Simulator is a software tool used for simulating the various conditions for checking the functionality of the application firmware.
- It is a host-based program that simulates the functionality and instruction set of the target processor.
- The features of Simulator based debugging are:
    - Purely software based
    - Doesn't require a real target system
    - Very primitive
    - Lack of Real –time behavior.
- Simulation is used whenever trying things in the physical world would be inconvenient, expensive.
- Simulation allows experimenter to try things with more control over parameters and better insight into the results.
- Simulating an embedded computer system can be broken down into five main parts:
- The computer board itself, the piece of hardware containing one or more processor, executing the embedded software.
- The software running on the computer board. This includes the user applications, also the boot ROM or BIOS, hardware drivers, OS and various libraries.

- The communication network or networks that the board is connected to hand over which the software communicates with software on other computers.
- The environment in which the computer operates and that it measures using sensors and affects using actuators.

Advantages of Simulator Based Debugging

- The simulator based debugging techniques are simple and straightforward.
- Other advantages are: No need for original Target Board
    - It is purely software oriented.
    - IDEs software support simulates the CPU of the target board.
    - User's only needs to know about the memory map of various devices within the target board and the firmware should be written on the bases of it.
    - Since real hardware is not required the firmware development can start well in advance immediately after the device interface and memory maps are finalized.
    - This saves development time.

Simulate I/O peripherals

- It provides the option to simulate various I/O peripherals.
- Using simulator's I/O support we can edit the values for I/O registers ad can be used as the input/output value in the firmware execution.
- Hence it eliminates the need for connection I/O devices for debugging the firmware.

Simulates Abnormal Conditions

- With simulator's simulation support we can input any desired value for any parameter during debugging the firmware and can observe the control flow of firmware.
- It helps the developer in simulating abnormal operational environment for firmware and helps the firmware developer to study the behavior of the firmware under abnormal input conditions.

Limitations OF Simulator Based Debugging

Deviation from Real Behavior

- Simulation-based firmware debugging is always carried out in a development environment where the developer may not be able to debug the firmware under all possible combinations of input.

- Under certain operating conditions we may get some particular result and it need not be the same when the firmware runs in a production environment.

Lack of Real timeliness

- The major limitation is that it is not real-time in behavior.
- The debugging is developer driven and it is no way capable of creating a real time behavior.
- Moreover in a real application the I/O condition may be varying or unpredictable.

EMULATOR

- It is a piece of hardware that exactly behaves like the real microcontroller chip with all its integrated functionality.
- It is the most powerful debugging of all.
- A microcontroller's functions are emulated in real-time and non-intrusively.
- All emulators contain 3 essential function:
    - The emulator control logic, including emulation memory
    - The actual emulation device
    - A pin adapter that gives the emulator's target connector the same "package" and pin out as the microcontroller to be emulated.
- An emulator is a piece of hardware that looks like a processor, has memory like a processor, and executes instructions like a processor but it is not a processor.
- The advantage is that we can probe points of the circuit that are not accessible inside a chip.
- It is a combination of hardware and software.

DEBUGGERS

- Debugging in embedded application is the process of diagnosing the firmware execution, monitoring the target processor's registers and memory while the firmware is running.
- Debugging is classified into two namely Hardware debugging and firmware debugging.
- Hardware debugging deals with the monitoring of various bus signals and checking the status lines of the target hardware.

- Firmware debugging deals with examining the firmware execution, execution flow, changes to various CPU registers and status registers on execution of the firmware to ensure that the firmware is running as per the design.
- It is a special program used to find errors or bugs in other programs.
- A debugger allows a programmer to stop a program at any point and examine and change the values of the variables.
- A debugger or debugging tool is a computer program that is used to test and debug other programs.
- Some of the debuggers offer two modes of operation like full or partial simulation.
- A crash happens when the program cannot normally continue because of a programming bug.
- Ex- The program might have tried to use an instruction not available on the current version of the CPU to access unavailable or protected memory.
- When program crashes or reaches a preset condition the debugger shows the position in the original code if it is a source-level debugger or symbolic debugger.

CODESIGN DEFINITION AND KEY CONCEPTS

CODESIGN

- The meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design

Key concepts

- Concurrent: hardware and software developed at the same time on parallel paths
- Integrated: interaction between hardware and software developments to produce designs that meet performance criteria and functional specifications

Motivations for Code sign

- Factors driving codesign (hardware/software systems):
    - Instruction Set Processors (ISPs) available as cores in many design kits (386s, DSPs, microcontrollers, etc.)
    - Systems on Silicon - many transistors available in typical processes (> 10 million transistors available in IBM ASIC process, etc.)

    - Increasing capacity of field programmable devices - some devices even able to be reprogrammed on-the-fly (FPGAs, CPLDs, etc.)
    - Efficient C compilers for embedded processors
    - Hardware synthesis capabilities
- The importance of codesign in designing hardware/software systems:
    - Improves design quality, design cycle time, and cost
- Reduces integration and test time
    - Supports growing complexity of embedded systems
    - Takes advantage of advances in tools and technologies
- Processor cores
- High-level hardware synthesis capabilities
- ASIC development

Categorizing Hardware/Software Systems

- Application Domain
    - Embedded systems
- Manufacturing control
- Consumer electronics
- Vehicles
- Telecommunications
- Defense Systems
    - Instruction Set Architectures
    - Reconfigurable Systems
- Degree of programmability
    - Access to programming
    - Levels of programming
- Implementation Features
    - Discrete vs. integrated components
    - Fabrication technologies

Categories of Codesign Problems

- Codesign of embedded systems
    - Usually consist of sensors, controller, and actuators
    - Are reactive systems
    - Usually have real-time constraints
    - Usually have dependability constraints
- Codesign of ISAs
    - Application-specific instruction set processors (ASIPs)
    - Compiler and hardware optimization and trade-offs
- Codesign of Reconfigurable Systems
    - Systems that can be personalized after manufacture for a specific application
    - Reconfiguration can be accomplished before execution or concurrent with execution (called evolvable systems)

Components of the Codesign Problem

- Specification of the system
- Hardware/Software Partitioning
    - Architectural assumptions - type of processor, interface style between hardware and software, etc.
    - Partitioning objectives - maximize speedup, latency requirements; minimize size, cost, etc.
    - Partitioning strategies - high level partitioning by hand, automated partitioning using various techniques, etc.
- Scheduling
    - Operation scheduling in hardware
    - Instruction scheduling in compilers
    - Process scheduling in operating systems
- Modeling the hardware/software system during the design process

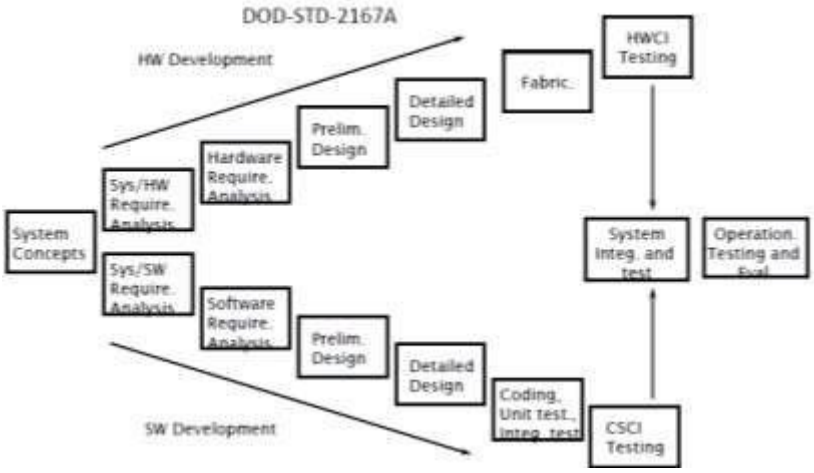


Figure: A Model of the Current Hardware/Software Design Process

Current Hardware/Software Design Process

- Basic features of current process:
  - System immediately partitioned into hardware and software components
  - Hardware and software developed separately
  - "Hardware first" approach often adopted
- Implications of these features:
  - HW/SW trade-offs restricted
- Impact of HW and SW on each other cannot be assessed easily
  - Late system integration
- Consequences these features:
  - Poor quality designs
  - Costly modifications
  - Schedule slippages

Incorrect Assumptions in Current Hardware/Software Design Process

- Hardware and software can be acquired separately and independently, with successful and easy integration of the two later
- Hardware problems can be fixed with simple software modifications
- Once operational, software rarely needs modification or maintenance

- Valid and complete software requirements are easy to state and implement in code

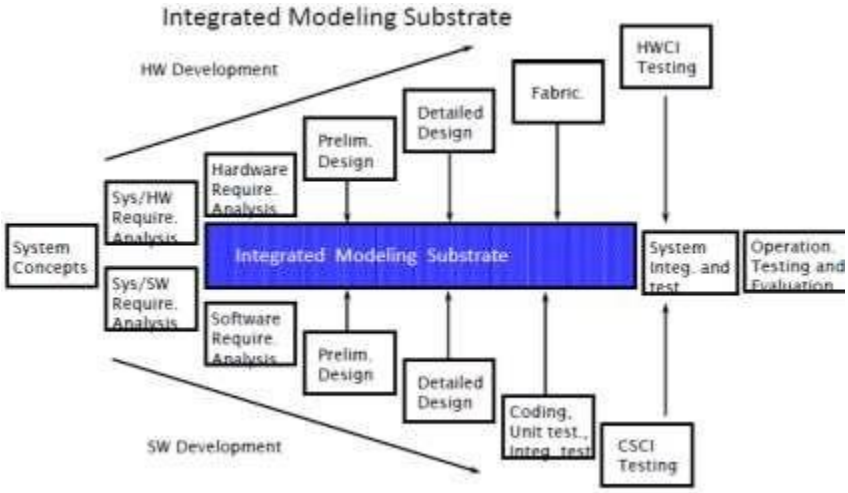


Figure: Directions of the HW/SW Design Process

Requirements for the Ideal Codesign Environment

- Unified, unbiased hardware/software representation
  - Supports uniform design and analysis techniques for hardware and software
  - Permits system evaluation in an integrated design environment
  - Allows easy migration of system tasks to either hardware or software
- Iterative partitioning techniques
  - Allow several different designs (HW/SW partitions) to be evaluated
  - Aid in determining best implementation for a system
  - Partitioning applied to modules to best meet design criteria (functionality and performance goals)
- Integrated modeling substrate
  - Supports evaluation at several stages of the design process
  - Supports step-wise development and integration of hardware and software
- Validation Methodology
  - Insures that system implemented meets initial system requirements

Cross-fertilization between Hardware and Software Design

- Fast growth in both VLSI design and software engineering has raised awareness of similarities between the two

- Hardware synthesis
- Programmable logic
- Description languages

- Explicit attempts have been made to “transfer technology” between the domains
- EDA tool technology has been transferred to SW CAD systems
  - Designer support (not automation)
  - Graphics-driven design
  - Central database for design information
  - Tools to check design behavior early in process
- Software technology has been transferred to EDA tools
  - Single-language design
- Use of 1 common language for architecture spec. and implementation of a chip
  - Compiler-like transformations and techniques
- Dead code elimination
- Loop unrolling
  - Design change management
- Information hiding
- Design families

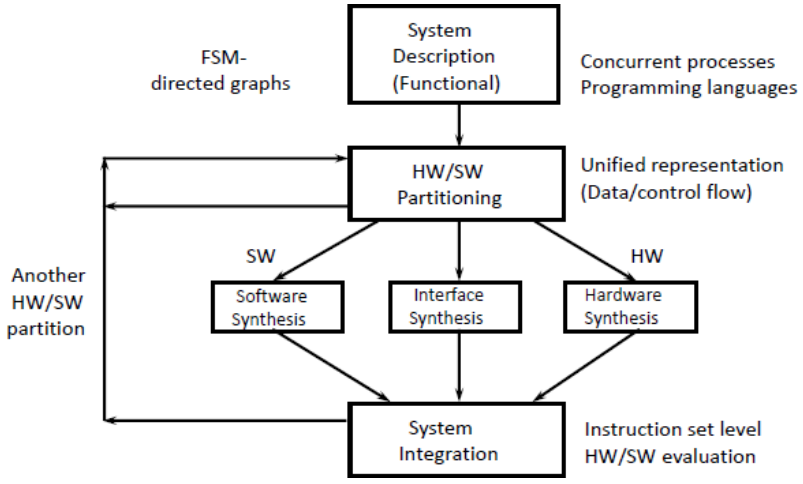


Figure: Typical Codesign Process

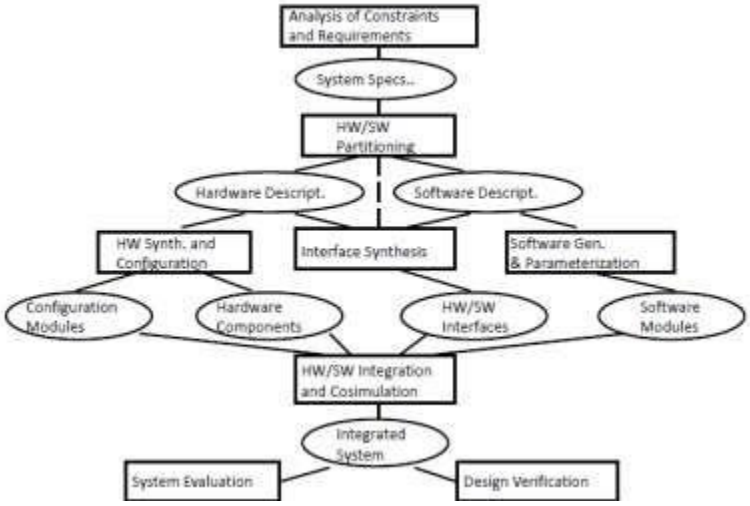


Figure: Conventional Codesign Methodology

Codesign Features

Basic features of a codesign process

- Enables mutual influence of both HW and SW early in the design cycle
    - Provides continual verification throughout the design cycle
    - Separate HW/SW development paths can lead to costly modifications and schedule slippages
- Enables evaluation of larger design space through tool interoperability and automation of codesign at abstract design levels
- Advances in key enabling technologies (e.g., logic synthesis and formal methods) make it easier to explore design tradeoffs

INTEGRATION AND TESTING OF EMBEDDED HARDWARE AND FIRMWARE.

Hardware Testing

- Hardware testing is to check/ensure the functionality, stability of hardware component and ensure that it should not have process fault. It also includes the heavy workload task for memory and CPU to check the performance and durability.
- Now a day's hardware design become much complex which demands the methods for testing to adhere and adapt to the challenges that arise, hence test development with

new standards for hardware become advance. There are many components involve in hardware testing like BIOS, CPU/Processor. Test the hardware to ensure its logical correctness and to ensure that follow appropriate standards. Using functional tests to determine whether met the test criteria. There are the few techniques commonly used for hardware testing.

  - Software-based self-testing
  - ATPG (Automatic test pattern generation)
  - BIST (Built-in self-test)

- ***The Software-Based Self-Testing:*** Modern microprocessors impose significant challenges to the testing hardware, because of their high complexity and heterogeneity. The software-based self-testing alternate way to hardware based self-testing, which cover the testing of a microprocessor using its instruction set. The benefit of software-based self test is that it can be applied in the normal operation mode of the microprocessor, thus applying the required tests at-speed.
- ***The ATPG (Automatic test pattern generation):*** Starting with a chip net list, inserting scan-chains (Scan chain is a technique used in design for testing), and generating vectors is the most direct and effortless approach and doesn't require complete knowledge of the DUT (Device under test). The recent EDA (Electronic Design Automation) tools are capable of deducing how to partition a design into blocks, and to isolate them by scan-chains.
- ***Built In Self Test (BIST):*** BIST is best for testing complex system, due to less accessibility to internal nets as design complexity increased, has spawned various design techniques that increases testability. BIST implementations are based on full scan architecture. This means that all the storage elements in the DUT concatenated to form several scan chains. These way test patterns can be serially shifted in and out of the storage elements. BIST requires no interaction with a large, expensive external test system. The testing is all built-in and only testers are needed to start the test.
- There are many tools available for hardware test and hardware diagnose like.
  - Automatic Test Equipment (ATE)
  - Sandra Lite – SiSoftware
- To execute load tests, simulate and observe variety of conditions and using or exceeding the amounts of data that could be expected in an actual situation. Following tools allows to measure different aspects of a system.

- Bonnie++
- IOZone
- Net pipe
- Linpack
- NFS Connectathon package

Firmware Testing

- Firmware is a computer program that is embedded in a hardware device that provides control, monitoring and data manipulation of engineered products and systems. The firmware contained devices provides the low-level control program for the device. Examples of devices containing firmware are embedded systems, computers, computer peripherals, mobile phones, etc.
- Importance of Firmware testing is the certification of firmware system meets its requirements with respect to functional correctness as well as performance, operational, and implementation properties. Then to reduce the risk and improve the performance.
- The firmware functionality changed from conventional instruction set emulators to more extensive and powerful instruction sets, diagnostic programs, interpreters for high level languages, and operating system functions. These are the three techniques of firmware testing.
    - Tests the micro program level that considers complete micro programs by analyzing their code or investigating the machine states after execution.
    - Tests the microinstruction level that considers single microinstructions by analyzing the assignment of micro-operations to them or investigating the machine states after execution.
    - Tests the micro-operation level that consider individual micro-operations by monitoring the execution
- The firmware testing is huge and complex task to complete, to overcome this challenge there are some automated tools available.
    - Firmware Test Suite (fwts): FWTS is a Linux tool that automates firmware checking. Tests that are designed to exercise and test different aspects of a machine's firmware – including ACPI, UEFI, hardware configuration, power-saving and so on.

UNIT-3

INTRODUCTION

- Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.
- Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are four main components used in IoT:

1. **Low-power embedded systems:** Less battery consumption, high performance is the inverse factors play a significant role during the design of electronic systems.
2. **Cloud computing:** Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.
3. **Availability of big data:** We know that IoT relies heavily on sensors, especially real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.
4. **Networking connection:** In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

There are two ways of building IoT:

- Form a separate internetwork including only physical objects.
- Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

IoT Enablers:

1. **RFIDs:** uses radio waves in order to electronically track the tags attached to each physical object.
2. **Sensors:** devices that are able to detect changes in an environment (ex: motion detectors).
3. **Nanotechnology:** as the name suggests, these are extremely small devices with dimensions usually less than a hundred nanometers.
4. **Smart networks:** (ex: mesh topology).

CHARACTERISTICS OF IOT

1. ***Interconnectivity:*** With regard to the IoT, anything can be interconnected with the global information and communication infrastructure.
2. ***Things-related services:*** The IoT is capable of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things. In order to provide thing-related services within the constraints of things, both the technologies in physical world and information world will change.
3. ***Heterogeneity:*** The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks.
4. ***Dynamic changes:*** The state of devices change dynamically, e.g., sleeping and waking up, connected and/or disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically.
5. ***Enormous scale:*** The number of devices that need to be managed and that communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet. Even more critical will be the management of the data generated and their interpretation for application purposes. This relates to semantics of data, as well as efficient data handling.
6. ***Safety:*** As we gain benefits from the IoT, we must not forget about safety. As both the creators and recipients of the IoT, we must design for safety. This includes the safety of our personal data and the safety of our physical well-being. Securing the endpoints, the networks, and the data moving across all of it means creating a security paradigm that will scale.

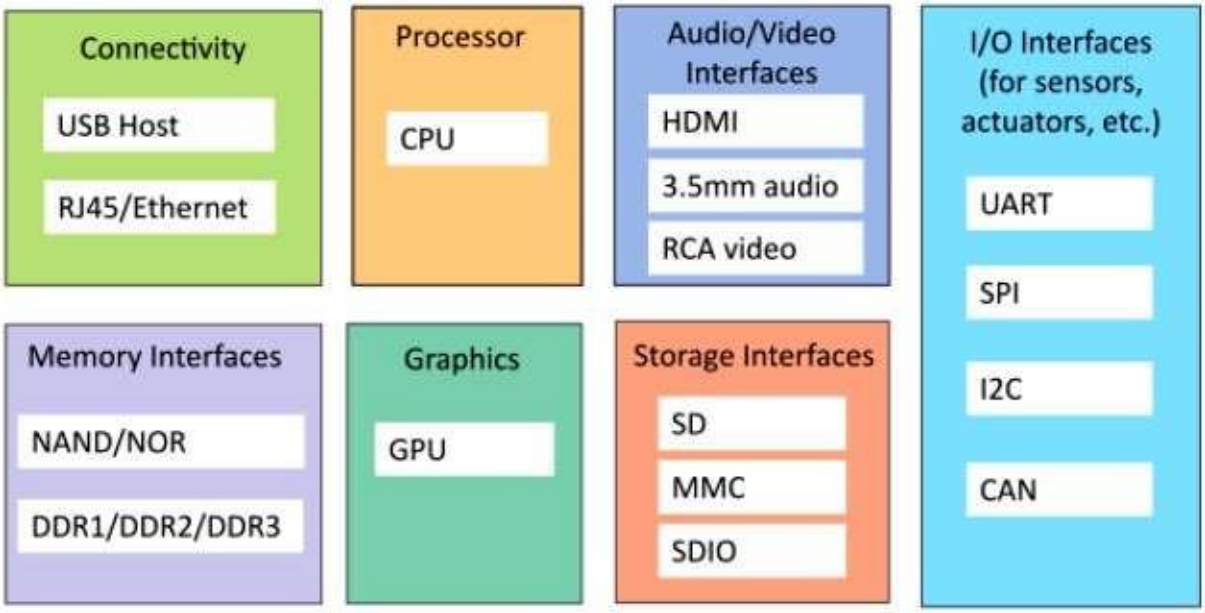
7. ***Connectivity:*** Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and produce data.

PHYSICAL DESIGN OF IOT

1. **Things in IoT**
2. **IoT Protocols**

Things in IoT:

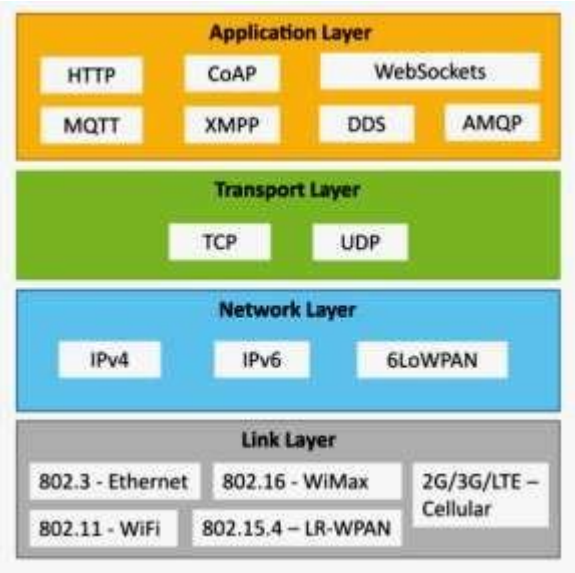
- Refers to IoT devices which have unique identities that can perform sensing, actuating and monitoring capabilities.
- IoT devices can exchange data with other connected devices or collect data from other devices and process the data either locally or send the data to centralized servers or cloud – based application back-ends for processing the data.



Generic Block Diagram of an IoT Device

- An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.
- I/O interfaces for sensors
- Interfaces for internet connectivity
- Memory and storage interfaces
- Audio/video interfaces

IOT PROTOCOLS



IoT Protocols-Link Layer-Ethernet

LOGICAL DESIGN OF IOT

- In this article we discuss Logical design of Internet of things. Logical design of IoT system refers to an abstract representation of the entities & processes without going into the low-level specifies of the implementation. For understanding Logical Design of IoT, we describes given below terms.
    1. IoT Functional Blocks
    2. IoT Communication Models
    3. IoT Communication APIs

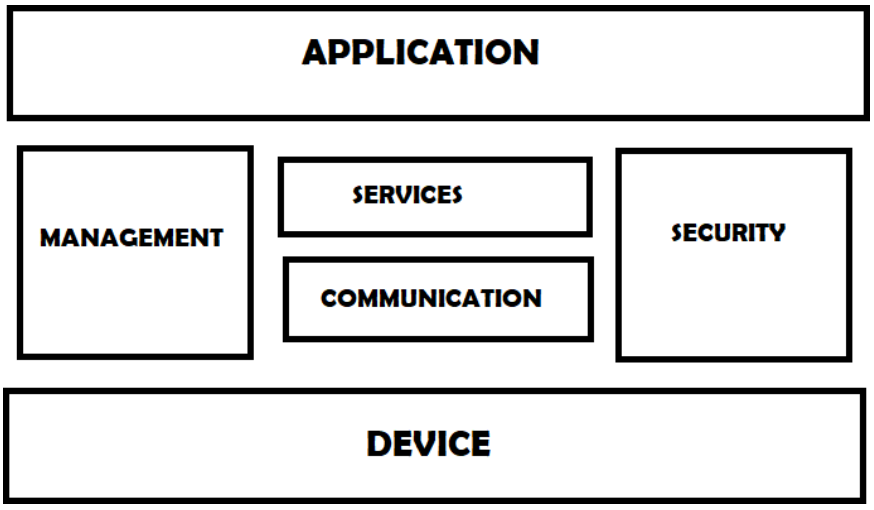
1. IoT Functional Blocks

- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication and management.

Functional blocks are:

- ***Device:*** An IoT system comprises of devices that provide sensing, actuation, and monitoring and control functions.
- ***Communication:*** Handles the communication for the IoT system.
- ***Services:*** services for device monitoring, device control service, data publishing services and services for device discovery.

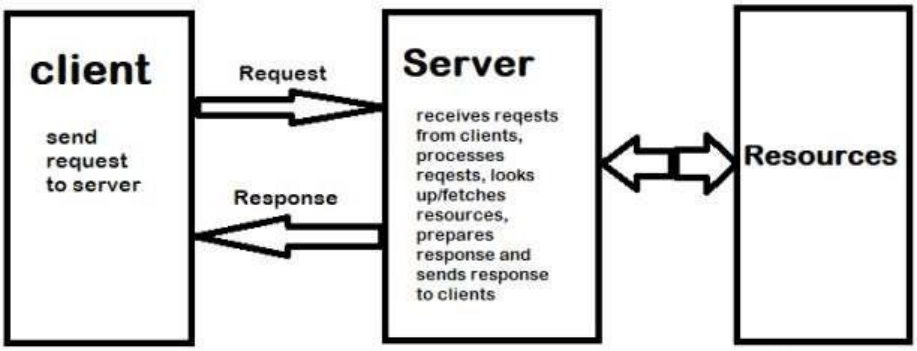
- ***Management:*** This block provides various functions to govern the IoT system.
- ***Security:*** this block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
- ***Application:*** This is an interface that the users can use to control and monitor various aspects of the IoT system. Application also allows users to view the system status and view or analyze the processed data.



2. IoT Communication Models:

Request-Response Model

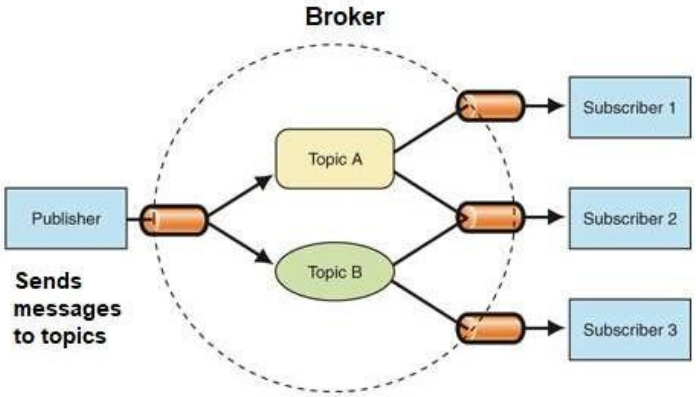
- Request-response model is communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representation, prepares the response, and then sends the response to the client. Request-response is a stateless communication model and each request-response pair is independent of others.
- HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server.
- Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.



Request-Response Communication Model

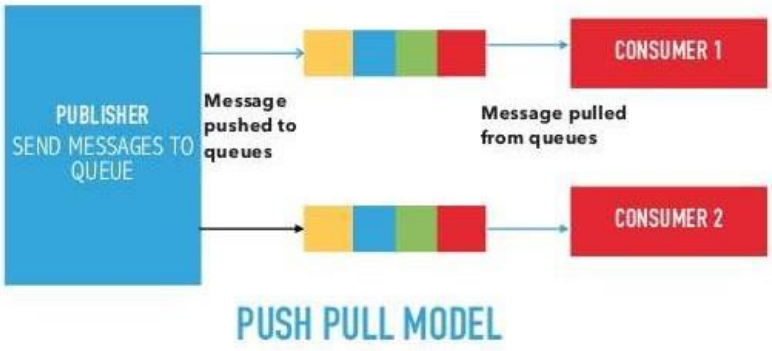
Publish-Subscribe Model

- Publish-Subscribe are a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.



Push-Pull Model

- Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the Queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the Producers and Consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumer pull data.



PUSH PULL MODEL

Exclusive Pair Model

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server. Connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. Exclusive pair is state full communication model and the server is aware of all the open connections.

EXCLUSIVE PAIR COMMUNICATION MODEL

IoT Communication APIs:

Generally we used Two APIs for IoT Communication. These IoT Communication APIs are:

- REST-based Communication APIs
- Web Socket-based Communication APIs

REST-based Communication APIs

Representational state transfer (REST) is a set of architectural principles by which you can design Web services the Web APIs that focus on systems resources and how resource states are addressed and transferred. REST APIs that follow the request response communication model,

the rest architectural constraint apply to the components, connector and data elements, within a distributed hypermedia system. The rest architectural constraint is as follows:

1. ***Client-server:*** The principle behind the client-server constraint is the separation of concerns. For example clients should not be concerned with the storage of data which is concern of the serve. Similarly the server should not be concerned about the user interface, which is concern of the client. Separation allows client and server to be independently developed and updated.
2. ***Stateless:*** Each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server. The session state is kept entirely on the client.
3. ***Cache-able:*** Cache constraints requires that the data within a response to a request be implicitly or explicitly leveled as cache-able or non cache-able. If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests. Caching can partially or completely eliminate some instructions and improve efficiency and scalability.
4. ***Layered system:*** layered system constraints, constrains the behavior of components such that each component cannot see beyond the immediate layer with they are interacting. For example, the client cannot tell whether it is connected directly to the end server or two an intermediary along the way. System scalability can be improved by allowing intermediaries to respond to requests instead of the end server, without the client having to do anything different.
5. ***Uniform interface:*** uniform interface constraints require that the method of communication between client and server must be uniform. Resources are identified in the requests (by URIs in web based systems) and are themselves is separate from the representations of the resources data returned to the client. When a client holds a representation of resources it has all the information required to update or delete the resource you (provided the client has required permissions). Each message includes enough information to describe how to process the message.
6. ***Code on demand:*** Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

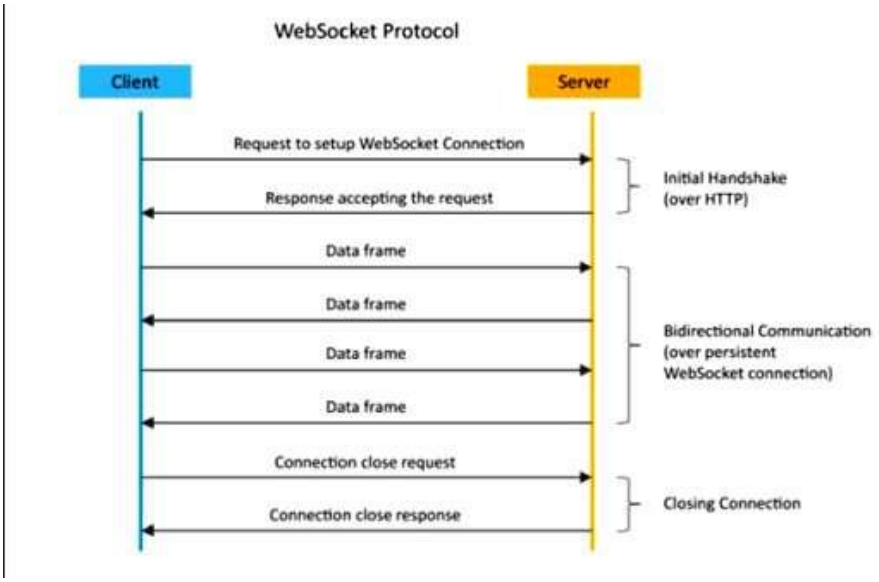
A Restful web service is a" Web API" implemented using HTTP and REST principles. REST is most popular IoT Communication APIs.

HTTP methods

Uniform Resource Identifier (URI)	GET	PUT	PATCH	POST	DELETE
Collection, such as https://api.example.com/resources/	List the URIs and perhaps other details of the collection's members.	Replace the entire collection with another collection.	Not generally used	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.	Delete the entire collection.
Element, such as https://api.example.com/resources/item5	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	Replace the addressed member of the collection, or if it does not exist, create it.	Update the addressed member of the collection.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry within it.	Delete the addressed member of the collection.

