2 Embedded IoT Devices

Syllabus

Sensors and actuators for IoT applications, IoT components and implementation, Programming of NodeMCU and Raspberry PI, Implementation of IoT with Edge devices, Reading sensor data and transmit to cloud, Controlling devices through cloud using mobile application and web application, Types and configurations of gatways, Specifications of IoT gateways (Practical aspects of this chapter should be covered during lab sessions)

Contents

2.1 Sensors

- Sensor converts a physical quantity into a corresponding voltage. Sensor is a device that when exposed to a physical phenomenon (temperature, displacement, force, etc.) produces a proportional output signal (electrical, mechanical, magnetic, etc.).
- The term transducer is often used synonymously with sensors. Sensor is a device that responds to a change in the physical phenomenon. On the other hand, a transducer is a device that converts one form of energy into another form of energy. Sensors are transducers when they sense one form of energy input and output in a different form of energy.
- Sensors can also be classified as passive or active. In passive sensors, the power required to produce the output is provided by the sensed physical phenomenon itself whereas the active sensors require external power source.
- In embedded system, sensor and actuators are used for controlling the system. Sensors are connected to input port. Actuators are connected to output port.
- Sensor captures the changes in the environmental variable. Middle system process the information. Actuators are changed according to the input variable. It displays the output.
- Example of control is air conditioner system. It controls the room temperature to a specified limit.
- Deflection : The signal produces some physical (deflection) effect closely related to the measured quantity and transuded to be observable.
- Null : The signal produced by the sensor is counteracted to minimize the deflection. That opposing effect necessary to maintain a zero deflection should be proportional to the signal of the measurand.
- Here, the output is usually an 'electrical quantity' and measurand is a 'physical quantity, property or condition which is to be measured'.
- A sensor is a device that responds to a physical stimulus, measures the physical stimulus quantity and converts it into a signal usually electrical, which can be read by an observer or by an instrument.
- A sensor can be very small and itself can be a trackable devices. The sensor itself, if not connected is not part of the IoT or WSN value chain.

Specifications of Sensor :

1. **Accuracy :** Error between the result of a measurement and the true value being measured.

2. **Resolution :** The smallest increment of measure that a device can make.
3. **Sensitivity :** The ratio between the change in the output signal to a small change in input physical signal. Slope of the input-output fit line.
4. **Repeatability/Precision :** The ability of the sensor to output the same value for the same input over a number of trials.
5. **Bandwidth :** The frequency range between the lower and upper cut-off frequencies, within which the sensor transfer function is constant gain or linear.

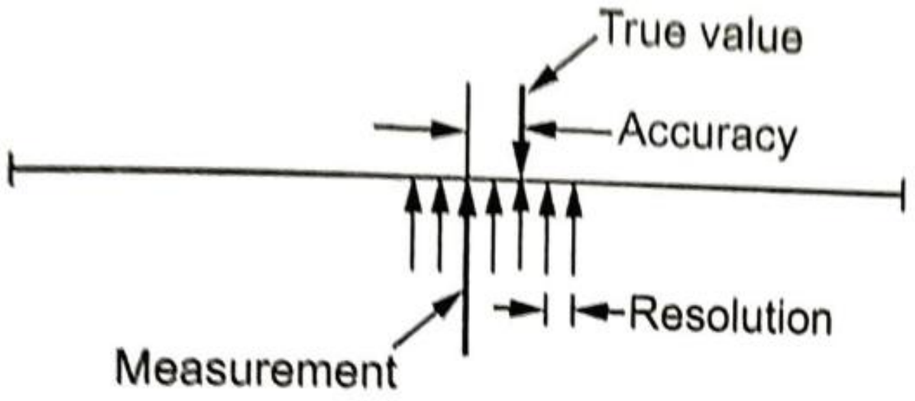


Fig. 2.1.1 Accuracy vs. Resolution

2.1.1 Types of Sensors

- Mechanical sensor : Any suitable mechanical / electrical switch may be adopted but because a certain amount of force is required to operate a mechanical switch it is common to use micro-switches.
- Pneumatic sensor : These proximity sensors operate by breaking or disturbing an air flow. The pneumatic proximity sensor is an example or a contact type sensor. These cannot be used where light components may be blown away.
- Optical sensor : In there simplest form, optimcal proximity sensors operate by breaking a light beam which falls onto a light sensitive device such as a photocell. These are examples of non contact sensors. Care must be exercised with the lighting environment of these sensors for example optical sensors can be blinded by flashes from are welding processes, airborne dust and smoke clouds may impede light transmission etc.
- Electrical sensor : Electrical proximity sensors may be contact or non-contact. Simple contact sensors operate by making the sensor and the component complete an electrical circuit. Non-contact electrical proximity sensors rely on the electrical principles of either induction for detecting metals or capacitance for detecting non metals as well.
- Range sensing : Range sensing concerns detecting how near or far a component is from the sensing position, although they can also be used as proximity sensors. Distance or range sensors use non-contact analog techniques. Short range sensing,

between a few millimetres and a few hundred millimetres is carried out using transmitted energy waves of various types e.g. radio waves, sound waves and lasers.

2.1.2 Sensor Data Communication Protocols

1. Direct transmission protocols :

- In direct communication protocol, each sensor sends its data directly to the base station. If the base station is far away from the nodes, direct communication will require a large amount of transmit power from each node.
- This will quickly drain the battery of the nodes and reduce the system lifetime. However, the only receptions in this protocol occur at the base station, so if either the base station is close to the nodes, or the energy required receiving data is large, this may be an acceptable method of communication.

2. Minimum transfer energy protocols :

- In these protocols, nodes act as routers for other nodes' data in addition to sensing the environment. These protocols differ in the way the routes are chosen.
- Some of these protocols only consider the energy of the transmitter and neglect the energy dissipation of the receivers in determining the routes.
- Depending on the real time costs of the transmit amplifier and the radio electronics, the total energy expended in the system might actually be greater using MTE routing than direct transmission to the base station.
- It is clear that in MTE routing, the nodes closest to the base station will be used to route a large number of data messages to the base station.
- Thus, these nodes will die out quickly, causing the energy required to get the remaining data to the base station to increase and more nodes to die.

2.1.3 Actuators

- A device or mechanism capable of performing a physical action. Actuators interact with the world. Sensors capture information from the world.
- The interface between the microcontroller and the sensors or the actuators is either analog or digital.
- An actuator requires a control signal and a source of energy. An actuator is the mechanism by which a control system acts upon an environment. The control system can be simple, software-based, a human, or any other input.
- When the actuation is a motion, motor have to be used for rotational or linear motion.

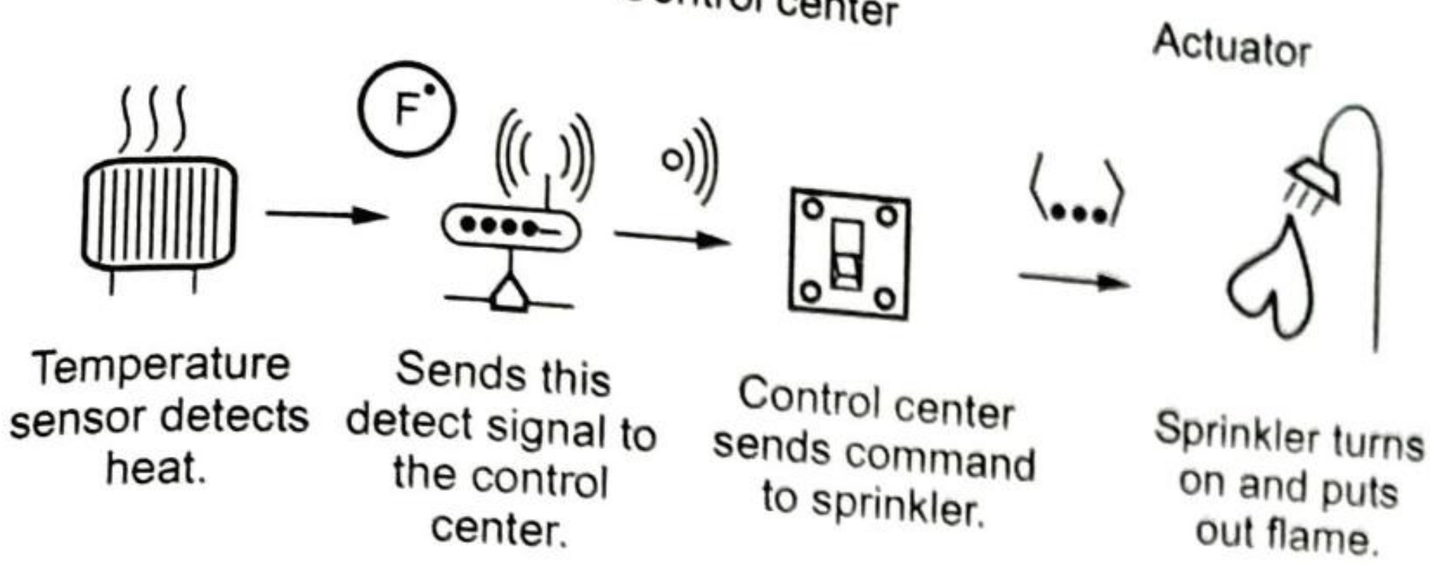


Fig. 2.1.2

- The selection of the proper actuator is more complicated than selection of the sensors, primarily due to their effect on the dynamic behaviour of the overall system. Furthermore, the selection of the actuator dominates the power needs and the coupling mechanisms of the entire system.
- In typical IoT systems, a sensor may collect information and route to a control centre where a decision is made and a corresponding command is sent back to an actuator in response to that sensed input.

2.2 Smart Objects

- Smart objects are an autonomous physical and/or digital object that have sensing, processing, and networking capabilities, and carry application logic.
- They make sense of their local environment and interact with human users.
- They sense, log, and interpret what's occurring within themselves and the world, act on their own, intercommunicate with each other, and exchange information with people.
- Smart objects are small computers with a sensor or actuator and a communication device, embedded in objects such as thermometers, car engines, light switches, and industry machinery.
- Smart objects enable a wide range of applications in areas such as home automation, building automation, factory monitoring, smart cities, structural health management systems, smart grid and energy management, and transportation.
- Until recently, smart objects were realized with limited communication capabilities, such as RFID tags, but the new generation of devices has bidirectional wireless communication and sensors that provide real-time data such as temperature, pressure, vibrations, and energy measurement.
- Smart objects can be battery-operated, but not always, and typically have three components: a CPU (8-, 16- or 32-bit micro-controller), memory and a low-power wireless communication device.
- The size is small and the price is low.

- Advantages in designing IoT systems based on smart objects are as follows :
  1. Energy saving is one of them. Smart objects are usually powered by battery.
  2. The second advantage is automation. IoT smart objects are autonomous and self-governed.
  3. They operate independently and can collaborate with other objects globally.
- Challenges of Using Smart Objects :
  1. Smart objects are often constrained devices and are usually powered by battery.
  2. Frequently they are working in real-time mode. These are the main causes of the challenges.
  3. Other challenge is connectivity. Currently a large number of networking technologies are being employed in connecting physical devices together and to the Internet.
  4. Security and privacy is of big concern for smart object based IoT systems.
  5. Diversity of communication technologies : Depending on the application and the environment in which the system is deployed, smart objects can use a wide range of communication technologies

2.2.1 Communication Patterns used for Smart Objects

- Smart object communication patterns can be divided into three categories : one-to-one, one-to-many and many-to-one.
- Smart objects have specific communication patterns based on their application
- Smart objects often communicate over unreliable communication channels. The radio transmission of a smart object with a radio transceiver may be disturbed by other radio senders in the vicinity.
  1. **One-to-one communication** pattern occurs when one smart object communicates with another smart object. The communication may involve other smart objects, however, as the communication may be routed through a network of smart objects. Fig 2.2.1 shows one-to-one communication in a smart object network.

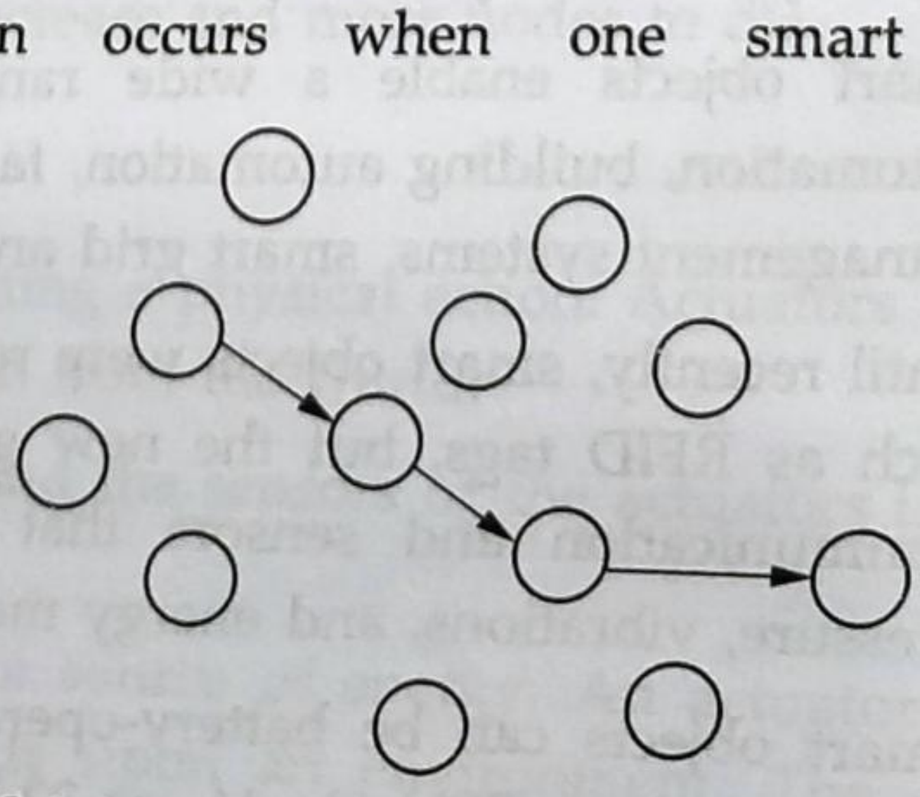


Fig. 2.2.1 One-to-one communication in a smart object network

2. **One-to-many communication** pattern is used for sending messages from one node to several other nodes and possibly all other nodes in the network. This can be used, for example, for sending a command to a set of nodes in the network. Fig 2.2.2 shows one-to-many communication in a smart object network.

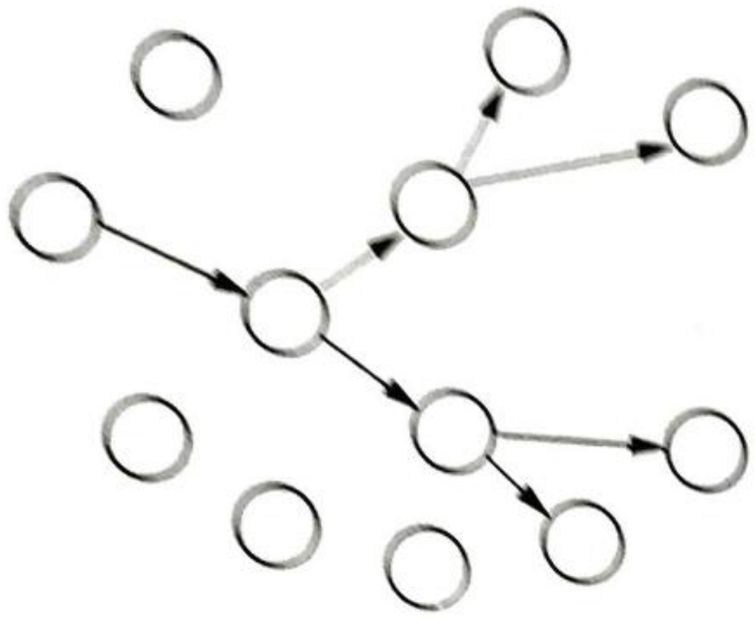


Fig. 2.2.2 One-to-many communication in a smart object network

3. **Many-to-one communication** occurs frequently in smart object networks that collect data from the nodes. In many-to-one communication, several nodes send data toward a single node. This node is often called a sink node. Fig 2.2.3 shows Many-to-one communication in a smart object network.

