

INTERNET OF THINGS

UNIT – 4

Introduction to Raspberry Pi

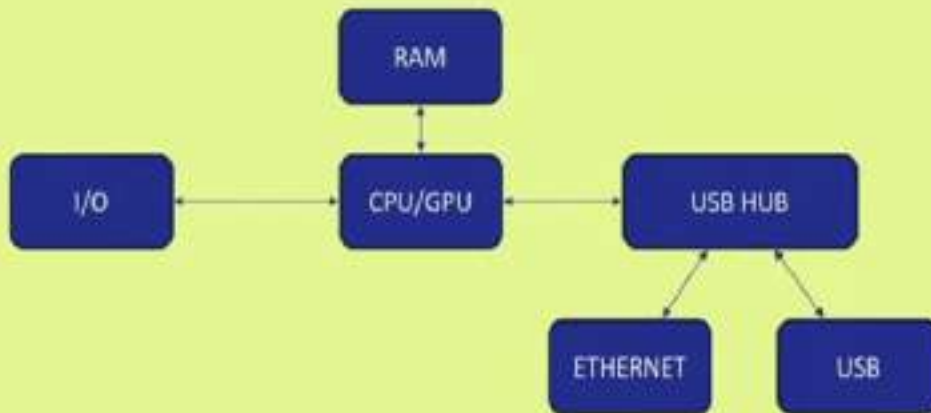
Raspberry Pi

- It is a micro sized computer and more specifically it is a single board computer which is very low-cost device and very easy to access.
- Raspberry pi compared to Arduino is more powerful, it is more powerful in terms of the computation or processing power.
- Additionally, it has better memory capacity and also it can integrate different types of sensors and actuators which makes it more attractive than compared to the similar kind of feature of Arduino.
- We can configure Raspberry pi as a web server, as an edge device and so on.

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

- There is a provision for Wi-Fi and Bluetooth only on mod pi 3.
- Generally video output is from HDMI port and there are 40 GPIO pins. So, these GPIO pins are mainly known as general purpose input output pins.

Basic Architecture



- This is the basic functional architecture of a raspberry pi.
- the center you have a CPU or GPU you have various input output ports connected to it
- connect an Ethernet from RAM and USB hub

Raspberry pi3 b+

- Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer.
- It is popularly used for real time Image/Video Processing; IoT based applications and Robotics applications.
- It is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at low power consumption.
- Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, etc.
- Raspbian OS is official Operating System available for free to use. This OS is efficiently optimized to use with Raspberry Pi.
- Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc.
- We should use SD card (minimum 16 GB recommended) to store the OS (operating System).
- Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e. GPIOs for developing an application.
- By accessing GPIO, we can connect devices like LED, motors, sensors, etc and can control them too.
- It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit).
- The CPU speed of Raspberry Pi varies from 700 MHz to 1.2 GHz. Also, it has on-board SDRAM that ranges from 256 MB to 1 GB.
- Raspberry Pi also provides on-chip SPI, I2C, and UART modules.
- There are different versions of raspberry pi available as listed below:

Raspberry Pi 1 Model A

Raspberry Pi 1 Model A+

Raspberry Pi 1 Model B
 Raspberry Pi 1 Model B+
 Raspberry Pi 2 Model B
 Raspberry Pi 3 Model B
 Raspberry Pi Zero

Out of the above versions of Raspberry Pi, more prominently use Raspberry Pi and their features are as follows:

| Features | Raspberry Pi Model B+ | Raspberry Pi 2 Model B | Raspberry Pi 3 Model B | Raspberry Pi zero |
|------------------------|------------------------------|-------------------------------|-------------------------------|----------------------------|
| SoC | BCM2835 | BCM2836 | BCM2837 | BCM2835 |
| CPU | ARM11 | Quad Cortex A7 | Quad Cortex A53 | ARM11 |
| Operating Freq. | 700 MHz | 900 MHz | 1.2 GHz | 1 GHz |
| RAM | 512 MB SDRAM | 1 GB SDRAM | 1 GB SDRAM | 512 MB SDRAM |
| GPU | 250 MHz Videocore IV | 250MHz Videocore IV | 400 MHz Videocore IV | 250MHz Videocore IV |
| Storage | micro-SD | Micro-SD | micro-SD | micro-SD |
| Ethernet | Yes | Yes | Yes | No |
| Wireless | WiFi and Bluetooth | No | No | No |