Web Socket based communication API

Web socket APIs allows bi-directional, full duplex communication between clients and servers. Web socket APIs follows the exclusive pair communication model. Unlike request-response model such as REST, the Web Socket APIs allow full duplex communication and do not require new connection to be setup for each message to be sent. Web socket communication begins with a connection setup request sent by the client to the server. The request (called web socket handshake) is sent over HTTP and the server interprets it is an upgrade request. If the server supports web socket protocol, the server responds to the web socket handshake response. After the connection setup client and server can send data/messages to each other in full duplex mode. Web socket API reduces the network traffic and latency as there is no overhead for connection setup and termination requests for each message. Web socket suitable for IoT applications that have low latency or high throughput requirements. So Web socket is most suitable IoT Communication APIs for IoT System.



IOT ENABLING TECHNOLOGIES

Wireless Sensor Networks

A wireless sensor network comprises of distributed device with sensor which are used to monitor the environmental and physical conditions. A WSN consists of a number of end-nodes and routers and a coordinator. End Nodes have several sensors attached to them in node can also act as routers. Routers are responsible for routing the data packets from end-nodes to the coordinator. The coordinator collects the data from all the nodes. Coordinator also acts as a gateway that connects the WSN to the internet. Some examples of WSNs used in IoT systems are described as follows:

- Weather monitoring system use WSNs in which the nodes collect temperature humidity and other data which is aggregated and analyzed.
- Indoor air quality monitoring systems use WSNs to collect data on the indoor air quality and concentration of various gases
- Soil moisture monitoring system use WSNs to monitor soil moisture at various locations.
- Surveillance system use WSNs for collecting Surveillance data (such as motion detection data)
- Smart grid use WSNs for monitoring the grid at various points.
- Structural health monitoring system use WSNs to monitor the health of structures (buildings, bridges) by collecting vibration data from sensor nodes de deployed at various points in the structure.

You may like also:

- LiteOS: an IoT operating system and middleware
- Contiki OS: The Open Source OS for IoT

Cloud Computing

- Cloud computing is a trans-formative computing paradigm that involves delivering applications and services over the Internet Cloud computing involves provisioning of computing, networking and storage resources on demand and providing these resources as metered services to the users, in a "pay as you go" model. C loud computing resources can be provisioned on demand by the users, without requiring interactions with the cloud service Provider. The process of provisioning resources is automated. Cloud computing resources can be accessed over the network using standard access mechanisms that provide platform independent access through the use of heterogeneous client platforms such as the workstations, laptops, tablets and smart phones.

Cloud computing services are offered to users in different forms:

- ***Infrastructure as a Service (IaaS):*** hardware is provided by an external provider and managed for you
- ***Platform as a Service (PaaS):*** in addition to hardware, your operating system layer is managed for you
- ***Software as a Service (SaaS):*** further to the above, an application layer is provided and managed for you – you won't see or have to worry about the first two layers.

Big Data Analytics

- Big Data analytics is the process of collecting, organizing and analyzing large sets of data (called Big Data) to discover patterns and other useful information. Big Data analytics can help organizations to better understand the information contained within the data and will also help identify the data that is most important to the business and future business decisions. Analysts working with Big Data typically want the knowledge that comes from analyzing the data.

Some examples of big data generated by IoT systems are described as follows:

- Sensor data generated by IoT system such as weather monitoring stations.
- Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting Failures.
- Health and fitness data generated by IoT devices such as wearable fitness bands
- Data generated by IoT systems for location and tracking of vehicles
- Data generated by retail inventory monitoring systems
- **Characteristics**
- Big data can be described by the following characteristics:
- **Volume:** The quantity of generated and stored data. The size of the data determines the value and potential insight and whether it can be considered big data or not.
- **Variety:** The type and nature of the data. This helps people who analyze it to effectively use the resulting insight. Big data draws from text, images, audio, video; plus it completes missing pieces through data fusion.
- **Velocity:** In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. Big data is often available in real-time. Compared to small data, big data are produced more continually. Two kinds of velocity related to Big Data are the frequency of generation and the frequency of handling, recording, and publishing.
- **Veracity:** It is the extended definition for big data, which refers to the data quality and the data value. The data quality of captured data can vary greatly, affecting the accurate analysis.

Communication protocols

- Communication protocols form the backbone of IoT systems and enable network connectivity and coupling to applications. Communication protocols allow devices to exchange data over the network. Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together are known as a protocol suite; when implemented in software they are a protocol stack.
- Internet communication protocols are published by the Internet Engineering Task Force (IETF). The IEEE handles wired and wireless networking, and the International Organization for Standardization (ISO) handles other types. The ITU-T handles telecommunication protocols and formats for the public switched telephone network (PSTN). As the PSTN and Internet converge, the standards are also being driven towards convergence.
- In IoT we used MQTT, COAP, AMQP etc. protocols. You can read in detail by given below links.
- **You may like also:**
    - IoT Data Protocols
    - Wireless IoT Network Protocols
    - IoT Open Source Development Tools

Embedded Systems:

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a controller programmed and controlled by a real-time operating system (RTOS) with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured to serve as embedded system component.

An embedded system has three components:

- It has hardware.
- It has application software.
- It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a

plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

IOT LEVELS

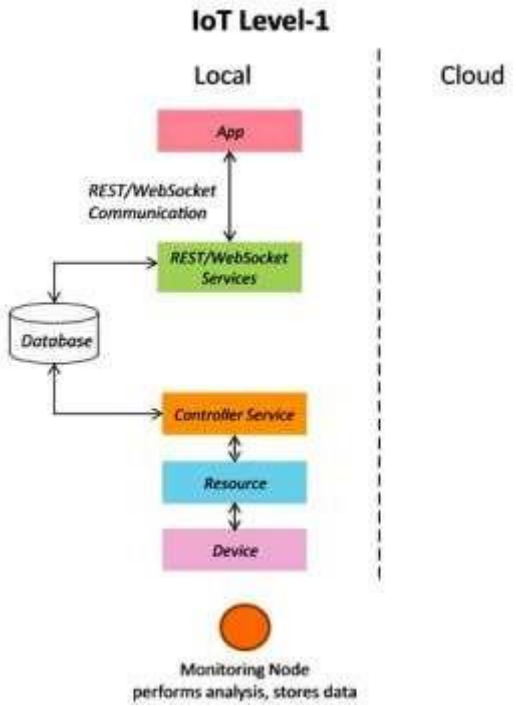
An IoT system comprises the following components: **Device, Resource, Controller Service, Database, and Web service, Analysis, Component and Application.**

- ***Device:*** An IoT device allows identification, remote sensing, and remote monitoring capabilities.
- ***Resource:***
    - Software components on the IoT device for
        - accessing, processing and storing sensor information,
        - Controlling actuators connected to the device.
        - Enabling network access for the device.
- ***Controller Service:*** Controller service is a native service that runs on the device and interacts with the web services. It sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.
- ***Database:*** Database can be either local or in the cloud and stores the data generated by the IoT device.
- ***Web Service:*** Web services serve as a link between the IoT device, application, and database and analysis components. It can be implemented using HTTP and REST principles (REST service) or using the Web Socket protocol (Web Socket service).
- ***Analysis Component:*** Analysis Component is responsible for analyzing the IoT data and generating results in a form that is easy for the user to understand.
- ***Application:*** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and the processed data.

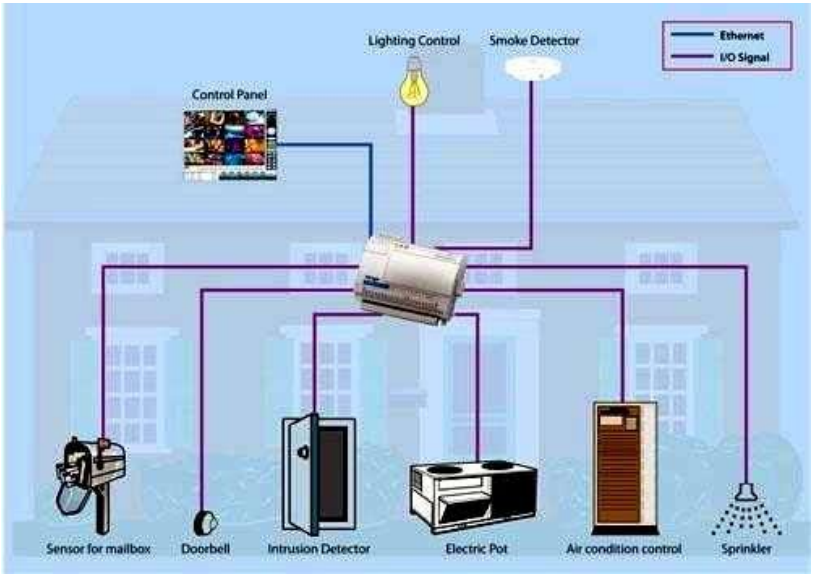
IoT Level-1

- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application.

- Level-1 IoT systems are suitable for modeling low- cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.

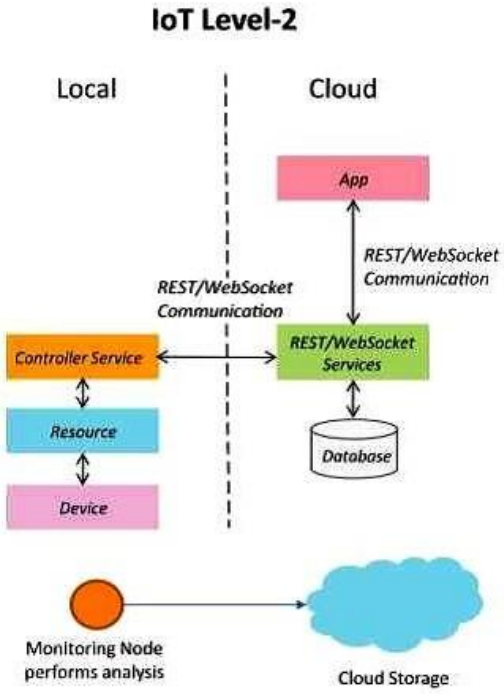


- IoT – Level 1 Example : Home Automation System



IoT Level-2

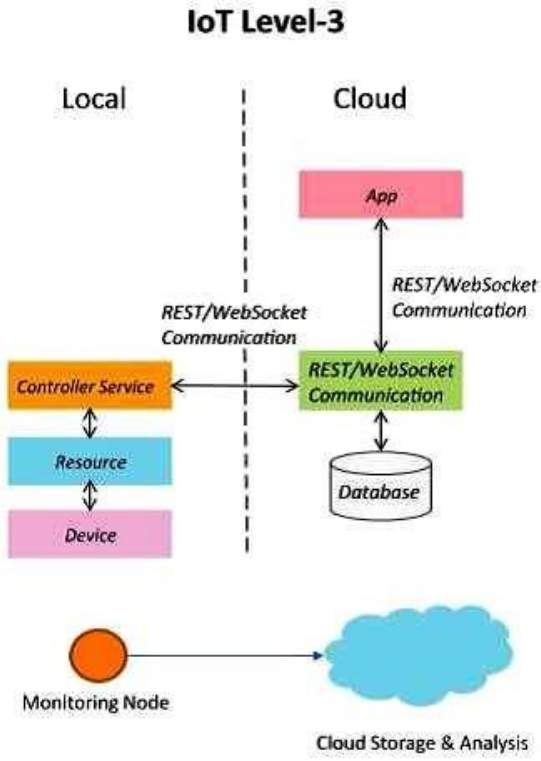
- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis. Data is stored in the cloud and the application is usually cloud-based.
- Level-2 IoT systems are suitable for solutions where the data involved is big; however, the primary analysis requirement is not computationally intensive and can be done locally



- **IoT – Level 2 Example: Smart Irrigation**

IoT Level-3

- A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and the application is cloud based.
- Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.



- IoT – Level 3 Example: Tracking Package Handling

Sensors used

Accelerometer sense movement or

Gyroscope

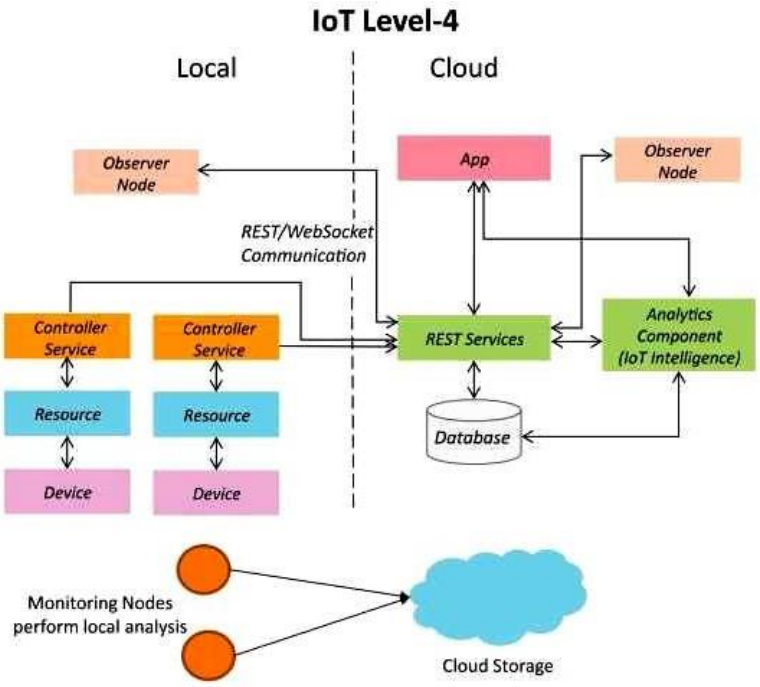
Gives orientation info

vibrations

- Web socket service is used because sensor data can be sent in real time.

IoT Level-4

- A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and the application is cloud-based.
- Level-4 contains local and cloud based observer nodes which can subscribe and receive information collected in the cloud from IoT devices.
- Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.

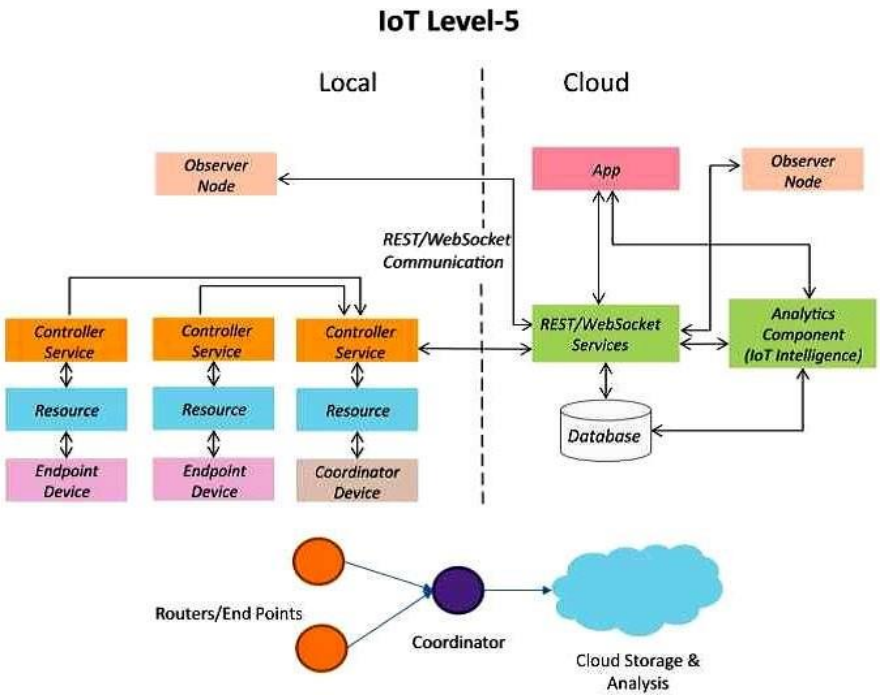


- IoT – Level 4 Example: Noise Monitoring

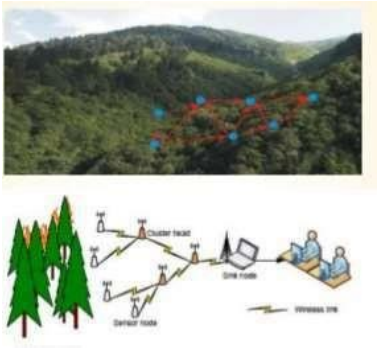
Sound Sensors are used

IoT Level-5

- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes perform sensing and/or actuation.
- The coordinator node collects data from the end nodes and sends it to the cloud.
- Data is stored and analyzed in the cloud and the application is cloud- based.
- Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is brigand the analysis requirements are computationally intensive.



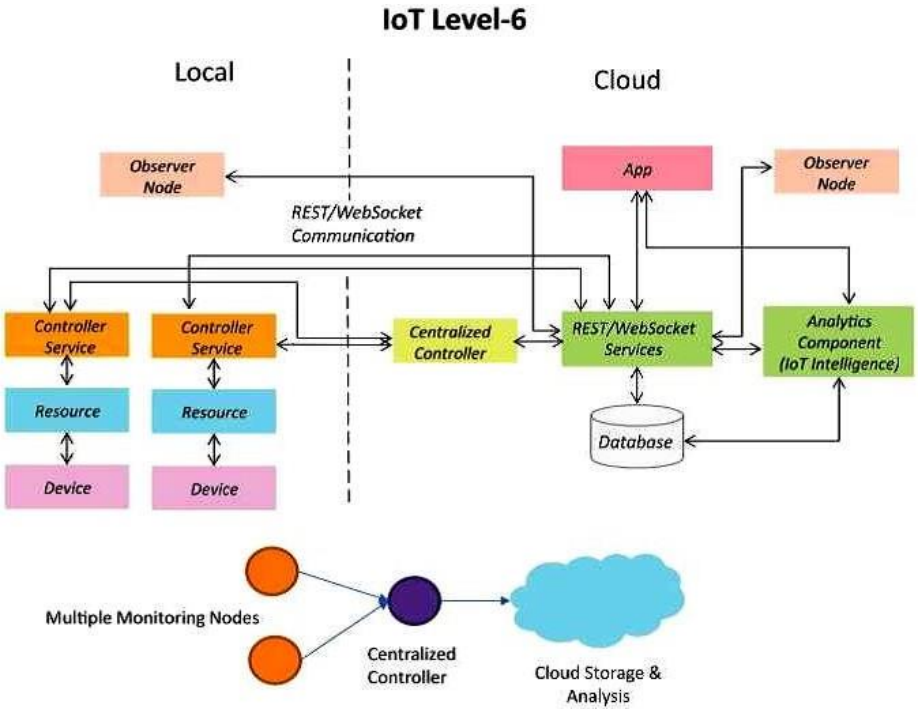
- IoT – Level 5 Example: Forest Fire Detection
- Detect forest fire in early stages to take action while the fire is still controllable. Sensors measure the temperature, smoke, weather, slope of the earth, wind speed, speed of fire spread, flame length



IoT Level-6

- A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- Data is stored in the cloud and the application is cloud-based.
- The analytics component analyzes the data and stores the results in the cloud database.
- The results are visualized with the cloud-based application.

- The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.



- IoT – Level 6 Example: Weather Monitoring System
- ***Sensors used :*** Wind speed and direction, Solar radiation, Temperature (air, water, soil), Relative humidity, Precipitation, Snow depth, Barometric pressure, Soil moisture

DOMAIN SPECIFIC IOTS

IoT Applications for:

1. Home
2. Cities
3. Environment
4. Energy Systems
5. Retail
6. Logistics
7. Industry
8. Agriculture
9. Health & Lifestyle

1. Home Automation

IoT applications for smart homes:

a). Smart Lighting

b). Smart Appliances

c). Intrusion Detection

d). Smoke / Gas Detectors

a). Smart Lighting

- Smart lighting achieves energy savings by sensing the human movements and their environments and controlling the lights accordingly.
- Key enabling technologies for smart lighting include :
  - Solid state lighting (such as LED lights)
  - IP-enabled lights
- Wireless-enabled and Internet connected lights can be controlled remotely from IoT applications such as a mobile or web application.
- Paper: Energy-aware wireless sensor network with ambient intelligence for smart LED lighting system control [IECON, 2011] presented controllable LED lighting system that is embedded with ambient intelligence gathered from a distributed smart WSN to optimize and control the lighting system to be more efficient and user-oriented.

b). Smart Appliances

- Smart appliances make the management easier and provide status information of appliances to the users remotely. E.g.: smart washer/dryer that can be controlled remotely and notify when the washing/drying cycle is complete.
- Open Remote is an open source automation platform for smart home and building that can control various appliances using mobile and web applications.
- It comprises of three components:
    - A Controller: manages scheduling and runtime integration between devices.
    - A Designer: allows creating both configurations for the controller and user interface designs.
    - Control Panel: allows interacting with devices and controlling them.
- Paper: An IoT-based Appliance Control System for Smart Home [ICICIP, 2013] [implemented an IoT based appliance control system for smart homes that uses a smart-central controller to set up a wireless sensor and actuator network and control modules for appliances.

c). Intrusion Detection

- Home intrusion detection systems use security cameras and sensors to detect intrusions and raise alerts.
- The form of the alerts can be in form:
    - SMS
    - Email
    - Image grab or a short video clip as an email attachment
- Papers :
    - Could controlled intrusion detection and burglary prevention stratagems in home automation systems [BCFIC, 2012] present a controlled intrusion detection system that uses location-aware services, where the geo-location of each node of a home automation system is independently detected and stored in the cloud?
    - An Intelligent Intrusion Detection System Based on UPnP Technology for Smart Living [ISDA, 2008] implement an intrusion detection system that uses image processing to recognize the intrusion and extract the intrusion subject and generate Universal-Plug-and-Play (UPnP-based) instant messaging for alerts.

d). Smoke / Gas Detectors

- Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire.
- It uses optical detection, ionization or air sampling techniques to detect smoke
- The form of the alert can be in form :
  - Signals that send to a fire alarm system
- Gas detector can detect the presence of harmful gases such as carbon monoxide (CO), liquid petroleum gas (LPG), etc.
- Paper: Development of Multipurpose Gas Leakage and Fire Detector with Alarm System [TIIEC, 2013] designed a system that can detect gas leakage and smoke and gives visual level indication.

2. Cities

- IoT applications for smart cities:

  a). Smart Parking

  b). Smart Lighting for Road

  c). Smart Road

  d). Structural Health Monitoring

  e). Surveillance

  f). Emergency Response

a). Smart Parking

- Finding the parking space in the crowded city can be time consuming and frustrating
- Smart parking makes the search for parking space easier and convenient for driver.
- It can detect the number of empty parking slots and send the information over the Internet to the smart parking applications which can be accessed by the drivers using their smart phones, tablets, and in car navigation systems.
- Sensors are used for each parking slot to detect whether the slot is empty or not, and this information is aggregated by local controller and then sent over the Internet to database.

- Paper :
    - Design and implementation of a prototype Smart Parking (SPARK) system using WSN [International Conference on Advanced Information Networking and Applications Workshop, 2009] designed and implemented a prototype smart parking system based on wireless sensor network technology with features like remote parking monitoring, automate guidance, and parking reservation mechanism.

b). Smart Lighting for Roads

- It can help in saving energy
- Smart lighting for roads allows lighting to be dynamically controlled and also adaptive to ambient conditions.
- Smart light connected to the Internet can be controlled remotely to configure lighting schedules and lighting intensity.
- Custom lighting configurations can be set for different situations such as a foggy day, a festival, etc.
- Paper :
    - Smart Lighting solutions for Smart Cities [International Conference on Advance Information Networking and Applications Workshop, 2013] described the need for smart lighting system in smart cities, smart lighting features and how to develop interoperable smart lighting solutions.

c). Smart Roads

- Smart Roads provides information on driving conditions, travel time estimates and alerts in case of poor driving conditions, traffic congestions and accidents.
- Such information can help in making the roads safer and help in reducing traffic jams
- Information sensed from the roads can be communicated via internet to cloud-based applications and social media and disseminated to the drivers who subscribe to such applications.
- Paper:
    - Sensor networks for smart roads [PerCom Workshop, 2006] proposed a distributed and autonomous system of sensor network nodes for improving driving safety on public roads, the system can provide the driver and passengers with a consistent view of the road situation a few hundred meters ahead of them or a few dozen miles away, so that they can react to potential dangers early enough.

d). Structural Health Monitoring

- It uses a network of sensors to monitor the vibration levels in the structures such as bridges and buildings.
- The data collected from these sensors is analyzed to assess the health of the structures.
- By analyzing the data it is possible to detect cracks and mechanical breakdowns, locate the damages to a structure and also calculate the remaining life of the structure.
- Using such systems, advance warnings can be given in the case of imminent failure of the structure.
- Paper:
  - Environmental Effect Removal Based Structural Health Monitoring in the Internet of Things [International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013] □ proposed an environmental effect removal based structural health monitoring scheme in an IoT environment.
  - Energy harvesting technologies for structural health monitoring applications [IEEE Conference on Technologies for Sustainability, 2013] □ Explored energy harvesting technologies of harvesting ambient energy, such as mechanical vibrations, sunlight, and wind.

e). Surveillance

- Surveillance of infrastructure, public transport and events in cities is required to ensure safety and security.
- City wide surveillance infrastructure comprising of large number of distributed and Internet connected video surveillance cameras can be created.
- The video feeds from surveillance cameras can be aggregated in cloud-based scalable storage solutions.
- Cloud-based video analytics applications can be developed to search for patterns of specific events from the video feeds.

f). Emergency Response

- IoT systems can be used for monitoring the critical infrastructure cities such as buildings, gas, and water pipelines, public transport and power substations.
- IoT systems for critical infrastructure monitoring enable aggregation and sharing of information collected from lager number of sensors.

- Using cloud-based architectures, multi-modal information such as sensor data, audio, video feeds can be analyzed I near real-time to detect adverse events.
- The alert can be in the form :
  - Alerts sent to the public
  - Re-rerouting of traffic
  - Evacuations of the affected areas

3. Environment

IoT applications for smart environments:

a). Weather Monitoring

b). Air Pollution Monitoring

c). Noise Pollution Monitoring

d). Forest Fire Detection

e). River Flood Detection

a). Weather Monitoring

- It collects data from a number of sensors attached such as temperature, humidity, pressure, etc and sends the data to cloud-based applications and store back-ends.
- The data collected in the cloud can then be analyzed and visualized by cloud-based applications.
- Weather alert can be sent to the subscribed users from such applications.
- AirPi is a weather and air quality monitoring kit capable of recording and uploading information about temperature, humidity, air pressure, light levels, UV levels, carbon monoxide, nitrogen dioxide and smoke level to the Internet.
- Paper:
  - PeWeMoS – Pervasive Weather Monitoring System [ICPCA, 2008] □ Presented a pervasive weather monitoring system that is integrated with buses to measure weather variables like humidity, temperature, and air quality during the bus path

b). Air Pollution Monitoring

- IoT based air pollution monitoring system can monitor emission of harmful gases by factories and automobiles using gaseous and meteorological sensors.

- The collected data can be analyzed to make informed decisions on pollutions control approaches.
- Paper:
  - Wireless sensor network for real-time air pollution monitoring [ICCSPA, 2013] [Presented a real time air quality monitoring system that comprises of several distributed monitoring stations that communicate via wireless with a backend server using machine-to machine communication.

c). Noise Pollution Monitoring

- Noise pollution monitoring can help in generating noise maps for cities.
- It can help the policy maker in making policies to control noise levels near residential areas, school and parks.
- It uses a number of noise monitoring stations that are deployed at different places in a city.
- The data on noise levels from the stations is collected on servers or in the cloud and then the collected data is aggregate to generate noise maps.
- Papers :
  - Noise mapping in urban environments: Applications at Suez city center [ICCIE, 2009] presented a noise mapping study for a city which revealed that the city suffered from serious noise pollution.
  - Sound Of City – Continuous noise monitoring for a health city [PerComW,2013] [Designed a Smartphone application that allows the users to continuously measure noise levels and send to a central server here all generated information is aggregated and mapped to a meaningful noise visualization map.

d). Forest Fire Detection

- IoT based forest fire detection system use a number of monitoring nodes deployed at different location in a forest.
- Each monitoring node collects measurements on ambient condition including temperature, humidity, light levels, etc.
- Early detection of forest fires can help in minimizing the damage.
- Papers:
  - A novel accurate forest fire detection system using wireless sensor networks [International Conference on Mobile Adhoc and Sensor Networks, 2011] Presented

a forest fire detection system based on wireless sensor network. The system uses multi-criteria detection which is implemented by the artificial neural network. The ANN fuses sensing data corresponding to, multiple attributes of a forest fire such as temperature, humidity, infrared and visible light to detect forest fires.

e). River Flood Detection

- IoT based river flood monitoring system uses a number of sensor nodes that monitor the water level using ultrasonic sensors and flow rate using velocity sensors.
- Data from these sensors is aggregated in a server or in the cloud, monitoring applications raise alerts when rapid increase in water level and flow rate is detected.
- Papers:
  - RFMS : Real time flood monitoring system with wireless sensor networks [MASS, 2008] Described a river flood monitoring system that measures river and weather conditions through wireless sensor nodes equipped with different sensors
  - Urban Flash Flood Monitoring, Mapping and Forecasting via a Tailored Sensor Network System [ICNSC, 2006] Described a motes-based sensor network for river flood monitoring that includes a water level monitoring module, network video recorder module, and data processing module that provides floods information n the form of raw data, predict data, and video feed.

4. Energy

IoT applications for smart energy systems:

a). Smart Grid

b). Renewable Energy Systems

c). Prognostics

a). Smart Grids

- Smart grid technology provides predictive information and recommendation s to utilize, their suppliers, and their customers on how best to manage power.
- Smart grid collect the data regarding :
  - Electricity generation
  - Electricity consumption
  - Storage
  - Distribution and equipment health data

- By analyzing the data on power generation, transmission and consumption of smart grids can improve efficiency throughout the electric system. Storage collection and analysis of smarts grids data in the cloud can help in dynamic optimization of system operations, maintenance, and planning.
- Cloud-based monitoring of smart grids data can improve energy usage levels viaenergy feedback to users coupled with real-time pricing information.
- Condition monitoring data collected from power generation and transmission systems can help in detecting faults and predicting outages.

b). Renewable Energy System

- Due to the variability in the output from renewable energy sources (such as solar and wind), integrating them into the grid can cause grid stability and reliability problems.
- IoT based systems integrated with the transformer at the point of interconnection measure the electrical variables and how much power is fed into the grid
- To ensure the grid stability, one solution is to simply cut off the overproductions.
- Paper:
    - Communication systems for grid integration of renewable energy resources [IEEE Network, 2011] -provided the closed-loop controls for wind energy system that can be used to regulate the voltage at point of interconnection which coordinate wind turbine outputs and provides reactive power support.

c). Prognostics

- IoT based prognostic real-time health management systems can predict performance of machines of energy systems by analyzing the extent of deviation of a system from its normal operating profiles.
- In the system such as power grids, real time information is collected using specialized electrical sensors called Phasor Measurement Units (PMU)
- Analyzing massive amounts of maintenance data collected from sensors in energy systems and equipment can provide predictions for impending failures.
- OpenPDC is a set of applications for processing of streaming time-series data collected from Phasor Measurements Units (PMUs) in real-time.

5. Retail

IoT applications in smart retail systems:

a). Inventory Management

b). Smart Payments

c). Smart Vending Machines

a). Inventory Management

- IoT system using Radio Frequency Identification (RFID) tags can help inventory management and maintaining the right inventory levels.
- RFID tags attached to the products allow them to be tracked in the real-time so that the inventory levels can be determined accurately and products which are low on stock can be replenished.
- Tracking can be done using RFID readers attached to the retail store shelves or in the warehouse.
- Paper:
  - RFID data-based inventory management of time-sensitive materials [IECON, 2005] l described an RFID data-based inventory management system for time-sensitive materials

b). Smart Payments

- Smart payments solutions such as contact-less payments powered technologies such as near field communication (NFC) and Bluetooth.
- NFC is a set of standards for smart-phones and other devices to communicate with each other by bringing them into proximity or by touching them
- Customer can store the credit card information in their NFC-enabled smart-phones and make payments by bringing the smart-phone near the point of sale terminals.
- NFC maybe used in combination with Bluetooth, where NFC initiates initial pairing of devices to establish a Bluetooth connection while the actual data transfer takes place over Bluetooth.

c). Smart Vending Machines

- Smart vending machines connected to the Internet allow remote monitoring of inventory levels, elastic pricing of products, promotions, and contact-less payments using NFC.
- Smart-phone applications that communicate with smart vending machines allow user preferences to be remembered and learned with time. E.g.: when a user moves from one vending machine to the other and pair the smart-phone, the user preference and favorite product will be saved and then that data is used for predictive maintenance.

- Smart vending machines can communicated each others, so if a product out of stock in a machine, the user can be routed to nearest machine
- For perishable items, the smart vending machines can reduce the price as the expiry date nears.

6. Logistic

IoT applications for smart logistic systems:

a). Fleet Tracking

b). Shipment Monitoring

c). Remote Vehicle Diagnostics

a). Fleet Tracking

- Vehicle fleet tracking systems use GPS technology to track the locations of the vehicles in the real-time.
- Cloud-based fleet tracking systems can be scaled up on demand to handle large number of vehicles,
- The vehicle locations and routers data can be aggregated and analyzed for detecting bottlenecks the supply chain such as traffic congestions on routes, assignments and generation of alternative routes, and supply chain optimization
- Paper:
    - A Fleet Monitoring System for Advanced Tracking of commercial Vehicles [IEEE International Conference in Systems, Man and Cybernetics, 2006] □ provided a system that can analyze messages sent from the vehicles to identify unexpected incidents and discrepancies between actual and planned data, so that remedial actions can be taken.

b). Shipment Monitoring

- Shipment monitoring solutions for transportation systems allow monitoring the conditions inside containers.
- E.g.: Containers carrying fresh food produce can be monitored to prevent spoilage of food. IoT based shipment monitoring systems use sensors such as temperature, pressure, humidity, for instance, to monitor the conditions inside the containers and send the data to the cloud, where it can be analyzed to detect food spoilage.