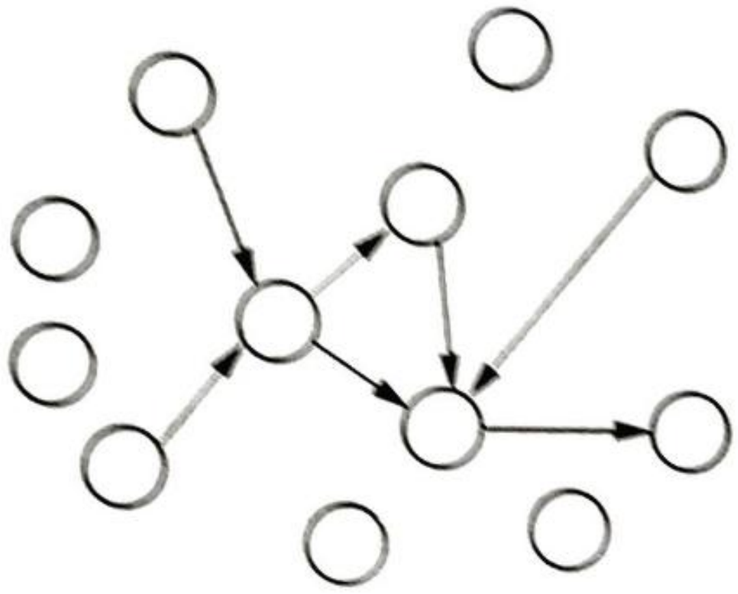


Fig. 2.2.3 Many-to-one communication in a smart object network

- Many-to-one communication can be used to collect sensor data, such as temperature data, from the nodes in the network, but it can also be used for network health status information.
- Nodes send periodic status reports to a sink node. The sink node then reports the overall performance of the network to an outside observer.

2.2.2 Connecting Smart Objects

- Range is one the communication criteria for smart objects. Range may be short, medium and long.

1. **Short range :** Classical example is serial cable. It supporting upto 10meters as maximum distance between two devices. Example of wireless technologies are Bluetooth and visible light communication.
2. **Medium range :** Its range is 10 to 100 meters. It is main categories of IoT access technologies. Maximum distance between two devices is less than 1 mile. Example is 802.11 Wi-Fi, IEEE 802.15.4 and 802.15.4g WPAN. For wired technology, 802.3 Ethernet is an example.
3. **Long range :** Here the distance is greater than 1 miles between two devices. Example of wireless technology are 2G, 3G, 4G etc.

Frequency bands :

- Frequency band is a specific range of frequencies in the radio frequency (RF) spectrum, which is divided among ranges from very low frequencies (vlf) to extremely high frequencies (ehf). Each band has a defined upper and lower frequency limit.
- Very low frequencies (vlf) range from 3 to 30 kilohertz (kHz). Time signals and standard frequencies are among the users of this band.
- Low frequencies (lf) range from 30 to 300 kHz. Fixed, maritime mobile and navigational systems and radio broadcasting are among the users of this band.
- Medium frequencies (mf) range from 300 to 3000 kHz. Land, maritime mobile and radio broadcasting are among the users of this band.
- High frequencies (hf) - also called shortwaves - range from 3 to 30 megahertz (MHz). Fixed, mobile, aeronautical and marine mobile, amateur radio, and radio broadcasting are among the users of this band.
- Cellular networks operate on different frequency bands including the 450 MHz band, 700 MHz band, 800 MHz band, 900 MHz band, 1800 MHz band, 2100 MHz band, and 2600 MHz band.
- The industrial, scientific, and medical radio band (ISM band) refers to a group of radio bands or parts of the radio spectrum that are internationally reserved for the use of radio frequency (RF) energy intended for scientific, medical and industrial requirements rather than for communications.
- In the U.S., the 902 - 928 MHz, 2.4 GHz and 5.7 - 5.8 GHz bands were initially used for machines that emitted radio frequencies, such as RF welders, industrial heaters and microwave ovens, but not for radio communications.

Power Consumption :

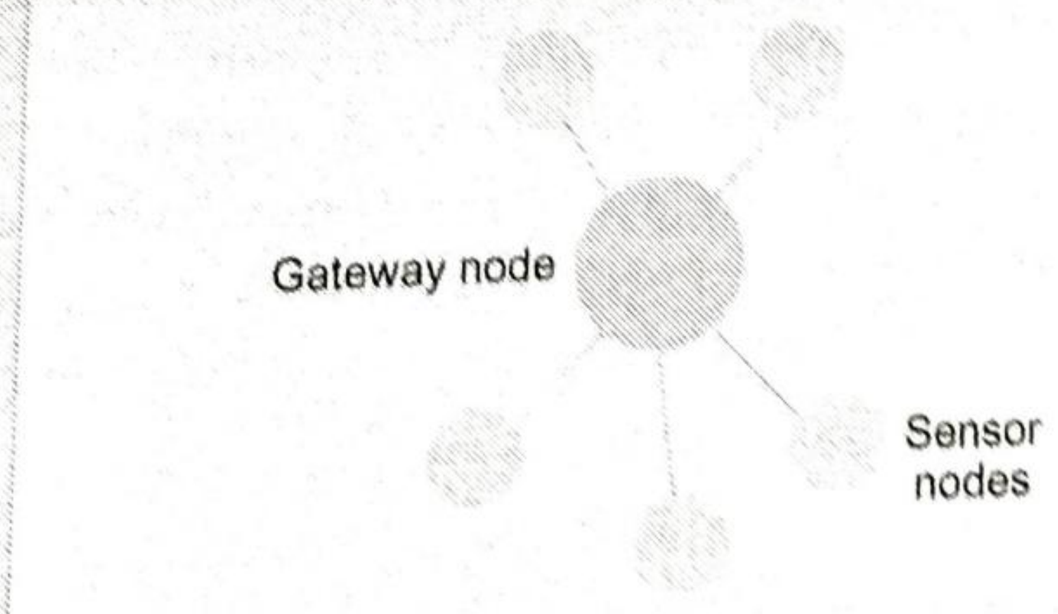
- IoT uses two types of devices : Powered node and battery powered node.

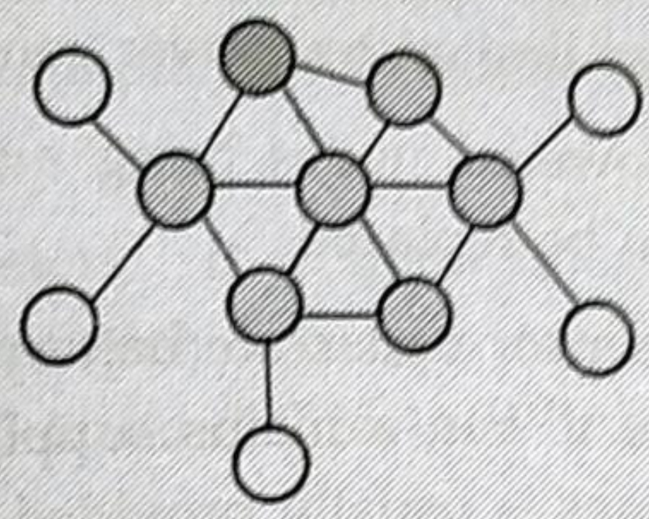
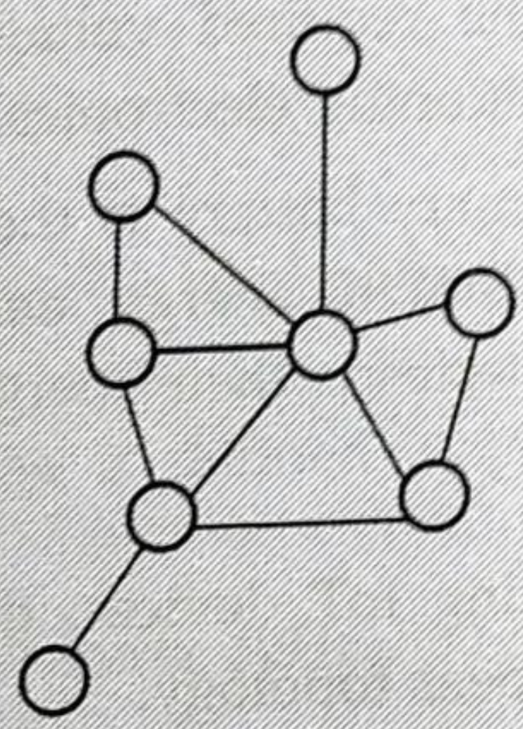
- Powered node has a direct connection to a power source and communications are usually not limited by power consumptions criteria.
- Battery powered nodes bring much more flexibility. Batteries can be changed or device can be replaced.
- Wireless sensor nodes are battery-powered devices, since it is generally difficult or impossible to run a mains supply to their deployment site. Power to the wireless sensor nodes is usually provided through primary batteries
- Primary batteries are typical power sources for sensor nodes. By supplying their energy at the required voltage levels, they eliminate the need for intermediate power conditioning electronics.
- When a primary battery is the single power source of a sensor node, the amount of initially stored energy determines the node's lifetime. The main metric of primary batteries is their energy density.

Topology :

- For connecting IoT devices, three main topologies are used. They are star, mesh and pee-to-peer topology.
- In the star network topology, a central node has a direct connection to all other nodes.
- In the mesh topology every node can be both an end device or a router, meaning that each node has several links to the coordinator. This means that if one of the routers goes offline, most of the network is still intact by rerouting through the remaining routers.
- Peer-to-peer topology rely on multiple full function devices.
- A network is said to be fully meshed if all nodes are directly connected to all other nodes, and partially meshed if only some nodes have multiple connections to others. Meshing to create multiple paths increases resiliency under failure, but increases cost.

Star topology



Mesh topology	
Peer to Peer topology	

- Peer-to-peer topology rely on multiple full function devices.
- A network is said to be fully meshed if all nodes are directly connected to all other nodes, and partially meshed if only some nodes have multiple connections to others. Meshing to create multiple paths increases resiliency under failure, but increases cost.

Constrained devices

- Following are the classes of constrained nodes as defined by RFC 7228

1. Class 0 devices : They are so severely constrained in memory and processing capabilities that most likely they will not have the resources required to communicate directly with the Internet in a secure manner. Class 0 devices will participate in Internet communications with the help of larger devices acting as proxies, gateways, or servers. Class 0 devices generally cannot be secured or managed comprehensively in the traditional sense. They will most likely be preconfigured with a very small data set. For management purposes, they could answer keepalive signals and send on/ off or basic health indications.

2. Class 1 devices are quite constrained in code space and processing capabilities, such that they cannot easily talk to other Internet nodes employing a full protocol stack such as using HTTP, Transport Layer Security (TLS), and related security protocols and XML-based data representations. However, they are capable enough to use a protocol stack specifically designed for constrained nodes and participate in meaningful conversations without the help of a gateway node. In particular, they can provide support for the security functions required on a large network.
3. Class 2 devices are less constrained and fundamentally capable of supporting most of the same protocol stacks as used on notebooks or servers. However, even these devices can benefit from lightweight and energy-efficient protocols and from consuming less bandwidth. Furthermore, using fewer resources for networking leaves more resources available to applications. Thus, using the protocol stacks defined for more constrained devices on Class 2 devices might reduce development costs and increase the interoperability.

2.3 IoT System Building Blocks

- The hardware utilized in IoT systems includes devices for a remote dashboard, devices for control, servers, a routing or bridge device, and sensors. These devices manage key tasks and functions such as system activation, action specifications, security, communication, and detection to support-specific goals and actions.
- Major components of IoT devices are as follows :
  1. **Control units :** A small computer on a single integrated circuit containing processor core, memory and a programmable I/O peripheral. It is responsible for the main operation.
  2. **Sensor :** Devices that can measure a physical quantity and convert it into a signal, which can be read and interpreted by the microcontroller unit. These devices consist of energy modules, power management modules, RF modules, and sensing modules. Most sensors fall into 2 categories: Digital or analog. An analog data is converted to digital value that can be transmitted to the Internet.
     a. Temperature sensors : Accelerometers
     b. Image sensors : Gyroscopes
     c. Light sensors : Acoustic sensors
     d. Micro flow sensors : Humidity sensors
     e. Gas RFID sensors : Pressure sensors

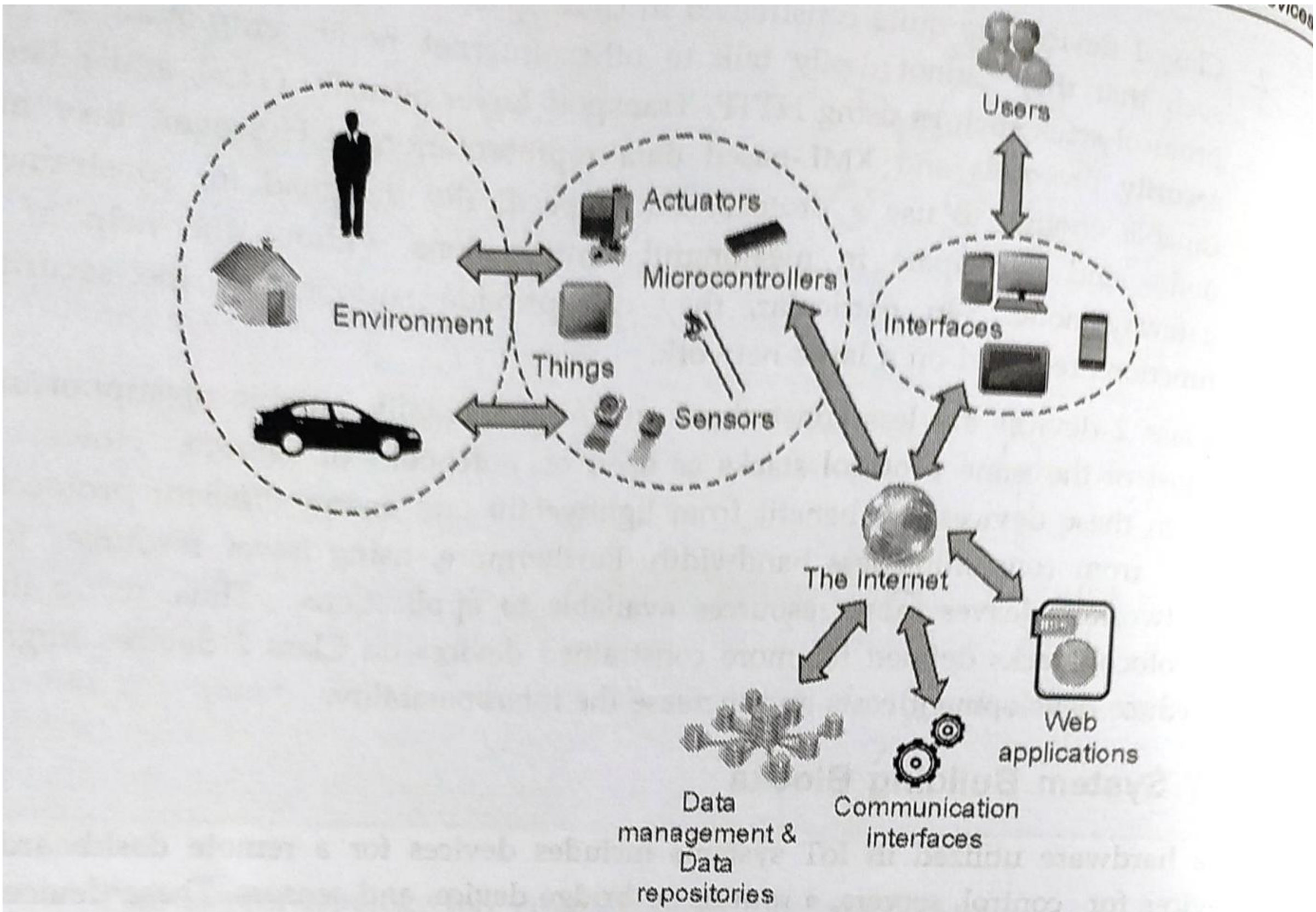


Fig. 2.3.1 Working of IoT

3. **Communication modules :** These are the part of devices and responsible for communication with rest of IoT platform. They provide connectivity according to wireless or wired communication protocol they are designed. The communication between IoT devices and the Internet is performed in two ways :

   A) There is an Internet-enable intermediate node acting as a gateway;

   B) The IoT Device has direct communication with the Internet.