Raspberry Pi 3 Technical Specifications

| | |
|--|--|
| Microprocessor | Broadcom BCM2837 64bit Quad Core Processor |
| Processor Operating Voltage | 3.3V |
| Raw Voltage input | 5V, 2A power source |
| Maximum current through each I/O pin | 16mA |
| Maximum total current drawn from all I/O pins | 54mA |
| Flash Memory (Operating System) | 16Gbytes SSD memory card |

| | |
|------------------------------|---|
| Internal RAM | 1Gbytes DDR2 |
| Clock Frequency | 1.2GHz |
| GPU | Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GLES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high- profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure. |
| Ethernet | 10/100 Ethernet |
| Wireless Connectivity | BCM43143 (802.11 b/g/n Wireless LAN and Bluetooth 4.1) |
| Operating Temperature | -40°C to +85°C |

Board Connectors

| Name | Description |
|-------------------|---|
| Ethernet | Base T Ethernet Socket |
| USB | 2.0 (Four sockets) |
| Audio Output | 3.5mm Jack and I2S |
| Video output | HDMI |
| Camera Connector | 15 pin MIPI Camera Serial Interface (CSI 2) |
| Display Connector | Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane. |
| Memory Card Slot | Push/Pull Micro SDIO |

Difference between Arduino and Raspberry Pi

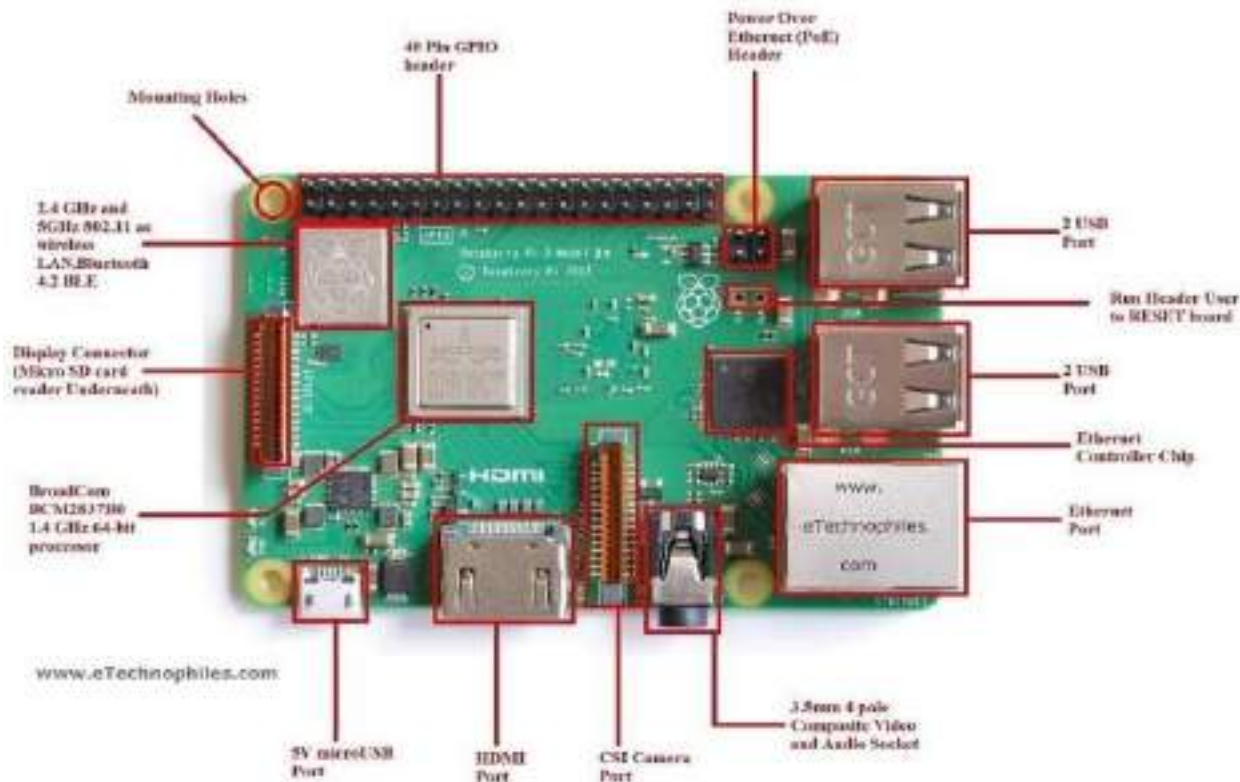
| <u>ARDUINO</u> | <u>RASPBERRY Pi</u> |
|--|--|
| Arduino is a microcontroller, which is a part of computer. It runs one program again and again | It is a mini computer with Raspbian OS. It can run one multiple programs at a time. |
| Arduino requires external hardware to connect to the internet | Raspberry Pi can easily connect to the internet using ethernet port and USB Wi-Fi dongles. |
| Arduino has only one USB port to connect to the computer | Raspberry Pi has 4 USB port to connect different devices |
| Processor used in Arduino is from AVR family Atmega32P | The Processor used is from ARM family |
| Arduino uses C/C++ type of coding | The recommended programming is python but C, C++, Python, Ruby are pre-installed |
| Arduino can provide onboard storage but limited memory | Raspberry did not have storage on board. It provides an SD card port. |
| NO inbuild Wi-Fi, Bluetooth facility | Inbuild Wi-Fi, Bluetooth facility |
| No on-board port for camera | on board port for camera |
| Low cost | Relatively high cost |