- Paper:
    - On a Cloud-Based Information Technology Framework for Data Driven Intelligent Transportation System [Journal of Transportation Technologies, 2013] -proposed a cloud based framework for real time fresh food supply tracking and monitoring
    - Container Integrity and Condition Monitoring using RF Vibration Sensor Tags [IEEE International Conference on Automation Science and Engineering, 2007]- proposed a system that can monitor the vibrations patterns of a container and its contents to reveal information related to its operating environment and integrity during transport, handling, and storage.

c). Remote Vehicle Diagnostics

- It can detect faults in the vehicles or warn of impending faults.
- These diagnostic systems use on-board IoT devices for collecting data on vehicle operation such as speed, engine RPM, coolest temperature, fault code number and status of various vehicle subsystem.
- Modern commercial vehicles support on-board diagnostic (OBD) standard such as OBD-II
- OBD systems provide real-time data on the status of vehicle sub-systems and diagnostic trouble codes which allow rapidly identifying the faults in the vehicle.
- IoT based vehicle diagnostic systems can send the vehicle data to centralize servers or the cloud where it can be analyzed to generate alerts and suggest remedial actions.

7. Agriculture

IoT applications for smart agriculture:

a). Smart Irrigation

b). Green House Control

a). Smart Irrigation

- Smart irrigation system can improve crop yields while saving water.
- Smart irrigation systems use IoT devices with soil moisture sensors to determine the amount of moisture on the soil and release the flow of the water through the irrigation pipes only when the moisture levels go below a predefined threshold.
- It also collects moisture level measurements on the server on in the cloud where the collected data can be analyzed to plan watering schedules.

- Cultivar's Rain Could is a device for smart irrigation that uses water valves, soil sensors, and a Wi-Fi enabled programmable computer. [http://ecultivar.com/rain-cloud-product-project/]

b). Green House Control

- It controls temperature, humidity, soil, moisture, light, and carbon dioxide level that are monitored by sensors and climatologically conditions that are controlled automatically using actuation devices.
- IoT systems play an importance role in green house control and help in improving productivity.
- The data collected from various sensors is stored on centralized servers or in the cloud where analysis is performed to optimize the control strategies and also correlate the productivity with different control strategies.
- Paper:
  - Wireless sensing and control for precision Green house management [ICST, 2012] I provided a system that uses wireless sensor network to monitor and control the agricultural parameters like temperature and humidity in the real time for better management and maintenance of agricultural production.

8. Industry

IoT applications in smart industry:

a). Machine Diagnosis & Prognosis

b). Indoor Air Quality Monitoring

a). Machine Diagnosis & Prognosis

- Machine prognosis refers to predicting the performance of machine by analyzing the data on the current operating conditions and how much deviations exist from the normal operating condition.
- Machine diagnosis refers to determining the cause of a machine fault.
- Sensors in machine can monitor the operating conditions such as temperature and vibration levels, sensor data measurements are done on timescales of few milliseconds to few seconds which leads to generation of massive amount of data.
- Case-based reasoning (CBR) is a commonly used method that finds solutions to new problems based on past experience.

- CBR is an effective technique for problem solving in the fields in which it is hard to establish a quantitative mathematical model, such as machine diagnosis and prognosis.

b). Indoor Air Quality Monitoring

- Harmful and toxic gases such as carbon monoxide (CO), nitrogen monoxide (NO), Nitrogen Dioxide, etc can cause serious health problem of the workers.
- IoT based gas monitoring systems can help in monitoring the indoor air quality using various gas sensors.
- The indoor air quality can be placed for different locations
- Wireless sensor networks based IoT devices can identify the hazardous zones, so that corrective measures can be taken to ensure proper ventilation.
- Papers:
  - A hybrid sensor system for indoor air quality monitoring [IEEE International Conference on Distributed Computing in Sensor System, 2013] presented a hybrid sensor system for indoor air quality monitoring which contains both stationary sensor and mobile sensors.
  - Indoor air quality monitoring using wireless sensor network [International Conference on Sensing Technology, 2012] provided a wireless solution for indoor air quality monitoring that measures the environmental parameters like temperature, humidity, gaseous pollutants, aerosol and particulate matter to determine the indoor air quality.

9. Health & Lifestyle

IoT applications in smart health & lifestyle:

a). Health & Fitness Monitoring

b). Wearable Electronics

a). Health & Fitness Monitoring

- Wearable IoT devices allow to continuous monitoring of physiological parameters such as blood pressure, heart rate, body temperature, etc than can help in continuous health and fitness monitoring.
- It can analyze the collected health-care data to determine any health conditions or anomalies.
- The wearable devices may can be in various form such as:
  - Belts

- Wrist-bands

- Papers:
  - Toward ubiquitous mobility solutions for body sensor network health care [IEEE Communications Magazine, 2012] - Proposed an ubiquitous mobility approach for body sensor network in health-care
  - A wireless sensor network compatible wearable u-healthcare monitoring system using integrated ECG, accelerometer and SpO2 [International Conference of the IEEE Engineering in Medicine and Biology Society, 2008]- Designed a wearable ubiquitous health-care monitoring system that uses integrated electrocardiogram (ECG), accelerometer and oxygen saturation (SpO2) sensors.

b). Wearable Electronics

- Wearable electronics such as wearable gadgets (smart watch, smart glasses, wristbands, etc) provide various functions and features to assist us in our daily activities and making us lead healthy lifestyles.
- Using the smart watch, the users can search the internet, play audio/video files, make calls, play games, etc.
- Smart glasses allows users to tae photos and record videos, get map directions, check flight status or search internet using voice commands
- Smart shoes can monitor the walking or running speeds and jumps with the help of embedded sensors and be paired with smart-phone to visualize the data.
- Smart wristbands can tract the daily exercise and calories burnt.

IOT VS M2M

- This article explains the basic difference between M2M and IoT. It will start from M2M and IoT basic principle along with its applications.
- M2M, or machine-to-machine, is a direct communication between devices using wired or wireless communication channels. M2M refers to the interaction of two or more devices/machines that are connected to each other. These devices capture data and share with other connected devices, creating an intelligent network of things or systems. Devices could be sensors, actuators, embedded systems or other connected elements.
- M2M technology could be present in our homes, offices, shopping malls and other places. Controlling electrical appliances like bulbs and fans using RF or Bluetooth from

your Smartphone is a simple example of M2M applications at home. Here, the electrical appliance and your Smartphone are the two machines interacting with each other.

- The Internet of Things (IoT) is the network of physical devices embedded with sensors, software and electronics, enabling these devices to communicate with each other and exchange data over a computer network. The things in the IoT refer to hardware devices uniquely identifiable through a network platform within the Internet infrastructure.

M2M versus the IoT

M2M	IoT
M2M is about direct communication between machines.	The IoT is about sensors automation and Internet platform.
It supports point-to-point communication.	It supports cloud communication.
Devices do not necessarily rely on an Internet connection.	Devices rely on an Internet connection.
M2M is mostly hardware-based technology.	The IoT is both hardware- and software-based technology.
Machines normally communicate with a single machine at a time.	Many users can access at one time over the Internet.
A device can be connected through mobile or other network.	Data delivery depends on the Internet protocol (IP) network.

- However, there is a lot of confusion between the IoT and M2M, as both refer to communicating and sharing data. M2M is about machines, smart phones and appliances, whereas the IoT is about sensors, cyber-based physical systems, Internet and so on. Some of the differences between M2M and the IoT are listed in the table.

IOT SYSTEMS MANAGEMENT

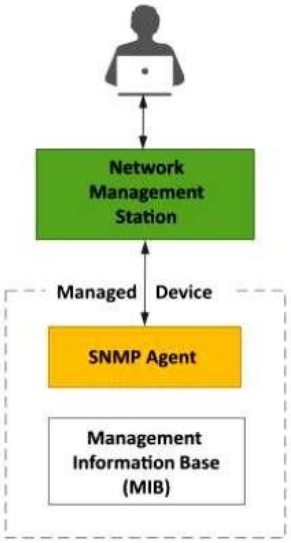
1. Need for IoT Systems Management
2. SNMP
3. Network Operator Requirements
4. NETCONF
5. YANG
6. IoT Systems Management with NETCONF-YANG

1. ***Need for IoT Systems Management:***

- Automating Configuration
- Monitoring Operational & Statistical Data
- Improved Reliability
- System Wide Configurations
- Multiple System Configurations
- Retrieving & Reusing Configurations

2. ***Simple Network Management Protocol (SNMP)***

- SNMP is a well-known and widely used network management protocol that allows monitoring and configuring network devices such as routers, switches, servers, printers, etc.
- SNMP component include
- Network Management Station (NMS)
- Managed Device
- Management Information Base (MIB)
- SNMP Agent that runs on the device



Limitations of SNMP

- SNMP is stateless in nature and each SNMP request contains all the information to process the request. The application needs to be intelligent to manage the device.

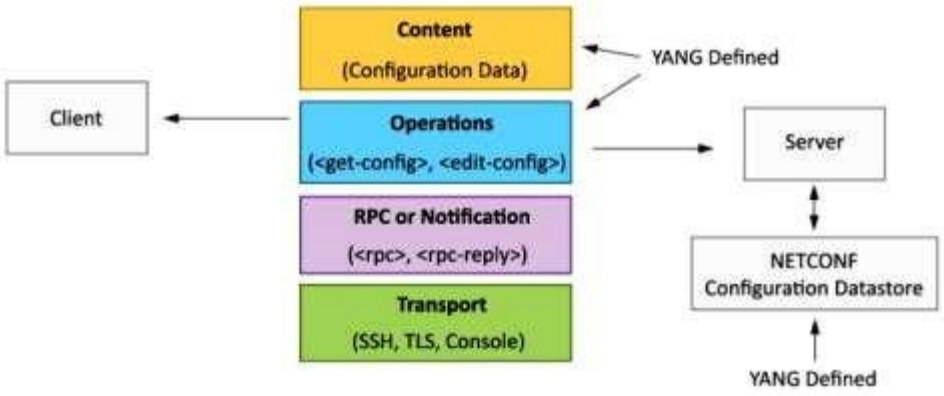
- SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.
- MIBs often lack writable objects without which device configuration is not possible using SNMP.
- It is difficult to differentiate between configuration and state data in MIBs.
- Retrieving the current configuration from a device can be difficult with SNMP.
- Earlier versions of SNMP did not have strong security features.

3. Network Operator Requirements

- Ease of use
- Distinction between configuration and state data
- Fetch configuration and state data separately
- Configuration of the network as a whole
- Configuration transactions across devices
- Configuration deltas
- Dump and restore configurations
- Configuration validation
- Configuration database schemas
- Comparing configurations
- Role-based access control
- Consistency of access control lists:
- Multiple configuration sets
- Support for both data-oriented and task oriented access control

4. Network Configuration Protocol (NETCONF)

- Network Configuration Protocol (NETCONF) is a session-based network management protocol.
- NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.



- NETCONF works on SSH transport protocol.
- Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.
- NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.
- The RPC layer provides mechanism for encoding of RPC calls and notifications.
- NETCONF provides various operations to retrieve and edit configuration data from network devices.
- The Content Layer consists of configuration and state data which is XML-encoded.
- The schema of the configuration and state data is defined in a data modeling language called YANG.
- NETCONF provides a clear separation of the configuration and state data.
- The configuration data resides within a NETCONF configuration data store on the server.

5. YANG

- YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol
- YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
- YANG modules define the data exchanged between the NETCONF client and server.
- A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.

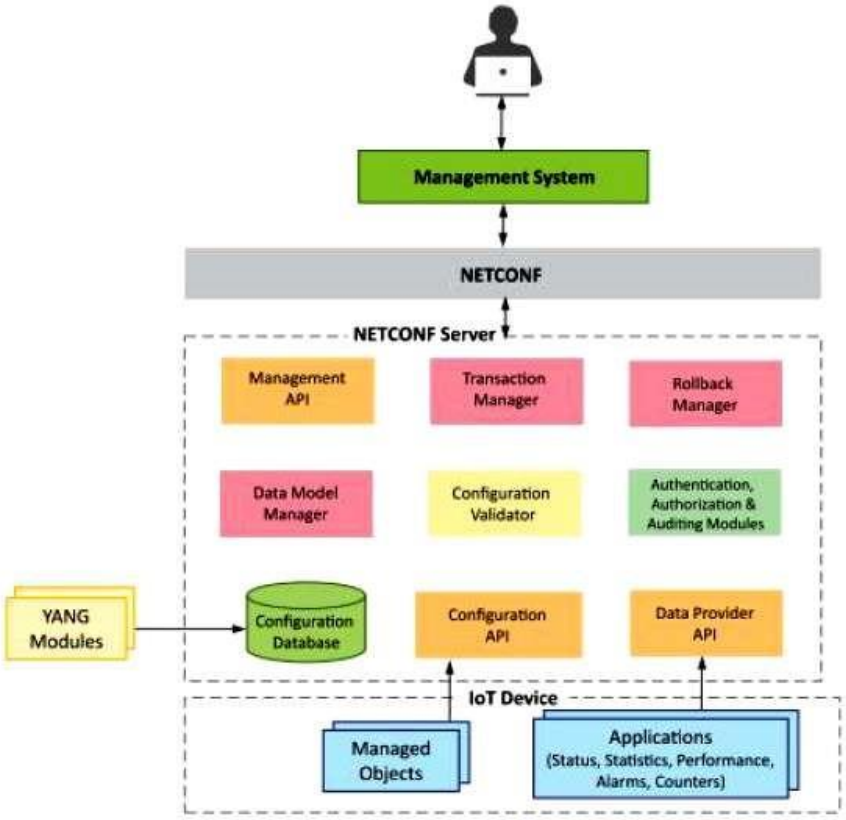
- The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.
- Leaf nodes are organized using 'container' or 'list' constructs.
- A YANG module can import definitions from other modules.
- Constraints can be defined on the data nodes, e.g. allowed values.
- YANG can model both configuration data and state data using the 'config' statement.

YANG Module Example

- This YANG module is a YANG version of the toaster MIB
- The toaster YANG module begins with the header information followed by identity declarations which define various bread types.
- The leaf nodes (‘toaster Manufacturer’, ‘toaster Model Number’ and Toaster Status’) are defined in the ‘toaster’ container.
- Each leaf node definition has a type and optionally a description and default value.
- The module has two RPC definitions (‘make-toast’ and ‘cancel-toast’).

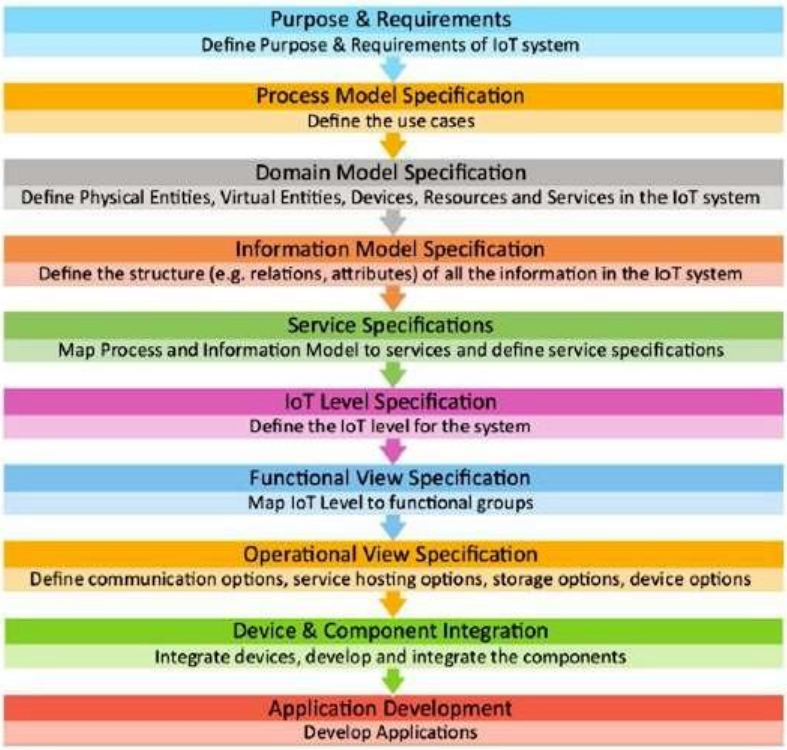
6. **IoT Systems Management with NETCONF-YANG**

- Management System
- Management API
- Transaction Manager
- Rollback Manager
- Data Model Manager
- Configuration Validator
- Configuration Database
- Configuration API
- Data Provider API



IOT DESIGN METHODOLOGY

1. IoT Design Methodology that includes:
2. Purpose & Requirements Specification
3. Process Specification
4. Domain Model Specification
5. Information Model Specification
6. Service Specifications
7. IoT Level Specification
8. Functional View Specification
9. Operational View Specification
10. Device & Component Integration
11. Application Development



Step 1: Purpose & Requirements Specification

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

Step 2: Process Specification

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

Step 3: Domain Model Specification

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

Step 4: Information Model Specification

- The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored. To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations.

Step 5: Service Specifications

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

Step 6: IoT Level Specification

- The sixth step in the IoT design methodology is to define the IoT level for the system. In Chapter-1, we defined five IoT deployment levels.

Step 7: Functional View Specification

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

Step 8: Operational View Specification

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

Step 9: Device & Component Integration

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

Step 10: Application Development

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

SPECIFICATIONS INTEGRATION AND APPLICATION DEVELOPMENT.

Requirements and Specifications

- Smart IoT services demand careful requirements capturing and specification development
- A comprehensive description of an IoT service and/or its elements is needed to support the development and verification process. 7layers supports these processes with formal description techniques.

Requirements capturing and requirements specification

- Requirements capturing is the first step in a requirements engineering process. The description of a capability or characteristic that provides value to a user or other stakeholder in an IoT Services process has been defined as a "requirement", whereas a set of specific requirements is called a "requirements specification". Requirements specifications are used as input into the design stage of an IoT process. Goal is to achieve a complete, valid and processable description of an IoT Service and/or its elements. To establish a complete requirements specification, 7layers performs the following activities:
- Initial requirements capturing and elicitation
    - For this purpose we interview IoT Services stakeholders and potential users about their demands, business cases, user stories etc. We also analyze conceptual papers, feasibility studies, already existing product or services descriptions from various sources.
- Requirements classification
    - Once the requirements have been captured, they are classified according to architectural / design requirements, functional and non-functional requirements.
- Structuring of requirements
    - Requirements are structured according to characteristics such as hardware, software, communications, interfaces, security, electrical mechanical etc.
- Description and documentation of IoT process requirements
    - After classification and structuring, the requirements are described using formal description techniques. Especially for processes as complex as IoT Services set-ups, the requirements should be documented in a system that allows for continuous requirements management.

IoT Services specification

- An appropriate subset of the established requirements will be used to define the basic IoT Service. In some cases additional aspects like environmental or legal requirements, or functional and design aspects may be included in the services specification. All specifications must be documented in a complete, consistent, correct, unambiguous and testable way.

Requirements and specification management

- Naturally, the requirements and specifications of the many elements an IoT Service is made of will evolve during its lifecycle. Initially this is due to the verification processes during the development, implementation and release phases. Once released however, an IoT Service and its elements have to be monitored continuously and adapted to aspects such as market, regulatory, technical or business developments. For this purpose, the feedback from various IoT Services stakeholders has to be documented, analyzed, priorities and agreed upon.
- 7layers support these processes by documenting and analyzing change requests, setting-up joint requirements development sessions, establishing a version management process for requirements and a release management process for requirements specifications.
- The more complex a product or service is, the more important it becomes to use proven software systems to support the lifecycle management processes of requirements and specifications management. Suitable systems are for example Interlab EVO, Rational DOORS, and HP Quality Center etc. At 7layers we take great care, that feedback arising from testing and verification processes can be traced back exactly to their respective requirements, thus easing the lifecycle management process considerably and helping improve the overall quality of service.

APPLICATION DEVELOPMENT

- We develop IoT applications for software product companies including those engaged with the development of SaaS applications and smart products, as well as design, develop and implement IoT-capable enterprise applications.

Needs analysis and requirements gathering

- If you have detailed requirements to an IoT application's functionality, we will closely analyze the requirements specification, define the project scope and complexity, and provide a delivery schedule.
- In case you have a vision of an IoT application and want to turn your idea into a ready-to-use IoT solution, we carry out a feasibility study and analyze legal, regulatory and other requirements to create a well-defined IoT application specification.

IoT application development

- We can implement an IoT application at once according to a Waterfall methodology. Alternatively, we can turn to Agile launching a minimum viable product or a first application version with a lean set of high-priority features within 3-6 months and releasing new features every 2-4 weeks.

UI design for web and mobile apps

- Our UI/UX designers wrap robust application functionality into a user-friendly and appealing interface to provide IoT application users with a convenient way to access IoT insights.

Integration

- For software product companies, we design, model, and implement APIs to allow you reliably integrate IoT functionality into your software product.
- For enterprises, our engineering team makes sure that the developed IoT application seamlessly integrates with relevant external systems, such as ERP, MES, OMS, and others

Quality assurance

- We place a premium on applications' quality from the very start of the project and focus on prevention and early detection of defects.
- At the application design stage, we conduct comprehensive verification of requirements and application architecture.
- At the development stage, we ensure quality with a well-structured and well-documented code, and perform regular code reviews and unit testing.
- At the testing stage, we resort to risk-based testing and pay special attention to critical application functionality, perform exploratory testing to detect non-obvious application defects, as well as employ a relevant degree of automation to increase test coverage and eliminate regression errors.

Security testing

- With 15 years in cyber security, we incorporate security as early as at the application design phase and conduct vulnerability assessment and penetration testing for all major releases.

User training

- For enterprise clients, we provide comprehensive training including step-by-step guides, onsite training, and workshops before an application or an application build goes live in an enterprise to help your employees get used to the new processes and ensure high user adoption.

IoT application launch

- For enterprises, we release an IoT application or its version into the existing enterprise IT environment.

After-launch support and application evolution

- We will carry out IoT application monitoring and performance management, proactively diagnose and fix application defects, and perform daily administration tasks including security updates, user and access management to make sure your IoT-capable product or enterprise IoT application runs without a hitch and stays up-to-date with your business needs.

Knowledge transfer

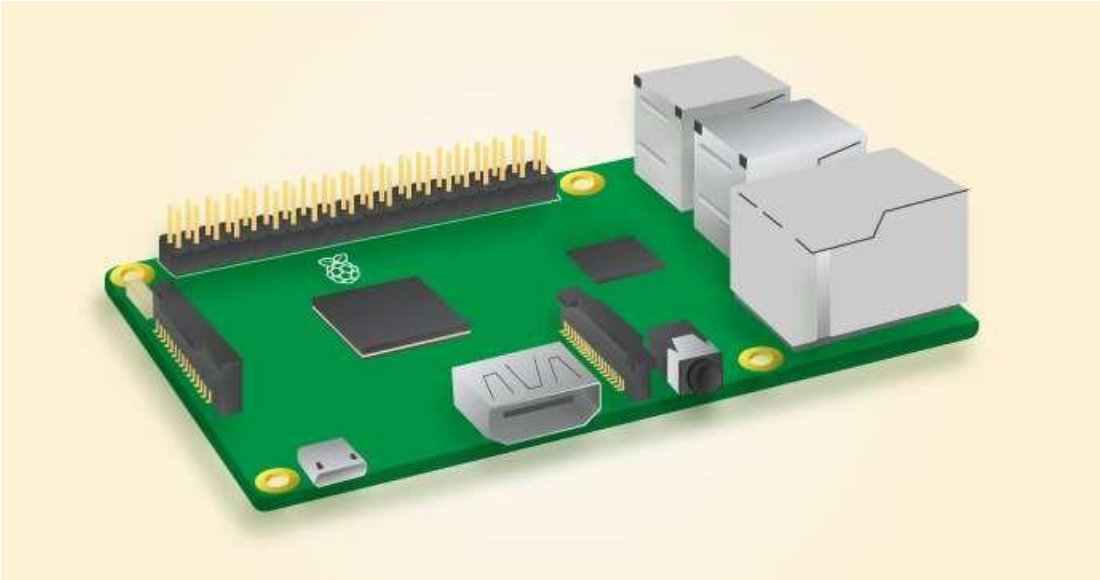
- We are ready to deliver a comprehensive knowledge transfer plan and share our expertise with your IT team to enable you manage and further develop an IoT application with in-house resources.

UNIT-4

PHYSICAL DEVICE RASPBERRY PI

The Raspberry Pi Computer:

- It may not look much like the computers you are used to, but this is because we are accustomed to seeing a computer in a case, with a monitor, keyboard and mouse attached.
- The Raspberry Pi comes without any of these peripheral input, output and storage devices. It is known as a single board computer, and the fact that it comes without any additional peripherals, and uses hardware components more usually found in mobile phones and tablets, means it can be sold for as little as $35.



- The Raspberry Pi has several ports that enable you to connect a variety of devices.
- Input devices let you send data to a computer. The two most common input devices are a keyboard and a mouse. You can plug a USB keyboard and mouse into two of the four USB ports on the Raspberry Pi.
- Output devices let the computer send data to a user. Two of the most common output devices are a monitor and speakers. You can connect an HDMI monitor or television to the Raspberry Pi using the single HDMI port. If you don't have an HDMI monitor, then you'll need to use an adaptor. You can connect speakers or headphones to the Raspberry Pi using the 3.5mm headphone port.
- Storage devices are used to store data. On most computers this would be handled by a hard drive. Most modern computers, tablets and mobile phones now use Solid State

storage devices. The Raspberry Pi uses a type of Solid State storage device called a microSD card. This will be used to store the Operating System, your software, and all the files you create.

- The last thing you'll need to do is provide your Raspberry Pi with power. For this, we use a micro-USB power supply.

RASPBERRY PI INTERFACES

- There are many peripherals that can be added to a microprocessor over the I2C and SPI serial interfaces. These include atmospheric sensors, EEPROMS, and several types of display.

The Pi Wedge helps access the I2C and SPI signals.

- This tutorial will walk you through getting the I2C and SPI interfaces of your Raspberry Pi working. These interfaces aren't enabled by default, and need some extra configuration before you can use them.

Background & Software Setup

- The Raspberry Pi has three types of serial interface on the GPIO header. You're probably already familiar with the UART serial port, which allows you to open a login session from a serial terminal application, such as PuTTY.
- The other two serial interfaces are the Serial Peripheral Interface (SPI) and Inter-Integrated-Circuit bus (I2C). SPI on the Pi allows for up to two attached devices, while I2C potentially allows for many devices, as long as their addresses don't conflict.

Software Details

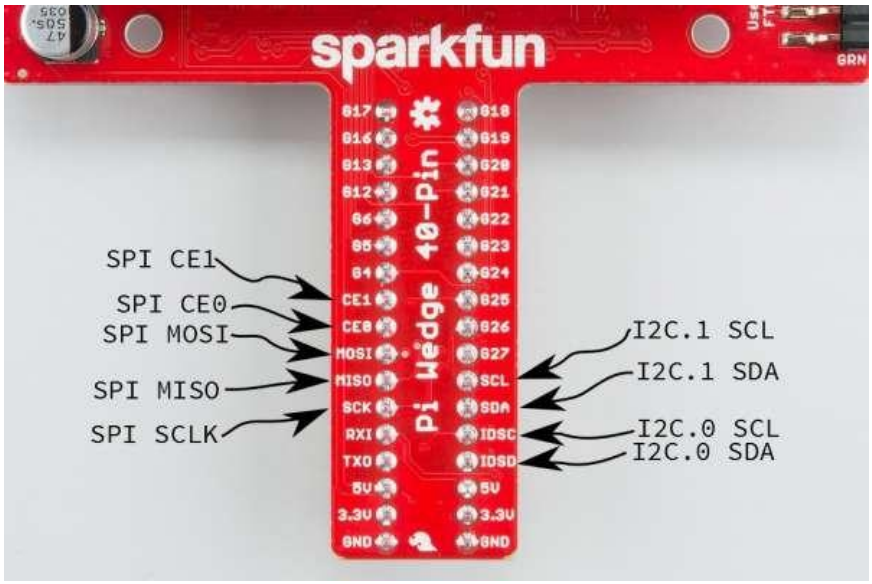
- The software landscape for the Raspberry Pi has evolved considerably since the introduction of the Pi. Many different operating systems have been ported to the Pi, and the device driver infrastructure has also changed quite a bit.
- For this tutorial, we'll be using a recent version of Raspbian (installed via NOOBS), and the wiringPi I/O library for C/C++ (or spidev/smbus for Python).
- With the implementation of device tree overlays in Raspbian, some of the specific interface enablement details have changed. If you're working with an older install, it might be worth backing up your SD card, and starting with a fresh install.

Connecting To the Ports

- Before we get into the configuration and software examples, let's locate the pins used by each of these interfaces.
- If you're directly connecting to the pins on the Pi, they're a little disorganized. I2C.1 is near one end, while SPI and I2C.0 are in the middle of the header. If you're connecting to these pins, be sure to count carefully.

Pi Serial Bus Pins

- The Pi Wedge adapter PCB rearranges the pins, and labels them clearly. We'll be using the Wedge for the following examples.



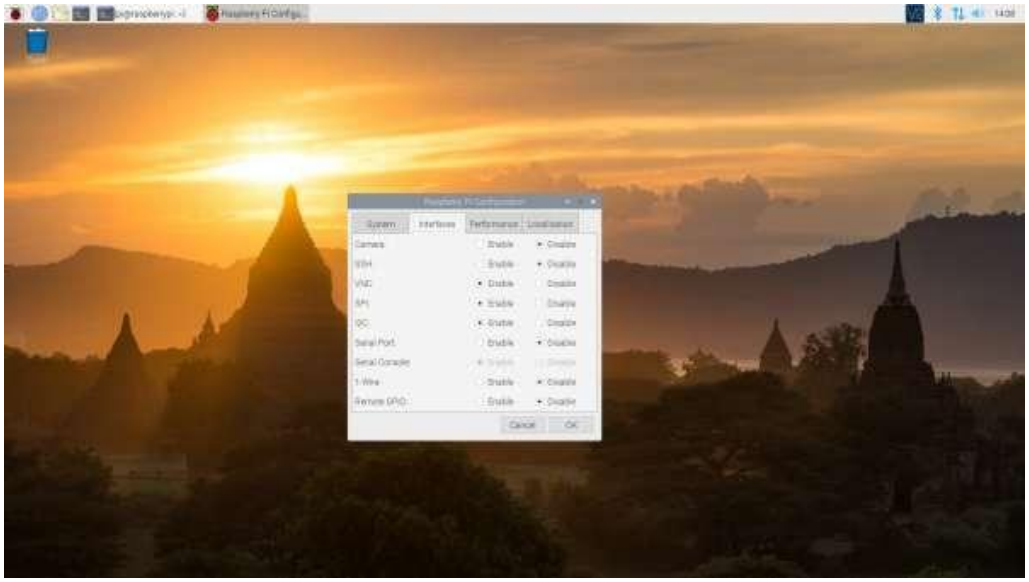
Wedge Serial Bus Pins

SPI on Pi

Configuration

- The SPI peripheral is not turned on by default. There are two methods to adjust the settings. To enable it, do the following.
- Raspberry Pi Configuration via Desktop GUI
- You can use the Desktop GUI by heading to the Pi Start Menu > Preferences > Raspberry Pi Configuration.

- A window will pop up with different tabs to adjust settings. What we are interested is the Interfaces tab. Click on the tab and select Enable for SPI. At this point, you can enable additional interfaces depending on your project needs. Click on the OK button to same.



- We recommend restarting your Pi to ensure that the changes to take effect. Click on the Pi Start Menu > Preferences > Shutdown. Since we just need to restart, click on the Restart button.

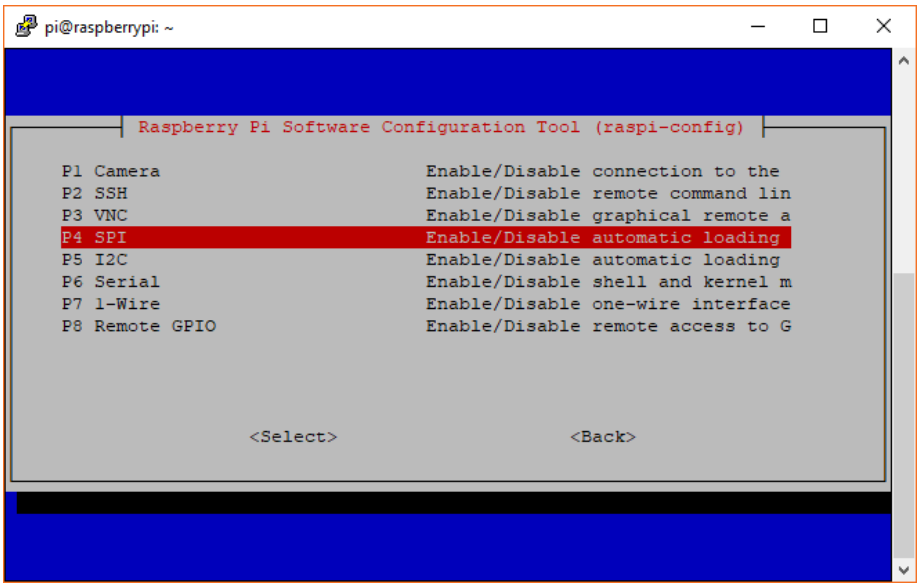
Shutdown

Turn Off, Restart, Log Off

raspi-config Tool

- If you are using a terminal, you will need to:
  1. Run sudo raspi-config.
  2. Use the down arrow to select 5 Interfacing Options
  3. Arrow down to P4 SPI.