- The communication between the main control unit and the communication module uses serial protocol in most cases.

4. **Power sources :** In small devices the current is usually produced by sources like batteries, thermocouples and solar cells. Mobile devices are mostly powered by lightweight batteries that can be recharged for longer life duration.

- **Communication Technology and Protocol :** IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

Working :

1. **Collect and transmit data :** The device can sense the environment and collect information related to it and transmit it to a different device or to the Internet.
2. **Actuate device based on triggers :** It can be programmed to actuate other devices based on conditions set by user.
3. **Receive information :** Device can also receive information from the network.
4. **Communication assistance :** It provides communication between two devices of same network or different network.

- Fig. 2.3.1 shows working of IoT.
- Sensors for various applications are used in different IoT devices as per different applications such as temperature, power, humidity, proximity, force etc.
- Gateway takes care of various wireless standard interfaces and hence one gateway can handle multiple techologies and multiple sensors. The typical wireless technologies used widely are 6LoWPAN, Zigbee, Zwave, RFID, NFC etc. Gateway interfaces with cloud using backbone wireless or wired technologies such as WiFi, Mobile , DSL or Fibre.

2.4 Arduino

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.
- The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

Features :

- Support fast computations, ARM based MCU
- AVR micro-controller clock is ATSAMSX8I
- Operating input voltages is 3.3 Volt
- It uses EEPROM, SRAM and Flash memory
- It also support USB and UART

- Fig. 2.4.1 shows Arduino board.

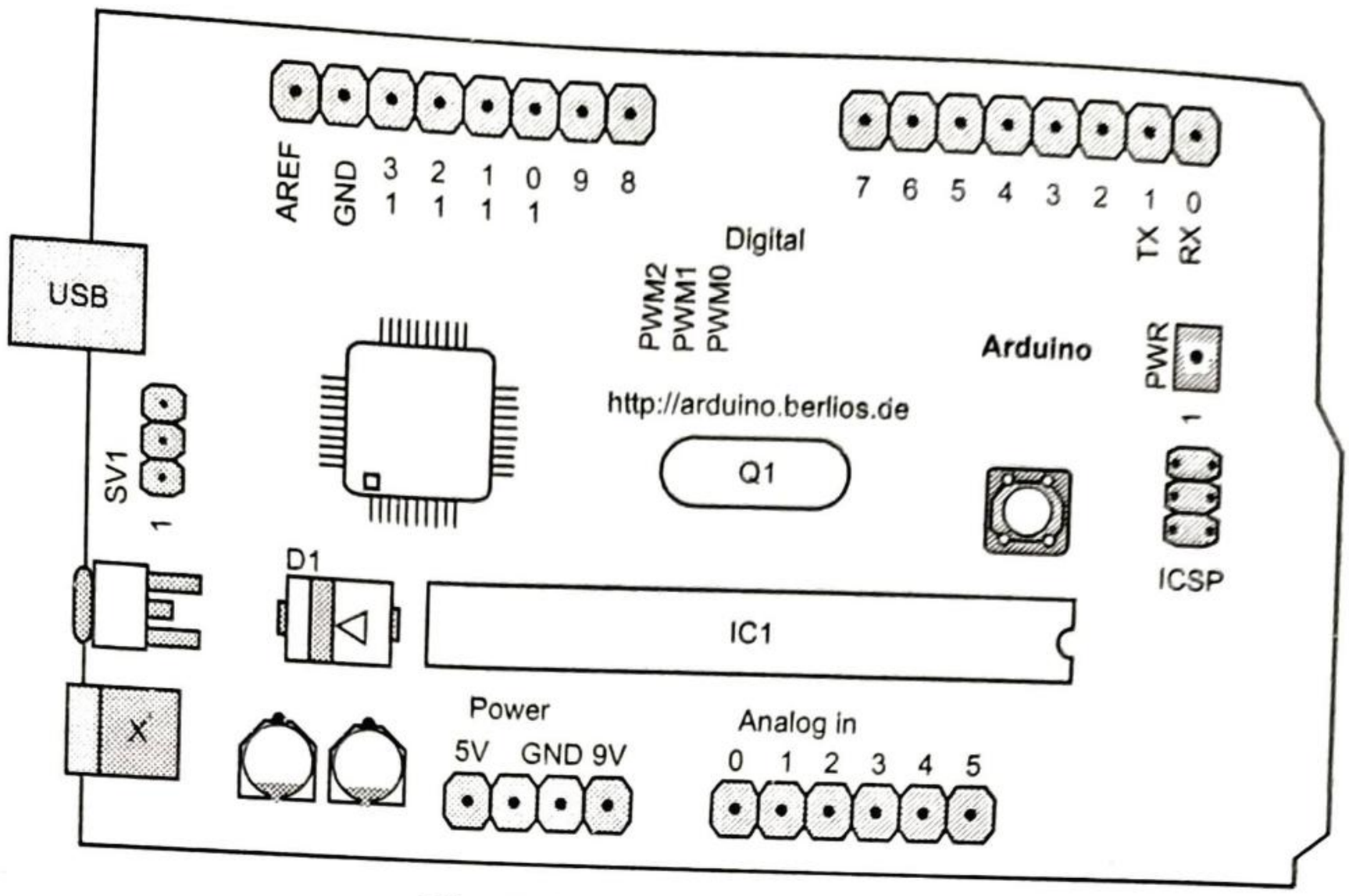


Fig. 2.4.1 Arduino Board

Starting clockwise from the top center,

- Analog Reference pin (1st pin)
- Digital Ground
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX : These pins cannot be used for digital I/O (digital Read and digital Write) if you are also using serial communication
- Reset Button - S1
- In-circuit Serial Programmer
- Analog In Pins 0-5
- Power and Ground Pins
- External Power Supply In (9-12VDC) - X1
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB

Digital Pins

- In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode( ), digitalRead( ), and digitalWrite( ) commands.
- Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite( ), when the pin is configured as an input. The maximum current per pin is 40 mA.

- **Serial :** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
- **External Interrupts :** 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM :** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT Reset :** 7. (Arduino BT-only) Connected to the reset line of the bluetooth module.
- **SPI :** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED :** 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog Pins

- In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function.
- Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.
- **I2C :** 4 (SDA) and 5 (SCL). Support I2C (TWI) communication.

Power Pins

- VIN (sometimes labelled "9 V"). The input voltage to the Arduino board when it's using an external power source.
- You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.

- **5 V :** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5 V supply.
- **3V3 :** (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- **GND :** Ground pins.

Other Pins

- **AREF :** Reference voltage for the analog inputs. Not currently supported by the Arduino software.
- **Reset :** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7 V, however, the 5 V pin may supply less than five volts and the board may be unstable. If using more than 12 V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Arduino Uno R3 Programming

- The programming of an Arduino Uno R3 can be done using IDE software. The microcontroller on the board will come with pre-burned by a boot loader that permits to upload fresh code without using an exterior hardware programmer.
- The communication of this can be done using a protocol like STK500.
- We can also upload the program in the microcontroller by avoiding the boot loader using the header like the In-Circuit Serial Programming.

2.5 Raspberry Pi

- A Raspberry Pi is a credit card-sized computer originally designed for education, inspired by the 1981 BBC Micro.
- Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at the pre-university level.
- The Raspberry Pi is slower than a modern laptop or desktop but is still a complete Linux computer and can provide all the expected abilities that implies, at a low-power consumption level.

Versions	Remarks
Raspberry Pi 1	• The original Raspberry Pi had 256 Mb of RAM, which increased to 512 MB in a later revision. • It has a 26-way GPIO connector
Pi Zero	• The Pi Zero includes the GPIO connector, but the header pins are not soldered
Raspberry Pi 2	• The Raspberry Pi 2 swapped the single-core processor for a much faster quad-core processor and increased the memory to 1 GB RAM
Raspberry Pi 3	• The Raspberry Pi 3 changes the processor to an even more powerful 64-bit processor. • It also adds Wi-Fi and bluetooth which previously needed to be added as a USB device. • The Raspberry Pi 3 Model B was launched in February 2016.

- To get the Raspberry Pi working an SD card needs to be prepared with the Linux operating system installed.
- Raspberry Pi users have made many creative and impressive projects using this device. It can also be programmed to assist in 'housekeeping' your network by functioning as NAS, LDAP server, web server, media server, DNS server etc.
- The Raspberry Pi Foundation recommends Python. Any language which will compile for ARMv6 can be used. Installed by default on the Raspberry Pi : C, C++, Java, Scratch and Ruby.

2.5.1 About the Board

- Fig 2.5.1 shows the Raspberry Pi board. The Raspberry Pi does not have a separate CPU, RAM or GPU. Instead they are all squeezed into one component called a system on Chip or SoC unit.
- Raspberry Pi is open hardware with the exception of its primary chip, the Broadcomm SoC which runs the main components of the board - CPU, graphics, memory, USB controller etc.

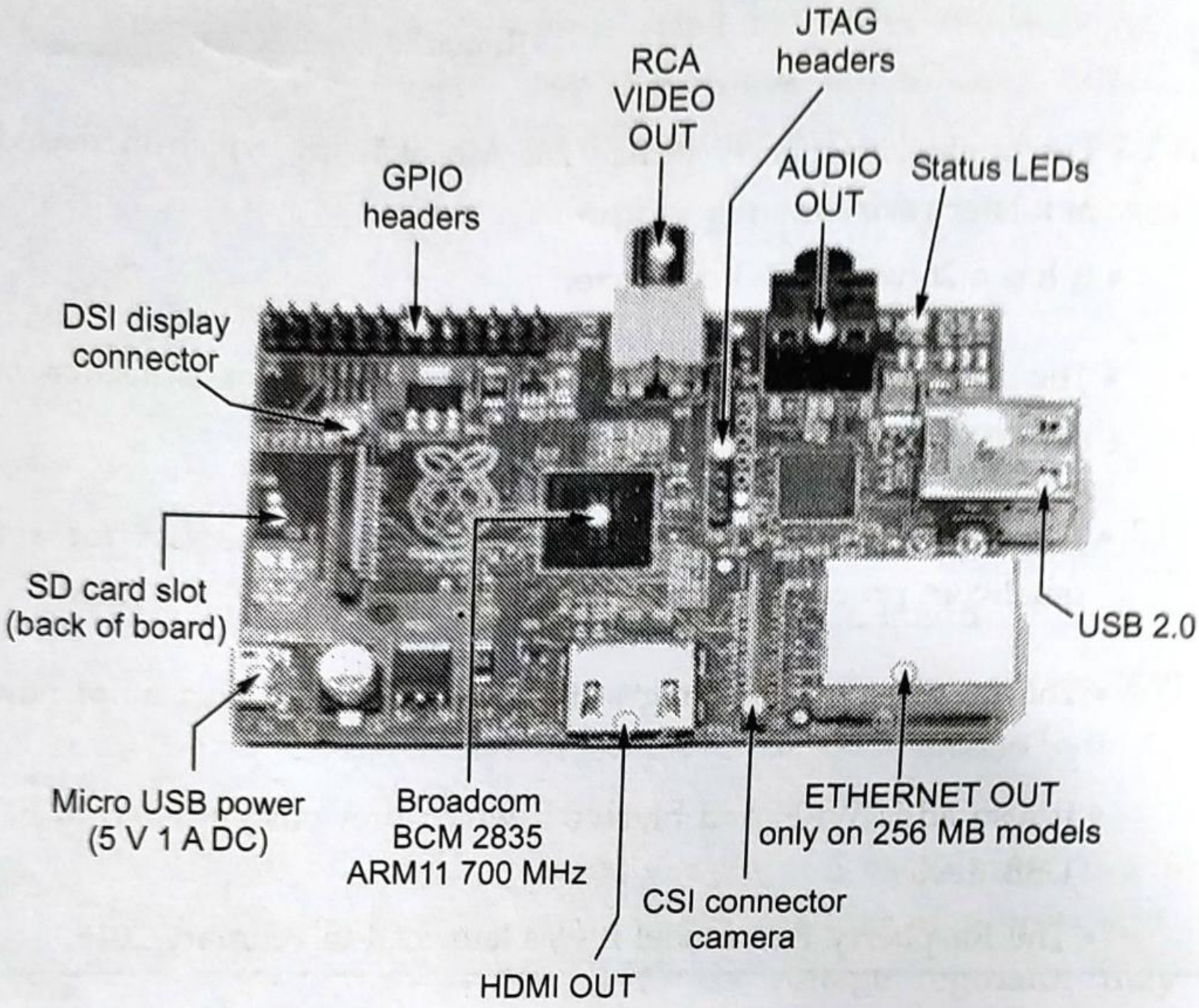


Fig 2.5.1 (a) Raspberry Pi circuit board

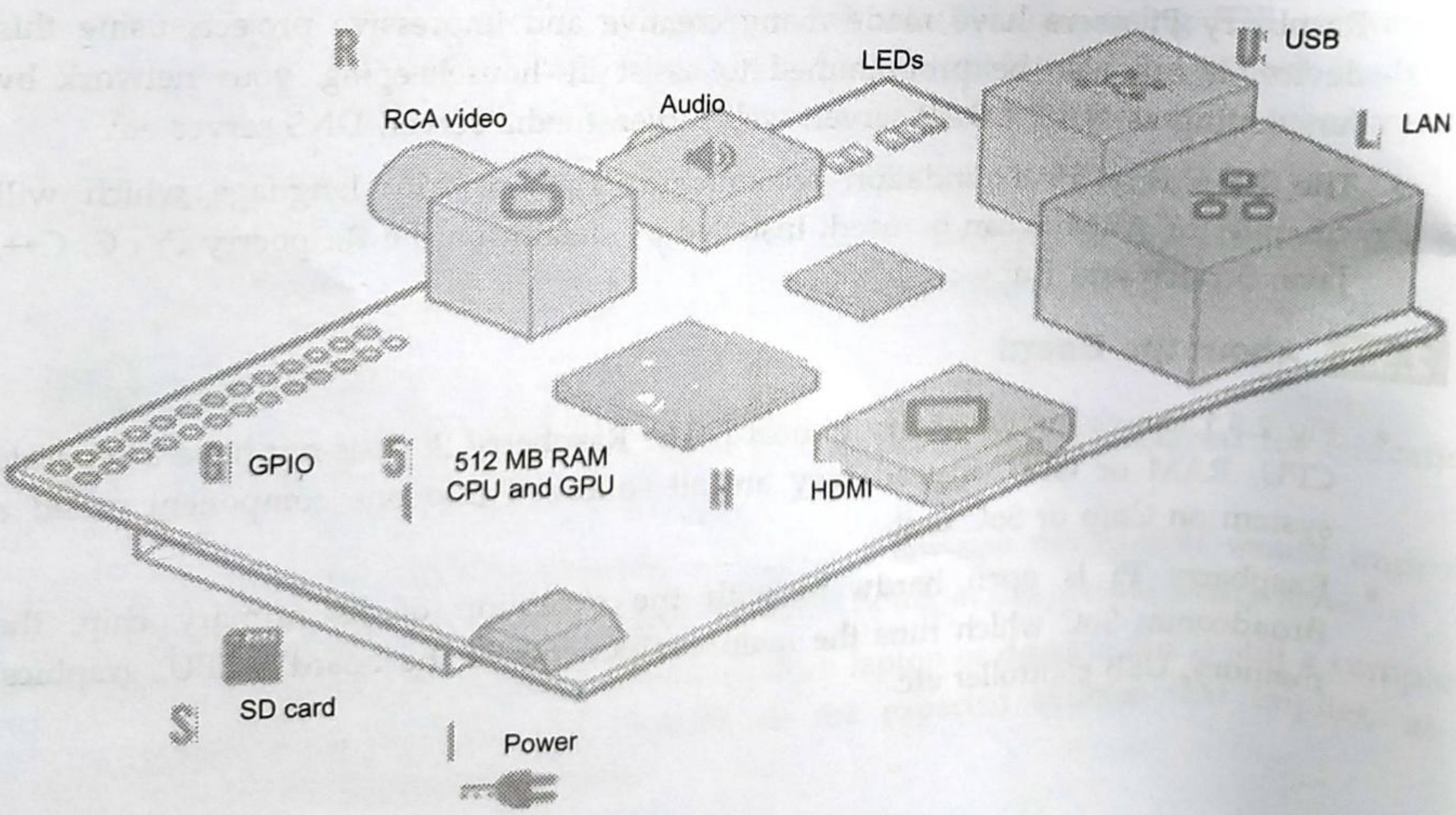


Fig 2.5.1 (b) Block diagram

- All of these Raspberry Pi Models share the following features :