INTERNET OF THINGS

UNIT – 4

INTRODUCTION TO RASPBERRY Pi

Raspberry Pi 3 model B Board Description



Processor & RAM: Raspberry Pi is based on an ARM processor. 64 bit ARMv7 Broadcom BCM2837 Quad Core Computer running at 1.2GHz,

USB Ports: Raspberry Pi comes with FOUR 2.0 ports. The USB ports on Raspberry Pi can provide a current up to 100mA.

Ethernet Ports: Raspberry Pi comes with a standard RJ45 Ethernet port. We can connect an Ethernet cable or a USB WiFi adapter to provide Internet connectivity.

HDMI Output: The HDMI port on Raspberry Pi provides both video and audio output. We can connect the Raspberry Pi to a monitor using an HDMI cable. For monitors that have a DVI (Digital Video interface) port but no HDMI port, we can use an HDMI to DVI adapter/cable.

Composite Video Output: Raspberry Pi comes with a composite video output with an RCA jack that supports both PAL and NTSC video output. The RCA jack can be used to connect old televisions that have an RCA input only.

Audio Output: Raspberry Pi has a 3.5 mm audio output jack. This audio jack is used for providing audio output to old televisions along with the RCA jack for video. The audio quality from this jack is inferior to the HDMI output.

GPIO Pins: Raspberry Pi comes with a number of general purpose input/output pins. There are four types of pins on Raspberry Pi- true GPIO pins, I2C interface pins, SPI interface pins and serial Rx and Tx pins.

Display Serial Interface(DSI): The DSI interface can be used to connect an LCD panel to Raspberry Pi.