4. Select yes when it asks you to enable SPI,
5. Also select yes if it asks about automatically loading the kernel module.
6. Use the right arrow to select the <Finish> button.
7. Select yes when it asks to reboot.



Raspi-config for SPI

- The system will reboot. When it comes back up, log in and enter the following command
  - **>ls /dev/*spi***
- The Pi should respond with
  - **/dev/spidev0.0 /dev/spidev0.1**
- These represent SPI devices on chip enable pins 0 and 1, respectively. These pins are hardwired within the Pi. Ordinarily, this means the interface supports at most two peripherals, but there are cases where multiple devices can be daisy-chained, sharing a single chip enable signal.

Programming Example

- Required Materials
  - The 40-pin Pi Wedge.
  - A Raspberry Pi B+ or Pi 2 Model B single board computer.
  - A Solderless Breadboard.
  - Some jumper wires.

- Headers of your choice.
- A Serial 7-Segment display.

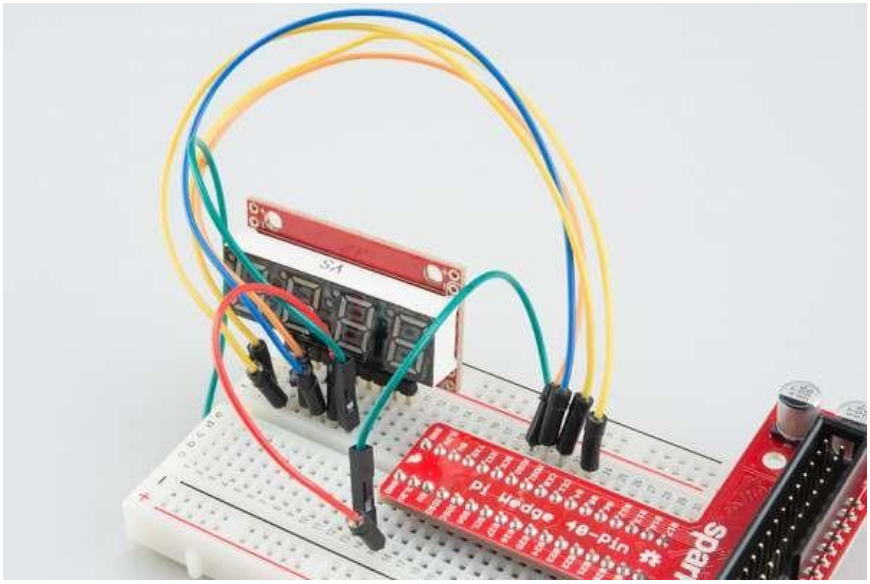
- The Serial 7-Segment display is particularly useful for testing serial interfaces, because it can accept command from a UART, SPI, or I2C. Make sure to solder header pins on the 7-segment display before wiring.

Hookup Table

- The display was connected to the Pi, via the Pi Wedge, as follows.

Raspberry Pi Signal	Serial 7-seg Signal
GND	GND
3.3V	VCC
CE1	SS (Shift Select)
SCK	SCK
MOSI	SDI
MISO	SDO

- The test hardware looked like this.



Serial 7-Segment connections for SPI

Sample Python Program

```
# spitest.py
# A brief demonstration of the Raspberry Pi SPI interface, using the Sparkfun
# Pi Wedge breakout board and a SparkFun Serial 7 Segment display:
# https://www.sparkfun.com/products/11629
import time
import spidev
# We only have SPI bus 0 available to us on the Pi
bus = 0
#Device is the chip select pin. Set to 0 or 1, depending on the connections
device = 1
# Enable SPI
spi = spidev.SpiDev()
# Open a connection to a specific bus and device (chip select pin)
spi.open(bus, device)
# Set SPI speed and mode
spi.max_speed_hz = 500000
spi.mode =  0
# Clear display
msg = [0x76]
spi.xfer2(msg)
time.sleep(5)
# Turn on one segment of each character to show that we can
# address all of the segments
i = 1
while i < 0x7f:
   # The decimals, colon and apostrophe dots
   msg = [0x77]
```

```
    msg.append(i)
    result = spi.xfer2(msg)
    # The first character
    msg = [0x7b]
    msg.append(i)
    result = spi.xfer2(msg)
    # The second character
    msg = [0x7c]
    msg.append(i)
    result = spi.xfer2(msg)
    # The third character
    msg = [0x7d]
    msg.append(i)
    result = spi.xfer2(msg)
    # The last character
    msg = [0x7e]
    msg.append(i)
    result = spi.xfer2(msg)
    # Increment to next segment in each character
    i <<= 1
    # Pause so we can see them
    time.sleep(5)
# Clear display again
msg = [0x76]
spi.xfer2(msg)
```

- Save the program with a name like spitest.py, and run it with:
    - **> python spitest.py**

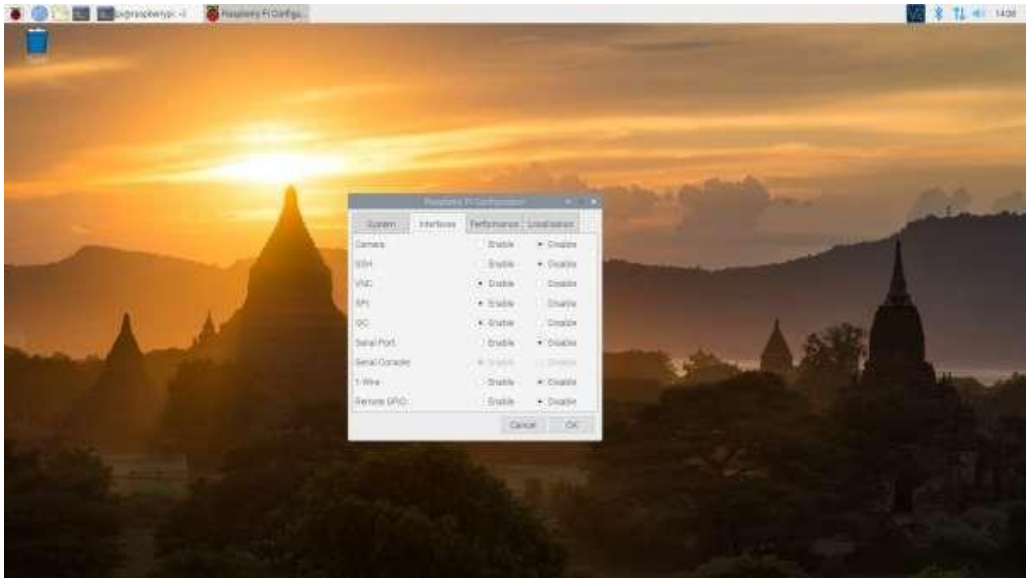
- This will illuminate each segment in each character for 5 seconds before moving on to the next segment. It should take about 40 seconds for the whole program to run.

I2C on Pi

Configuration

- The I2C peripheral is not turned on by default. There are two methods to adjust the settings just like the SPI. To enable it, do the following.
- Raspberry Pi Configuration via Desktop GUI
- You can use the Desktop GUI by heading to the Pi Start Menu > Preferences > Raspberry Pi Configuration.

- A window will pop up with different tabs to adjust settings. What we are interested is the Interfaces tab. Click on the tab and select Enable for I2C. At this point, you can enable additional interfaces depending on your project needs. Click on the OK button to same.



- Click on image for a closer view.
- We recommend restarting your Pi to ensure that the changes to take effect. Click on the Pi Start Menu > Preferences > Shutdown. Since we just need to restart, click on the Restart button.

Shutdown

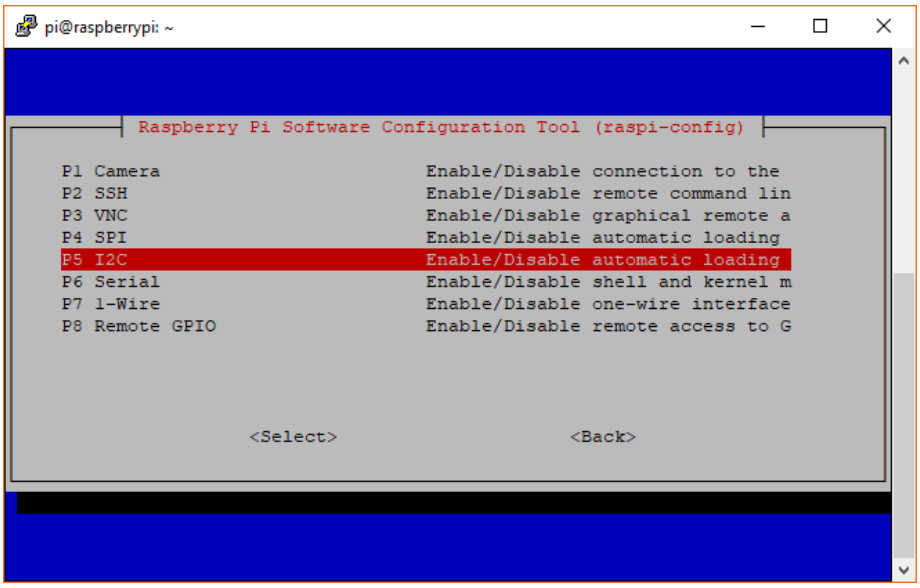
Turn Off, Restart, Log Off

raspi-config Tool

Like the SPI peripheral, I2C is not turned on by default. Again, we can use raspi-config to enable it.

1. Run sudo raspi-config.
2. Use the down arrow to select 5 Interfacing Options
3. Arrow down to P5 I2C.
4. Select yes when it asks you to enable I2C

5. Also select yes if it asks about automatically loading the kernel module.
6. Use the right arrow to select the <Finish> button.
7. Select yes when it asks to reboot.



Raspi-config for I2C

- The system will reboot. when it comes back up, log in and enter the following command
  - **>ls /dev/*i2c***
- The Pi should respond with
  - **/dev/i2c-1**
- Which represents the user-mode I2C interface.
- Utilities
- There is a set of command-line utility programs that can help get an I2C interface working. You can get them with the apt package manager.
  - **sudo apt-get install -y i2c-tools**
- In particular, the i2cdetect program will probe all the addresses on a bus, and report whether any devices are present.

pi@raspberrypi:~/$ i2cdetect -y 1

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: -- -- -- -- -- -- -- -- -- -- -- -- --

10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

70: -- -- -- -- -- -- -- --

- This map indicates that there is a peripheral at address 0x60. We can try to read and write its registers using the i2cget, i2cset and i2cdump commands.

Programming Example

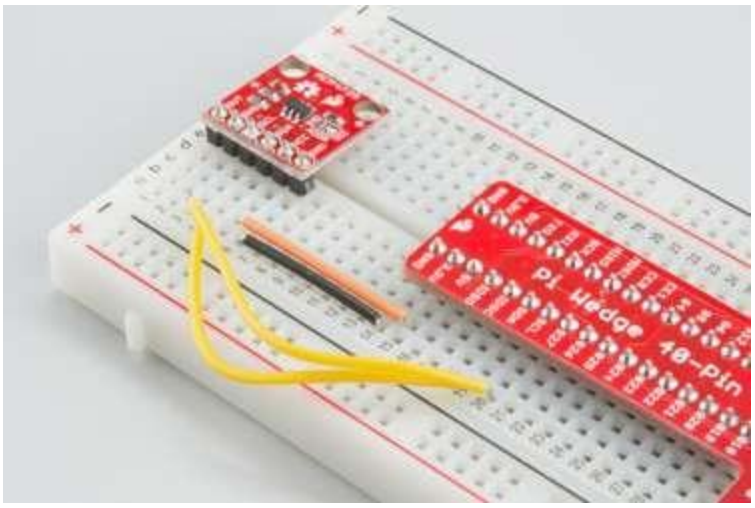
- Required Materials
- The 40-pin Pi Wedge.
- A Raspberry Pi B+ or Pi 2 Model B single board computer.
- A Solderless Breadboard.
- Some jumper wires.
- Header pins of your choice.
- An MCP4725 digital-to-analog converter.

Hookup Table

- The display was connected to the Pi, via the Pi Wedge, as follows.

Raspberry Pi Signal	MCP4725
GND	GND
3.3V	VCC
SCL	SCL
SDA	SDA

- The test hardware looked like this.



DAC on a Breadboard

Sample Python Program

```
# i2ctest.py
# A brief demonstration of the Raspberry Pi I2C interface, using the Sparkfun
# Pi Wedge breakout board and a SparkFun MCP4725 breakout board:
# https://www.sparkfun.com/products/8736
import smbus
# I2C channel 1 is connected to the GPIO pins
channel = 1
# MCP4725 defaults to address 0x60
address = 0x60
# Register addresses (with "normal mode" power-down bits)
reg_write_dac = 0x40
# Initialize I2C (SMBus)
bus = smbus.SMBus(channel)
# Create a sawtooth wave 16 times
for i in range(0x10000):
    # Create our 12-bit number representing relative voltage
    voltage = i & 0xfff
    # Shift everything left by 4 bits and separate bytes
```

msg = (voltage & 0xff0) >> 4

msg = [msg, (msg & 0xf) << 4]

 # Write out I2C command: address, reg_write_dac, msg[0], msg[1]

bus.write_i2c_block_data(address, reg_write_dac, msg)

- Save the program with a name like i2ctest.py, and run it with the command:
  - **python i2ctest.py**

APIS / PACKAGES

- Wouldn't it be interesting to have an API to control our hardware, like Arduino or Raspberry Pi? We now have the tools to make this possible and easy.
- Having an API for our hardware allows us to remotely control and/or monitor our hardware with code. With this power, we can even build apps on top of our hardware!
- We'll walk through creating an API for the Raspberry Pi Zero W to control four LEDs, each connected to its own GPIO. The same idea works for other hardware too.
- First, we'll need to set up our Raspberry Pi Zero W. Unlike typical Pi setups, we can do this without a monitor. If you need help with this, check out this tutorial.
- To begin, we'll need to make sure our device is updated and the proper libraries are installed.

Update and Install Libraries

- First, let's update our Pi:
  - **sudo apt-get upgrade && sudo apt-get update**
- Next, install the libraries we need:
  - **sudo apt-get install git python-pip python-gpiozero python-pkg-resources**

Clone the Code

- Using Git, clone the code from GitHub:
  - **git clone https://github.com/Losant/example-raspberry-pi-api.git**
- Change to the new project directory:
  - **cd example-raspberry-pi-api**
- In the folder, you'll see a file named "index.py". This is where the magic happens. You can see the contents of this file in the Code section.

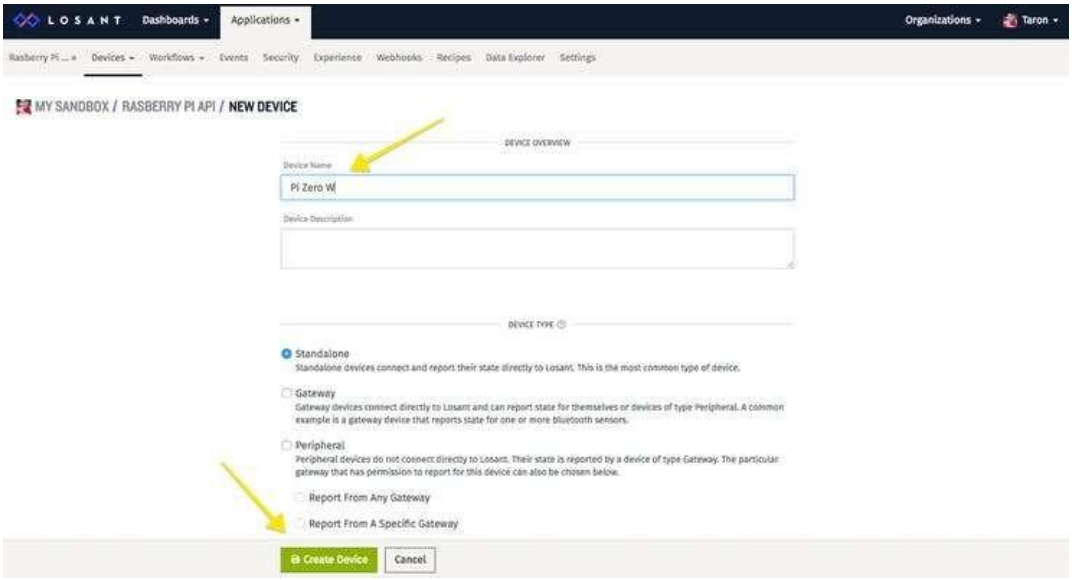
- We still need to configure one line of the code before we can run:
    - **device = Device("my-device-id", "my-app-access-key", "my-app-access-secret")**
- You can obtain the device ID, access key, and access secret from Losant.

Losant

- Losant is an easy-to-use and powerful developer platform designed to help you quickly build connected applications.

1. ***Log in and create an application in Losant.***
2. ***Create a device.***

- In Losant, a device could be Raspberry Pi, Arduino, smart bulb, or any custom hardware. Devices can contain many sensors or attached peripherals.



Create a device in Losant

3. ***Create an Access Key and Secret.***

- To connect your devices to the Losant Platform, you must use a set of security credentials called access keys. Access keys consist of a generated key and secret pair.

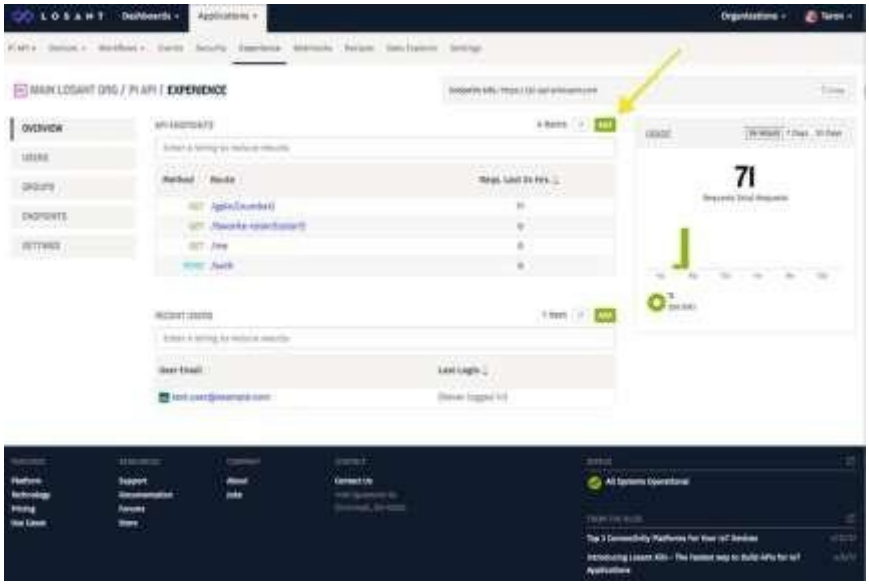
4. Create a Losant Experience.

- Writing an API service, implementing the entire user authentication, and hosting the result somewhere is a lot of work. An Experience in Losant brings all of this functionality directly inside your Losant application.
- Experiences are the key to building fully functional API allowing users to interact with our device.

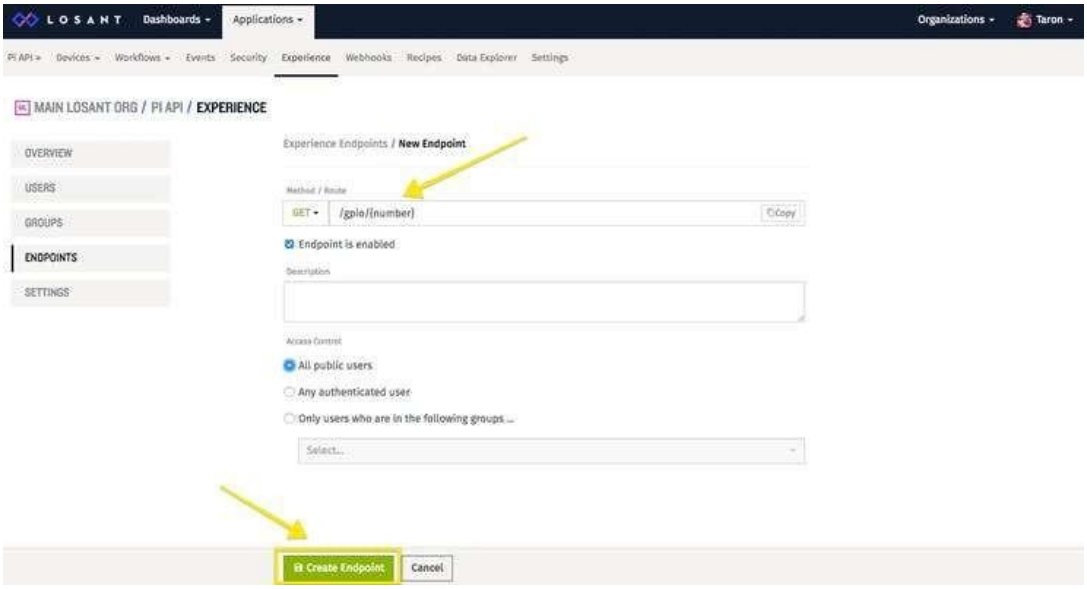
- After creation, a lot happens. We created a Walkthrough Video that walks you through the process and gives some background to what's going on.

5. Create an Endpoint

- An Endpoint is a combination of an HTTP method and a route that, when invoked by an HTTP request, can fire a workflow. That workflow does some work, like control an LED, and responds to the request.



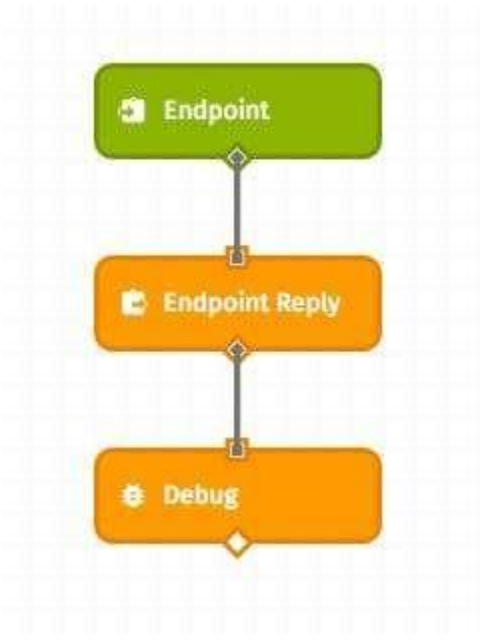
- For this example, we're going to create a GET request to toggle a GPIO pin. If the URL for this route would look something like this:
  - **https://example.onlosant.com/gpio/{number}**
- The method and route, in this case, would be:
  - **GET /gpio/{number}**
- "Number" in the route is a path parameter. We'll replace this with the GPIO we want to toggle.
- You can configure this in Losant like so:



- Note: Every endpoint has access control. To make things easy, our routes will be public. However, you are able to add authentication to endpoints.

6. Create a Workflow for the new endpoint

- Every Endpoint is powered by a Workflow. If you go down to the bottom of the endpoint configuration, you'll see the option to create a workflow for that route.
- Then, you'll have a basic workflow:

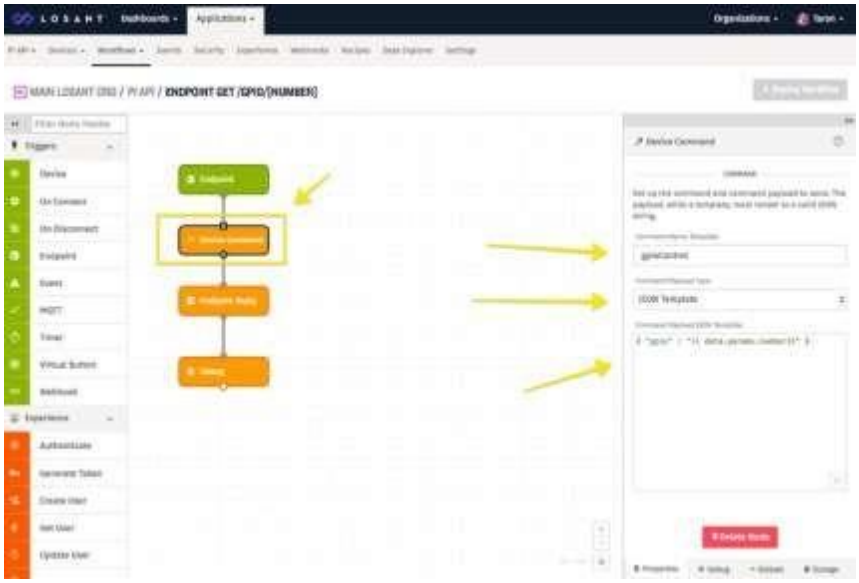


- Every endpoint workflow will start with an Endpoint Trigger and end with an Endpoint Reply. The Endpoint Trigger is fired when you make an HTTP request. The Endpoint

Reply responds to the HTTP request. So, this workflow isn't doing anything but responding immediately.

7. Update workflow to send device command.

- If you take a peek in the Python code, there is an "on_command" function that's attached to an event.
  - **device.add_event_observer("command", on_command)**
- We can trigger this function by sending a Device Command to our Pi. Drag-and-Drop the Device Command node:
- In the code, we are also listening for a command called "gpioControl" and looking for a payload variable called "gpio".
- Select the Device Command node to configure it.
  - Command Name Template: gpioControl
  - Command Payload Type: JSON Template
  - Command Payload JSON Template:
  - **{ "gpio" : "{{ data.params.number}}" }**

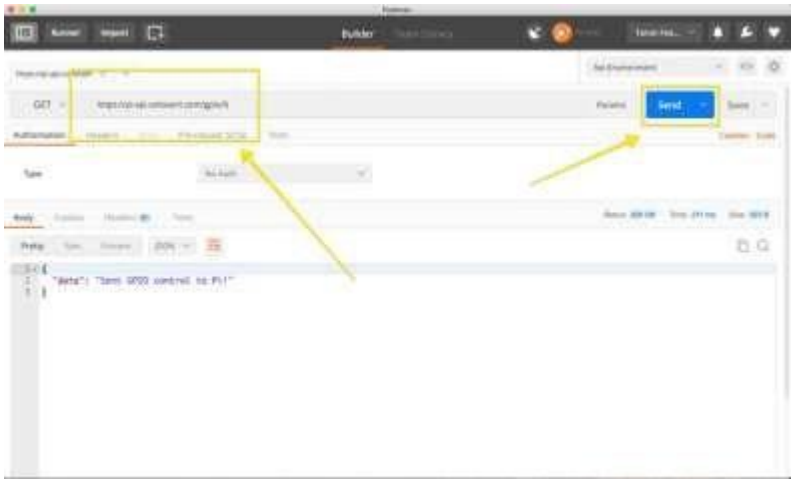


Wiring

- Now, you should wire the Raspberry Pi to the four 4 LEDs. This wiring diagram is supplied in the Wiring section.
- We'll connect the LEDs to 6, 13, 19, and 26.

Testing

- That's it! Now you are ready to test.
- To start your code:
  - **python index.py**
- To trigger an API request, we can use an app called Postman.
- Enter your URL and send the request:



- Once you fire the request, use 6, 13, 19, or 26 as the parameter and watch that LED light up!

WEB SERVICES

Various web servers are available, with different advantages for usage:

1. Apache
2. NGINX

1. *Apache*

- Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages.
- On its own, Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

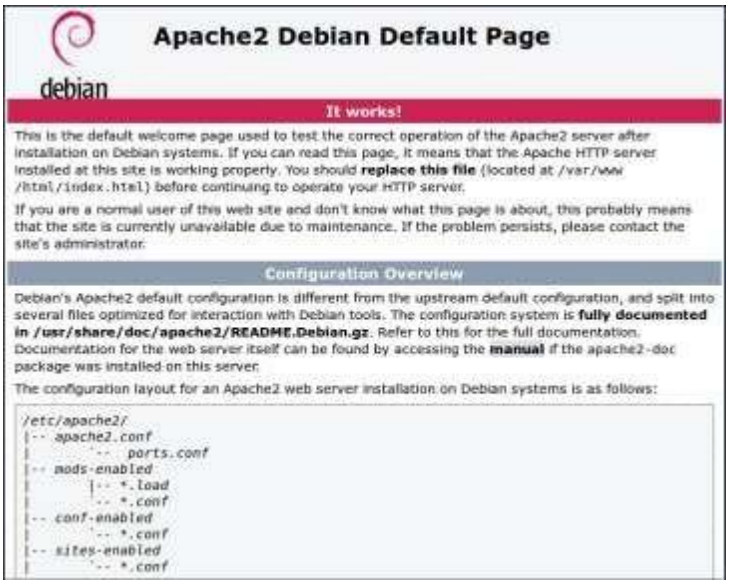
Install Apache

- First, update the available packages by typing the following command into the Terminal:
  - **sudo apt update**

- Then, install the apache2 package with this command:
    - **sudo apt install apache2 -y**

Test the web server

- By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to http://localhost/ on the Pi itself, or http://192.168.1.10 (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type hostname -I at the command line (or read more about finding your IP address).
- Browse to the default web page either on the Pi or from another computer on the network and you should see the following:



- This means you have Apache working!

Changing the default web page

- This default web page is just an HTML file on the file system. It is located at
    - **/var/www/html/index.html.**
- Navigate to this directory in a terminal window and have a look at what's inside:
    - **cd /var/www/html**
    - **ls -al**

This will show you:

- **total 12**
- **drwxr-xr-x 2 root root 4096 Jan 8 01:29 .**

- **drwxr-xr-x 12 root root 4096 Jan 8 01:28 ..**
- **-rw-r--r-- 1 root root 177 Jan 8 01:29 index.html**

- This shows that by default there is one file in /var/www/html/ called index.htmland it is owned by the root user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default pi user is assumed here) using sudo chown pi: index.html.
- You can now try editing this file and then refreshing the browser to see the web page change.

Your own website

- If you know HTML you can put your own HTML files and other assets in this directory and serve them as a website on your local network.

Additional - install PHP

- To allow your Apache server to process PHP files, you'll need to install the latest version of PHP and the PHP module for Apache. Type the following command to install these:
  - **sudo apt install php libapache2-mod-php -y**
- Now remove the index.html file:
  - **sudo rm index.html**
- and create the file index.php:
  - **sudo nano index.php**
- Put some PHP content in it:
  - **<?php echo "hello world"; ?>**
- Now save and refresh your browser. You should see "hello world". This is not dynamic but still served by PHP. Try something dynamic:
  - **<?php echo date('Y-m-d H:i:s'); ?>**
- or show your PHP info:
  - **<?php phpinfo(); ?>**

2. NGINX

- NGINX (pronounced engine x) is a popular lightweight web server application you can install on the Raspberry Pi to allow it to serve web pages.

- Like Apache, NGINX can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

Refresh database of available packages

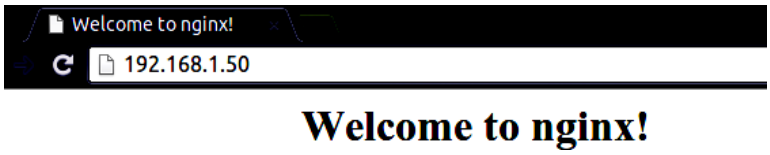
- Ensure that the package manager has up-to-date information about which packages are available:
  - **sudo apt update**
- You only need to do this occasionally, but it's the most likely solution if subsequent steps fail with messages like:
  - **404 Not Found [IP: 93.93.128.193 80]**

Install NGINX

- First install the nginx package by typing the following command in to the Terminal:
  - sudo apt install nginx
- and start the server with:
  - sudo /etc/init.d/nginx start

Test the web server

- By default, NGINX puts a test HTML file in the web folder. This default web page is served when you browse to http://localhost/ on the Pi itself, or http://192.168.1.10 (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type hostname -I at the command line (or read more about finding your IP address).
- Browse to the default web page either on the Pi or from another computer on the network and you should see the following:



Changing the default web page

- NGINX defaults its web page location to /var/www/html on Raspbian. Navigate to this folder and edit or replace index.nginx-debian.html as you like. You can confirm the default page location at /etc/nginx/sites-available on the line which starts with 'root', should you need to.