1. Operating systems : Raspbian RaspBMC, Arch Linux, Rise OS, OpenELEC Pidora
2. Video output : HDMI Composite RCA
3. Supported resolutions : 640x350 to 1920x1200, including 1080p, PAL and NTSC standards
4. Power source : Micro USB

Components	Description
Processor	• Raspberry Pi uses an ARM processor which is also installed in a wide variety of mobile phones. • This CPU is single core, however it does have a co-processor to perform floating point calculations
Memory	• Model B Raspberry Pi has 512 MB SDRAM (Synchronous Dynamic RAM). • It store programs that are currently being run in the CPU
USB ports	• Board has two USB ports. USB port can provide a current upto 100 mA • Using powered hub, it is possible to connect more devices.
HDMI Output	• High Definition Multimedia Interface (HDMI) supports high-quality digital video and audio through a single cable. • It is also possible to connect a computer monitor with a DVI connection to HDMI using a converter.
Composite Video Output	• It support composite video output with RCA jack and also support PAL and NTSC. • The TVDAC pin can be used to output composite video.
Audio Output	• Audio output jack is 3.5 mm. • This jack is used for providing audio output to old television along with the RCA jack for video
GPIO Pins	• Both models have a total of 26 GPIO pins, organized into one pin header, named the P1 header • The newer Raspberry Pi (model B revision 2) adds 8 more GPIO pins in a new pin header called P5

	• Not all the GPIO pins are programmable. Some of them are 5.0 VDC or 3.3 VDC positive power pins, some of them are negative ground pins and a few of them are marked DNC (do not connect). • The P1 header has 17 programmable pins and the P5 header adds 4 more. • Fig 2.5.2 shows GPIO pin header. • Reading from various environmental sensors. Writing output to dc motors, LEDs for status.
Power Input	• Micro-USB connector is used for power input
Status LED	• It has five status LED.
CSI	• Camera Serial Interface (CSI) can be used to connect a camera module to Raspberry Pi
SD Card Slot	• This card is used for loading operating system

- The Raspberry Pi comes with a set of 26 exposed vertical pins on the board. These pins are a General Purpose Input /Output interface that is purposely not linked to any specific native function on the Raspberry Pi board.

Raspberry Pi P1 header

PIN #	Name			Name	PIN #
	3.3 VDC power	1	2	5.0 VDC power	
8	SDA0 (I2C)	3	4	DNC	
9	SCL0 (I2C)	5	6	0 V (Ground)	
7	GPIO 7	7	8	TxD	15
	DNC	9	10	RxD	16
0	GPIO 0	11	12	GPIO1	1
2	GPIO 2	13	14	DNC	
3	GPIO 3	15	16	GPIO4	4
	DNC	17	18	GPIO5	5
12	MOSI	19	20	DNC	
13	MISO	21	22	GPIO6	6
14	SCLK	23	24	CE0	10
	DNC	25	26	CE1	11

Fig. 2.5.2 GPIO pin header

- Instead, the GPIO pins are there explicitly for the end user to have low-level hardware access directly to the board for the purposes of attaching other hardware boards, peripherals, LCD display screens and other hardware devices to the Pi.

The Status LEDs

Status LED	Color	Functions
ACT	Green	Lights when the SD card is accessed (marked OK on earlier boards)
PWR	Red	Hooked up to 3.3 V power
FDX	Green	On if network adapter is full duplex
LNK	Green	Network activity light
100	Yellow	On if the network connection is 100 Mbps

- The Raspberry Pi draws its power from a microUSB port and requires a microUSB-to-AC adapter. Because the Pi is a micro computer and not simply a cell phone getting a battery topped off, you need to use a high quality charger with stable power delivery that provides a consistent 5 V with at least 700 mA minimum output for older model units and 2.5 A for the Pi 3.

2.5.2 Linux on Raspberry Pi

- There are several unix like operating systems for the RPI and there is an operating system called RISC OS that has its origin at the developers of the first ARM chips.
- The Raspberry Pi Foundation recommends the use of the following Linux Distributions :
  1. Debian 7
  2. Raspbian
  3. Arch Linux ARM
  4. QtonPi
- Raspbian is a free operating system based on Debian optimized for the Raspberry Pi (RPI) hardware.
- The default command prompt on the Pi consists of four components shown in Fig. 2.5.3.
- Raspbian is the desired operating system for the Raspberry Pi. In order to download and install the operating system onto our Raspberry Pi ; you will need Raspbian, Win32DiskImager and USB memory card reader.

1. Download both Raspbian and Win32DiskImager and save somewhere easily accessible
2. Plug the USB memory card reader into your computer
3. Open Win32DiskImager
4. Find the location of the image file and the memory card
5. Click "Write"

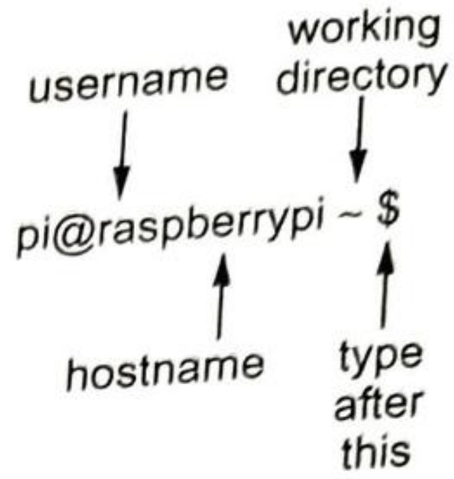


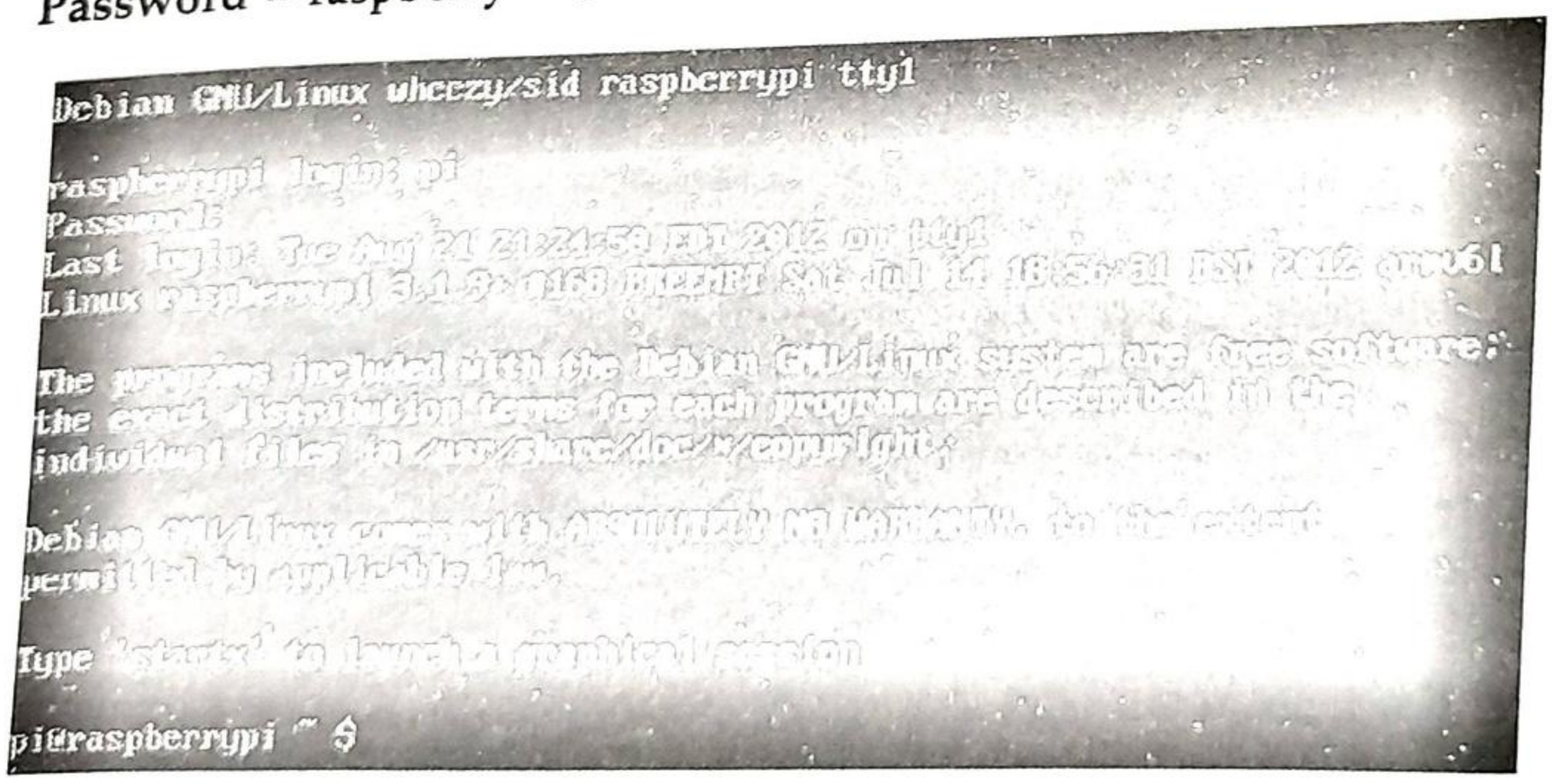
Fig. 2.5.3 Command prompt

Logging In

- Now it is time to turn on our Raspberry Pi. When the memory card, HDMI lead, Ethernet cable, mouse and keyboard are plugged in, plug in the power lead.
- As soon as you do this. You screen should be black and filled with white text. This will be visible every time you turn on your raspberry pi.
- Wait until your screen reads "raspberrypi login :"

  Username = pi [ENTER]

  Password = raspberry [ENTER]



Starting the Raspbian GUI

- *GUI stands for Graphical User Interface and is a type of operating system. It is the most common type of user interface as it is a very 'friendly' way for people to interact with the computer. It makes use of pictures, graphics, icons and pointers, hence the name 'Graphical' User Interface. Fig. 2.5.4 shows Rasbian Linux desktop*

1. **Type the line : "startx"**

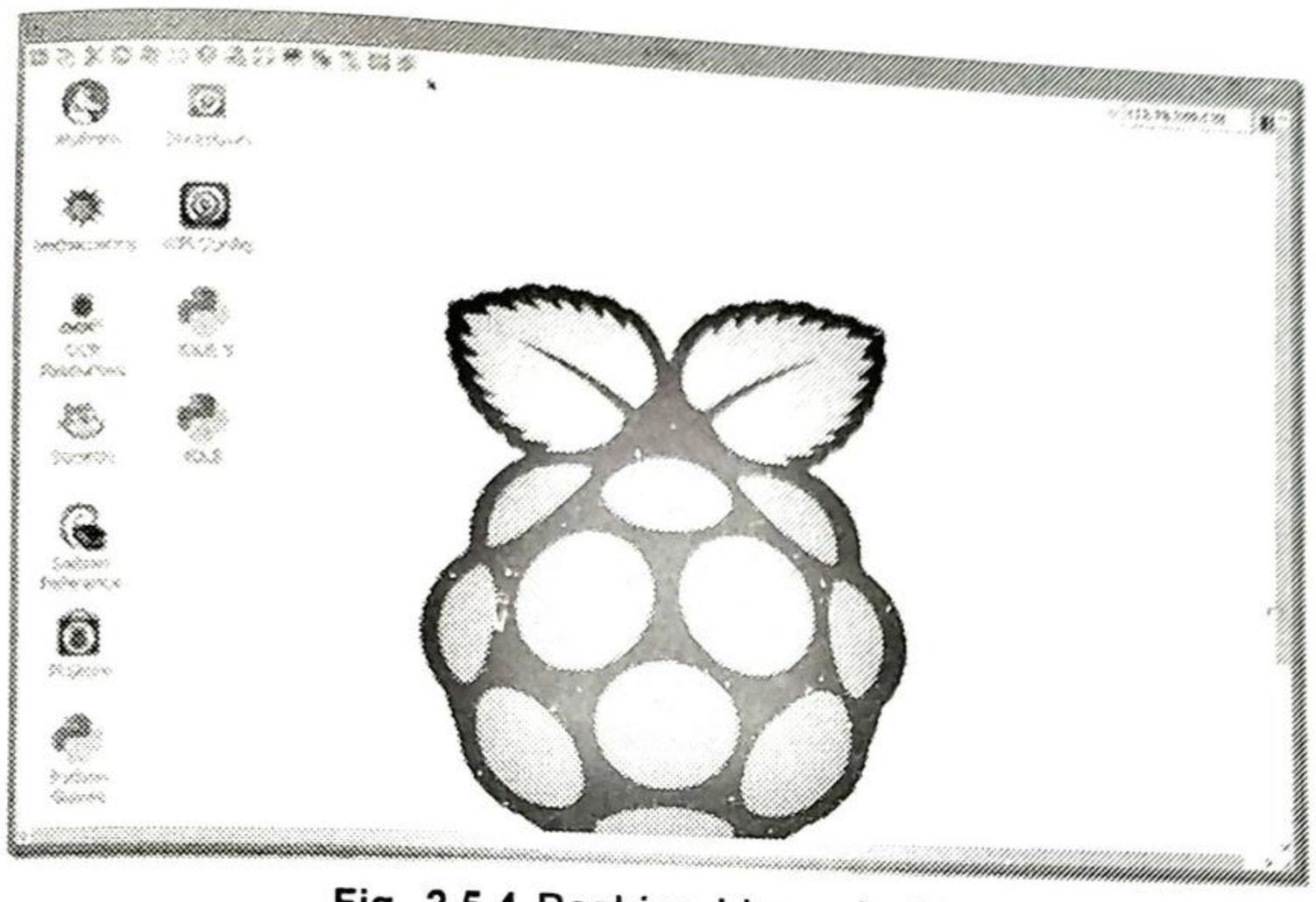


Fig. 2.5.4 Rasbian Linus desktop

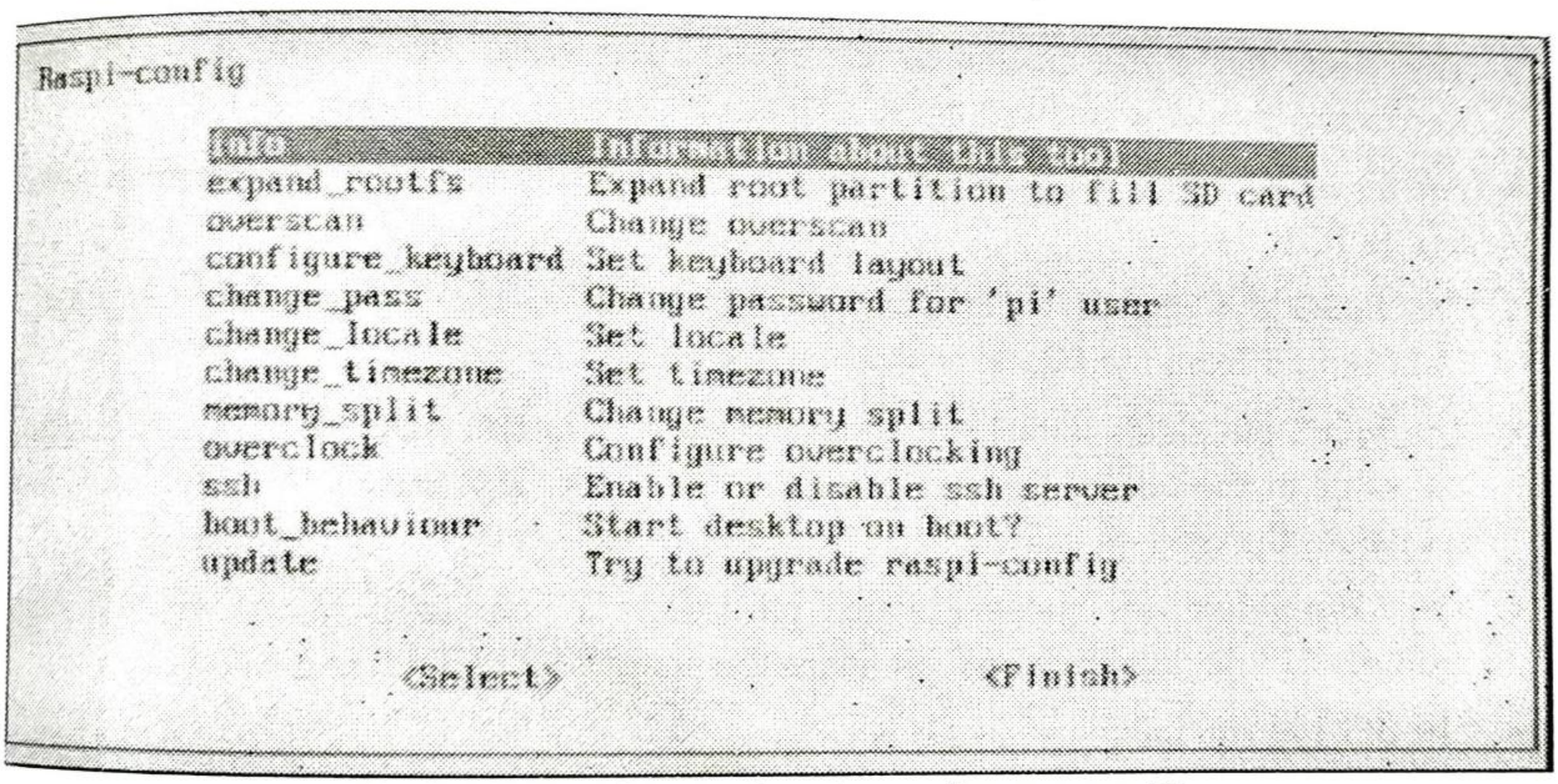


Fig. 2.5.5 First boot : time to configure your Pi

2.5.3 Difference between Raspberry Pi is and Desktop Computers

- In Raspberry Pi, operating system is installed on SD card whereas in desktop computer, operating system is installed in hard disk.
- Raspberry Pi does not have their own CPU and RAM.
- Processing power of Raspberry Pi is less as compared to desktop computers.
- Raspberry Pi uses less power than desktop computers.