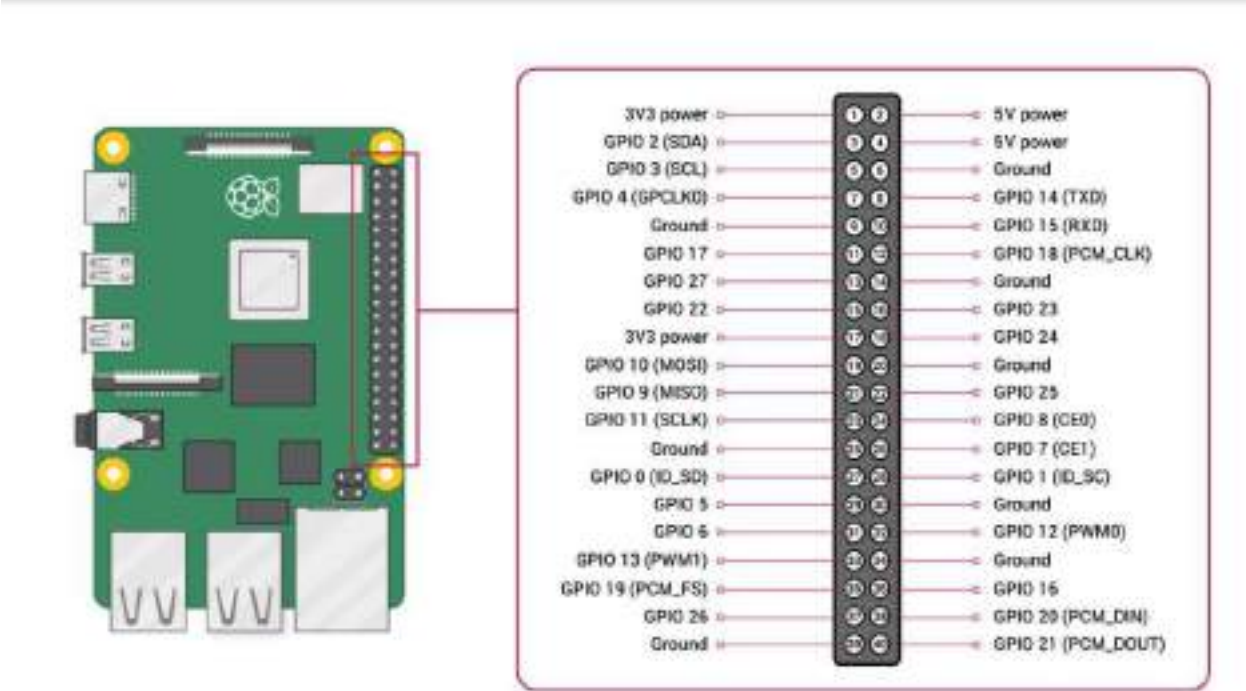
Camera Serial Interface (CSI): This interface can be used to connect a camera module to Raspberry Pi.

Status LEDs: Raspberry Pi has five status LEDs.

| Status LED | Function |
|------------|---------------------------|
| ACT | SD card access |
| PWR | 3.3V Power is present |
| FDX | Full duplex LAN connected |
| LNK | Link/Network activity |
| 100 | 100 Mbit LAN connected |

SD Card Slot: Raspberry Pi does not have a built in operating system and storage. **SD card slot** on the **Raspberry Pi 3** is located just below the Display Serial Adapter on the other side. We can plug-in an SD card loaded with a Linux image to the SD card slot.

Power Input: Raspberry Pi has a micro-USB connector for power input.



HDMI cable is used to connect the monitor and the Raspberry pi. you will require a keyboard and mouse a basic 5 volt adapter to power up the pi LAN cable. memory card include the operating system on it.

Raspberry Pi on board Serial Communication standards

UART

UART is multi master communication protocol. This protocol is quite easy to use and very convenient for communicating between several boards: Raspberry Pi to Raspberry Pi, or Raspberry Pi to Arduino, etc.



For using UART you need 3 pins:

GND that will connect to the global GND of your circuit.

RX for Reception. It will connect this pin to the TX pin of the other component.

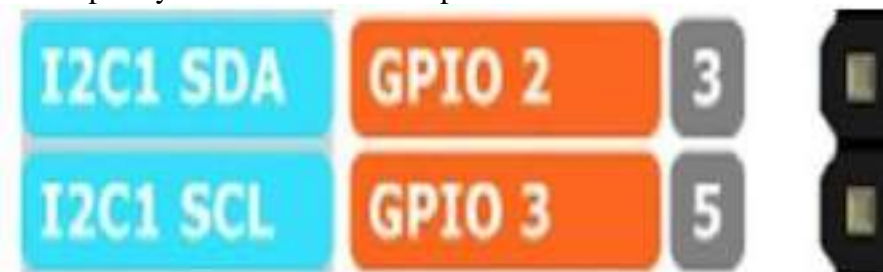
TX for Transmission. It will connect this pin to the RX of the other component.

By using a UART to USB converter, you can communicate between your laptop and Raspberry Pi with UART.

I2C

I2C is a 2 WIRE master-slave bus protocol (it can have multiple masters but mostly used with one master and multiple slaves). The most common use of I2C is to read data from sensors and actuate some components.