Additional - Install PHP

sudo apt install php-fpm

- Enable PHP in NGINX

  cd /etc/nginx

  sudo nano sites-enabled/default

- find the line

  index index.html index.htm;

- roughly around line 25 (Press CTRL + C in nano to see the current line number)
- Add index.php after index to look like this:

  index index.php index.html index.htm;

- Scroll down until you find a section with the following content:

  # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

  #

  # location ~ \.php$ {

- Edit by removing the # characters on the following lines:

  location ~ \.php$ {

  include snippets/fastcgi-php.conf;

  fastcgi_pass unix:/var/run/php5-fpm.sock;

  }

- It should look like this:

  # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

  #

  location ~ \.php$ {

  include snippets/fastcgi-php.conf;

  # With php-fpm (or other unix sockets):

  fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;

  # With php-cgi (or other tcp sockets):

  # fastcgi_pass 127.0.0.1:9000;

}

- Reload the configuration file

  sudo /etc/init.d/nginx reload

- Test PHP
- Rename index.nginx-debian.html to index.php:

  cd /var/www/html/

  sudo mv index.nginx-debian.html index.php

- Open index.php with a text editor:

  sudo nano index.php

- Add some dynamic PHP content by replacing the current content:

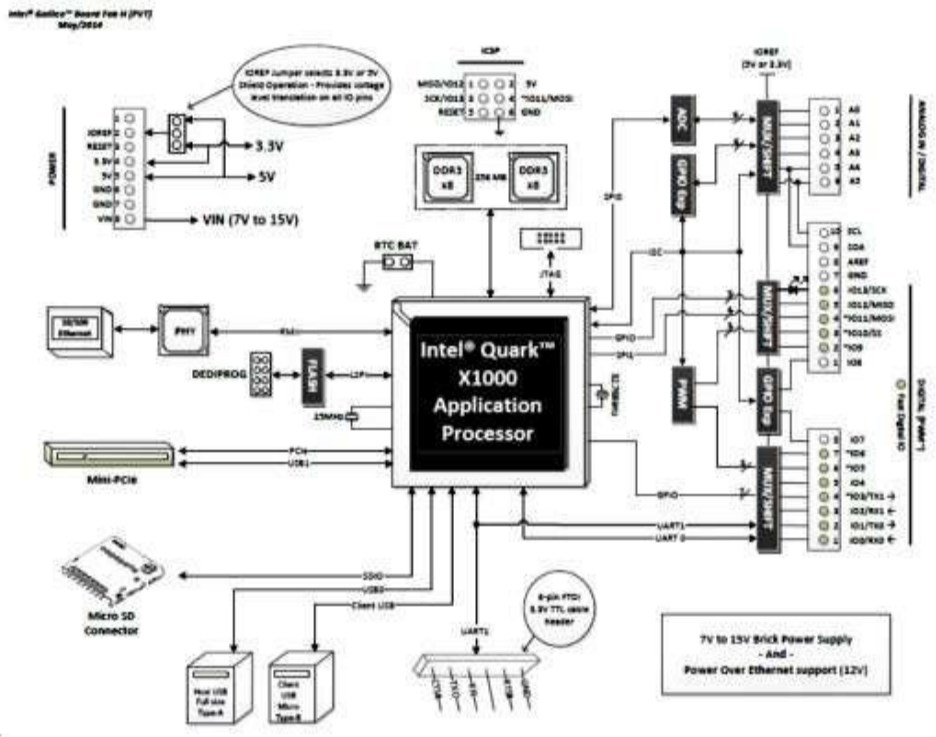
  <?php echo phpinfo(); ?>

- Save and refresh your browser. You should see a page with the PHP version, logo and current configuration settings.

INTEL® GALILEO GEN2 WITH ARDUINO

Overview

- The Intel® Galileo Gen2 is a board based on the Intel® Quark™ SoC X1000, a 32-bit Intel® Pentium® processor-class system on a chip (SoC), operating at speeds up to 400MHz.
- The Quark processor supports the Yocto 1.4 Poky Linux distribution.
- The board has built-in Ethernet with support for Power Over Ethernet(PoE), a USB 2.0 Host Port, micrso-SD slot, PCI Express mini-card slot, 20 digital input/output pins (of which 6 can be used as PWM outputs with 8/12-bits of resolution and 6 as analog inputs with 12 bits of resolution), a micro USB connection, an ICSP header, a JTAG header, and 2 reset buttons.
- There is also an integrated Real Time Clock (RTC), with an optional 3V “coin cell” battery for operation between turn on cycles.
- The Intel® Galileo Gen2 supports shields that operate at either 3.3v or 5v. The board is designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. Digital pins 0 to 13 (and the adjacent AREF and GND pins), Analog inputs 0 to 5, the power header, ICSP header, and the UART port pins (0 and 1), are all in the same locations as on the Arduino Uno R3.



Supported Arduino libraries

The software release supports the following Arduino libraries:

- SPI
- EEPROM
- UART
- GPIO
- Wi-Fi
- Servo
- USB Host

Microcontroller	SoC Quark X1000
Operating Voltage	3.3V / 5V

Input Voltage	7-15V
Digital I/O Pins	14 (of which 6 provide 8/12-bit PWM output)
Analog Input Pins	6
Flash Memory	512 kB
RAM	256 MB DDR3
SRAM	512 kB
Flash Storage	8MB
EEPROM	8kB
Clock Speed	400 MHz
PoE compatible	-
Length	124 mm
Width	72 mm

Differences between the Intel® Galileo Gen2 and the Intel® Galileo

- The earlier Intel Galileo did not have an on-board regulator, so the power supply had to be exactly 5V. The Intel Galileo Gen2 has on-board regulator, so it may be powered with any suitable supply providing 7-15 VDC.

Differences between the Arduino Yùn and the Intel® Galileo Gen2

- The Arduino Yùn has two processors: One runs Linux, and an Atmel microcontroller runs the Arduino sketch. The Intel® Galileo Gen2 has one processor. In addition to running Linux, this processor runs the Arduino sketch.

Differences between the Intel® Galileo Gen2 and all Arduinos

- Note that a sketch loaded into the Intel® Galileo Gen2 is lost after a power cycle. It is possible to boot the Intel® Galileo Gen2 from a USD card, and in that case to restore a sketch from the same card. Instructions for doing so are forthcoming.

On-board Linux

- The Yocto 1.4 Poky Linux distribution is installed on your Intel Galileo Gen2. You can access various Linux functions with the system() call.

Power

- The Intel® Galileo Gen2 can be powered only via an external power supply. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. The board can operate on an external supply of 7 to 15 volts. The provided power supply provides 12v.

The power pins are as follows:

- VIN: The input voltage to the Intel® board. You can access the voltage supplied via the power jack through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board.
- 3.3V: A 3.3 volt supply generated by the on-board regulator. This regulator also provides the power supply to the Quark microcontroller.
- GND: Ground pins.
- IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. This can be 3.3V or 5V based on the IOREF jumper position.
- 12V power-over-Ethernet (PoE) capable

Buttons

There are 2 reset buttons with different functions on the board:

- Reboot: resets the Quark X1000 processor
- Reset: resets sketch and any attached shield

Memory

- The Quark X1000 has 512kB of embedded SRAM.
- The board has an additional 256 MB DDR3 Ram amd 8MB of Flash to store firmware and the Arduino sketch. It's possible to update the firmware using the IDE. The on board uSD supports uSD card up 32G, and can be used to provide a complete Yocto 1.4 Poky Linux image.

Input and Output

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.

- Digital I/O: Digital pins 0 through 13 and Analog pins A0 through A5 can be used as a digital input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They can operate at 3.3 or 5 volts.

Each pin can provide (source) or receive (sink) a current of 16mA @ 5v, or a current of 8mA@3.3V

- PWM: Pins 3,5,6,9,10, and 11
    - Provide 8/12bit PWM output with the analogWrite() function. The resolution of the PWM can be changed with the analogWriteResolution() function.
- SPI: SPI header (ICSP header on other Arduino boards)
    - These pins support SPI communication using the SPI library.
- Analog Inputs: pins A0 through A5
    - Each analog input provide 10/12 bits of resolution. The resolution can be changed with the analogReadResolution() function.
- SDA and SCL: Support TWI communication using the Wire library.

There are 2 reset buttons with different functions on the board:

- Reboot:
    - resets the Quark X1000 processor
- Reset:
    - resets sketch and attached shield

Communication

- The Intel® Galileo Gen2 has a number of facilities for communicating with a computer, another Galileo, Arduino or other microcontrollers, and different devices like phones, tablets, cameras and so on.
- It provides 2 UART Controllers : UART 0 to Galileo headers 0, 1; UART 1 to 6-pin 3.3V USB TTL FTDI header; optionally directed to Galileo headers 2, 3 .
- The Native USB port can also act as a USB host for connected peripherals such as mice, keyboards, and smart phones. To use these features, see the USB Host reference pages. The onboard microSD card reader is accessible through the SD Library. The communication between Galileo and the SD card is provided by an integrated SD controller and does not require the use of the SPI interface like other Arduino boards.

- The onboard Ethernet interface is fully supported, use the Ethernet Library. It does not require the use of the SPI interface like existing Arduino shields.
- The Arduino software includes a Wire library to simplify use of the TWI/I2C bus; see the documentation for details. For SPI communication uses the SPI library.
- The board provides also a mini PCI Express (mPCIe) slot. This slot allows full size and half size (with adapter) mPCIe modules to be connected to the board and also provides an additional USB Host port via the slot. Any standard mPCIe module can be connected and used to provide applications such as WiFi, Bluetooth or Cellular connectivity.

Programming

- The Intel® Galileo Gen2 can be programmed with this special version of the Arduino software. It's possible to make requests of the Linux kernel with system() calls. This gives your Arduino sketch access to powerful utilities like Python, Node.js, OpenCV, and all sorts of fun Linux stuff.
- Note that Intel® Galileo Gen2 forgets the sketch after powering down. It is possible to booth the Galileo Gen2 from the uSD card, and in that case, to restore a sketch from the same card. Instructions for that will be forthcoming.

ARDUINO INTERFACES

GPS Module Interfacing With Arduino UNO

- Interfaced GPS receiver module with Arduino UNO and display the Time, Latitude, Longitude, and Altitude info on the Serial window. Arduino read the data serially from GPS receiver using USART communication with 9600 Baud rate.

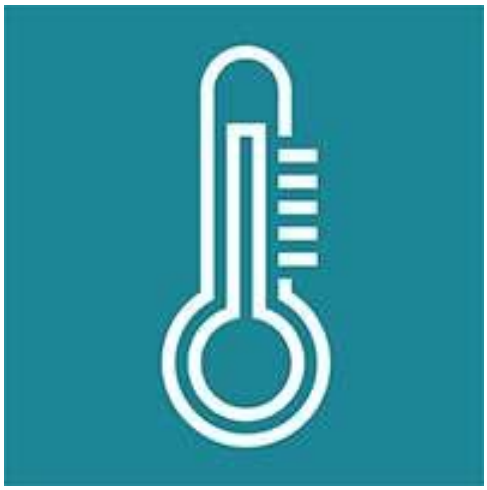
HC-05 Bluetooth Module Interfacing With Arduino UNO

- Bluetooth is a wireless communication protocol used to communicate over short distances. HC-05 Bluetooth module uses serial communication to talk with microcontrollers.

Analog Joystick Interfacing With Arduino UNO

- Analog Joystick is an input device used to control the pointer movement in 2-Dimensional axes. Generally, joystick is used for getting angular movements.

LM35 Interfacing With Arduino UNO

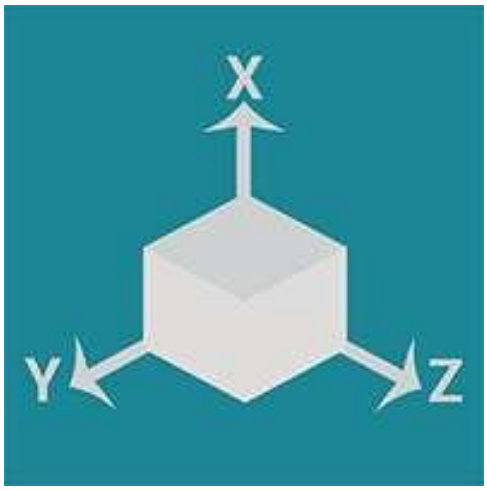


- LM35 is a sensor which is used to measure temperature. It provides an electrical output proportional to the temperature (in Celsius).

4x4 Keypad Interfacing With Arduino UNO

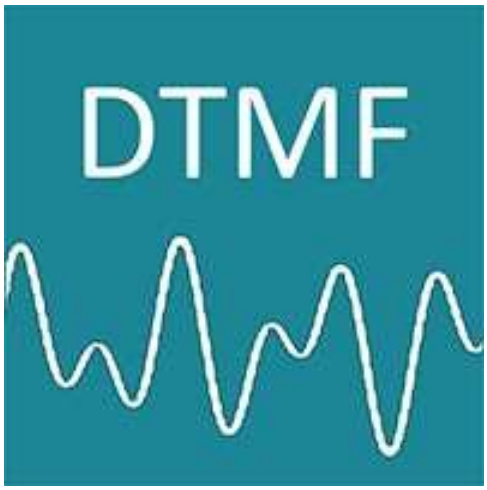
- Keypad is an input device which is generally used in applications such as calculators, ATM machines, computers etc.

ADXL335 Accelerometer Interfacing With Arduino Uno



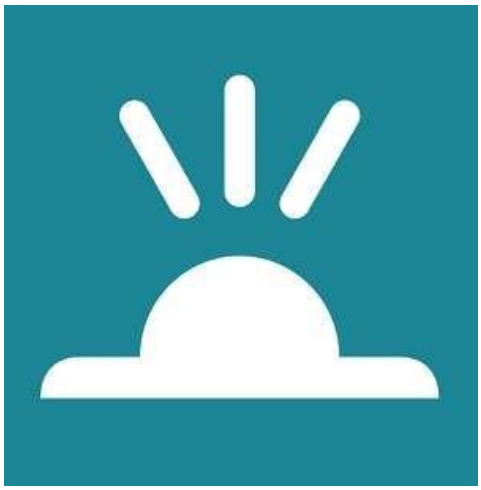
- ADXL335 accelerometer sensor measures acceleration due to gravity. It is used to measure the angle of tilt or inclination in application systems such as in Mobile devices, Gaming applications, Laptops, Digital cameras, Airplanes etc.

MT8870 DTMF Decoder Interfacing With Arduino UNO



- MT8870 is a DTMF (Dual Tone Multi-Frequency) receiver, which decodes the dial tone generated from a telephone keypad. It used in Interactive Voice Response Systems (IVRS), Remote control, Credit card systems etc.

IR Communication Using Arduino UNO



- IR (infrared) communication is a wireless communication technology, used for short distance data/control transmission. It is commonly used in TV remotes, mobile phones, computers, and PDAs etc.

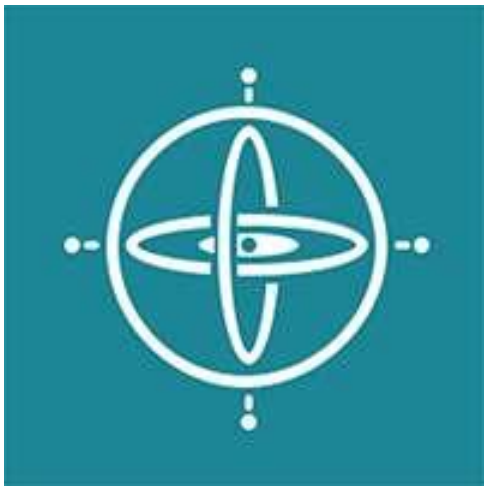
Magnetometer HMC5883L Interfacing With Arduino UNO

- HMC5883L is a triple axis magnetometer developed by Honeywell. It provides the direction of heading. Magnetometer is used as a compass in Mobiles Phones and Navigation systems in vehicles to indicate directions.

PIR Sensor Interfacing With Arduino UNO

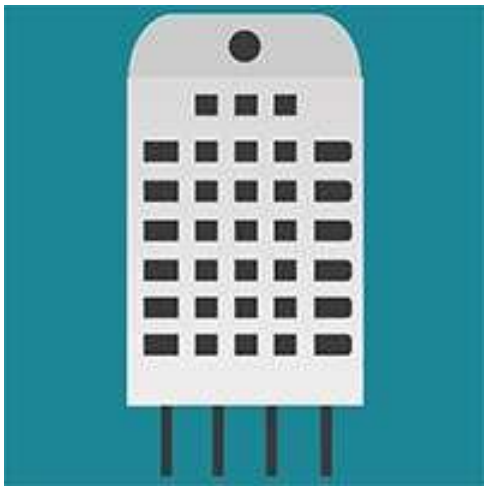
- PIR motion sensor senses Infrared signals. It is generally used to detect the motion of humans or animals.

MPU6050 Interfacing with Arduino UNO



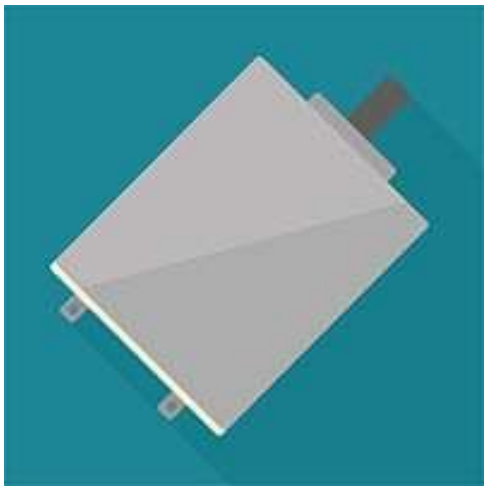
- MPU6050 sensor module is a combination of 3-axis Gyroscope, 3-axis Accelerometer and Temperature sensor with on-board Digital Motion Processor (DMP). It is used in mobile devices, motion enabled games, 3D mice, gesture (motion command) control technology etc.

DHT11 Sensor Interfacing With Arduino UNO



- Interface single wire DHT11 sensor with Arduino Uno to read the values of Temperature and Humidity. Display these Temperature and Humidity values on a serial window.

DC Motor Interfacing With Arduino UNO



- DC Motor is a device which converts electrical energy into mechanical energy. It is used in robotics field, toys, quad copters etc.

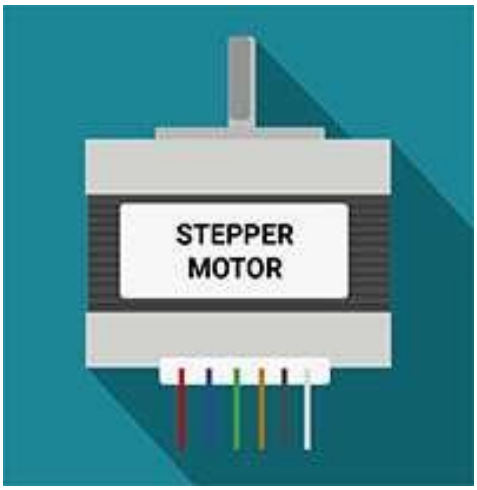
DS1307 RTC Module Interfacing With Arduino UNO

- DS1307 is a two wire (I2C) serial interface RTC (Real Time Clock) with 56 byte of nonvolatile RAM. This provides clock and calendar with second, minute, hour, day, date, month and year.

LCD 16x2 Interfacing With Arduino Uno

- LCD16x2 has two lines with 16 characters in each line. LCD16x2 is generally used for printing values and strings in embedded application

Stepper Motor Interfacing With Arduino UNO



- Rotate the Stepper Motor Clockwise and anti-clockwise by interfacing it with Arduino. Here, we are using a ULN2003 driver which is used to drive a stepper motor.

7-Segment Display Interfacing With Arduino UNO

- 7-segment LED displays are used for displaying numerical values from 0 to 9 and few characters like A, b, C, d, e, F, H, L, O, P, U etc.7-segment displays are widely used in digital clock

Servo Motor Interfacing With Arduino Uno

- Servo motor is an electromechanical device which consists of motor, gear assembly and feedback circuitry. It is used in Robotics applications, airplanes, rudders, quad copters, etc.

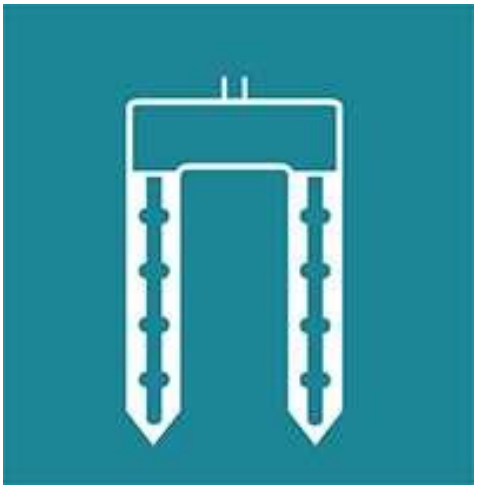
TCP Client Using SIM900A GPRS and Arduino UNO

- Transmission Control Protocol (TCP) is standard transport layer internet protocol in between server and client. Using SIM900A GPRS module, we can implement TCP server/client over GPRS for IoT applications

HTTP Client Using Sim900A GPRS And Arduino UNO

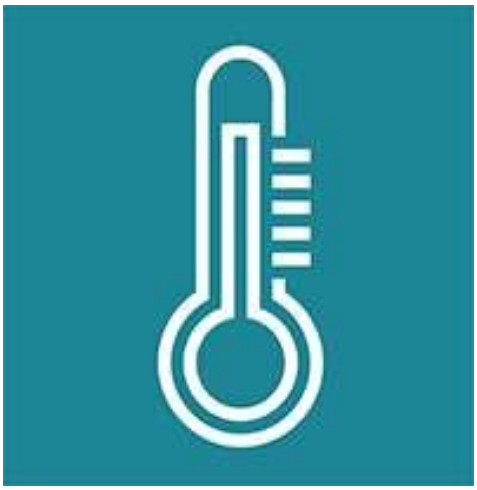
- Hypertext Transfer Protocol (HTTP) is standard application layer protocol in between server and client. Using SIM900A GPRS module, we can implement HTTP server/client over GPRS for IoT applications

Soil Moisture Sensor Interfacing With Arduino UNO



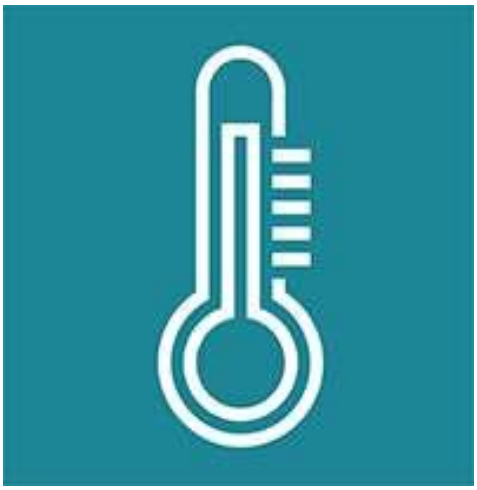
- Soil moisture sensor is used to measure the water content (moisture) in the soil. It is used in agriculture applications, irrigation and gardening systems, etc.

Thermistor Interfacing With Arduino UNO



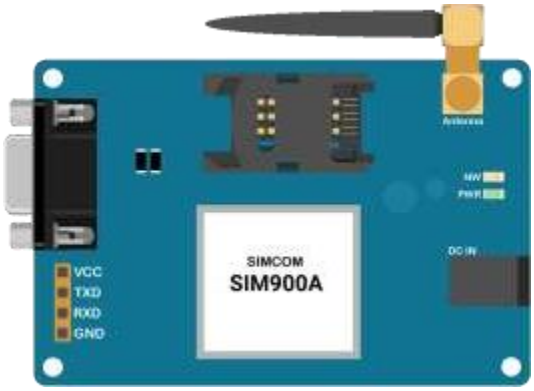
- Thermistor is a type of resistor whose resistance changes in accordance with change in temperature. It is used to measure the temperature over small range typically -100 °C to 300 °C

Thermocouple Interfacing With Arduino UNO



- Thermocouple is a device consisting of two dissimilar metals connected together creating a junction which is used for measuring the temperature

Sim900A GSM Module Interfacing With Arduino UNO

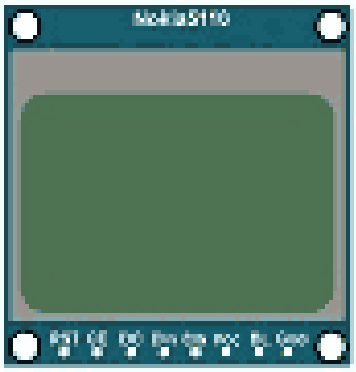


- SIM900A GSM module is a communication device which is used to make or receive calls, send or receive SMS, connect to the internet over GPRS

ESP8266 WiFi Module Interfacing With Arduino UNO

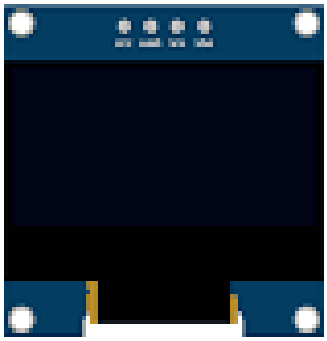
- ESP8266 is Wi-Fi enabled system on chip (SoC) module developed by Espressif system. It is mostly used for development of IoT (Internet of Things) embedded applications.

Nokia5110 Graphical Display Interfacing With Arduino UNO



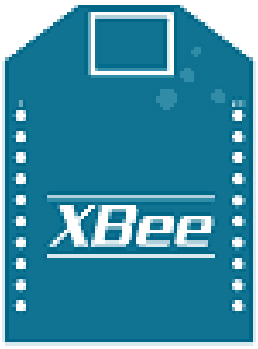
- Nokia5110 is 48x84 dot LCD display with Serial Peripheral Interface (SPI) Connectivity. It was designed for cell phones, also used in embedded applications

OLED Graphic Display Interfacing With Arduino UNO



- OLED graphic display modules are compact and have high contrast pixels which make these displays easily readable. They do not require backlight since the display creates its own light. Hence they consume less power. Both I2C and SPI based OLED modules are available in market

XBee S2 (ZigBee) Interfacing With Arduino UNO



- XBee is a radio module developed by Digi International. It is a popular wireless transceiver used to send or receive data

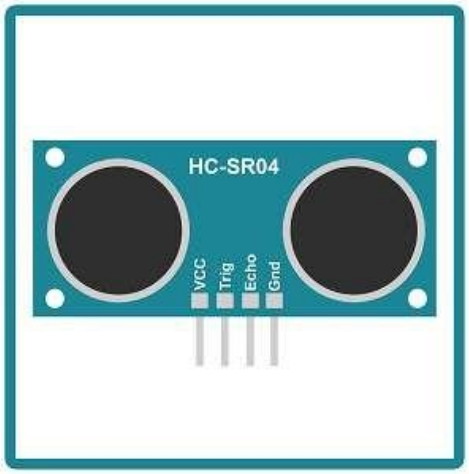
NRF24L01 Interfacing With Arduino UNO

- nRF24L01 is ultra-low power RF transreceiver radio module for 204GHz ISM band which is developed by Nordic Semiconductor.

MicroSD Card Interfacing With Arduino

- MicroSD (Secure Digital) card is an electronic digital data storage device, which provides storage in a gigabyte (GB) at low cost and small size. It is used in various applications like Mobile Phones, Laptops, Digital Camera, etc.

Ultrasonic Sensor HC-SR04 Interfacing With Arduino Uno



- Ultrasonic Sensor HC-SR04 interfaced with Arduino for finding a distance to an object. It can operate in the range of 2cm-400cm.

ARDUINO IDE

- The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

1. Writing Sketches

- Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.
- NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.
    - ***Verify:*** Checks your code for errors compiling it.
    - ***Upload:*** Compiles your code and uploads it to the configured board. See uploading below for details.
    - ***Note:*** If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"
    - ***New:*** Creates a new sketch.
    - ***Open:*** Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.
    - ***Note:*** due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.
    - ***Save:*** Saves your sketch.
    - ***Serial:*** Monitor Opens the serial monitor.
- Additional commands are found within the five menus: File, Edit, Sketch, Tools, and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

2. File

- ***New:*** Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- ***Open:*** Allows loading a sketch file browsing through the computer drives and folders.
- ***Open Recent:*** Provides a short list of the most recent sketches, ready to be opened.
- ***Sketchbook:*** Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- ***Examples:*** Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- ***Close:*** Closes the instance of the Arduino Software from which it is clicked.
- ***Save:*** Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- ***Save as:*** Allows saving the current sketch with a different name.
- ***Page Setup:*** It shows the Page Setup window for printing.
- ***Print:*** Sends the current sketch to the printer according to the settings defined in Page Setup.
- ***Preferences:*** Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- ***Quit:*** Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

3. **Edit**

- ***Undo/Redo:*** Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- ***Cut:*** Removes the selected text from the editor and places it into the clipboard.
- ***Copy:*** Duplicates the selected text in the editor and places it into the clipboard.
- ***Copy for Forum:*** Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- ***Copy as HTML:*** Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- ***Paste:*** Puts the contents of the clipboard at the cursor position, in the editor.

- ***Select All:*** Selects and highlights the whole content of the editor.
- ***Comment/Uncomment:*** Puts or removes the // comment marker at the beginning of each selected line.
- ***Increase/Decrease Indent:*** Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- ***Find:*** Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- ***Find Next:*** Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- ***Find Previous:*** Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

4. Sketch

- ***Verify/Compile:*** Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- ***Upload:*** Compiles and loads the binary file onto the configured board through the configured Port.
- ***Upload Using Programmer:*** This will overwrite the boot loader on the board; you will need to use Tools > Burn Boot loader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so Tools -> Burn Boot loader command must be executed.
- ***Export Compiled Binary:*** Saves a .hex file that may be kept as archive or sent to the board using other tools.
- ***Show Sketch Folder:*** Opens the current sketch folder.
- ***Include Library:*** Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- ***Add File:*** Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

5. ***Tools***

- ***Auto Format:*** This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- ***Archive Sketch:*** Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- ***Fix Encoding & Reload:*** Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- ***Serial Monitor:*** Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- ***Board:*** Select the board that you're using. See below for descriptions of the various boards.
- ***Port:*** This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- ***Programmer:*** For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a boot loader to a new microcontroller, you will use this.
- ***Burn Boot loader:*** The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a boot loader). Ensure that you've selected the correct board from the Boards menu before burning the boot loader on the target board. This command also set the right fuses.

6. ***Help***

- Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.
- ***Find in Reference:*** This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

- The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.
- Beginning with version 1.0, files are saved with an .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

- Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

- Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for a Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the sports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.
- When you upload a sketch, you're using the Arduino boot loader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The boot loader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to

the microcontroller. The boot loader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

- Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.
- There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch.

Third-Party Hardware

- Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, boot loaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

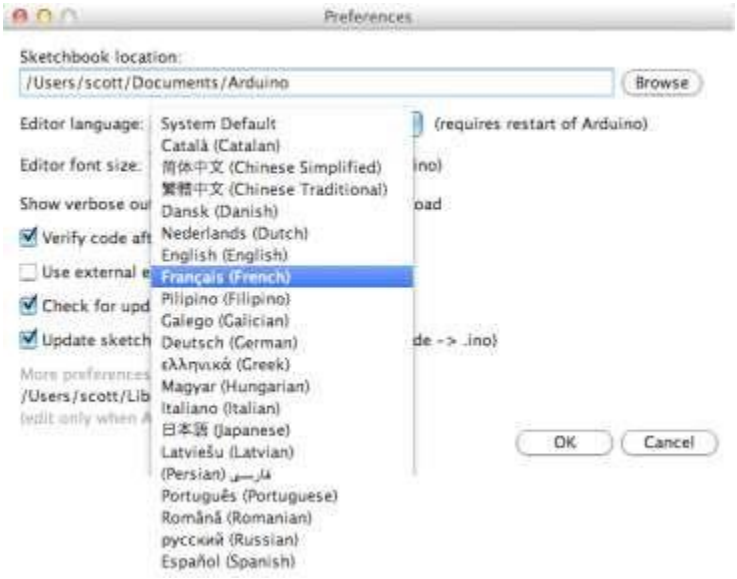
Serial Monitor

- This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

Preferences

- Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

Language Support



- Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)
- If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.
- You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

Boards

- The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn boot loader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the boot loader. You can find a comparison table between the various boards here.

- Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows adding support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, and Galileo and so on.
- ***Arduino Yùn:*** An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- ***Arduino/Genuino Uno:*** An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino Diecimila or Duemilanove w/ ATmega168:*** An ATmega168 running at 16 MHz with auto-reset.
- ***Arduino Nano w/ ATmega328P:*** An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
- ***Arduino/Genuino Mega 2560:*** An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- ***Arduino Mega:*** An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- ***Arduino Mega ADK:*** An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- ***Arduino Leonardo:*** An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- ***Arduino/Genuino Micro:*** An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- ***Arduino Esplora:*** An ATmega32u4 running at 16 MHz with auto-reset.
- Arduino Mini w/ ATmega328P: An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino Ethernet:*** Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino Fio:*** An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino BT w/ ATmega328P:*** ATmega328P running at 16 MHz. The boot loader burned (4 KB) includes codes to initialize the on-board Bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..

- ***LilyPad Arduino USB:*** An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- ***LilyPad Arduino:*** An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P:*** An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino NG or older w/ ATmega168:*** An ATmega168 running at 16 MHz without auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the boot loader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
- ***Arduino Robot Control:*** An ATmega328P running at 16 MHz with auto-reset.
- ***Arduino Robot Motor:*** An ATmega328P running at 16 MHz with auto-reset.
- ***Arduino Gemma:*** An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

ARDUINO PROGRAMING

- Arduino programs are written in the Arduino Integrated Development Environment (IDE). Arduino IDE is special software running on your system that allows you to write sketches (synonym for program in Arduino language) for different Arduino boards. The Arduino programming language is based on a very simple hardware programming language called processing, which is similar to the C language. After the sketch is written in the Arduino IDE, it should be uploaded on the Arduino board for execution.
- The first step in programming the Arduino board is downloading and installing the Arduino IDE. The open source Arduino IDE runs on Windows, Mac OS X, and Linux. Download the Arduino software (depending on your OS) from the official website and follow the instructions to install.