2.6 Raspberry Pi Interface

- Three types of interface is supported by Raspberry Pi.

1. Serial

- It uses serial peripherals for serial communication.
- Transmit (Tx) and Receive (Rx) pin is used for serial communication.

2. Serial Peripheral Interface (SPI)

- SPI is a communication protocol used to transfer data between micro-computers like the Raspberry Pi and peripheral devices. These peripheral devices may be either sensors or actuators.
- SPI uses 4 separate connections to communicate with the target device. These connections are the serial clock (CLK), Master Input Slave Output (MISO), Master Output Slave Input (MOSI) and Chip Select (CS).
- The clock pin sense pulses at a regular frequency, the speed at which the Raspberry Pi and SPI device agree to transfer data to each other.
- For the ADC, clock pulses are sampled on their rising edge, on the transition from low to high.
- The MISO pin is a data pin used for the master to receive data from the ADC. Data is read from the bus after every clock pulse.
- The MOSI pin sends data from the Raspberry Pi to the ADC. The ADC will take the value of the bus on the rising edge of the clock. This means the value must be set before the clock is pulsed.
- The Chip Select line chooses which particular SPI device is in use. If there are multiple SPI devices, they can all share the same CLK, MOSI, and MISO.
- The SPI has the following features :
  1. 16-bit shift register
  2. 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation alias register (SPIEMU)
  3. 16-bit Transmit data register (SPIDAT0) and 16-bit Transmit data and format selection register (SPIDAT1)
  4. 8-bit baud clock generator
  5. Serial clock (SPICLK) I/O pin
  6. Slave in, master out (SPISIMO) I/O pin
  7. Slave out, master in (SPISOMI) I/O pin

8. Multiple slave chip select (SPISCS[n]) I/O pins (4 pin mode only)
9. Programmable SPI clock frequency range
10. Programmable character length (2 to 16 bits)
11. Programmable clock phase (delay or no delay)
12. Programmable clock polarity (high or low)
13. Interrupt capability
14. DMA support (read/write synchronization events)
15. Up to 66 MHz operation

Master-slave configuration of SPI :

- Fig. 2.6.1 shows SPI system. SPI bus is composed by four signals, namely the Master Out Slave In (MOSI), Master In Slave Out (MISO), serial clock (SCK) and active low slave select (\SS).
- MOSI : This pin is used to transmit data out of the SPI module when it is configured as a Master and receive data when it is configured as Slave.
- MISO : This pin is used to transmit data out of the SPI module when it is configured as a Slave and receive data when it is configured as Master.
- /SS : This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when its configured as a Master and its used as an input to receive the slave select signal when the SPI is configured as Slave.
- SCLK : This pin is used to output the clock with respect to which the SPI transfers data or receive clock in case of Slave.

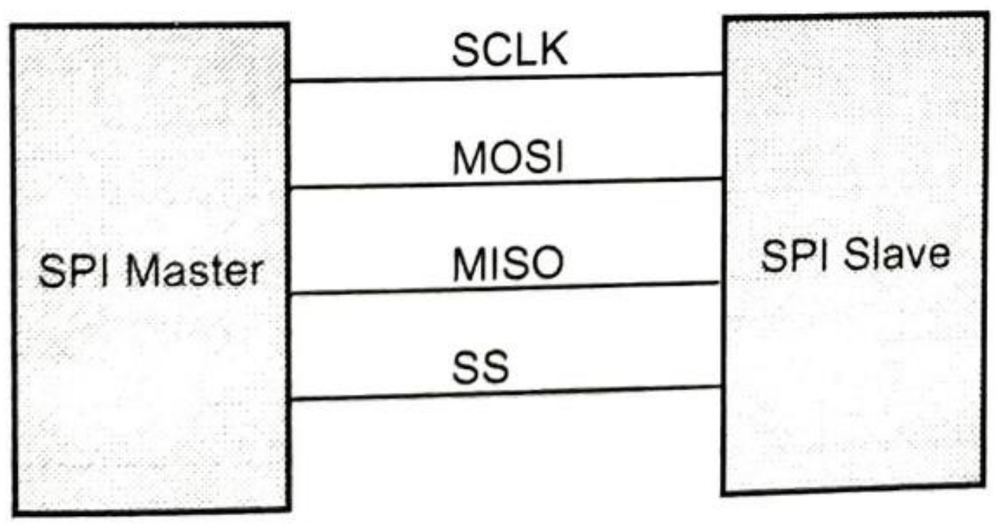


Fig. 2.6.1 SPI

- SCK master device will generate a pulse and the data will be synchronized in both master and slave devices. There are four different clock types to define SIP protocol, depending on what the SCK polarity and phase may be. It must ensure these signals between the master and slave devices compatible with each other.

- SPI is a Synchronous protocol. The clock signal is provided by the master to provide synchronization. The clock signal controls when data can change and when it is valid for reading.
- SPI creates a data loop between two devices. Data leaving the master exits on the SDO (serial data output) line. Data entering the master enters on the serial data input, SDI line.
- A clock (SCK), is generated by the master device. It controls when and how quickly data is exchanged between the two devices.
- SS allows a master device to control when a particular slave is being addressed. This allows the possibility of having more than one slave and simplifies the communications. When the SS signal goes low at a slave device, only that slave is accessed by SPI
- For SPI, there are Serial Clocks (SCLK), Chip Select lines (CS), Serial Data In (SDI) and Serial Data Out(SDO). There is only one master, there number of slaves depends on the number of chip select lines of the master.
- Synchronous operation, latch on rising or falling edge of clock, SDI on rising edge, SDO on falling edge. It operates in 1 to 2 MHz range.
- Master sends out clocks and chip selects. Activates the slaves it wants to communicate with.
- Fig. 2.6.2 master with multiple slave interface.

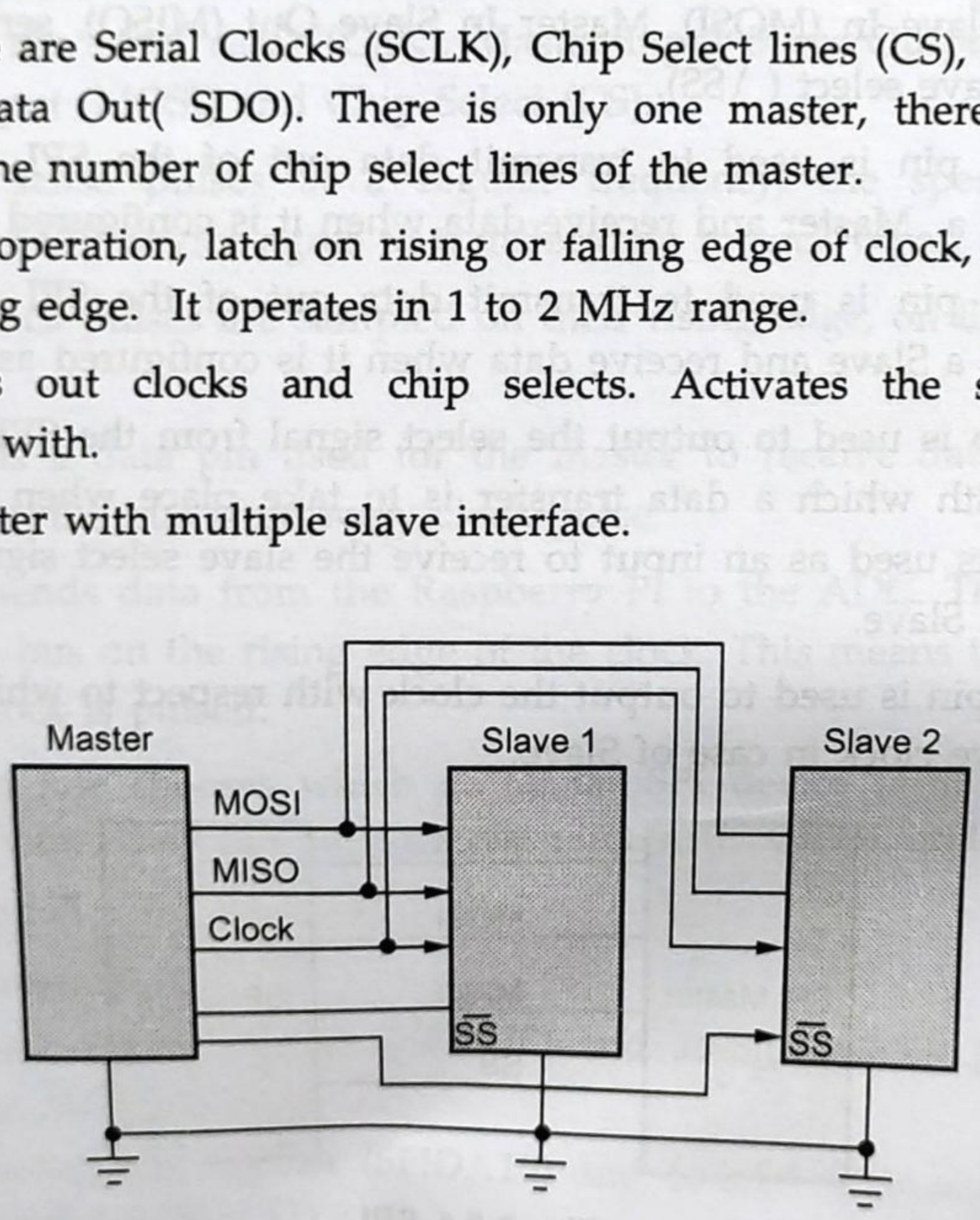


Fig. 2.6.2 Multiple slave interface

- SPI data transmit and data receive register are the main elements of the SPI. When the communication takes place the data on the transmit register are transferred into the shift register.

- The shift register in the master of width (8,16,32) and the shift register in the slave are linked by MOSI and MISO pins to form a distributed 16,32,64 bit register respectively.
- When the data transfer operation needs to be performed these 16,32,64- bit registers are serially shifted eight, sixteen, thirty-two bit positions by the serial clock generated by the master so that the data can be exchanged between the master and the selected slave.
- Data on the master SPI data transmit register becomes the input data for the slave read from the MOSI and the data read from the master SPI data receive register was the data send from the slave from MISO.
- Data on the shift registers are transferred into data receive register when the transfer completes and this data may be read from the data receive register any time before next transfer has completed.

3. I2C

- I2C is a communication protocol that the Raspberry Pi can use to speak to other embedded devices (temperature sensors, displays, accelerometers, etc).
- I2C is a useful bus that allows data exchange between microcontrollers and peripherals with a minimum of wiring.
- I2C is a two wire bus, the connections are called SDA (Serial Data) and SCL (Serial Clock). Each I2C bus has one or more masters (Raspberry Pi) and one or more slave devices, like the I/O Expander.
- As the same data and clock lines are shared between multiple slaves, we need some way to choose which device to communicate with.

2.7 Raspberry Pi with Python

- General Purpose Input/Output (GPIO) is a generic pin on a chip whose behavior can be controlled by the user at run time. The GPIO connector has a number of different types of connection :

1. True GPIO pins that you can use to turn LEDs on and off etc.
2. I2C interface pins that allow you to connect hardware modules with just two control pins.
3. SPI interface with SPI devices, a similar concept to I2C but uses a different standard.
4. Serial Rx and Tx pins for communication with serial peripherals.

2.7.1 Controlling LED with Raspberry Pi

- Fig 2.7.1 shows diagram of connecting LED to Raspberry Pi. The LED will initially be off because the GPIO pins are initialized as inputs at power-on.
- Install Python 2 library Rpi.GPIO. A library that will let us control the GPIO pins. Install commands :

sudo apt - get update

sudo apt - get install python - dev

sudo apt - get install python - rpi.gpio

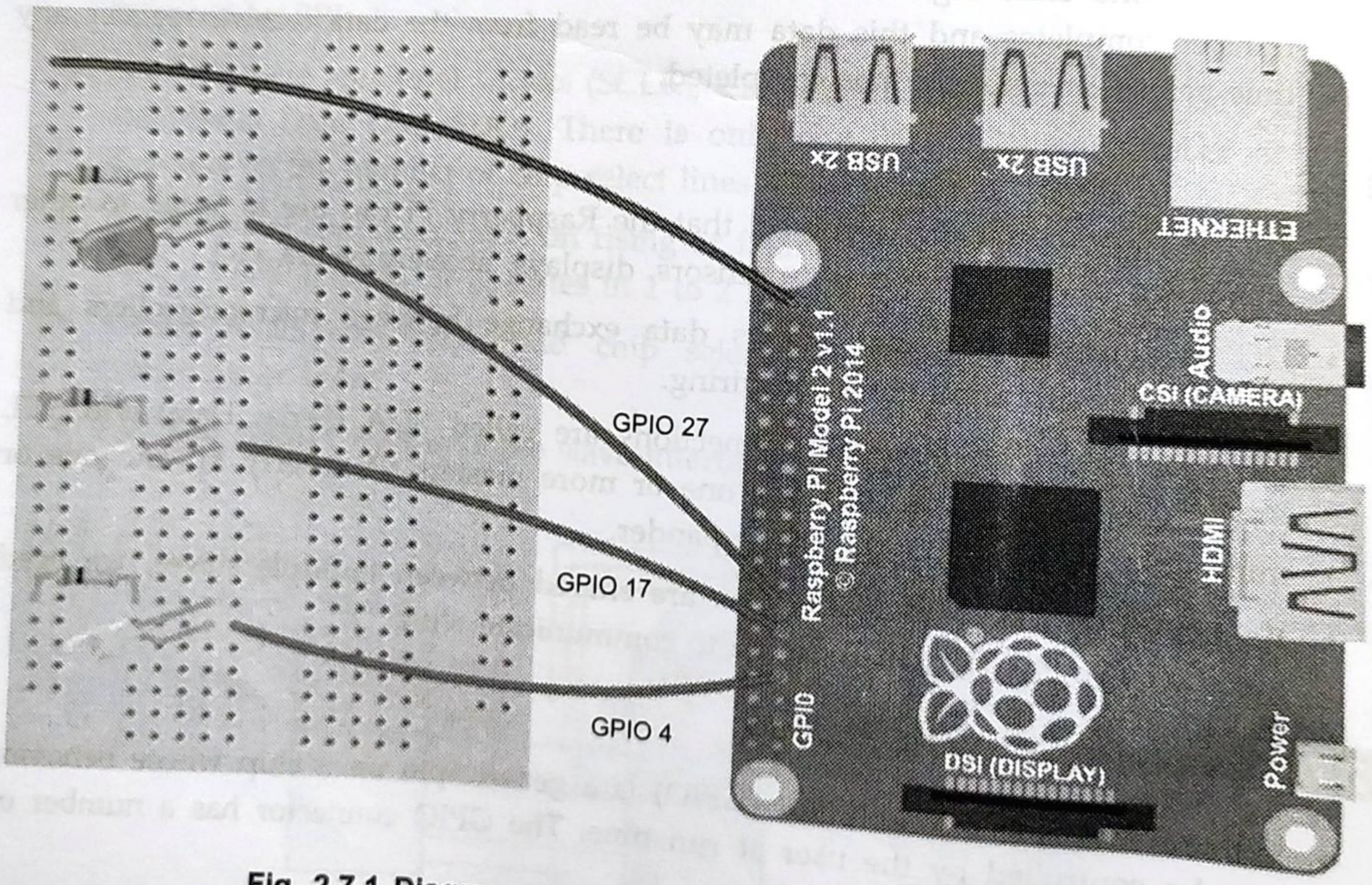


Fig. 2.7.1 Diagram of connecting LED to Raspberry Pi

- Simple LED circuit is shows below :