The master is the Raspberry Pi, and the slaves are all connected to the same bus. Each slave has a unique ID, so the Raspberry Pi knows which component it should talk to.



For using I2C you'll need 3 pins:

GND:

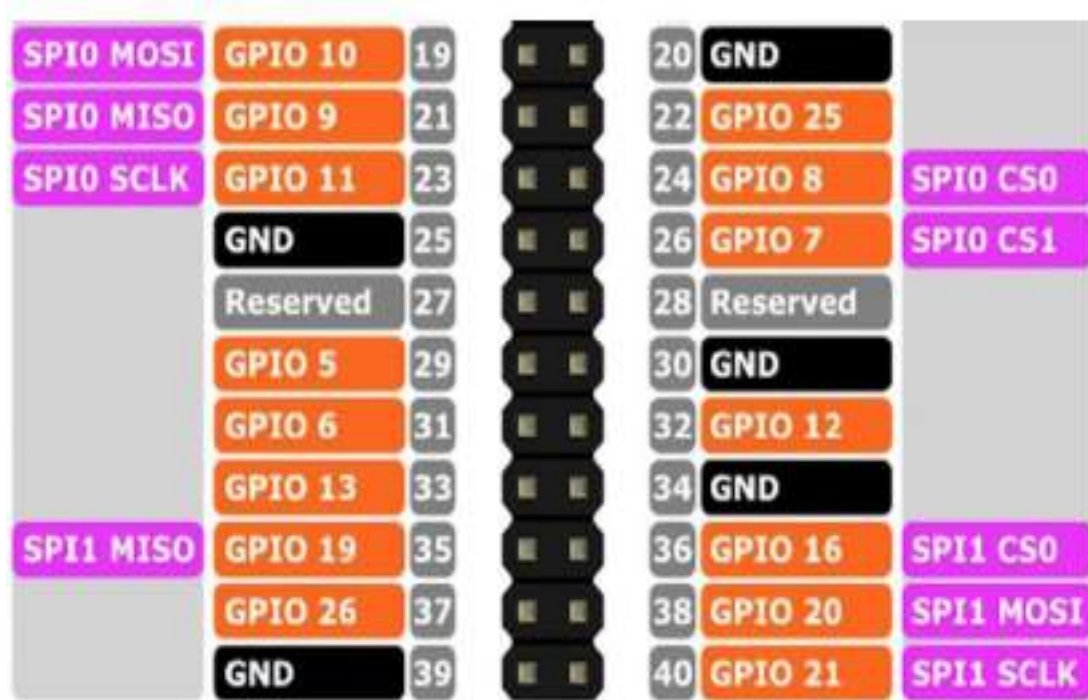
SCL (serial clock line): clock of the I2C. Connect all the slaves SCL to the SCL bus.

SDA (serial data line): exchanged data. Connect all the slaves SDA to the SDA bus.

Note that the SDA and SCL pins on the Raspberry Pi are alternate functions for GPIO 2 and 3. When you use a library (Python, Cpp, etc) for I2C, those two GPIOs will be configured so they can use their alternate function.

SPI (Serial Peripheral Interface): is a synchronous serial data protocol used for communicating with one or more peripheral devices.

SPI is yet another hardware communication protocol. It is a master-slave bus protocol and require more wires than I2C, but can be configured to run faster.



2 SPIs pins by default are: SPI0 and SPI1

For I2C, SPI uses the alternate functions of GPIOs. For using SPI you'll need 5 pins:

GND: what a surprise! Make sure you connect all GND from all your slave components and the Raspberry Pi together.

SCLK: clock of the SPI. Connect all SCLK pins together.

MOSI: means Master out Slave In. This is the pin to send data from the master to a slave.

MISO: means Master in Slave Out. This is the pin to receive data from a slave to the master.

CS: means Chip Select need one CS per slave on your circuit. By default you have two CS pins (CS0 – GPIO 8 and CS1 – GPIO 7). You can configure more CS pins from the other available GPIOs.

In your code, you can use the spidev library for Python, and WiringPi for Cpp.

Raspberry Pi Serial Communication standards (Serial Transfer Schemes)

Serial communication uses two methods

- Synchronous.
- Asynchronous.