The basics of Arduino programming.

- The structure of Arduino program is pretty simple. Arduino programs have a minimum of 2 blocks,
- Preparation & Execution
- Each block has a set of statements enclosed in curly braces:

```
void setup( )
{
statements-1;
.
.
.
statement-n;
}
void loop ( )
{
statement-1;
.
.
.
statement-n;
}
```

- Here, setup ( ) is the preparation block and loop ( ) is an execution block.
- The setup function is the first to execute when the program is executed, and this function is called only once. The setup function is used to initialize the pin modes and start serial communication. This function has to be included even if there are no statements to execute.

```
void setup ( )
{
pinMode (pin-number, OUTPUT); // set the 'pin-number' as output
pinMode (pin-number, INPUT); // set the 'pin-number' as output
}
```

- After the setup ( ) function is executed, the execution block runs next. The execution block hosts statements like reading inputs, triggering outputs, checking conditions etc..

- In the above example loop ( ) function is a part of execution block. As the name suggests, the loop( ) function executes the set of statements (enclosed in curly braces) repeatedly.

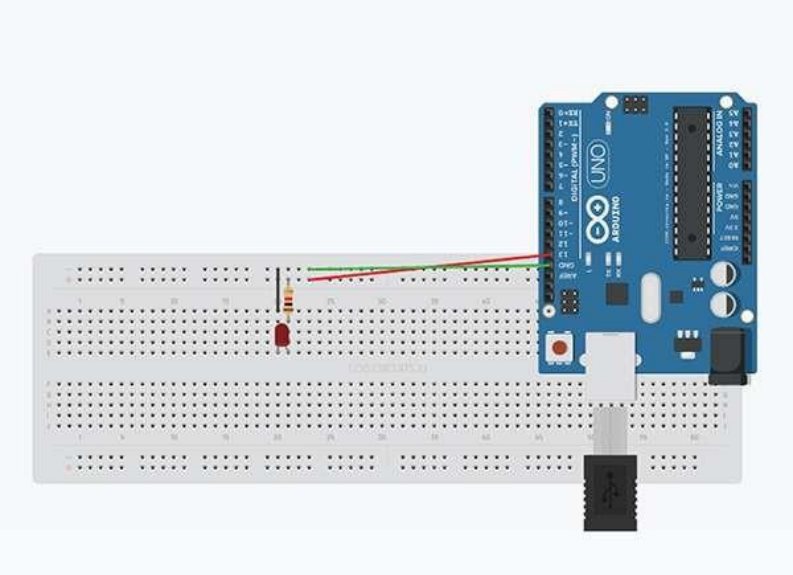
```
Void loop ( )
{
digitalWrite (pin-number,HIGH); // turns ON the component connected to 'pin-number'
delay (1000); // wait for 1 sec
digitalWrite (pin-number, LOW); // turns OFF the component connected to 'pin-number'
delay (1000); //wait for 1sec
}
```

- Note: Arduino always measures the time duration in millisecond. Therefore, whenever you mention the delay, keep it in milli seconds.
- Now, let's take a giant leap and do some experiments with Arduino
  1. Blinking the LED
  2. Fade-in and fade-out the LED
- In the process of experimenting with Arduino, writing the Arduino program is not the only important thing, building the breadboard circuit is equally important.
- Let's take a look at how the breadboard circuit has to be built for both the experiments.
- Components required:
  - Arduino UNO R3 -1
  - Breadboard -1
  - Breadboard connectors -3
  - LED -1
  - 1K resistor -1

1. Blinking LED

- Steps in building a breadboard connection:
- ***Step-1:*** Connect the Arduino to the Windows / Mac / Linux system via a USB cable
- ***Step-2:*** Connect the 13th digital pin of Arduino to the positive power rail of the breadboard and GND to the negative

- ***Step-3:*** Connect the positive power rail to the terminal strip via a 1K ohm resistor
- ***Step-4:*** Fix the LED to the ports below the resistor connection in the terminal strip
- ***Step-5:*** Close the circuit by connecting the cathode (the short chord) of the LED to the negative power strip of the breadboard



Arduino program for LED blink (Version-1)

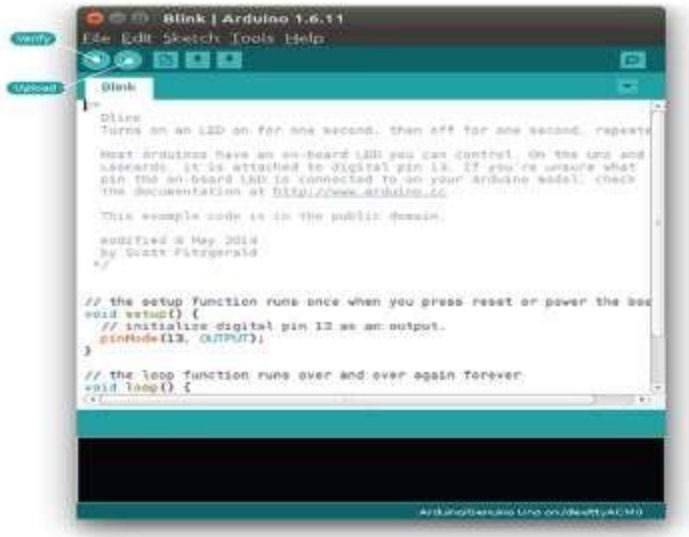
```
int LED =13; // The digital pin to which the LED is connected
 void setup ( )
{
 pinMode (LED, OUTPUT); //Declaring pin 13 as output pin
 }
 void loop( ) // The loop function runs again and again
{
 digitalWrite (LED, HIGH); //Turn ON the LED
delay(1000); //Wait for 1sec
digitalRead (LED, LOW); // Turn off the LED
delay(1000); // Wait for 1sec
}
```

Arduino program for LED blink (Version-2)

```
void setup ( )
```

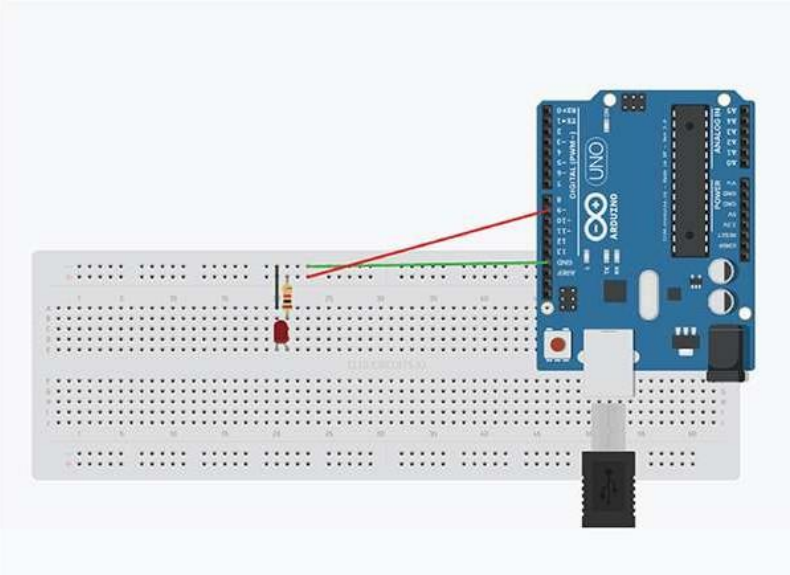
```
{
pinMode (13, OUTPUT); //pin 13 is set as output pin
 }
 void loop( ) // The loop function runs again and again
{
 digitalWrite (13,HIGH); // Turn ON the LED on pin 13
delay (1000); //Wait for 1sec
digitalWrite (13, LOW); //Turn OFF the LED on pin 13
 }
```

- In version-1 of the LED blink program LED is declared globally and is set to pin number 13. This will reduce the number of iterations required to update the pin number in the program when you connect the LED to the other digital pin. Whereas, the pin number has to be changed in 3 different statements in version-2.
- The magic happens when you click the upload icon in the Arduino IDE. The program will be uploaded into the microcontroller of Arduino board and LED in the circuit starts blinking.



2. Fade-in and fade-out the LED

- The step for the LED bling experiment are the same as those followed for building the breadboard circuit except that in step-2, you connect the 9th digital pin to the positive power rail of the breadboard.



Arduino program for LED fade-in and fade-out (Version-1)

```
int led = 9; // The digital pin to which the LED is connected
int brightness = 0; // Brightness of LED is initially set to 0
int fade = 5; // By how many points the LED should fade
void setup()
{
pinMode(led, OUTPUT); //pin 10 is set as output pin
}
void loop() // The loop function runs again and again
{
analogWrite(led, brightness); // set the brightness of LED
brightness = brightness + fade; //Increase the brightness of LED by 5 points
 if (brightness <= 0 || brightness >= 255) // check the level of brightness
 {
 fade = -fade;
 }
delay(30); // Wait for 30 milliseconds
}
```

Arduino program for LED fade-in and fade-out (Version-2)

```
int led=19; // The digital pin to which the LED is connected
void setup()
{
pinMode(led, OUTPUT); //pin 10 is set as output pin
}
 void loop() // The loop function runs again and again
{
 for (int fade=0; fade<=255; fade=fade+5)
{
 analogWrite (led, fade); // Change the brightness of LED by 5 points
delay (30);
 }
}
```

- In both the versions of the LED fade-in and fade-out programs, the analog Write statement is used for a led connected to a digital pin. The reason for this is that, digital pin 10 is a PWM pin. As discussed in the previous article, A tour of the Arduino UNO board, a PWM pin is capable of generating Analog output. In both the programs pin 10 is used as analog output pin.

VARIOUS REAL TIME APPLICATIONS OF IOT

Overview

- Understand the varied applications of the Internet of Things (IoT)
- Includes detailed study about smart homes, connected devices & many other work areas

Introduction

- Do you know what separates humans from other living beings?
- Curiosity. Humans are curious. We question a lot. We are the ones who challenge the status quo of existing rules and strive to build/produce something better. Such curiosity

& efforts have promised us a life where electronic devices & machines will probably become our best friend.

- Yes, you read it correctly the vision to make machines smart enough to reduce human labor to almost nil. The idea of inter-connected devices where the devices are smart enough to share information with us, to cloud based applications and to each other (device to device).
- Smart devices or "Connected devices" as they are commonly called, are designed in such a way that they capture and utilize every bit of data which you share or use in everyday life. And these devices will use this data to interact with you on daily basis and complete tasks.

1. Smart Home

- With IoT creating the buzz, 'Smart Home' is the most searched IoT associated feature on Google. But, what is a Smart Home?
- Wouldn't you love if you could switch on air conditioning before reaching home or switch off lights even after you have left home? Or unlock the doors to friends for temporary access even when you are not at home. Don't be surprised with IoT taking shape companies are building products to make your life simpler and convenient.
- Smart Home has become the revolutionary ladder of success in the residential spaces and it is predicted Smart homes will become as common as smart phones.
- The cost of owning a house is the biggest expense in a homeowner's life. Smart Home products are promised to save time, energy and money. With Smart home companies like Nest, Ecobee, Ring and August, to name a few, will become household brands and are planning to deliver a never seen before experience.

2. Wearable's

- Wearables have experienced an explosive demand in markets all over the world. Companies like Google, Samsung have invested heavily in building such devices. But, how do they work?
- Wearable devices are installed with sensors and software's which collect data and information about the users. This data is later pre-processed to extract essential insights about user.
- These devices broadly cover fitness, health and entertainment requirements. The pre-requisite from internet of things technology for wearable applications is to be highly energy efficient or ultra-low power and small sized.

3. Connected Cars

- The automotive digital technology has focused on optimizing vehicles internal functions. But now, this attention is growing towards enhancing the in-car experience.
- A connected car is a vehicle which is able to optimize its own operation, maintenance as well as comfort of passengers using onboard sensors and internet connectivity.
- Most large auto makers as well as some brave startups are working on connected car solutions. Major brands like Tesla, BMW, Apple, and Google are working on bringing the next revolution in automobiles.

4. Industrial Internet

- Industrial Internet is the new buzz in the industrial sector, also termed as Industrial Internet of Things (IIoT). It is empowering industrial engineering with sensors, software and big data analytics to create brilliant machines.
- According to Jeff Immelt, CEO, GE Electric, IIoT is a "beautiful, desirable and investable" asset. The driving philosophy behind IIoT is that, smart machines are more accurate and consistent than humans in communicating through data. And, this data can help companies pick inefficiencies and problems sooner.
- IIoT holds great potential for quality control and sustainability. Applications for tracking goods, real time information exchange about inventory among suppliers and retailers and automated delivery will increase the supply chain efficiency. According to GE the improvement industry productivity will generate $10 trillion to $15 trillion in GDP worldwide over next 15 years.

5. Smart Cities

- Smart city is another powerful application of IoT generating curiosity among world's population. Smart surveillance, automated transportation, smarter energy management systems, water distribution, urban security and environmental monitoring all are examples of internet of things applications for smart cities.
- IoT will solve major problems faced by the people living in cities like pollution, traffic congestion and shortage of energy supplies etc. Products like cellular communication enabled Smart Belly trash will send alerts to municipal services when a bin needs to be emptied.
- By installing sensors and using web applications, citizens can find free available parking slots across the city. Also, the sensors can detect meter tampering issues, general malfunctions and any installation issues in the electricity system.

6. IoT in agriculture

- With the continuous increase in world's population, demand for food supply is extremely raised. Governments are helping farmers to use advanced techniques and research to increase food production. Smart farming is one of the fastest growing fields in IoT.
- Farmers are using meaningful insights from the data to yield better return on investment. Sensing for soil moisture and nutrients, controlling water usage for plant growth and determining custom fertilizer are some simple uses of IoT.

7. Smart Retail

- The potential of IoT in the retail sector is enormous. IoT provides an opportunity to retailers to connect with the customers to enhance the in-store experience.
- Smartphone's will be the way for retailers to remain connected with their consumers even out of store. Interacting through Smartphone's and using Beacon technology can help retailers serve their consumers better. They can also track consumer's path through a store and improve store layout and place premium products in high traffic areas.

8. Energy Engagement

- Power grids of the future will not only be smart enough but also highly reliable. Smart grid concept is becoming very popular all over world.
- The basic idea behind the smart grids is to collect data in an automated fashion and analyze the behavior or electricity consumers and suppliers for improving efficiency as well as economics of electricity use.
- Smart Grids will also be able to detect sources of power outages more quickly and at individual household levels like nearby solar panel, making possible distributed energy system.

9. IOT in Healthcare

- Connected healthcare yet remains the sleeping giant of the Internet of Things applications. The concept of connected healthcare system and smart medical devices bears enormous potential not just for companies, but also for the well-being of people in general.
- Research shows IoT in healthcare will be massive in coming years. IoT in healthcare is aimed at empowering people to live healthier life by wearing connected devices.

- The collected data will help in personalized analysis of an individual's health and provide tailor made strategies to combat illness. The video below explains how IoT can revolutionize treatment and medical help.

10. IoT in Poultry and Farming

- Livestock monitoring is about animal husbandry and cost saving. Using IoT applications to gather data about the health and well being of the cattle, ranchers knowing early about the sick animal can pull out and help prevent large number of sick cattle.
- With the help of the collected data and ranchers can increase the poultry production. Watch this interesting video.

CONNECTING IOT TO CLOUD

- The Internet of Things (IoT) will continue to transform the business landscape as well as the way we live. Cloud computing is the backbone of this transformation. Increased cloud adoption has acted as a springboard for many IoT applications and business models, offering the ability for companies to reduce time-to-market and total cost of ownership.

Cloud and Connectivity Synergy for Enterprise IoT

- Cloud computing allows companies to store and manage data over cloud platforms, providing scalability in the delivery of applications and Software as a Service (SaaS). IoT devices can generate a significant amount of data per second, with Cisco estimating that IoT will generate 847 zettabytes per year by 2021.
- IoT devices are often sensors that collect data and send it to be processed. In the domain of IoT, physical sensors are virtualized before the data is uploaded to the cloud. While IoT devices can generate lots of data, cloud computing paves the way for this data to travel. This data enables better workflow for developers, who can store and access data remotely, which allows them to implement projects without delay.
- Cloud computing enables the storage and analysis of data to be done quickly and in real-time, allowing enterprises to get the maximum benefit. This is supported by an industry survey from InformationWeek where 65% of respondents said "the ability to quickly meet business demands" was one of the most important reasons a business should move to a cloud environment.
- With cloud computing, organizations do not have to deploy extensive hardware or configure and manage networks and infrastructure. Cloud computing also enables enterprises to scale up the infrastructure, depending on their needs, without setting up additional hardware and infrastructure. This not only helps speed up the development

process but can also cut down on development costs. Half of all CIOs and IT leaders surveyed by the cloud-security company Bitglass, reported cost savings from using cloud-based applications.

- Devices are connected to the cloud through many different methods depending on the device connectivity capabilities. These methods include cellular, satellite, Wi-Fi, Low Power Wide Area Networks (LPWAN) (e.g. NB-IoT), and direct connection to the Internet via Ethernet. Cellular connectivity offers an excellent choice for uninterrupted data transfer between devices, applications, and the cloud.

Cellular Connectivity: The Reliable and Secure Choice for IoT Projects

- Cellular technologies have been designed for reliability, security, and scalability. Cellular Internet of Things (CIoT), based on 3GPP standards, utilizes existing infrastructure to provide excellent IoT coverage and a fast time-to-market.
- Cellular IoT projects will include everything from devices, to a dedicated SIM card and management platform. CIoT hardware is always equipped with SIM cards and can connect to networks via 2G, 3G, or LTE connectivity, depending on which cell towers are available in different regions. For example, 2G switch-off has already begun in many parts of the world, as detailed on our previous blog post.

Cellular benefits include:

- ***Coverage:*** Cellular data coverage today is extensive and growing and has the added benefit of reaching underground spaces, buildings, and rural environments.
- ***Security:*** End-to-end security is supported and aided by the SIM credentials. This keeps data secure from the device to the cloud.
- ***Bandwidth:*** Cellular technologies based on 4G LTE are now as fast as 1 Gbps, with 5G expected to have speeds of up to 10 Gbps.
- For example, fleet tracking relies on secure coverage across boundaries with no risk of losing connection. To ensure this doesn't happen, cellular connectivity is better equipped than alternative connectivity methods, e.g. NB-IoT is designed for low throughput devices where a delay in communications will not hinder the service. In fleet management, with vehicles driving across borders, it's essential to have a robust and constant connection so that devices can be tracked in real time.

EMnify – Bringing the Cloud and Cellular Connectivity Together

- EMnify is an IoT connectivity provider that enables cellular IoT products and services worldwide. The company serves the growing needs of developers and enterprises that require secure, global connectivity for IoT/M2M applications. Its multi-

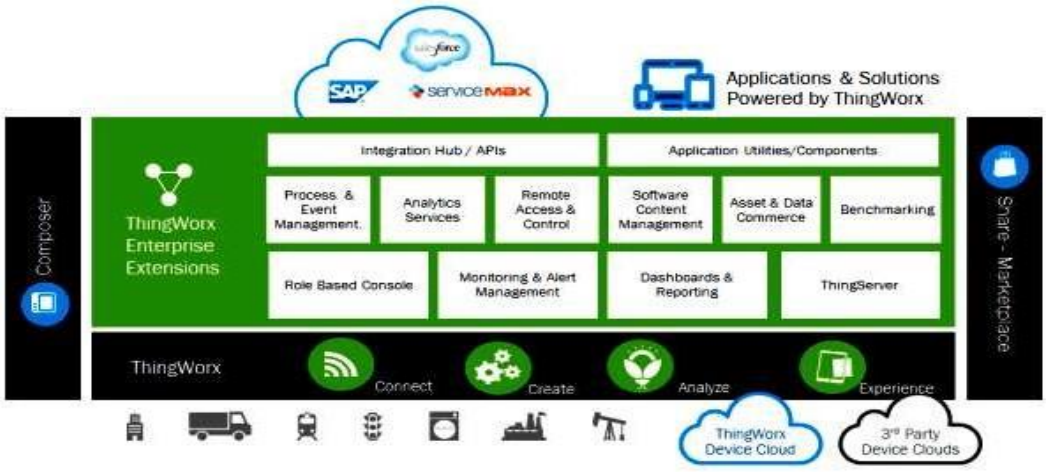
network EMnify SIM can operate instantly in over 180 countries across 540 mobile networks.

- EMnify connected devices benefit from a purpose-built and fully virtualized mobile core network that enables features like virtual private device cloud and dynamic regional internet breakouts. The EMnify management platform gives an oversight of IoT device connectivity as well as access to analytics with an easy to use and intuitive UI. The EMnify SIM is multi-carrier meaning it will automatically connect to the best cellular network and ensure the connection remains stable and continuously connected all around the world.
- In August 2018, EMnify achieved the AWS IoT Competency which differentiates EMnify as an AWS Partner Network (APN) member that has demonstrated relevant technical proficiency and proven customer success delivering solutions seamlessly in the AWS Cloud environment. Learn more about the successful transition to a Cloud Mobile Network and its benefits in the EMnify case study prepared for Telecom Liechtenstein.

CLOUD STORAGE FOR IOT

1. Thingworx 8 IoT Platform

- Thingworx is one of the leading IoT platforms for industrial companies, which provides easy connectivity for devices. It enables the experience from today's connected world. Thingworx 8 is a better, faster, easier platform, providing the functionality to build, deploy, and extend industrial projects and apps.



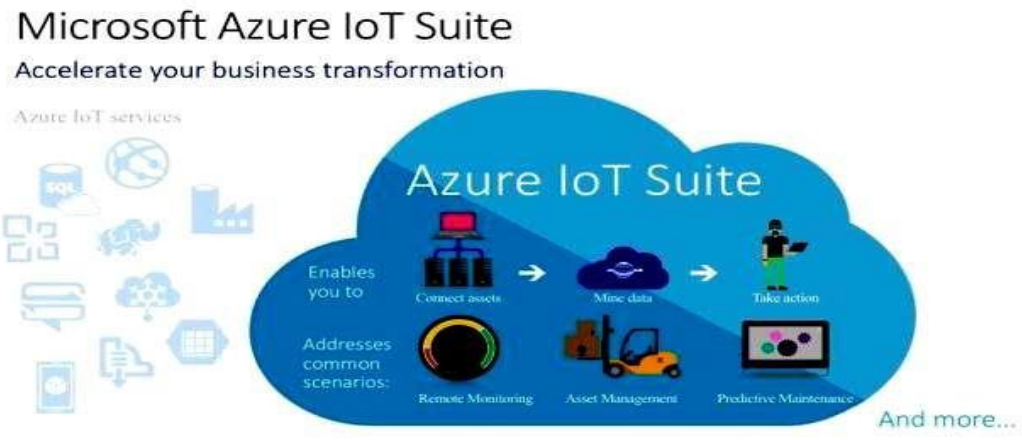
- Thingworx is an IoT platform designed by PTC for development of enterprise app development. It offers basic features, such as:
  - Easy connectivity with electronic devices, like sensors and RFIDs
  - You can work remotely once you are done with the setup

- Pre-built widgets for the dashboard
- Remove Complexity of the project
- Integrated machine learning

- **Pros**
  - Easy web page designs for customers
  - Easy to manage devices
  - Simple connectivity solutions
- **Cons**
  - Difficult to use with custom programs in C#
  - Hard to manage complex systems.
  - The limitation to install edge program on a custom platform.

2. Microsoft Azure IoT Suite

- Microsoft Azure provides multiple services to create IoT solutions. It enhances your profitability and productivity with pre-built connected solutions. It analyzes untapped data to transform business. This provides the solutions for a small PoC to Rolling out your ideas. Azure Suite can easily analyze and act on new data.



- Azure IoT Suite provides features like:
  - Easy Device Registry.
  - Rich Integration with SAP, Salesforce, Oracle, WebSphere, etc
  - Dashboards and visualization
  - Real-time streaming

- **Pros**
  - Offers third-party services
  - Secure and scalable
  - High availability
- **Cons**
  - Requires management
  - Expensive
  - No support for bugs

3. Google Cloud's IoT Platform

- Google's platform is among the best platforms we currently have. Google has an end-to-end platform for Internet-of-Things solutions. It allows you to easily connect, store, and manage IoT data. This platform helps you to scale your business.
- Their main focus is on making things easy and fast. Pricing on Google Cloud is done on a per-minute basis, which is cheaper than other platforms.
- Google Cloud's IoT platform provides features, including:
  - Provides huge storage
  - Cuts cost for server maintenance
  - Business through a fully protected, intelligent, and responsive IoT data
  - Efficient and scalable
  - Analyze big data
- **Pros**
  - Fastest input/output
  - Lesser access time
  - Provides integration with other Google services
- **Cons**
  - Most of the components are Google technologies
  - Limited programming language choices

4. IBM Watson IoT Platform

- IBM Watson is a powerful platform backed by IBM 's the Bluemix and hybrid cloud PaaS (platform as a service) development platform. By providing easy sample apps and interfaces for IoT services, they make it accessible to beginners. You can easily try out their sample to see how it works, which makes it stand out from other platforms.

- Users can get the following features:
  - Real-time data exchange
  - Secure Communication
  - Cognitive systems
  - Recently added data sensor and weather data service
- **Pros**
  - Process untapped data
  - Handle huge quantities of data
  - Improve customer services
- **Cons**
  - Need a lot of maintenance.
  - Take time for Watson integration
  - High switching cost.

5. *AWS IoT Platform*

- Amazon made it much easier for developers to collect data from sensors and Internet-connected devices. They help you collect and send data to the cloud and analyze that information to provide the ability to manage devices.
- You can easily interact with your application with the devices even they are offline.

- Main features of the AWS IoT platform are:
  - Device management
  - Secure gateway for devices
  - Authentication and encryption
  - Device shadow
- **Pros**
  - Good integration with laas offering.
  - Price dropped over the last six years
  - Open and flexible
- **Cons**
  - A big learning curve for AWS
  - Three outages in the last 2 years
  - Not secure for hosting critical enterprise applications

6. *Cisco IoT Cloud Connect*

- Cisco Internet of Things accelerates digital transformation and actions from your data. Cisco IoT Cloud Connect is a mobile, cloud-based suite. It offers solutions for

mobile operators to provide phenomenal IoT experience. It provides flexible deployment options for your devices.

- The main feature of the Cisco Cloud Connect:
  - Data and voice connectivity
  - Device and IP session report
  - Billing is customizable
  - Flexible deployment options

7. Salesforce IoT Cloud

- Salesforce IoT Cloud is powered by Salesforce Thunder. It gathers data from devices, websites, applications, and partners to trigger actions for real-time responses. Salesforce combined with IoT delivers improved customer service.

- Key features of Salesforce IoT Cloud:
  - Enhanced data collection
  - Improved customer engagement
  - Real-time event processing
  - Technology optimization
- **Pros**
  - Scale to billions of devices and messages.
  - Easy UI designs to connect with customers.
- **Cons**
  - Security liability
  - Flexibility limitations

8. Kaa IoT Platform

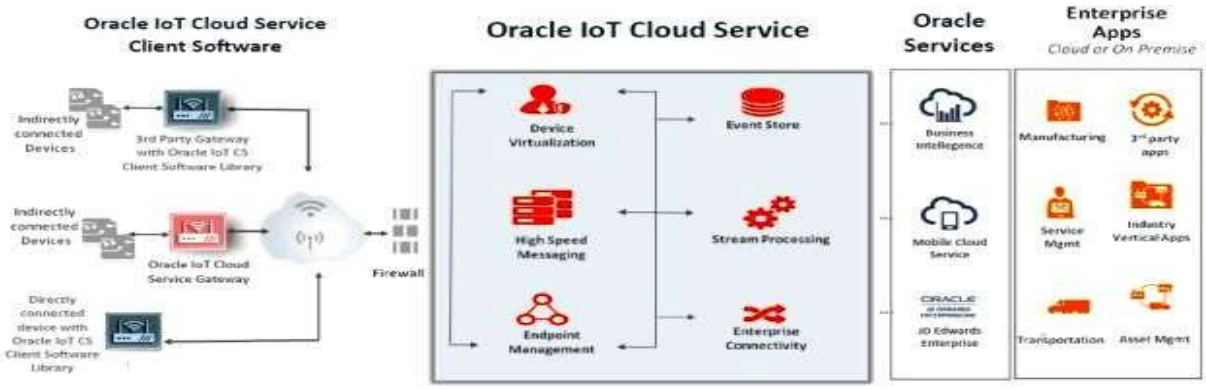
- Kaa is an open-source, multipurpose, middleware platform for complete end-to-end IoT development and smart devices. It reduces cost, risk, and market time. Also, Kaa offers a range of IoT tools that can be easily plugged in and implemented in IoT use cases.

- It provides many features that make it unique, such as:
  - Reduce development time
  - Open source and free
  - Easy and direct device implementation
  - Reduce marketing time
  - Handle millions of devices

- **Pros**
  - Ease of use
  - Third-party integration
  - Data security
- **Cons**
  - Not able to deploy applications based on the PaaS model

9. Oracle IoT Platform

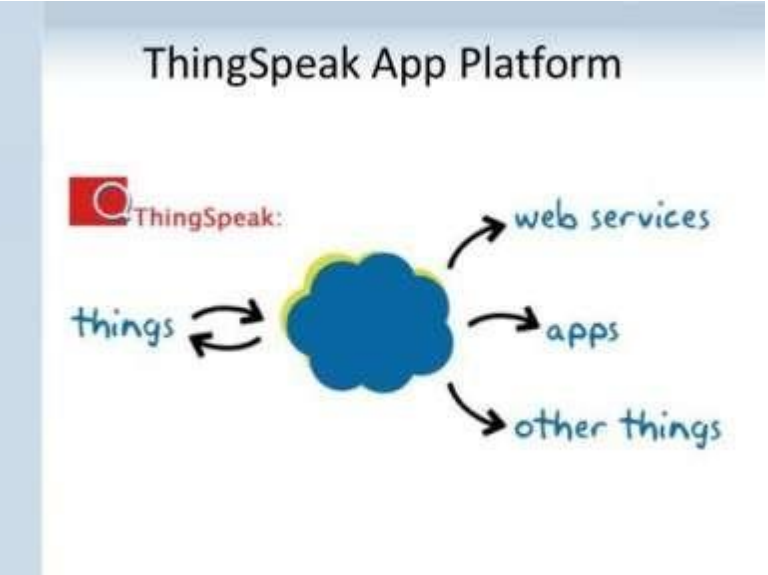
- Oracle offers real-time Internet of Things data analysis, endpoint management, and high speed messaging where the user can get real-time notification directly on their devices. Oracle IoT cloud service is a Platform as a Service (PaaS), cloud-based offering that helps you to make critical business decisions.



- Features offering to users
  - Secure and scalable
  - Real-time insight
  - Integrated
  - Faster to market
- **Pros**
  - Device visualization
  - High speed messaging
  - Event store

10. Thingspeak IoT Platform

- Thingspeak is an open-source platform that allows you to collect and store sensordata to the cloud. It provides you the app to analyze and visualize your data in Matlab. You can use Arduino, Raspberry Pi, and Beaglebone to send sensor data. You can create a separate channel to store data.



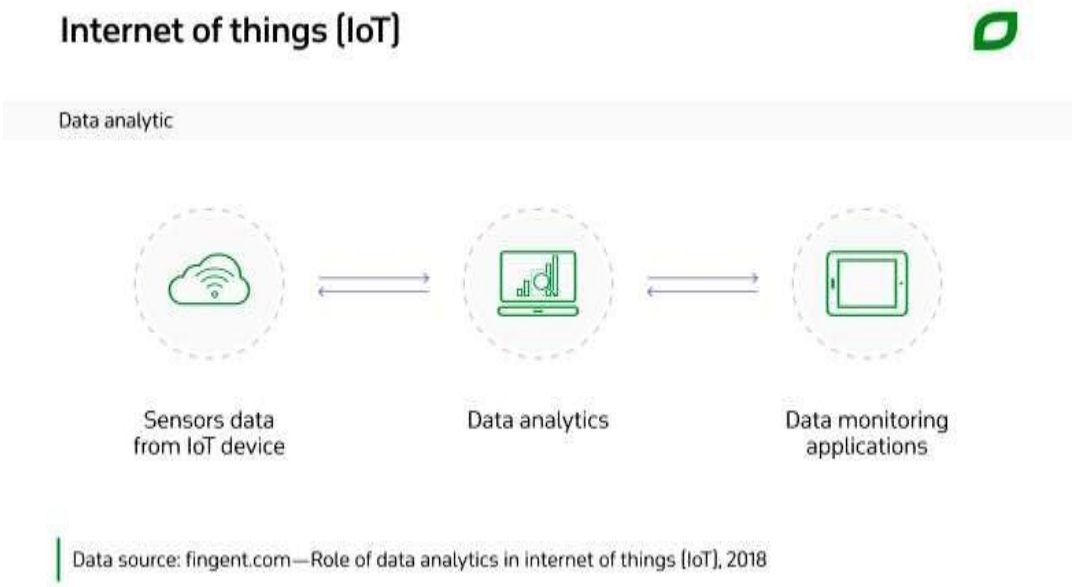
- Features of Thingspeak:
  - Collect data in private channels
  - App integration
  - Event scheduling
  - MATLAB analytics and visualization
- **Pros**
  - Free hosting for channels
  - Easy visualization
  - Provides additional features for Ruby, Node.js, and Python
- **Cons**
  - Limited data uploading for API
  - ThingSpeak API can be a hurdle for beginners

11. GE Predix IoT Platform

- Predix is the world's first industrial platform. Predix was designed to target factories and provides simple ecosystem. It can directly analyze data from the machine and store. GE wants to provide the growing industrial Internet of Things for its cloud platform. This platform is secure and scalable.