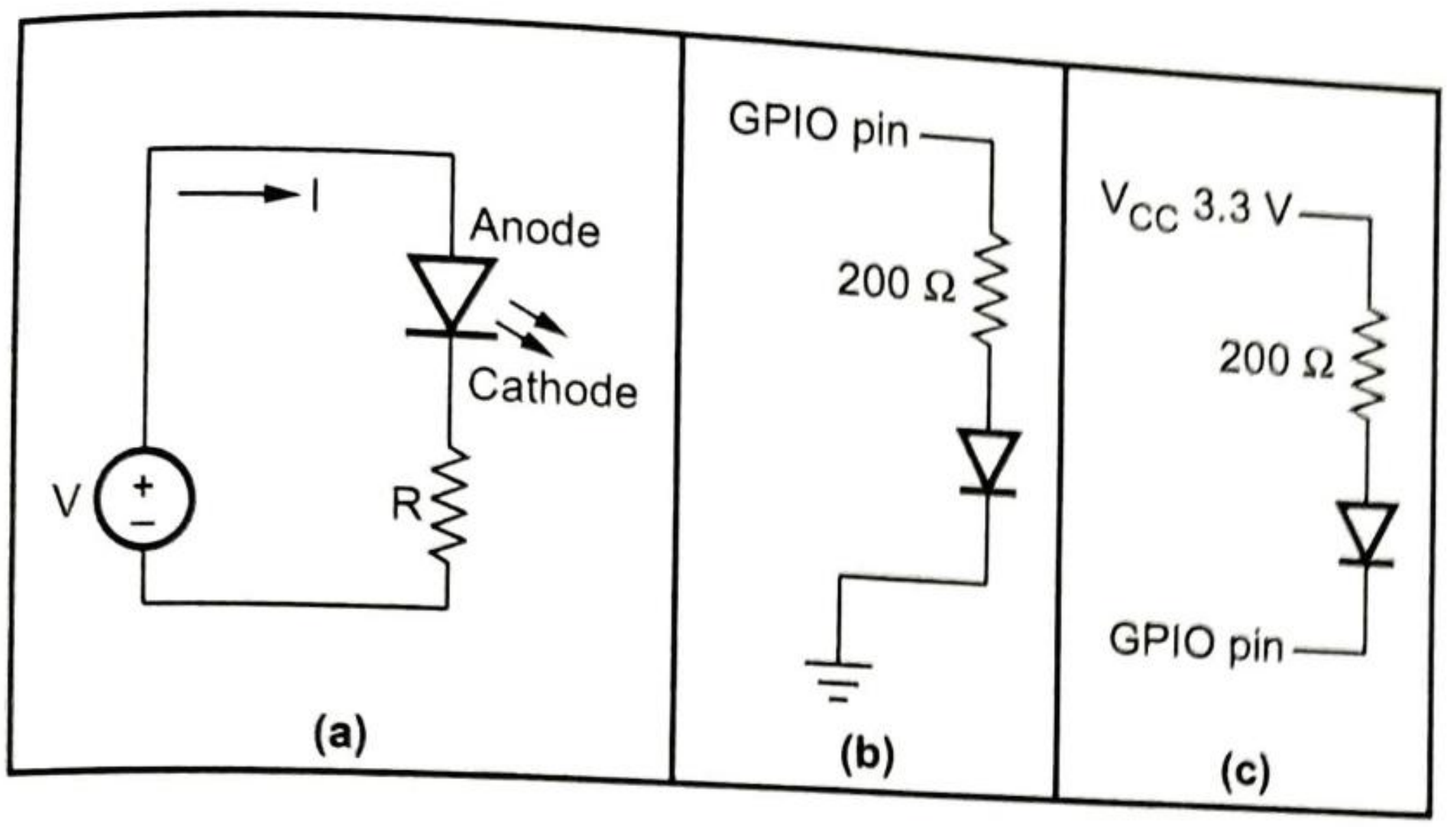


Fig. 2.7.2

- Current flows from the anode (+) to cathode (–). Anode is longer pin and cathode is shorter pin.
- Open up IDLE, the Python programming software and create a New file. Save it as led.py and input the code from the code listing. What the code does is first tell Python to use the GPIO module so we can connect to the GPIO pins, by importing the module.
- We then import the time module so we can create a delay between commands. We then tell the code to treat the GPIO pins as the number they are on the board and to turn the seventh pin into an output.
- We alternate between True and False so that it turns the pin on and off. Once it's cycled a few times, it will print the message 'Done' into IDLE and finally turn off the GPIO pins.

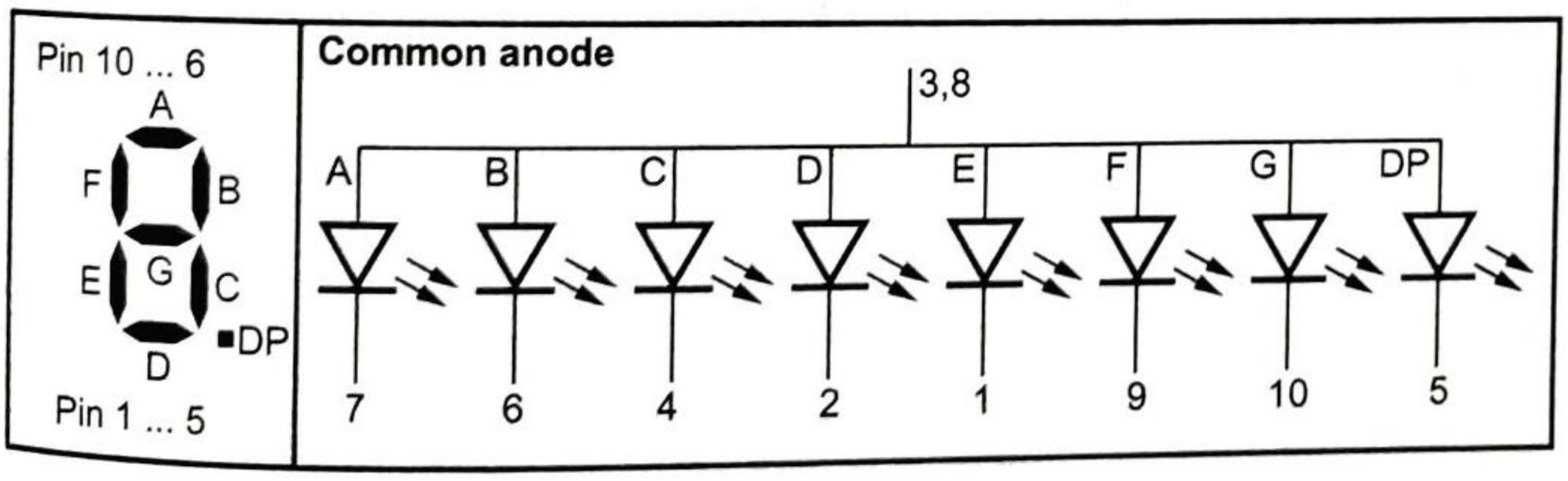


Fig. 2.7.3

```
Import RPi.GPIO as GPIO
Import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
GPIO.output(7,True)
time.sleep(1)
GPIO.output(7,False)
time.sleep(1)
GPIO.output(7,True)
time.sleep(1)
GPIO.output(7,False)
print"Done"
GPIO.cleanup()
```

Task 1 : Turn LED on for 2 seconds and off for 1 second, loop forever. Code is given below :

(In this example, we use diagram (b), i.e. controlling the LED by controlling the voltage at the anode (+)).

```
import RPi.GPIO as GPIO
import time
def main( ) :
GPIO.cleanup( )
GPIO.setmode(GPIO.BOARD)  # to use Raspberry Pi board pin numbers
GPIO.setup(11, GPIO.OUT)  # set up GPIO output channel
while True :
GPIO.output(11, GPIO.LOW)  # set RPi board pin 11 low. Turn off LED.
time.sleep(1)
GPIO.output(11, GPIO.HIGH)  # set RPi board pin 11 high. Turn on LED.
time.sleep(2)
main( )
```

- Example : Display digit on 7-segment LED. It is most direct way to control display :

1. Connect pin 3/8 of 7-seg-LED to Vcc

2. Connect the other 8 pins to 8 GPIO pins
3. Configure the 8 GPIO pins as out

- For example : display "2". Turn on segments A, B, D, E, G and turn off segments C, F, DP. Set A, B, D, E, G to LOW and set C, F, DP to HIGH. Set Pin 7, 6, 2, 1, 10 LOW and Set pin 4, 9, 5 HIGH (Refer Fig. 2.7.3)

2.7.2 Interfacing an LED and Switch with Raspberry PI

- When the switch is not pushed : GPIO detects Vcc (HIGH)
- When the switch is pushed : GPIO detects GND (LOW)

GPIO Input Sample Code

```
import RPi.GPIO as GPIO
# Use the pin numbers from the ribbon cable board
GPIO.setmode (GPIO.BCM)
# Set up this pin as input.
GPIO.setup (17, GPIO.IN)
# Check the value of the input pin
GPIO.input (17)
# Hold down the button, run the command again. The output should be "true".
GPIO.input(17)
```

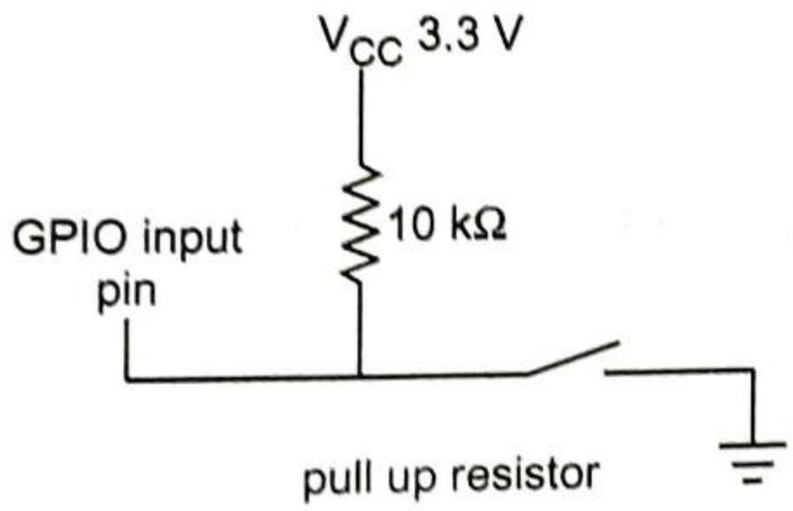


Fig. 2.7.4

2.7.3 Interfacing Light Sensor

- Unlike some other devices the Raspberry Pi does not have any analogue inputs. All 17 of its GPIO pins are digital. They can output high and low levels or read high and low levels.
- For sensors that act as a variable resistor such as LDRs (Light Dependent Resistors) or thermistors (temperature sensors) there is a simple solution. It allows

you to measure a number of levels using a single GPIO pin. In the case of a light sensor this allows you to measure different light levels.

- Fig 2.7.5 shows diagram of connecting an LDR to Raspberry Pi.

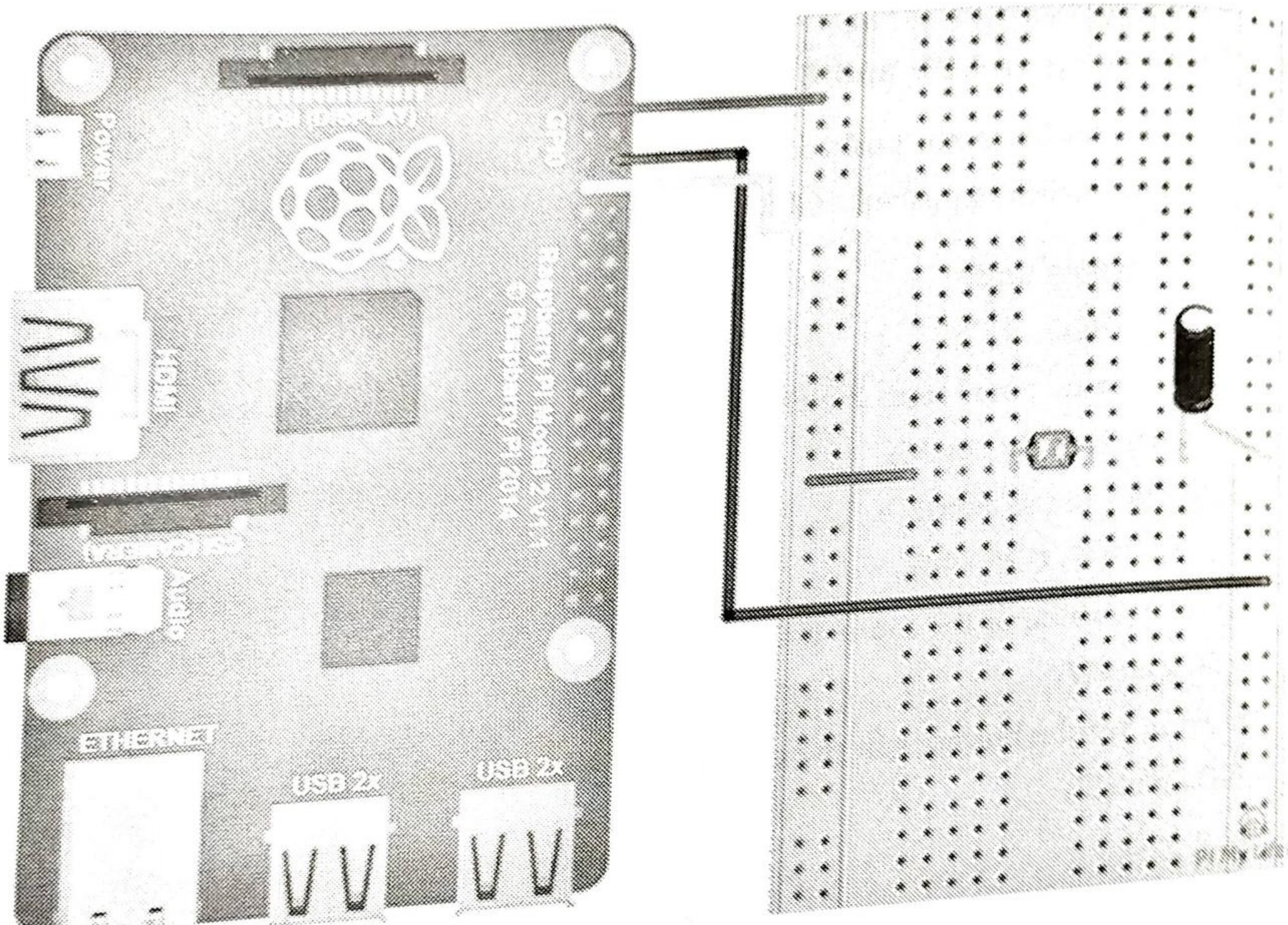


Fig. 2.7.5 Diagram of connecting an LDR to Raspberry Pi

- Following are steps :

1. First connect pin number 1 (3v3) to the positive rail on the breadboard.
2. Next connect pin number 6 (ground) to the ground rail on the breadboard.
3. Now place the LDR sensor onto the board and have a wire go from one end to the positive rail.
4. On the other side of the LDR sensor place a wire leading back to the Raspberry Pi. Hook this to pin number 7.
5. Finally place the capacitor from the wire to the negative rail on the breadboard. Make sure you have the negative pin of the capacitor in the negative rail.

- Fig 2.7.6 shows circuit diagram for above configuration.

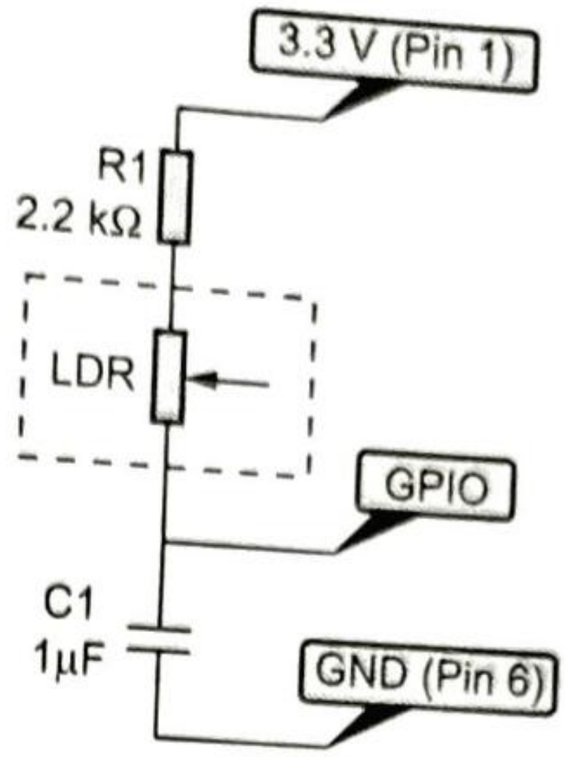


Fig. 2.7.6 Circuit diagram for LDR

The sequence of events :

1. Set the GPIO pin as an output and set it Low. This discharges any charge in the capacitor and ensures that both sides of the capacitor are 0 V.
2. Set the GPIO pin as an input. This starts a flow of current through the resistors and through the capacitor to ground. The voltage across the capacitor starts to rise. The time it takes is proportional to the resistance of the LDR.
3. Monitor the GPIO pin and read its value. Increment a counter while we wait.
4. At some point the capacitor voltage will increase enough to be considered as a High by the GPIO pin (approx 2v). The time taken is proportional to the light level seen by the LDR.
5. Set the GPIO pin as an output and repeat the process as required.

Python Code :

```
#!/usr/local/bin/python

# Reading an analogue sensor with a single GPIO pin

import RPi.GPIO as GPIO, time

# Tell the GPIO library to use

# Broadcom GPIO references

GPIO.setmode(GPIO.BCM)

# Define function to measure charge time

def RCtime (PiPin) :

measurement = 0

# Discharge capacitor

GPIO.setup(PiPin, GPIO.OUT)

GPIO.output(PiPin, GPIO.LOW)

time.sleep(0.1)
```

```
GPIO.setup(PiPin, GPIO.IN)
# Count loops until voltage across
# capacitor reads high on GPIO
while (GPIO.input(PiPin) == GPIO.LOW) :
 measurement += 1
return measurement

# Main program loop
while True :
 print RCtime(4) # Measure timing using GPIO4
```

2.8 Implementation of IoT with Edge Devices

- An edge device is a device that provides an entry point into enterprise or service provider core networks. Examples include routers, routing switches, Integrated Access Devices (IADs), multiplexers, and a variety of Metropolitan Area Network (MAN) and Wide Area Network (WAN) access devices.
- Edge devices also provide connections into carrier and service provider networks. An edge device that connects a local area network to a high speed switch or backbone may be called an edge concentrator.
- Edge Devices : A device which produces data is edge devices like machines and sensors, or any devices through which information is collected and delivered.
- Edge Gateway : It's a buffer where edge computing processing is done. The gateway is the window into the environment beyond the edge of the network.
- Edge computing aims to speed up critical data processing closer to where the device is located. It implies that data processing is done more efficiently. There is no need for the centralisation of attributes. In other words, edge computing is a mesh interface to process and store micro data centres locally.
- One of the most common types of edge devices is an edge router. Usually deployed to connect a campus network to the internet or a WAN, edge routers chiefly function as gateways between networks.
- Firewalls can also be classified as edge devices, as they sit on the periphery of one network and filter data moving between internal and external networks.
- Edge devices are comprised of hardware that performs the two essential functions of providing physical connectivity and enabling traffic between networks. A full range of edge device functions might include the transmission, routing, processing, monitoring, filtering, translation and storage of data between networks.

- Edge computing was developed due to the exponential growth of IoT devices, which connect to the internet for either receiving information from the cloud or delivering data back to the cloud. And many IoT devices generate enormous amounts of data during the course of their operations.
- Data produced by the IoT devices to be processed where it is created instead of taking away to the routes to data centers with the help of edge computing.

Why Edge Computing Matters ?

a) When IoT devices have poor connectivity.

b) Not efficient for IoT devices to be in constant touch with the central cloud.

c) The latency factor reduces latency because data doesn't have to traverse over a network to a central cloud for processing.

d) Where latencies are untenable like manufacturing or financial services.

e) As soon as data is produced, it doesn't need to send over a network; instead, it compiles the data and sends daily reports to the cloud for long term storage, i.e., reduces the data traversing.

f) Direct access to gateway into the telecom provider's network, which connects to a public IaaS cloud provider.

- Fig. 2.8.1 shows edge-centric IoT architecture.

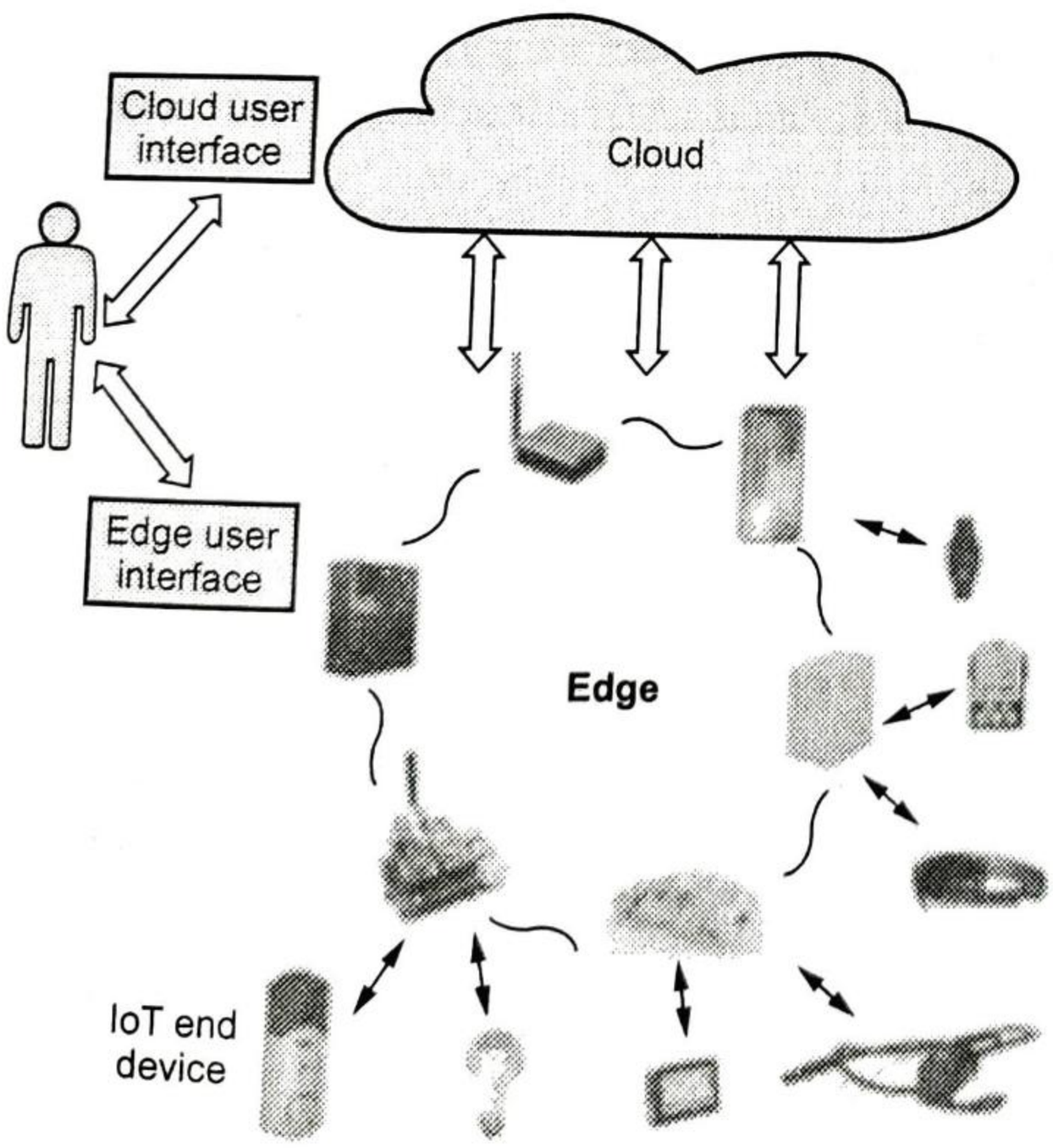


Fig. 2.8.1 Edge-centric IoT architecture

- The edge-centric IoT architecture contains four major parts : The cloud, the IoT end devices, the edge, and the users.
- The IoT end devices are deeply embedded in the physical world. They sense the physical world and take actions to control the physical world.
- In the edge-centric IoT architecture, IoT users submit queries to access IoT data or commands to control IoT devices. These queries and commands will eventually arrive at the edge layer through a web or mobile app-based interface provided by the cloud or the edge.
- Then they are handled by the edge layer, who will either forward them to IoT end devices or handle them at the edge layer on behalf of IoT end devices.
- Interacting with IoT end devices, the edge layer not only bridges them with the users and the cloud, but also can store data collected and uploaded from IoT end devices and offload substantial computational needs, such as big data analysis.
- In addition, many existing services for IoT end devices can be migrated from the cloud to the edge, and can be customized based on the needs of IoT end devices.

2.9 Reading Sensor Data and Transmit to Cloud

- Summary of most important terms which are used widely in sensor network is provided in the following :

a) **Sensor :** A transducer that converts a physical phenomenon such as heat, light, sound or motion into electrical or other signal that may be further processed by other devices.

b) **Sensor node :** The basic unit constituting a sensor network; it embeds a processor, a memory, a wireless interface and a local autonomous power supply.

c) **Network topology :** A graph, where nodes are sensor nodes and edges are communication links.

d) **Routing :** The process of forwarding data of interest along a network path from the source node to its final destination.

e) **Resource :** This term is used to address the sensors, the communication links, the computational capabilities, the data storage and the energy amount of a node

- Sensor includes a processor which is tough enough to configure the data uploading to cloud. The Ethernet connection would connect to wired Internet service. Some places don't have wired Internet. The processor could also have the ability to update or modify the functions of the sensor. There is no involvement of radio link.

- Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
- In a cloud computing platform, instead of having local servers for collecting and managing information coming from applications, remotely located servers are dynamically provisioned and (re)configured, according to the actual needs.
- As an example, if we consider a wind or solar farm, the weather - related information collected by a WSN can be processed together with grid - related information to improve plant efficiency and better satisfy the power demand.
- Cloud computing also describes those applications that can be remotely accessed through the Internet. Such cloud applications exploit large data centers and powerful servers similarly to Web applications and Web services we use every day.

Cloud computing features :

1. **Service on demand :** Clients requests are satisfied automatically without the intervention of human operator.
2. **Elasticity of demand :** Available resources are used by the clients according to their own needs for a flexible period of time.
3. **Abstraction :** Actual resources (hardware and/or software) are hidden to clients. Customers of the service exploits resources offered by the provider without knowing the locations from where processed data will arrive or where will be stored.
4. **Service measurement :** The provider exploits tools for measuring the actual usage of offered services.
5. **Resource pooling :** Available services, constituting a pool of services, are assigned dynamically according to clients' requests.
6. **Network access :** The client application can run over various platforms by means of Internet access exploiting devices as mobile phones, tablets, laptops and so on.

- According to its implementation, the cloud can provide different services levels.
  1. **Infrastructure as a Service (IaaS) :** This model provides basic storage and computing capabilities as standardized services over the network. Consequently, the user (on the client side) does not need to buy its own hardware. The user would typically execute its own applications exploiting the workload offered by the infrastructure. As an example, consider services offered by Amazon EC2.

2. **Platform as a Service (PaaS) :** This model provides application software and/or development environment as a service; in addition, other higher level software applications can be executed exploiting the same service.

   Consequently, the user has the freedom to create his own applications, that are executed by the infrastructure of the provider. As an example, consider the Google's App Engine or Windows Azure.

3. **Software as a Service (SaaS) :** This model provides services to clients according to their requests. A single instance of the service is executed on the cloud and can be used by multiple end users. There is no need for investment on the client side for hardware or software licenses.

- Fig. 2.9.1 shows Sensor to Mobile - Phone Network to Cloud

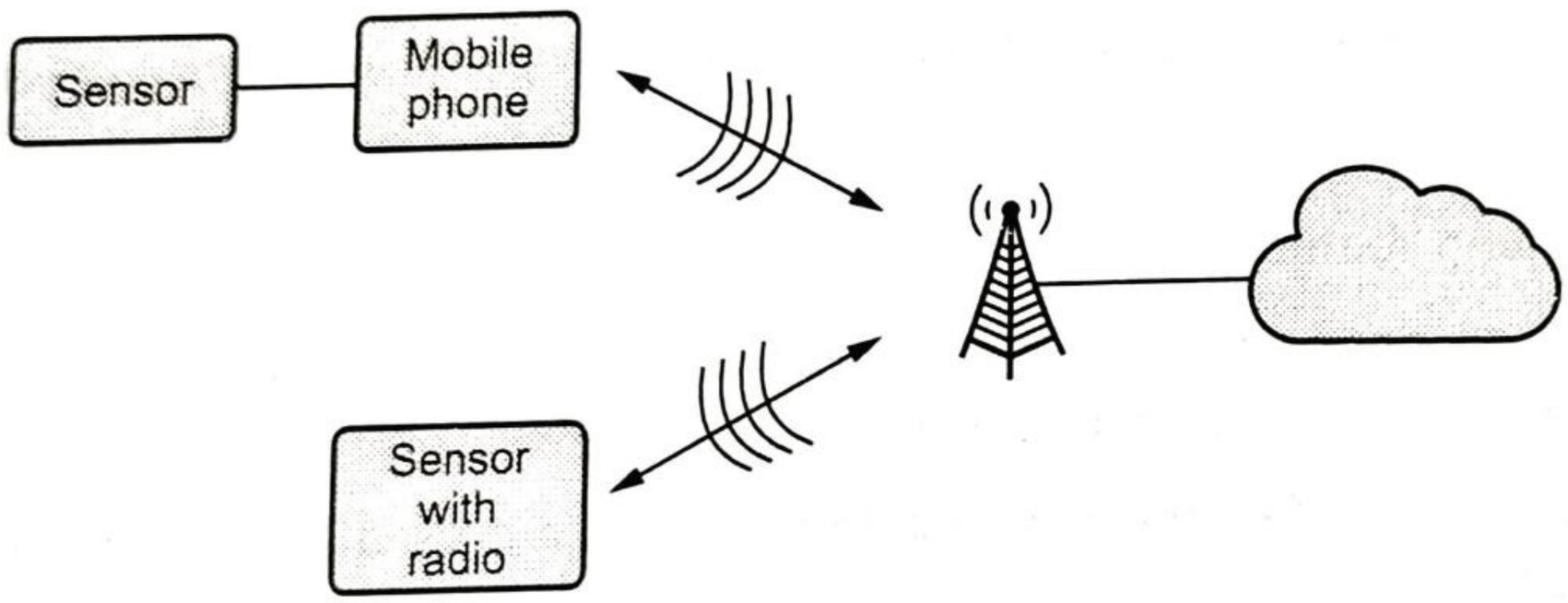


Fig. 2.9.1 Sensor to Mobile - Phone Network to Cloud

- The IoT, i.e. the capability to interconnect every possible device, opens new scenarios in WSNs. Cloud computing services and the availability of powerful and inexpensive smart devices allow to optimize information management, sharing measurement results and improving quality of services.
- The communication patterns for these devices can be categorized in the following categories : Telemetry, inquiries, commands and notifications.