- According to the customers their IoT platform can:
  - Optimize assets and operations
  - Provides key performance data
  - Reduces unplanned downtime
  - Real-time operational data

DATA ANALYTICS FOR IOT



- Simply put, IoT data analytics is the analysis of huge data volumes generated by connected devices. Organizations can derive a number of benefits from it: optimize operations, control processes automatically, engage more customers, and empower employees. The combination of IoT and data analytics has already proven to be beneficial in retail, healthcare, telematics, manufacturing, and smart cities. However, its true value for organizations has yet to be fully realized.
- At least a decade ago, it was exceedingly difficult and expensive to analyze massive volumes of information provided by a variety of connected devices. As time goes on, the cost to store data is going down, and analytics capabilities are making huge leaps forward. This creates favorable conditions for organizations to start investing in and implementing IoT data analytics.
- As the accessibility of IoT data analytics grows, more and more organizations are seeing the benefits of having it. Widely-known companies such as Microsoft, GE, Amazon, SAP, and Salesforce have already started implementing IoT data analytics into their day-to-day processes.

The value of IoT data for organizations

- A huge variety of devices connect to the internet and share data through sensors every day. This data is worthless without analysis. However, with an IoT analytics solution put in place, the data that organizations produce is effectively collected, analyzed, and stored. As a result, it allows organizations to optimize their operations at all levels, improve decision making, and achieve a number of benefits.

Better human productivity

- Some organizations install smart sensors throughout their facilities to collect data on employee engagement, performance ratings, and other work-related activities. This data is later used to improve day-to-day business operations and help utilize employee time and energy more efficiently.
- As an example, Humanyze has developed a system that uses smart employee badges with sensors that capture more than 100 data points to measure the productivity of workers. It analyzes how people interact, how much they gesture or listen, and what their tone of voice is. At one point, this data helped discover that bringing call-center workers together for lunch at the same time instead of staggering them so that somebody is always available to answer the phone, significantly improved productivity due to:
    - Job stress level decreased by 19%
    - Productivity increased by 23%

- Retention rate increased by 28%

- If IoT data analytics is evaluated and implemented properly, it can have a positive impact on employees' productivity and overall business success. However, there is a privacy concern: Accenture reports that only 32% of employees consent to their company using their workplace data, and 55% of businesses don't ask for consent.

Improved Equipment Maintenance

- The combination of IoT sensors and data analytics may help companies, especially in the manufacturing industry, to determine when equipment requires maintenance by measuring vibration, heat, and other important figures. Smart equipment can also send messages to operators about potential breakdowns, wear, and delivery schedules. This not only facilitates regular equipment maintenance but also contributes to predictive maintenance.

- Sensor data is used to predict when assets need to be serviced, which allows maintenance to be scheduled at the optimal time, thus reducing breakdowns and saving maintenance costs.

- IoT allows workers to see exactly how their machines are performing in real-time, and alerts them to any issues that might be arising. Being able to prevent unscheduled downtime by using predictive maintenance can provide significant benefits. According to PwC's Predictive Maintenance 4.0 study, utilizing predictive maintenance can reduce costs by 12%, improve uptime by 9%, and reduce safety, health, environment, and quality risks by 14% on average. Additionally, it can extend the lifetime of an aging asset by 20%.

Operations optimization and automation

- With IoT and analytics working in tandem, organizations can automatically control processes that previously could only be tracked manually. For example, it gives manufacturers a comprehensive view of what's going on at every point in production. This allows them to maintain a continuous flow of final products, identify bottlenecks in real-time, and avoid defects. It also reduces the risk of human error.

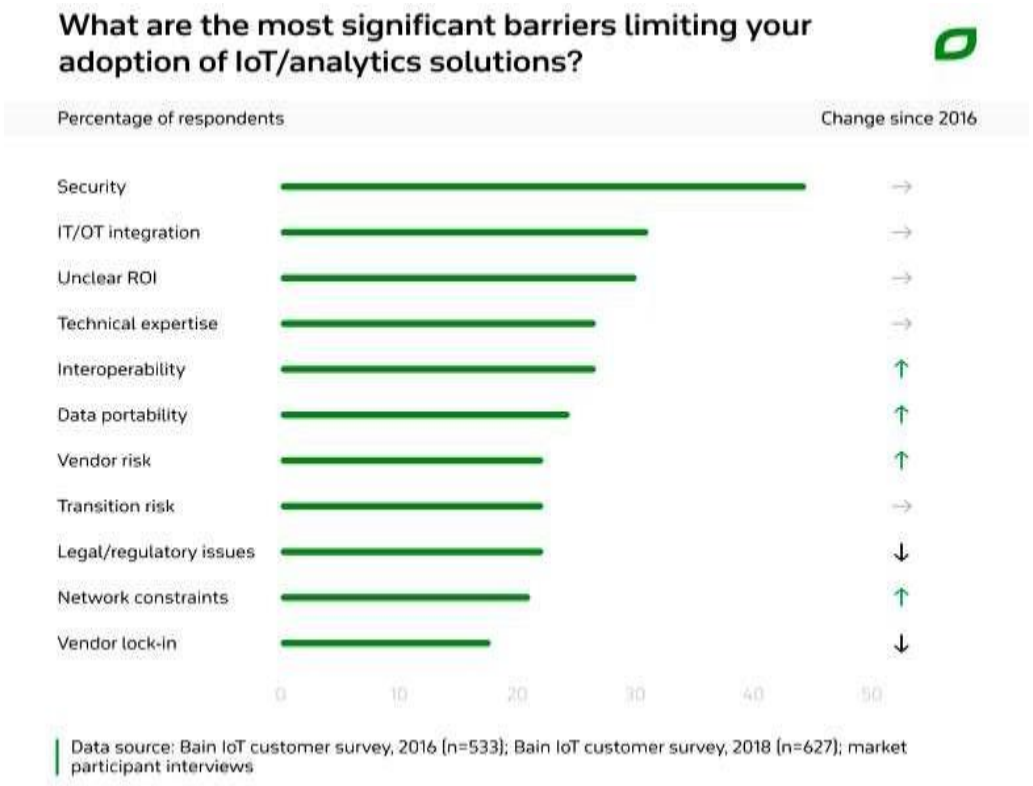
- General Motors, for instance, monitors humidity with IoT sensors to optimize painting. When conditions for painting are unfavorable, they send the detail to another place with more appropriate humidity levels, thereby reducing downtime and the need for repainting.

- When another major manufacturing company, Bosch Rexroth, started utilizing IoT, they saw a significant increase in productivity of both workers and equipment.

Enhanced customer experience

- Be it a retail shop or a healthcare center, each organization strives to create a better and more personalized customer experience. The implementation of IoT data analytics can help with this onerous task. IoT data reveals a wealth of customer behaviors and preferences that can be analyzed and used to predict customer needs.
- For example, when a customer enters a store, the IoT data analytics system can guide the shopper to the jeans they have been looking at online. It can also send them a personalized coupon to make the purchase in-store that day. In addition, retailers, restaurant chains, and makers of consumer goods can use IoT data to do targeted marketing and promotions.
- In healthcare, hospitals can use real-time IoT data analytics to manage increased patient traffic and improve general operational efficiency.
- This is beneficial to both parties; consumers gain more value through convenience and time saving, while organizations increase their revenue and stay more attractive to customers.

Challenges and barriers



- IoT data analytics can certainly provide benefits in many business areas, like reducing maintenance costs, avoiding equipment failures, and improving customer experiences and human productivity. But despite the proven benefits of applying IoT data, many organizations don't derive value from their data assets simply because they don't know what to do with the technology.
- Also, sometimes employees get overwhelmed with the amount of data coming in, preventing them from sifting through it all and finding efficient ways of utilizing it. With multiple sensors providing information as frequently as every 30 seconds, it could result in data overload for the workers or the computers, if the right ones aren't being used.
- Other possible barriers to IoT data analytics adoption include security issues and high costs. Since it requires multiple machines and devices working together and exchanging information, if one system is subjected to a security breach, it could spread throughout all systems.
- Additionally, because IoT data analytics is still fairly new, the implementation costs can be high. This can deter business owners from adopting it, especially if it is difficult to see the long-term benefits of making that investment.

IoT data analytics on the rise, still

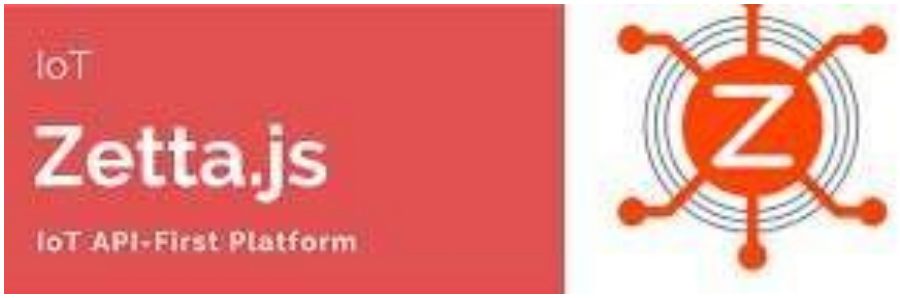
- At a time when organizations are looking to gain a competitive edge, many are starting to look toward the internet of things. It offers an opportunity to gather even more information than before, which can improve processes, drive innovation, and enhance customer experience. While it hasn't reached every corner of business and operations yet, it is already being used to make workplaces safer and more efficient.
- Having the ability to receive diagnostic and predictive data in real-time can be a game-changer, especially for companies who rely heavily on their equipment working properly and at full capacity. IoT data analytics allows workers to schedule downtimes that are convenient or know exactly what is wrong without having to bring in a technician to examine the machine. That all adds up to a less expensive and more efficient operation.
- IoT data analytics can also be used to make customers happier and enhance their shopping experiences. By gathering data on the preferences of their audiences, companies can give customers what they are looking for and even know how much they are willing to pay for it. It can also gather useful data from product reviews, giving companies a better idea of what their customers are looking for or would like to see done differently.
- Even though there are still significant challenges to overcome, IoT data analytics continues to grow and gain popularity. It has the ability to offer unprecedented insights

that have never before been so readily available. It is reasonable to think that we will see a time in the near future where the general consensus is that its pros outweigh its cons.

SOFTWARE & MANAGEMENT TOOLS FOR IOT

- IoT platforms and tools are considered as the most significant component of the IoT ecosystem. Any IoT device permits to connect to other IoT devices and applications to pass on information using standard Internet protocols. IoT platforms fill the gap between the device sensors and data networks. IoT platforms connect the data to the sensor system and give insights using back-end applications to create a sense of the plenty of data developed by the many sensors.
- The Internet of Things (IoT) is the future of technology that helps the Artificial intelligence (AI) to regulate and understand the things in a considerably stronger way.
- We have picked up a mix of best known IoT platforms and tools that help you to develop the IoT projects in an organized way.

1. Zetta



- Zetta is API based IoT platform based on Node.js. It is considered as a complete toolkit to make HTTP APIs for devices. Zetta combines REST APIs, Web Sockets to make data-intensive and real-time applications. The following are some notable features.
  - It can run on the cloud, or a PC, or even modest development boards.
  - Easy interface and necessary programming to control sensors, actuators, and controllers.
  - Allows developers to assemble Smartphone apps, device apps, and cloud apps.
  - It is developed for data-intensive and real-time applications.
  - Turns any machine into an API.

2. Arduino

- If you are seeking to make a computer that can perceive and exercise stronger control over the real world when related to your ordinary stand-alone computer, then Arduino can be your wise preference.
- Offering an appropriate blend of IoT hardware and software, Arduino is a simple-to-use IoT platform. It operates through an array of hardware specifications that can be given to interactive electronics. The software of Arduino comes in the plan of the Arduino programming language and Integrated Development Environment (IDE).

3. Node-RED

- Node-RED is a visual tool for lining the Internet of Things, i.e., wiring together hardware devices, APIs, and online services in new ways. Built on Node.js, Node-RED describes itself as “a visual means for wiring the Internet of Things.”
- It provides developers to connect devices, services, and APIs using a browser-based flow editor. It can run on Raspberry Pi, and further 60,000 modules are accessible to increase its facilities.

4. Flutter

- Flutter is a programmable processor core for electronics projects, designed for students, and engineers. Flutter's take to glory is its long-range. This Arduino-based board includes a wireless transmitter that can show up to more than a half-mile. Plus, you don't require a router; flutter boards can interact with each other quickly.
- It consists of 256-bit AES encryption, and it's simple to use. Some of the other features are below.
    - Fast Performance
    - Expressive and Flexible UI
    - Native Performance
    - Visual finish and functionality of existing widgets.

5. M2MLabs Mainspring

- M2MLabs Mainspring is an application framework for developing a machine to machines (M2M) applications such as remote control, fleet administration, or smart terminal. Its facilities include flexible design of devices, device structure, connection between machines and applications, validation and normalization of data, long-term data repository, and data retrieval functions.

- It's based on Java and the Apache Cassandra NoSQL database. M2M applications can be modeled in hours rather than weeks and subsequently passed on to a high-performance execution environment made on top of a standard J2EE server and the highly-scalable Apache Cassandra database.

6. ThingsBoard

- ThingsBoard is for data collection, processing, visualization, and device management. It upholds all standard IoT protocols like CoAP, MQTT, and HTTP as quickly as cloud and on-premise deployments. It builds workflows based on design life cycle events, REST API events, RPC requests.
- Let's take a look at the following ThigsBoard features.
  - A stable platform that is combining scalability, production, and fault-tolerance.
  - Easy control of all connected devices in an exceptionally secure system
  - Transforms and normalizes device inputs and facilitates alarms for generating alerts on all telemetry events, restores, and inactivity.
  - Enables use-state specific features using customizable rule groups.
  - Handles millions of devices at the same time.
  - No single moment of failure, as every node in the bundle is exact.
  - Multi-tenant installations out-of-the-wrap.
  - Thirty highly customized dashboard widgets for successful user access.

7. Kinoma

- Kinoma, a Marvell Semiconductor hardware prototyping platform, involves three different open source projects. Kimona Create is a DIY construction kit for prototyping electronic devices. Kimona Studio is the development environment that functions with Set up and the Kinoma Platform Runtime. Kimona Connect is a free iOS and Android app that links smart phones and stands with IoT devices.

8. Kaa IoT Platform

- Kaa is a production-ready, flexible, multi-purpose middleware platform for establishing end-to-end IoT solutions, connected applications, and smart devices. It gives a comprehensive way of carrying out effective communication, deals with, and interoperation capabilities in connected and intelligent devices.
- It mounts from tiny startups to a great enterprise and holds advanced deployment models for multi-cloud IoT solutions. It is primarily based on flexible micro services and readily conforms to virtually any need and application — some other features as below.
    - Facilitates cross-device interoperability.
    - Performs real-time device control, remote device provisioning, and structure.
    - Create cloud services for smart products
    - Consists of topic-based warning systems to provide end-users to deliver messages of any predefined format to subscribed endpoints.
    - Perform real-time device monitoring
    - Manage an infinite quantity of connected devices
    - Collect and analyze sensor data

9. Site Where

- Site Where platform offers the ingestion, repository, processing, and assimilation of device inputs. It runs on Apache Tomcat and provides highly tuned MongoDB and HBase implementations. You can deploy Site Where to cloud platforms like AWS, Azure, GCP, or on-premises. It also supports Kubernetes cluster provisioning.
- The following are some of the other features.
  - Run any estimate of IoT applications on a single Site Where instance
  - Spring brings the root configuration framework
  - Add widgets through self-registration, REST services, or in batches
  - InfluxDB for event data storage
  - Connect devices with MQTT, Stomp, AMQP and other protocols
  - Integrates third-party integration frameworks
  - Eclipse Californium for CoAP messaging
  - HBase for the non-relational datastore
  - Grafana to visualize Site Where data

10. DSA

- Distributed Services Architecture (DSA) is for implementing inter-device communication, logic, and efforts at every turn of the IoT infrastructure. It allows cooperation between devices in a distributed manner and sets up a network engineer to share functionality between discrete computing systems.
- You can manage node attributes, permission, and links from DSLinks.

11. Thinger

- Thinger.io provides a scalable cloud base for connecting devices. You can deal with them quickly by running the admin console or combine them into your project logic using their REST API. It supports all types of hacker's boards such as Raspberry Pi, Intel Edison, and ESP8266.
- Thinger can be integrated with IFTT, and it provides real-time data on a beautiful dashboard.

UNIT-5

INTERNET OF EVERYTHING (IOE)

- IoE is the intelligent connection of people, process, data and things.
- It describes a world where billions of objects have sensors to detect measure and assess their status; all connected over public or private networks using standard and proprietary protocols.
- It is based on the idea that in the future, internet connections will not be restricted to laptop or desktop computers
- A handful of tablets, as in previous decades. Instead, machines will generally become smarter by having more access to data and expanded networking opportunities.
- Major applications range from digital sensor tools/interfaces used for remote appliances to smarter and more well connected mobile devices, industrial machine learning systems and other types of distributed hardware that have recently become more intelligent and automated.
- A concept that extends the Internet of Things (IoT) emphasis on machine-to-machine (M2M) communications to describe a more complex system that also encompasses people and processes.

Features of IoE

Main features of IoE fall under two main categories:

- ***Input:*** Allows analog or external data to be put into a piece of hardware

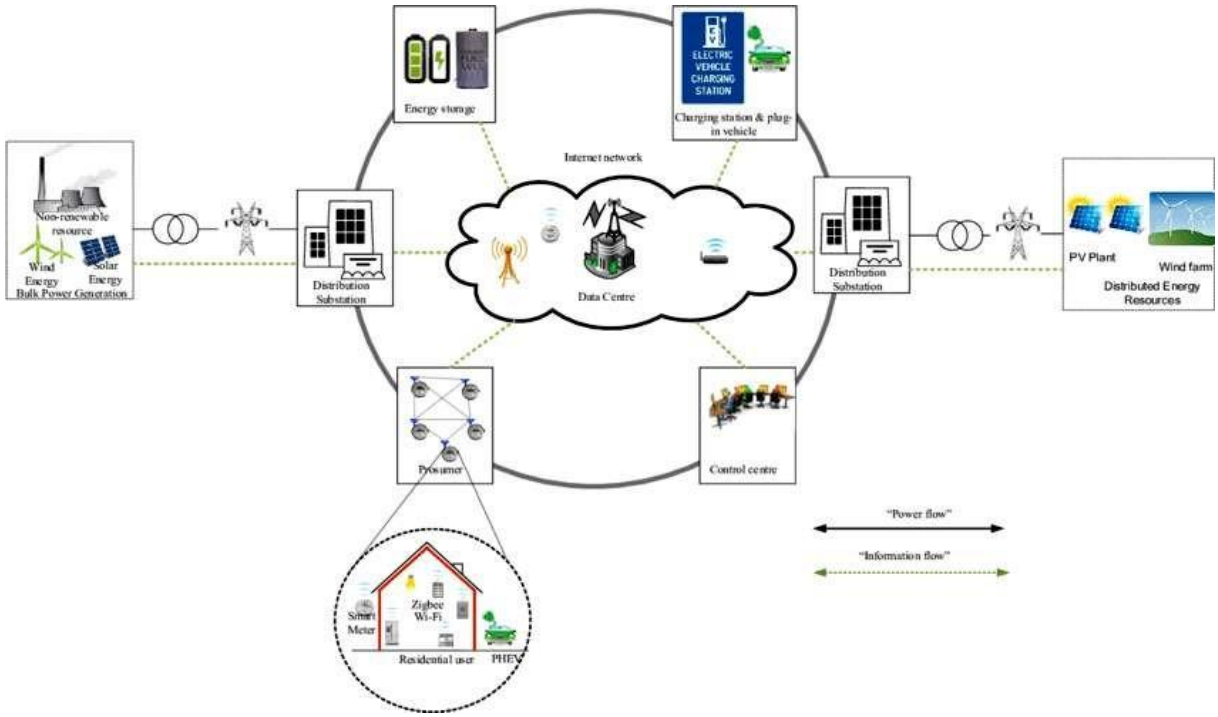
- ***Output:*** Allows a piece of hardware to be put back into the internet
- ***Decentralization and moving to the edge:*** data is processed not in a single center, but in numerous distributed nodes
- ***Data input and output:*** external data can be put into devices and given back to other components of the network
- ***Relation to every technology in the process of digital transformation:*** cloud computing, fog computing, AI, ML, IoT, Big Data, etc. Actually, a rise in Big Data and the IoE technology development are interconnected

Four Pillars of IoE

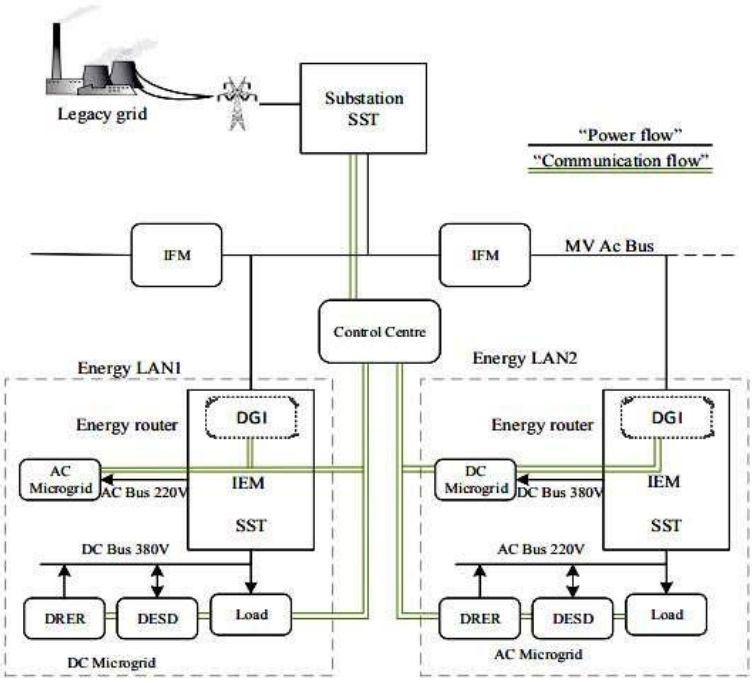
- "Internet of Everything" has four important pillars:
  1. people
  2. process
  3. data and
  4. things
- ***People:*** Connecting people in more relevant, valuable ways.
  - People provide their personal insights via websites, applications or connected devices they use (such as social networks, healthcare sensors and fitness trackers);
  - AI algorithms and other smart technologies analyze this data to "understand" human issues and deliver relevant content according to their personal or business needs that helps them quickly solve issues or make decisions.
- ***Data:*** Converting data into intelligence to make better decisions.
  - The raw data generated by devices has no value. But once it is summarized, classified and analyzed, it turns into priceless information that can control various systems and empower intelligent solutions.
- ***Process:*** Delivering the right information to the right person (or machine) at the right time.
  - Different processes based on artificial intelligence, machine learning, social networks or other technologies ensure that the right information is sent to the right person at the right time.
  - The goal of processes is to guarantee the best possible usage of Big Data.
- ***Things:*** Physical devices and objects connected to the Internet and each other for intelligent decision making; often called Internet of Things (IoT).

- Various physical items embedded with sensors and actuators generate data on their status and send it to the needed destination across the network.

ARCHITECTURE OF THE INTERNET OF ENERGY



The IoE will allow energy exchange between a wide variety of sources and loads, including renewable energy sources, distributed energy storage, plug-in electric vehicles, domestic and industrial "prosumers", etc. The energy network will be monitored and managed via the internet. In principle, peer-to-peer exchange of both information and energy may occur in this system, though some level of central or hierarchical control may also be required. The general principle is that energy is directed from the source(s) to the load(s), similar to the routing of information in the internet.



Various detailed architectures have been proposed for the IoE. For example, FREEDM system has published a roadmap for achieving an automated and flexible electricity distribution system, or energy internet. Key enabling technologies in this system are the "energy router" or intelligent energy management (IEM) unit, the plug-and-play interface, and distributed grid intelligence (DGI). An intelligent fault management unit (IFM) isolates the energy LAN if an electrical fault occurs or islanding is required. The IEM and IFM must communicate via a reliable and secure communication network. The energy router is capable of AC and DC supply, and contains a Distributed Grid Intelligence unit (DGI).

SMART OBJECTS AND LLNS

Characteristics of smart objects

- These devices are highly constrained in terms of
  - Physical size
  - CPU power
  - Memory (few tens of kilobytes)
  - Bandwidth (Maximum of 250 KB/s, lower rates the norm)
- Power consumption is critical
  - If it is battery powered then energy efficiency is paramount, batteries might have to last for years
- May operate in harsh environments

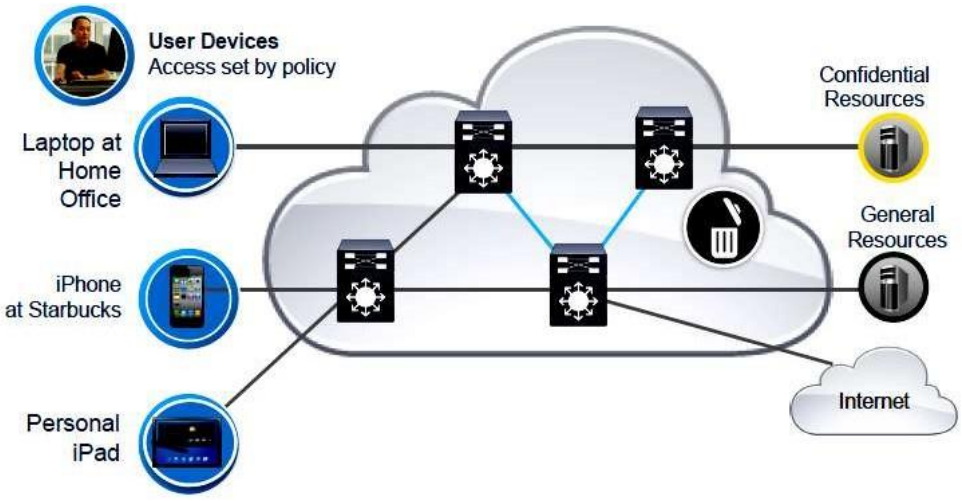
- Challenging physical environment (heat, dust, moisture, interference)

- Wireless capabilities based on Low Power & Lossy Network (LLNs) technology
  - Predominantly IEEE 802.15.4 (2.4 GHz and 900 MHz)
  - Newer RF technologies IEEE 802.15.4g Smart Utility Network (SUN)
- May also run over wired technologies such as IEEE P1901.2 PLC (Power Line)

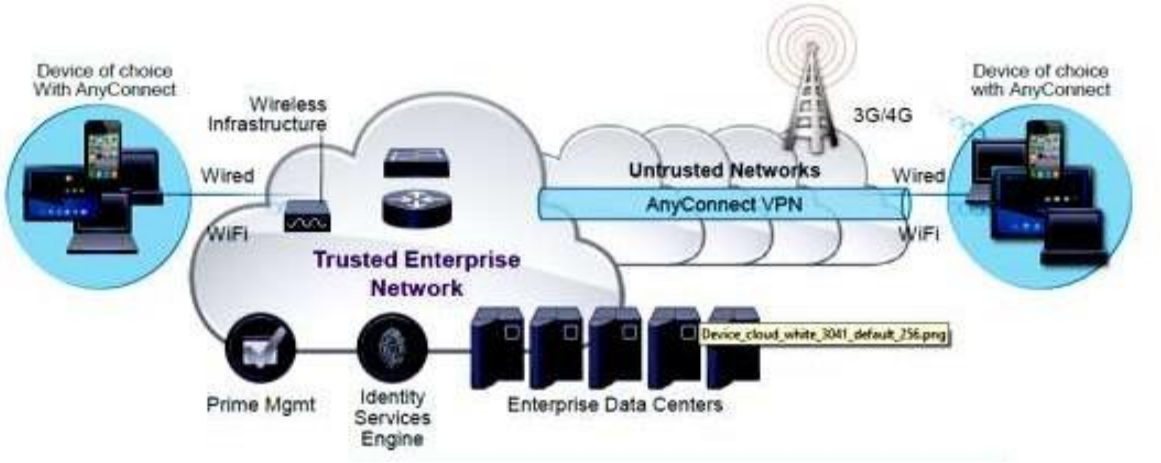
Low Power Lossy Network (LLN)

- LLNs comprise a large number of highly constrained devices (smart objects) interconnected by predominantly wireless links of unpredictable quality
- LLNs cover a wide scope of applications
  - Industrial Monitoring, Building Automation, Connected Home, Healthcare, Environmental Monitoring, Urban Sensor Networks, Energy Management, Asset Tracking, Refrigeration
- Several IETF working groups and Industry Alliance addressing LLNs
  - IETF - Core, 6Lowpan, ROLL
  - Alliances - IP for Smart Objects Alliance (IPSO)

SECURE MOBILITY



- Extend access to trusted networks and resources across untrusted boundaries—anywhere, anytime, and any device



- ***Benefits:***
  - mobile uses work easily and predictably whenever and from wherever
  - Network managers easily set policy and manage the network to allow secure access to the appropriate resources

HOME AUTOMATION

IoT applications for smart homes:

e). Smart Lighting

f). Smart Appliances

g). Intrusion Detection

h). Smoke / Gas Detectors

e). Smart Lighting

- Smart lighting achieves energy savings by sensing the human movements and their environments and controlling the lights accordingly.
- Key enabling technologies for smart lighting include :
  - Solid state lighting (such as LED lights)
  - IP-enabled lights
- Wireless-enabled and Internet connected lights can be controlled remotely from IoT applications such as a mobile or web application.

- Paper: Energy-aware wireless sensor network with ambient intelligence for smart LED lighting system control [IECON, 2011] presented controllable LED lighting system that is embedded with ambient intelligence gathered from a distributed smart WSN to optimize and control the lighting system to be more efficient and user-oriented.

f). Smart Appliances

- Smart appliances make the management easier and provide status information of appliances to the users remotely. E.g.: smart washer/dryer that can be controlled remotely and notify when the washing/drying cycle is complete.
- Open Remote is an open source automation platform for smart home and building that can control various appliances using mobile and web applications.
- It comprises of three components:
    - A Controller: manages scheduling and runtime integration between devices.
    - A Designer: allows creating both configurations for the controller and user interface designs.
    - Control Panel: allows interacting with devices and controlling them.
- Paper: An IoT-based Appliance Control System for Smart Home [ICICIP, 2013] implemented an IoT based appliance control system for smart homes that uses a smart-central controller to set up a wireless sensor and actuator network and control modules for appliances.

g). Intrusion Detection

- Home intrusion detection systems use security cameras and sensors to detect intrusions and raise alerts.
- The form of the alerts can be in form:
    - SMS
    - Email
    - Image grab or a short video clip as an email attachment
- Papers :
    - Could controlled intrusion detection and burglary prevention stratagems in home automation systems [BCFIC, 2012] present a controlled intrusion detection system that uses location-aware services, where the geo-location of each node of a home automation system is independently detected and stored in the cloud?

- An Intelligent Intrusion Detection System Based on UPnP Technology for Smart Living [ISDA, 2008] implement an intrusion detection system that uses image processing to recognize the intrusion and extract the intrusion subject and generate Universal-Plug-and-Play (UPnP-based) instant messaging for alerts.

h). Smoke / Gas Detectors

- Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire.
- It uses optical detection, ionization or air sampling techniques to detect smoke
- The form of the alert can be in form :
  - Signals that send to a fire alarm system
- Gas detector can detect the presence of harmful gases such as carbon monoxide (CO), liquid petroleum gas (LPG), etc.
- Paper: Development of Multipurpose Gas Leakage and Fire Detector with Alarm System [TIIEC, 2013] designed a system that can detect gas leakage and smoke and gives visual level indication.

CITIES: SMART PARKING

- Finding the parking space in the crowded city can be time consuming and frustrating
- Smart parking makes the search for parking space easier and convenient for driver.
- It can detect the number of empty parking slots and send the information over the Internet to the smart parking applications which can be accessed by the drivers using their smart phones, tablets, and in car navigation systems.
- Sensors are used for each parking slot to detect whether the slot is empty or not, and this information is aggregated by local controller and then sent over the Internet to database.
- Paper :
  - Design and implementation of a prototype Smart Parking (SPARK) system using WSN [International Conference on Advanced Information Networking and Applications Workshop, 2009] designed and implemented a prototype smart parking system based on wireless sensor network technology with features like remote parking monitoring, automate guidance, and parking reservation mechanism.

ENVIRONMENT: WEATHER MONITORING

- It collects data from a number of sensors attached such as temperature, humidity, pressure, etc and sends the data to cloud-based applications and store back-ends.
- The data collected in the cloud can then be analyzed and visualized by cloud-based applications.
- Weather alert can be sent to the subscribed users from such applications.
- AirPi is a weather and air quality monitoring kit capable of recording and uploading information about temperature, humidity, air pressure, light levels, UV levels, carbon monoxide, nitrogen dioxide and smoke level to the Internet.
- Paper:
  - PeWeMoS – Pervasive Weather Monitoring System [ICPCA, 2008] □ Presented a pervasive weather monitoring system that is integrated with buses to measure weather variables like humidity, temperature, and air quality during the bus path

AGRICULTURE: SMART IRRIGATION

- Smart irrigation system can improve crop yields while saving water.
- Smart irrigation systems use IoT devices with soil moisture sensors to determine the amount of moisture on the soil and release the flow of the water through the irrigation pipes only when the moisture levels go below a predefined threshold.
- It also collects moisture level measurements on the server on in the cloud where the collected data can be analyzed to plan watering schedules.
- Cultivar's Rain Could is a device for smart irrigation that uses water valves, soil sensors, and a Wi-Fi enabled programmable computer. [http://ecultivar.com/rain-cloud-product-project/]

SOFTWARE & MANAGEMENT TOOLS FOR IOT CLOUD STORAGE MODELS & COMMUNICATION APIS

Popular Models are

1. Amazon Web Service (AWS)
2. Xively Cloud (PAAS)

1. Amazon Web Service (AWS)

AWS is a comprehensive cloud services platform that offers compute power, storage, content delivery, and other functionality that organizations can use to deploy applications and services cost-effectively—with flexibility, scalability, and reliability. AWS self-service means that you can proactively address your internal plans and react to external demands when you choose.

a). Compute & Networking

- ***Amazon Elastic Compute Cloud (Amazon EC2):*** is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers and system administrators. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers and system administrators the tools to build failure resilient applications and isolate themselves from common failure scenarios.
- ***Auto Scaling:*** Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you're using increases seamlessly during demand spikes to maintain performance, and decreases automatically during demand lulls to minimize costs. Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Auto Scaling is enabled by Amazon Cloud Watch and available at no additional charge beyond Amazon Cloud Watch fees.
- ***Elastic Load Balancing:*** Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance.
- ***Amazon Workspaces:*** Amazon Workspaces is a fully managed desktop computing service in the cloud. Amazon Workspaces allows customers to easily provision cloud-based desktops that allow end-users to access the documents, applications and

resources they need with the device of their choice, including laptops, iPad, Kindle Fire, or Android tablets. With a few clicks in the AWS Management Console, customers can provision a high-quality desktop experience for any number of users at a cost that is highly competitive with traditional desktops and half the cost of most virtual desktop infrastructure (VDI) solutions.

- ***Amazon Virtual Private Cloud (Amazon VPC):*** Amazon Virtual Private Cloud lets you provision a logically isolated section of the Amazon Web Services (AWS) Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can easily customize the network configuration for your Amazon VPC. For example, you can create a public-facing subnet for your web servers that have access to the Internet, and place your backend systems such as databases or application servers in a private-facing subnet with no Internet access. You can leverage multiple layers of security (including security groups and network access control lists) to help control access to Amazon EC2 instances in each subnet additionally; you can create a hardware virtual private network (VPN) connection between your corporate data center and your VPC and leverage the AWS cloud as an extension of your corporate data center.
- ***Amazon Route 53:*** Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications by translating human readable names, such as www.example.com, into the numeric IP addresses, such as 192.0.2.1, that computers use to connect to each other. Route 53 effectively connects user requests to infrastructure running in AWS, such as an EC2 instance, an elastic load balancer, or an Amazon S3 bucket. Route 53 can also be used to route users to infrastructure outside of AWS. Amazon Route 53 is designed to be fast, easy to use, and cost effective. It answers DNS queries with low latency by using a global network of DNS servers. Queries for your domain are automatically routed to the nearest DNS server, and thus are answered with the best possible performance. With Route 53, you can create and manage your public DNS records with the AWS Management Console or with an easy-to-use API. It's also integrated with other Amazon Web Services. For instance, by using the AWS Identity and Access Management (IAM) service with Route 53, you can control who in your organization can make changes to your DNS records. Like other Amazon Web Services, there are no long-term contracts or minimum usage requirements for using Route 53—you pay only for managing domains through the service and the number of queries that the service answers.

- ***AWS Direct Connect:*** AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS. Using AWS Direct Connect, you can establish private connectivity between AWS and your data center, office, or co-location environment, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections. AWS Direct Connect lets you establish a dedicated network connection between your network and one of the AWS Direct Connect locations. Using industry standard 802.1Q virtual LANS (VLANs), this dedicated connection can be partitioned into multiple logical connections. This allows you to use the same connection to access public resources such as objects stored in Amazon S3 using public IP address space, and private resources such as Amazon EC2 instances running within an Amazon VPC using private IP space, while maintaining network separation between the public and private environments. Logical connections can be reconfigured at any time to meet your changing needs.

b). Storage & Content Delivery Network

- ***Amazon Simple Storage Service (Amazon S3):*** Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. The container for objects stored in Amazon S3 is called an Amazon S3 bucket. Amazon S3 gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of websites. The service aims to maximize benefits of scale and to pass those benefits on to developers.
- ***Amazon Glacier:*** Amazon Glacier is an extremely low-cost storage service that provides secure and durable storage for data archiving and backup. In order to keep costs low, Amazon Glacier is optimized for data that is infrequently accessed and for which retrieval times of several hours are suitable. With Amazon Glacier, customers can reliably store large or small amounts of data for as little as $0.01 per gigabyte per month, a significant savings compared to on-premises solutions. Companies typically over-pay for data archiving. First, they're forced to make an expensive upfront payment for their archiving solution (which does not include the ongoing cost for operational expenses such as power, facilities, staffing, and maintenance). Second, since companies have to guess what their capacity requirements will be, they understandably over-provision to make sure they have enough capacity for data redundancy and unexpected growth. This set of circumstances results in under-utilized capacity and wasted money. With Amazon Glacier, you pay only for what you use. Amazon Glacier changes the game for data archiving and backup because you pay nothing up front, pay a very low price for

storage, and can scale your usage up or down as needed, while AWS handles all of the operational heavy lifting required to do data retention well. It only takes a few clicks in the AWS Management Console to set up Amazon Glacier, and then you can upload any amount of data you choose.

- ***Amazon Elastic Block Storage (EBS)***: Amazon Elastic Block Store (EBS) provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are network-attached, and persist independently from the life of an instance. Amazon EBS provides highly available, highly reliable, predictable storage volumes that can be attached to a running Amazon EC2 instance and exposed as a device within the instance. Amazon EBS is particularly suited for applications that require a database, file system, or access to raw block level storage.
- ***AWS Storage Gateway:*** AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure. The service enables you to securely upload data to the AWS cloud for cost-effective backup and rapid disaster recovery. AWS Storage Gateway supports industry-standard storage protocols that work with your existing applications. It provides low-latency performance by maintaining data on your on-premises storage hardware while asynchronously uploading this data to AWS, where it is encrypted and securely stored in Amazon Simple Storage Service (Amazon S3) or Amazon Glacier. Using AWS Storage Gateway, you can back up point-in-time snapshots of your on-premises application data to Amazon S3 for future recovery. In the event you need replacement capacity for disaster recovery purposes, or if you want to leverage Amazon EC2's on-demand compute capacity for additional capacity during peak periods, for new projects, or as a more cost-effective way to run your normal workloads, you can use AWS Storage Gateway to mirror your on-premises data to Amazon EC2 instances.
- ***AWS Import/Export:*** AWS Import/Export accelerates moving large amounts of data into and out of AWS using portable storage devices for transport. AWS transfers your data directly onto and off of storage devices using Amazon's high-speed internal network and bypassing the Internet. For significant data sets, AWS Import/Export is often faster than Internet transfer and more cost effective than upgrading your connectivity. AWS Import/Export supports importing and exporting data into and out of Amazon S3 buckets in the US East (N. Virginia), US West (Oregon), US West (Northern California), EU (Ireland), and Asia Pacific (Singapore) Regions. The service also supports importing data into Amazon EBS snapshots in the US East (N. Virginia), US West (Oregon), US West (Northern California), EU (Ireland), and Asia Pacific (Singapore) Regions. In addition, AWS Import/Export supports importing data into Amazon Glacier in the US East (N.

Virginia), US West (Oregon), US West (Northern California), and EU (Ireland).AWS Import/Export supports importing and exporting data into and out of Amazon S3 buckets in the US Standard, US West (Oregon), US West (Northern California), EU (Ireland), and Asia Pacific (Singapore) regions. The service also supports importing data into Amazon Elastic Block Store (Amazon EBS) snapshots in the US East (N. Virginia), US West (Oregon), and US West (Northern California) regions.

- ***Amazon Cloud Front:*** Amazon Cloud Front is a content delivery web service. It integrates with other Amazon Web Services to give developers and businesses an easy way to distribute content to end users with low latency, high data transfer speeds, and no commitments. Amazon Cloud Front can be used to deliver your entire website, including dynamic, static and streaming content using a global network of edge locations. Requests for objects are automatically routed to the nearest edge location, so content is delivered with the best possible performance. Amazon Cloud Front is optimized to work with other Amazon Web Services, like Amazon S3 and Amazon EC2. Amazon Cloud Front also works seamlessly with any origin server, which stores the original, definitive versions of your files. Like other Amazon Web Services, there are no contracts or monthly commitments for using Amazon Cloud Front—you pay only for as much or as little content as you actually deliver through the service.

c). Database

- ***Amazon Relational:*** Database Service (Amazon RDS) Amazon Relational Database Service (Amazon RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you up to focus on your applications and business. Amazon RDS gives you access to the capabilities of a familiar MySQL, Oracle, SQL Server or PostgreSQL database. This means that the code, applications, and tools you already use today with your existing databases can be used with Amazon RDS. Amazon RDS automatically patches the database software and backs up your database, storing the backups for a retention period that you define and enabling point-in-time recovery. You benefit from the flexibility of being able to scale the computer resources or storage capacity associated with your relational database instance by using a single API call. In addition, Amazon RDS makes it easy to use replication to enhance availability and reliability for production databases and to scale out beyond the capacity of a single database deployment for read-heavy database workloads.
- ***Amazon Dynamo DB:*** Amazon Dynamo DB is a fast, fully managed NoSQL database service that makes it simple and cost-effective to store and retrieve any amount of data, and serve any level of request traffic. All data items are stored on Solid State Drives

(SSDs), and are replicated across 3 Availability Zones for high availability and durability. With Dynamo DB, you can offload the administrative burden of operating and scaling a highly available distributed database cluster, while paying a low price for only what you use Amazon Dynamo DB is designed to address the core problems of database management, performance, scalability, and reliability. Developers can create a database table that can store and retrieve any amount of data, and serve any level of request traffic. Dynamo DB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent, fast performance. All data items are stored on solid state drives (SSDs) and are automatically replicated across multiple Availability Zones in a Region to provide built-in high availability and data durability. Amazon Dynamo DB enables customers to offload the administrative burden of operating and scaling a highly available, distributed database cluster while only paying a low variable price for the resources they consume.

- ***Amazon Elastic ache:*** Amazon Elastic ache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from a fast, managed, in-memory caching system, instead of relying entirely on slower disk-based databases. Elastic ache supports two open-source caching engines. Memcached - a widely adopted memory object caching system. Elastic ache is protocol compliant with• Memcached, so popular tools that you use today with existing Memcached environments will work seamlessly with the service. Redis – a popular open-source in-memory key-value store that supports data structures such as sorted sets and• lists. Elastic ache supports Redis master / slave replication which can be used to achieve cross AZ redundancy. Amazon Elastic ache automatically detects and replaces failed nodes, reducing the overhead associated with self managed infrastructures and provides a resilient system that mitigates the risk of overloaded databases, which slow website and application load times. Through integration with Amazon Cloud Watch, Amazon Elastic ache provides enhanced visibility into key performance metrics associated with your Memcached or Redis nodes.
- ***Amazon Red shift:*** Amazon Red shift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. It is optimized for datasets ranging from a few hundred gigabytes to a petabyte or more and costs less than $1,000 per terabyte per year, a tenth the cost of most traditional data warehousing solutions. Amazon Red shift delivers fast query and I/O performance for virtually any size dataset by using columnar storage technology and parallelizing and distributing queries across

multiple nodes. We've made Amazon Red shift easy to use by automating most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse. Powerful security functionality is built-in. Amazon Red shift supports Amazon VPC out of the box and you can encrypt all your data and backups with just a few clicks. Once you've provisioned your cluster, you can connect to it and start loading data and running queries using the same SQL-based tools you use today.

d). Analytics

- ***Amazon Elastic Map Reduce***: (Amazon EMR) Amazon Elastic Map Reduce (Amazon EMR is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics. Customers launch millions of Amazon EMR clusters every year.
- ***Amazon Kinesis:*** Amazon Kinesis is a fully managed service for real-time processing of streaming data at massive scale. Amazon Kinesis can collect and process hundreds of terabytes of data per hour from hundreds of thousands of sources, allowing you to easily write applications that process information in real-time, from sources such as web site click-streams, marketing and financial information, manufacturing instrumentation and social media, and operational logs and metering data. With Amazon Kinesis applications, you can build real-time dashboards, capture exceptions and generate alerts, drive recommendations, and make other real-time business or operational decisions. You can also easily send data to a variety of other services such as Amazon Simple Storage Service (Amazon S3), Amazon Dynamo DB, or Amazon Red shift. In a few clicks and a couple of lines of code, you can start building applications which respond to changes in your data stream in seconds, at any scale, while only paying for the resources you use.
- ***AWS Data Pipeline***: AWS Data Pipeline is a web service that helps you reliably process and move data between different AWS compute and storage services as well as on-premise data sources at specified intervals. With AWS Data Pipeline, you can regularly access your data where it's stored, transform and process it at scale, and efficiently transfer the results to AWS services such as Amazon S3, Amazon RDS, Amazon Dynamo DB, and Amazon Elastic Map Reduce (EMR). AWS Data Pipeline helps you easily create complex data processing workloads that are fault tolerant, repeatable, and highly available. You don't have to worry about ensuring resource availability, managing inter-task dependencies, retrying transient failures or timeouts in individual tasks, or creating

a failure notification system. AWS Data Pipeline also allows you to move and process data that was previously locked up in on-premise data silos.

e). Application Services

- ***Amazon AppStream:*** Amazon AppStream is a flexible, low-latency service that lets you stream resource intensive applications and games from the cloud. It deploys and renders your application on AWS infrastructure and streams the output to mass-market devices, such as personal computers, tablets, and mobile phones. Because your application is running in the cloud, it can scale to handle vast computational and storage needs, regardless of the devices your customers are using. You can choose to stream either all or parts of your application from the cloud. Amazon AppStream enables use cases for games and applications that wouldn't be possible running natively on mass-market devices. Using Amazon AppStream, your games and applications are no longer constrained by the hardware in your customer's hands. Amazon AppStream includes a SDK that currently supports streaming applications from Microsoft Windows Server 2008 R2 to devices running FireOS, Android, iOS, and Microsoft Windows. A Mac OS X SDK is planned for 2014.
- Amazon Simple Queue Service (Amazon SQS) Amazon Simple Queue Service (Amazon SQS) is a fast, reliable, scalable, fully managed message queuing service. SQS makes it simple and cost-effective to decouple the components of a cloud application. You can use SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be always available. With SQS, you can offload the administrative burden of operating and scaling a highly available messaging cluster, while paying a low price for only what you use. Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Notification Service (Amazon SNS) Amazon Simple Notification Service (SNS) is a fast, flexible, fully managed push messaging service. SNS makes it simple and cost-effective to push to mobile devices such as iPhone, iPad, Android, Kindle Fire, and internet connected smart devices, as well as pushing to other distributed services. Besides pushing cloud notifications directly to mobile devices, SNS can also deliver notifications by SMS text message or email, to Simple Queue Service (SQS) queues, or to any HTTP endpoint. To prevent messages from being lost, all messages published to Amazon SNS are stored redundantly across multiple availability zones. Amazon Simple Workflow Service (Amazon SWF)
- Amazon Simple Workflow Service (Amazon SWF) is a task coordination and state management service for cloud applications. With Amazon SWF, you can stop writing complex glue-code and state machinery and invest more in the business logic that

makes your applications unique. Our APIs, ease-of-use libraries, and control engine give developers the tools to coordinate, audit, and scale applications across multiple machines – in the AWS Cloud and other data centers. Whether automating business processes for finance applications, building big-data systems, or managing cloud infrastructure services, Amazon SWF helps you develop applications with processing steps that are resilient to failure – steps that can be scaled independent of each other and be audited even when they touch many different systems. Using Amazon SWF, you structure the various processing steps in an application that runs across one or more machines as a set of "tasks." Amazon SWF manages dependencies between the tasks, schedules the tasks for execution, and runs any logic that needs to be executed in parallel. The service also stores the tasks, reliably dispatches them to application components, tracks their progress, and keeps their latest state. As your business requirements change, Amazon SWF makes it easy to change application logic without having to worry about the underlying state machinery, task dispatch, and flow control, and like other AWS Services, you only pay for what you use.

- ***Amazon Simple Email Service (Amazon SES):*** Amazon Simple Email Service (Amazon SES) is a highly scalable and cost-effective bulk and transactional email sending service for organizations and developers. Amazon SES eliminates the complexity and expense of building an in-house email solution or licensing, installing, and operating a third-party email service. The service integrates with other AWS services, making it easy to send emails from applications that are hosted on services such as Amazon EC2. With Amazon SES there is no long-term commitment, minimum spend, or negotiation required. Organizations can utilize a free usage tier and after that enjoy low fees for the number of emails sent plus data transfer fees. Building large-scale email solutions to send marketing and transactional messages is often a complex and costly challenge for organizations. To optimize the percentage of emails that are successfully delivered, organizations must deal with email server management and network configuration, plus they must meet rigorous Internet service provider (ISP) standards for email content. Additionally, many third-party email solutions require contract and price negotiations, as well as significant up-front costs. Amazon SES eliminates these challenges and enables organizations to benefit from the years of experience and sophisticated email infrastructure Amazon.com has built to serve its own large-scale customer base. Using SMTP or a simple API call, an organization can now access a high-quality, scalable email infrastructure to efficiently and inexpensively communicate to their customers. For high email deliverability, Amazon SES uses content filtering technologies to scan an organization's outgoing email messages to help ensure that the content meets ISP standards. The email message is then either queued for sending or routed back to the sender for corrective action. To help organizations further improve the quality of email

communications with their customers, Amazon SES provides a built-in feedback loop, which includes notifications of bounce backs, failed and successful delivery attempts, and spam complaints.

- ***Amazon Cloud Search:*** Amazon Cloud Search is a fully-managed service in the AWS Cloud that makes it easy to set up, manages, and scales a search solution for your website or application. Amazon Cloud Search enables you to search large collections of data such as web pages, document files, forum posts, or product information. With Amazon Cloud Search, you can quickly add search capabilities to your website without having to become a search expert or worry about hardware provisioning, setup, and maintenance. With a few clicks in the AWS Management Console, you can create a search domain, upload the data you want to make searchable to Amazon Cloud Search, and the search service automatically provisions the required technology resources and deploys a highly tuned search index. As your volume of data and traffic fluctuates, Amazon Cloud Search seamlessly scales to meet your needs. You can easily change your search parameters, fine tune search relevance, and apply new settings at any time without having to reupload your data. Amazon Cloud Search enables customers to offload the administrative burden and expense of operating and scaling a search service. With Amazon Cloud Search, there's no need to worry about hardware provisioning, data partitioning, or software patches.
- ***Amazon Elastic Transcoder***: Amazon Elastic Transcoder is media transcoding in the cloud. It is designed to be a highly scalable, easy to use and a cost effective way for developers and businesses to convert (or "transcode") media files from their source format into versions that will playback on devices like smart phones, tablets and PCs. Amazon Elastic Transcoder manages all aspects of the transcoding process for you transparently and automatically. There's no need to administer software, scale hardware, tune performance, or otherwise manage transcoding infrastructure. You simply create a transcoding "job" specifying the location of your source video and how you want it transcoded. Amazon Elastic Transcoder also provides transcoding presets for popular output formats, which means that you don't need to guess about which settings work best on particular devices. All these features are available via service APIs and the AWS Management Console.

f). Deployment and Management

- ***AWS Identity and Access Management (IAM):*** AWS Identity and Access Management (IAM) enable you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups and use permissions to allow and deny their access to AWS resources. IAM allows you to: Manage IAM users and their access - You can create users in IAM, assign them individual

security credentials (i.e., access keys, passwords, and Multi-Factor Authentication devices) or request temporary security credentials to provide users access to AWS services and resources. You can manage permissions in order to control which operations a user can perform. Manage IAM roles and their permissions - You can create roles in IAM, and manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role. You can also define which entity is allowed to assume the role. Manage federated users and their permissions - You can enable identity federation to allow existing identities (e.g. users) in your enterprise to access the AWS Management Console, to call AWS APIs, and to access resources, without the need to create an IAM user for each identity.

- ***AWS Cloud Trail:*** AWS Cloud Trail is a web service that records AWS API calls for your account and delivers log files to you. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. With Cloud Trail, you can get a history of AWS API calls for your account, including API calls made via the AWS Management Console, AWS SDKs, command line tools, and higher-level AWS services (such as AWS Cloud Formation). The AWS API call history produced by Cloud Trail enables security analysis, resource change tracking, and compliance auditing.
- ***Amazon Cloud Watch***: Amazon Cloud Watch provides monitoring for AWS cloud resources and the applications customers run on AWS. Developers and system administrators can use it to collect and track metrics, gain insight, and react immediately to keep their applications and businesses running smoothly. Amazon Cloud Watch monitors AWS resources such as Amazon EC2 and Amazon RDS DB Instances, and can also monitor custom metrics generated by a customer's applications and services. With Amazon Cloud Watch, you gain system-wide visibility into resource utilization, application performance, and operational health. Amazon Cloud Watch provides a reliable, scalable, and flexible monitoring solution that you can start using within minutes. You no longer need to set up, manage, or scale your own monitoring systems and infrastructure. Using Amazon Cloud Watch, you can easily monitor as much or as little metric data as you need. Amazon Cloud Watch lets you programmatically retrieve your monitoring data, view graphs, and set alarms to help you troubleshoot, spot trends, and take automated action based on the state of your cloud environment.
- ***AWS Elastic Beanstalk:*** AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with popular programming languages such as Java, .NET, PHP, Node.js, Python and Ruby. You simply upload your application and Elastic Beanstalk automatically handles the deployment details of

capacity provisioning, load balancing, auto-scaling and application health monitoring. At the same time, with Elastic Beanstalk, you retain full control over the AWS resources powering your application and can access the underlying resources at any time. Most existing application containers or platform-as-a-service solutions, while reducing the amount of programming required, significantly diminish developers' flexibility and control. Developers are forced to live with all the decisions predetermined by the vendor - with little to no opportunity to take back control over various parts of their application's infrastructure. However, with Elastic Beanstalk, you retain full control over the AWS resources powering your application. If you decide you want to take over some (or all) of the elements of their infrastructure, you can do so seamlessly by using Elastic Beanstalk's management capabilities. To ensure easy portability of your application, Elastic Beanstalk is built using familiar application/web servers such as Apache HTTP Server, Apache Tomcat, Nginx, Passenger and IIS 7.5/8.

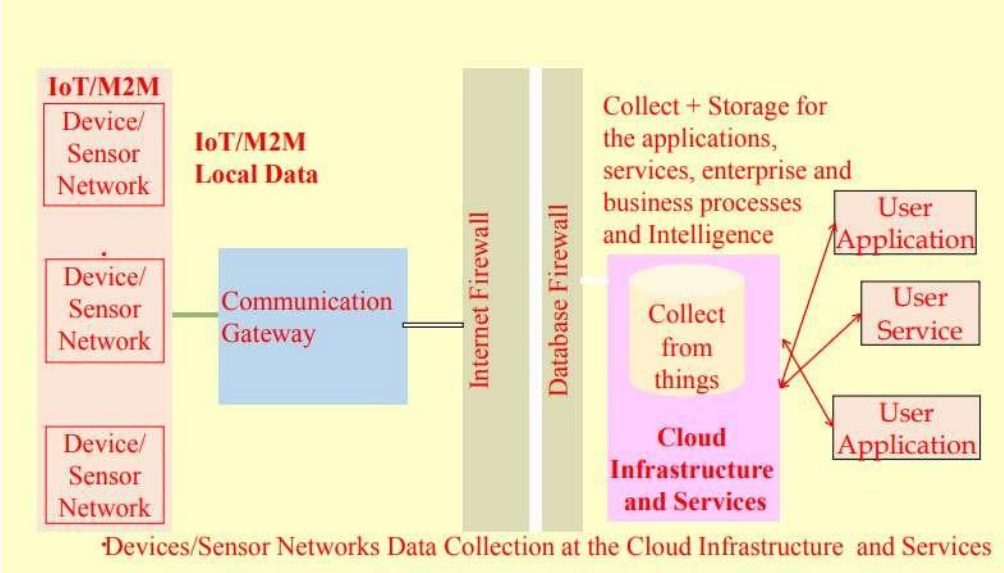
- ***AWS Cloud Formation:*** AWS Cloud Formation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. You can use AWS Cloud Formation's sample templates or create your own templates to describe the AWS resources, and any associated dependencies or runtime parameters, required to run your application. You don't need to figure out the order in which AWS services need to be provisioned or the subtleties of how to make those dependencies work. AWS Cloud Formation takes care of this for you. Once deployed, you can modify and update the AWS resources in a controlled and predictable way. This allows you to version control your AWS infrastructure in the same way as you version control your software. You can deploy and update a template and its associated collection of resources (called a stack) using the AWS Management Console, AWS Cloud Formation command line tools, or Cloud Formation API. AWS Cloud Formation is available at no additional charge, and you pay only for the AWS resources needed to run your applications.
- ***AWS OpsWorks***: AWS OpsWorks AWS OpsWorks is an application management service that makes it easy for DevOps users to model and manage the entire application from load balancers to databases. Start from templates for common technologies like Ruby, Node.JS, PHP, and Java, or build your own using Chef recipes to install software packages and perform any task that you can script. AWS OpsWorks can scale your application using automatic load-based or time-based scaling and maintain the health of your application by detecting failed instances and replacing them. You have full control of deployments and automation of each component.
- ***AWS Cloud HSM:*** The AWS Cloud HSM service helps you meet corporate, contractual and regulatory compliance requirements for data security by using dedicated Hardware

Security Module (HSM) appliances within the AWS cloud. AWS and AWS Marketplace partners offer a variety of solutions for protecting sensitive data within the AWS platform, but for applications and data subject to rigorous contractual or regulatory requirements for managing cryptographic keys, additional protection is sometimes necessary. Until now, your only option was to store the sensitive data (or the encryption keys protecting the sensitive data) in your on-premise datacenters. Unfortunately, this either prevented you from migrating these applications to the cloud or significantly slowed their performance. The AWS Cloud HSM service allows you to protect your encryption keys within HSMs designed and validated to government standards for secure key management. You can securely generate, store, and manage the cryptographic keys used for data encryption such that they are accessible only by you. AWS Cloud HSM helps you comply with strict key management requirements without sacrificing application performance. The AWS Cloud HSM service works with Amazon Virtual Private Cloud (VPC). Cloud HSMs are provisioned inside your VPC with an IP address that you specify, providing simple and private network connectivity to your Amazon Elastic Compute Cloud (EC2) instances. Placing Cloud HSMs near your EC2 instances decreases network latency, which can improve application performance. AWS provides dedicated and exclusive access to Cloud HSMs, isolated from other AWS customers. Available in multiple Regions and Availability Zones (AZs), AWS Cloud HSM allows you to add secure and durable key storage to your Amazon EC2 applications.

2. Xively Cloud (PAAS)

Use of Cloud IoT cloud-based service

- The service provides for the data collection, data points, messages and calculation objects.
- The service also provisions for the generation and communication of alerts, triggers and feeds to the user.
- A user is an application or service. The user obtains responses or feeds from the cloud service.

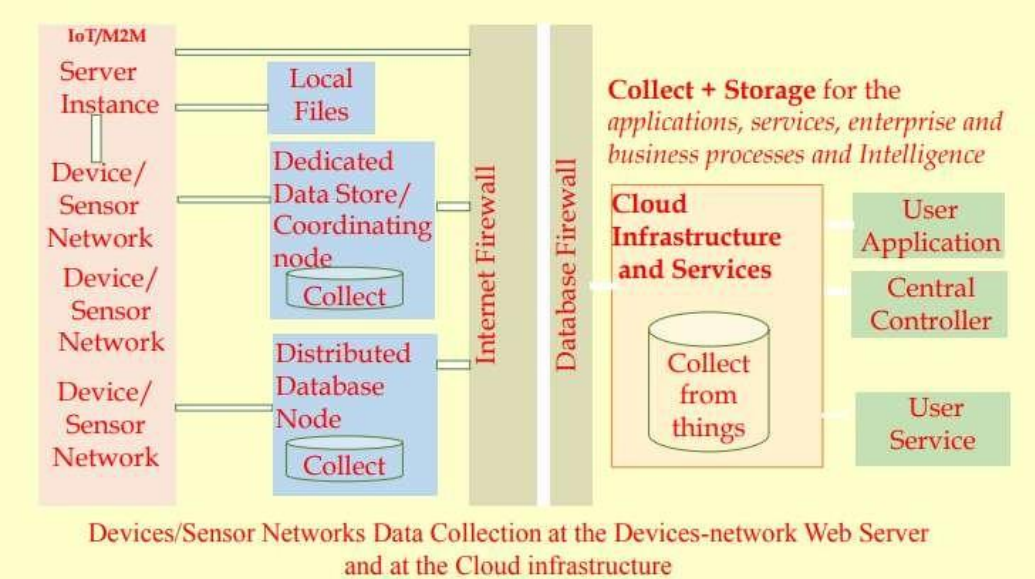


Xively

- Pachube platform: for data capture in real-time over the Internet
- Cosm: a changed domain name, where using a concept of console, one can monitor the feeds
- Xively is the latest domain name
- A commercial PaaS for the IoT/M2M
- A data aggregator and data mining website often integrated into the Web of Things
- An IoT PaaS for services and business services.

Cloud service instance

- Cloud service deploys an instance of server at Arduino and other IoT sensor nodes platforms
- Provision for real-time data collection, data visualization, graphical plots, HTTP based APIs and feed



Xively PaaS services

- Data visualization for data of connected sensors to IoT devices.
- Graphical plots of collected data.
- Generates alerts.
- Access to historical data
- Generates feeds which can be real-world objects of own or others.
- Enables services
- Business services platform which connects the products, including collaboration products
- Rescue, Bold chat, joins. me, and operations to Internet
- Data collection in real-time over Internet

Xively HTTP based APIs

- Easy to implement on device hardware acting as clients to Xively web services
- APIs connect to the web service and send data.
- APIs provides services for logging, sharing and displaying sensor data of all

Xively Support

- •The platform supports the REST, Web Sockets and MQTT protocols and connects the devices to Xively Cloud Services
- Native SDKs for Android, Arduino, ARM mbed, Java, PHP, Ruby, and Python languages
- Developers can use the workflow of prototyping, deployment and management through the tools provided at Xively

Xively APIs

- Enable interface with Python, HTML5, HTML5 server, tornado
- Interface with Web Socket Server and Web Sockets
- Interface with an RPC (Remote Procedure Call)..

Xively Methods for IoT Devices Data

- •Concept of users, feeds, data streams, data points and triggers
- Data feed typically a single location (e.g. a device or devices network),
- Data streams are of individual sensors associated with that location (for example, ambient lights, temperatures, power consumption).
- Pull or Push (Automatic or Manual Feed)

Xively Data formats and Structures

- Number of data formats and structures enable the interaction, data collection and services
- Support exists for JSON , XML (Section 3.3.3) and CSV
- Structures: Tabular, spreadsheet, Excel, Data numbers and Text with a comma-separated values in file

Xively Uses in IoT/M2M

- Private and Public Data Access
- Data streams, Data points and Triggers
- Creating and Managing Feeds
- Visualizing Data

Communication API

- Cloud Models are relied on Communication API
- Communication API facilitate data transfer, control information transfer from application to cloud, one service to another
- It also exist in the form of Communication Protocols
- It supports RPC, PUBSUB and WAMP
- Egg. Popular API is Restful API (communication in cloud model)
- Django web framework is used to implement Communication API

REFERENCES:

1. Shibu K.V, "Introduction to Embedded System", Tata McGraw-Hill, 2014.
2. David E. Simon, "An Embedded Software Primer", Pearson Education Asia, Addison Wesley, 2001.

3. Marilyn Wolf, Computers as Components, Principles of Embedded Computing System Design", Morgan Kaufmann Publishers, Third edition, 2012.
4. Arshdeep Bahga, Vijay Madisetti, "Internet of Things – A hands-on approach", Universities Press, 2015.
5. Manoel Carlos Ramon, "Intel Galileo and Intel Galileo Gen 2: API Features and Arduino Projects for Linux Programmers", Apress, 2014.
6. Marco Schwartz, "Internet of Things with the Arduino Yun", Packt Publishing, 2014.