1. **Telemetry :** A client device sends data one way to a cloud service.
2. **Inquiry :** A client device sends a query to the cloud service and receives a response.
3. **Command :** A cloud service issues a command to a client device and the client device returns a success or failure response.
4. **Notification :** A cloud service issues a one-way-out-of-band notification to a client device that's important for the device's operation.

2.10 Controlling Devices through Cloud using Mobile Application and Web Application

- Modern mobile devices, such as smartphones and tablets, have made many pervasive computing dreams come true. Still, many mobile applications do not perform well due to the shortage of resources for computation, data storage, network bandwidth, and battery capacity.
- While such applications can be re-designed with client - server models to benefit from cloud services, the users are no longer in full control of the application, which has become a serious concern for data security and privacy.
- Since the Internet became popular, a mobile device might overcome the constraints by offloading portions of application workload onto a server machine via the network to save execution time and conserve energy.
- Cloud computing has changed software infrastructures and business models of Internet services with technologies to provide and manage abundant resources of computation and data storage over the network at relatively low amortized operation costs.
- Fig. 2.10.1 shows Sensor to Mobile Phone to Cloud.

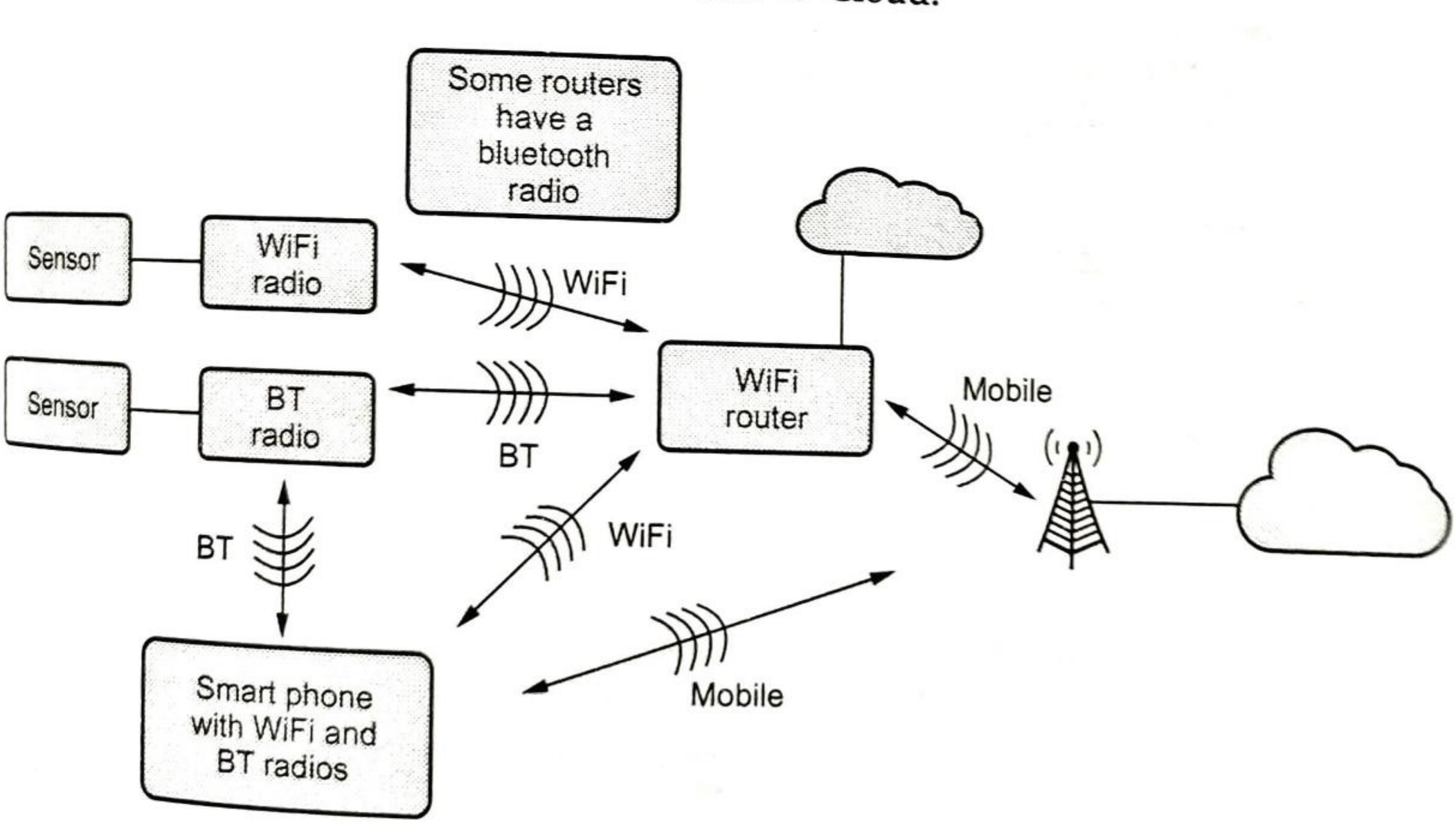


Fig. 2.10.1 Sensor to Mobile Phone to Cloud

- Application re-design and deployment : Traditional client - server models have been successfully used for mobile applications to leverage the resources in the cloud, but additional efforts, such as application partitioning and deployment of services, are required to enable dynamic, fine - grain client - server collaboration,

where the client may decide to offload a piece of work to the server during the runtime when the resources are available, if the offloading is beneficial.

- Network condition and service availability : The quality of the mobile networks is not adequate for delivering satisfactory user experiences via the collaboration between mobile devices and cloud services.
- Mobile and web applications usually require a back - end server. Web applications require a web server to deliver content. Applications also need to store user profiles and media such as images and videos. Communication between the application and the server is often done using an API, which is usually REST.

2.11 IoT Gateways

- IoT Gateway, as a hardware device or a virtual software code, acts as a communication bridge between IoT Sensor Network and Cloud Server. IoT gateway device has a layered architecture.
- Fig. 2.11.1 shows IoT gateway.

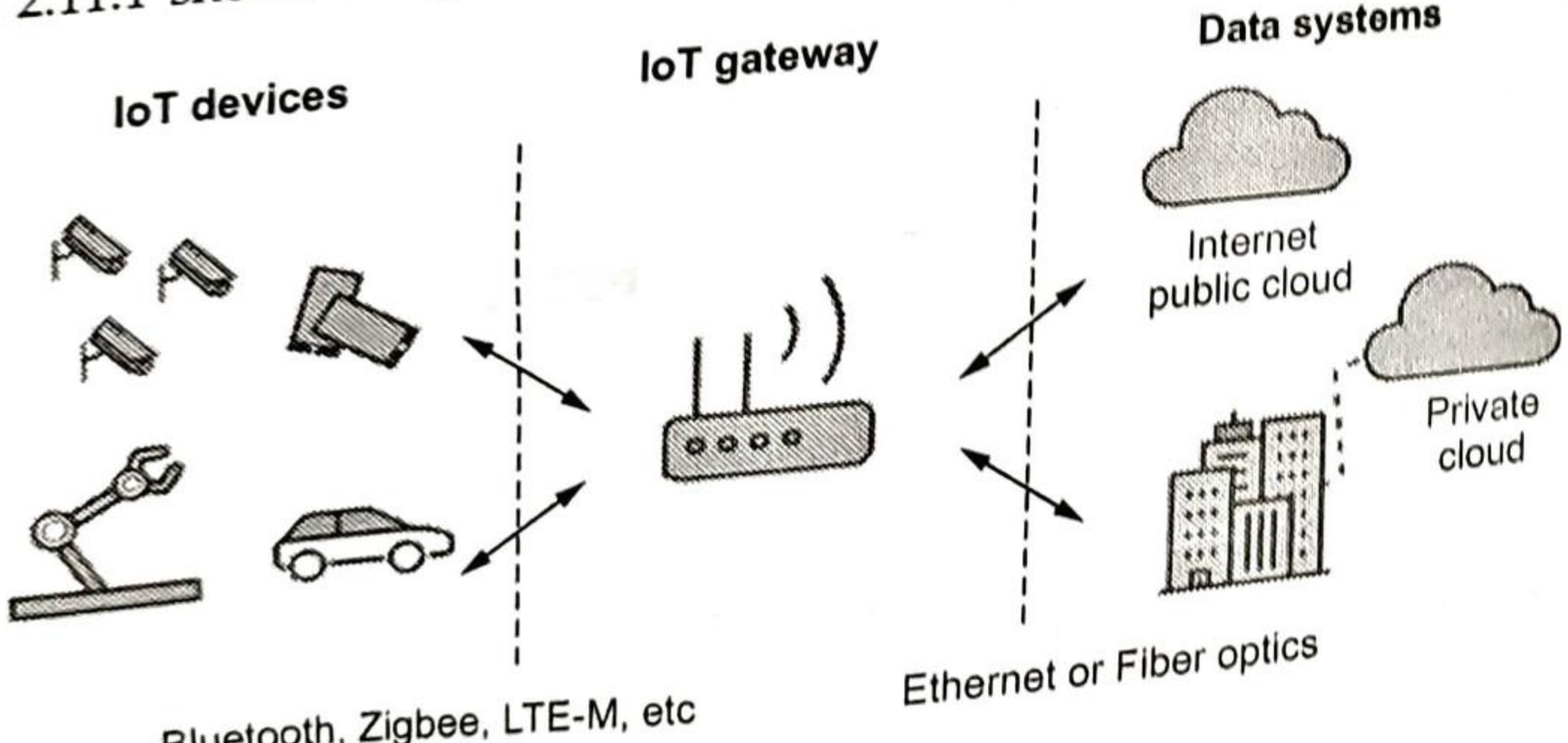


Fig. 2.11.1 IoT gateway

- IoT gateway development process :
  1. **Hardware platform** : This defines the processing power & memory specifications of the IoT Gateway. This is the gateway powerhouse and a hardware platform is selected based on the complexity of IoT application(s) that need to be deployed.
  2. **Operating system** : The decision of opting for a particular OS depends on the legacy systems. It is a best practice to continue to use the OS compatible with the existing systems in order to save costs and hassle-free integration.
  3. **Analytics engine** : This layer ensures raw data is converted to actionable insights.

4. **Integrated application development platform and device drivers** : This layer supports development and/or addition of new devices, applications or systems to the IoT network.

- Features of IoT gateway :

a) Facilitating communication with legacy or non-internet connected devices.
b) Data caching, buffering and streaming.
c) Data pre-processing, cleansing, filtering and optimization.
d) Some data aggregation.
e) Device to device communications.
f) Networking features and hosting live data.
g) Data visualization and basic data analytics via IoT gateway applications.
h) Short term data historian features.
i) Security - manage user access and network security features.
j) Device configuration management.
k) System diagnostics.

- The IoT gateway follows this simple process.

a) Pre - processes, cleans, and filters raw data.
b) Translates protocols for encryption and communication.
c) Sends data to a destination on the Internet or Intranet.

2.12 Fill in the Blanks

Q.1 Raspberry Pi also allows interfacing sensors and actuators through the general purpose _____.

Q.2 Raspberry Pi is based on an _____ processor.

Q.3 Raspberry Pi has _____ status LED.

Q.4 _____ port on Raspberry Pi provides both video and audio output.

Q.5 Raspberry Pi comes with a standard _____ Ethernet port.

Q.6 USB ports on Raspberry Pi can provide a current upto _____.

Q.7 IoT Gateway, as a hardware device or a virtual software code, acts as a communication bridge between IoT Sensor Network and _____.

Q.8 IoT gateway device has a _____ architecture.

Q.9 Sensor converts physical energy into _____ energy.

Q.10 ARM means _____.

Q.11 The disadvantages of Raspberry Pi is, it does not have a _____ associated with it.

Q.12 In GPIO, DNC stands for _____.

Q.13 The BCM is also referred _____.

Q.14 GPIO stands for _____.

Q.15 ARM means _____.

Q.16 In ARM, word means _____ bits.

Q.17 CPSR stands for _____.

Q.18 The Raspberry Pi has a _____ interface to allow it to perform serial data communications.

Q.19 The _____ rate is calculated from bits per second, which includes the start, data, parity, and stop bits.

Q.20 Linux _____ command is used to create empty files or change time stamp for existing files.

Q.21 The head command displays the _____ of a file.

2.13 Multiple Choice Questions

Q.1 Arduino Uno has _____ digital I/O pins and _____ analog input pins.

a 8, 16 b 4, 8 c 14, 6 d 6, 14

Q.2 _____ pin is used to provide an input voltage when the Arduino is using an external power source.

a VIN b VOUT c USB d REST

Q.3 The pins A5 and A4 can support _____ communication.

a serial b parallel c analog d TWI

Q.4 Raspberry Pi uses an _____ processor.

a embedded b UNO c ARM d microprocessor

Q.5 Model B Raspberry Pi has _____ SDRAM.

a 64 MB b 128 MB c 256 MB d 512 MB

Q.6 The Raspberry Pi models A support _____ number of USB port.

a 1 b 2 c 4 d 8

Q.7 An Accelerated Processing Unit (APU) combines the _____ CPU and _____ GPU onto a single chip

a CPU, GPU b CPU, RAM c GPU, RAM d GPU, SoC

Q.8 Raspberry Pi is a _____ pin expansion header.

a 16 b 24 c 40 d 48

Q.9 The _____ command displays the beginning of a file.

a ls b cp c tail d head

Q.10 _____ is a server-side platform built on Google Chrome's JavaScript Engine.

a Python b Node.js c GPIO d SPI

Q.11 Which of the following is NOT Raspberry Pi Interface ?

a UART b GPIO c SPI d FPGA

Q.12 What the distributions are supported by Raspberry Pi ?

a Arch Linux b Debian

c Fedora Remix d Arch Linux, Debian, and Fedora Remix

Q.13 Which instruction set architecture is used in Raspberry Pi ?

a X86 b MSP c AVR d ARM

Q.14 Raspberry Pi comes with a standard _____ Ethernet port.

a RJ11 b RJ45

c RJ11 and RJ45 d RJ5

Q.15 Which of the following is NOT Raspberry Pi Interface ?

a UART b GPIO c SPI d FPGA

Q.16 Raspberry Pi is a _____ pin expansion header.

a 16 b 24 c 40 d 48

Q.17 Raspberry Pi is a low cost mini - computer with the physical size of _____.

a credit card b micro card

c nano card d SIM card

Q.18 The disadvantages of Raspberry Pi is, it does not have a _____ associated with it.

a memory b hard disk c USB port d ALU

Answer Keys for Fill in the Blanks

1.	I/O pins	2.	ARM
3.	five	4.	HDMI
5.	RJ45	6.	100 mA
7.	Cloud Server	8.	layered
9.	electrical	10.	Advanced RISC Machines
11.	hard disk	12.	Do Not Connect
13.	Broadcom chip-specific pin numbers	14.	General Purpose Input/Output
15.	Advanced RISC machines	16.	32
17.	Current Program Status Register	18.	UART
19.	baud	20.	touch
21.	beginning		

Answer Keys for Multiple Choice Questions

Q.1	c	Q.2	a	Q.3	d	Q.4	c
Q.5	d	Q.6	a	Q.7	a	Q.8	c
Q.9	d	Q.10	b	Q.11	d	Q.12	d
Q.13	d	Q.14	b	Q.15	d	Q.16	c
Q.17	a	Q.18	b				