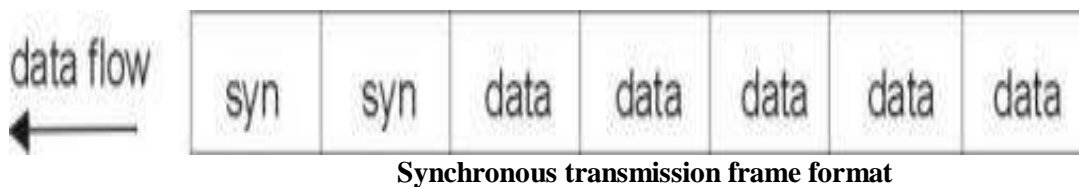
Synchronous:

- Transfers a block of data (packets or frame) at a time.
- Requires clock signal
- Synchronous transmission is fast.
- Synchronous transmission is costly.

In Synchronous transmission, time interval of transmission is constant.

In Synchronous transmission, there is no gap present between data.

The data blocks are grouped and spaced in regular intervals and are preceded by special characters called syn or synchronous idle characters. See the following illustration.



After the sync characters are received by the remote device, they are decoded and used to synchronize the connection. After the connection is correctly synchronized, data transmission may begin.

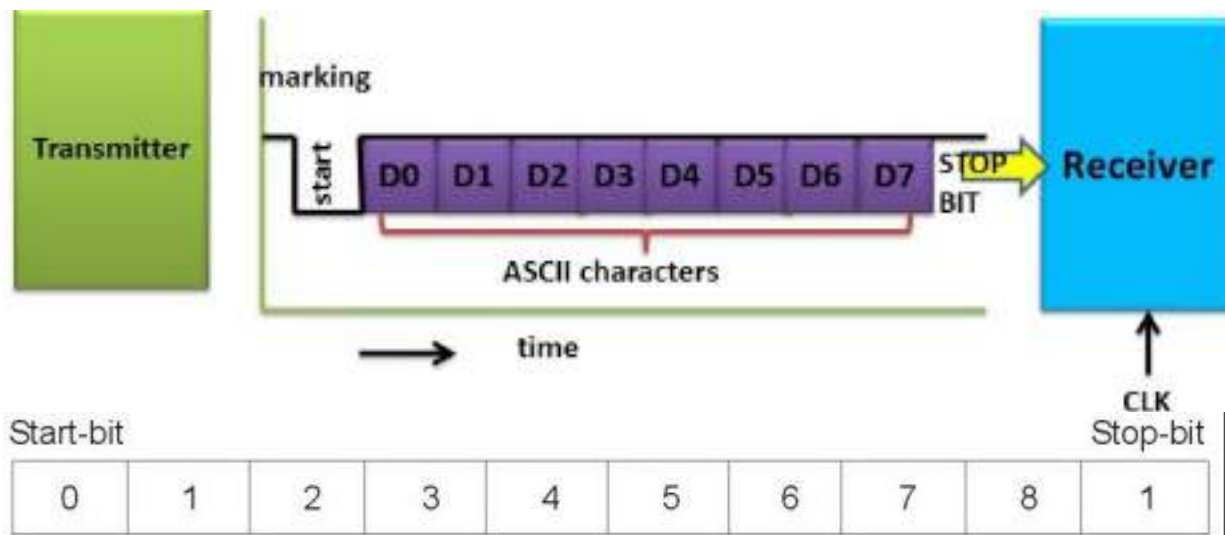
Example: 1. SPI (serial peripheral interface),
I2C (inter integrated circuit).
Ethernet

Asynchronous:

- transfers single byte at a time
- No need of clock signal.
- Asynchronous transmission is slow.
- Asynchronous transmission economical.

In asynchronous transmission, time interval of transmission is not constant, it is random.

In asynchronous transmission, there is present gap between data.



Example: 1. UART (universal asynchronous receiver transmitter)
 2. USB
 3. CAN

INTERNET OF THINGS

UNIT – 4

Raspberry Pi UART Communication

UART (Universal Asynchronous Receiver/Transmitter)

U: universal means this protocol can be applied to any transmitter and receiver

A: Asynchronous means here we do not use clock signal for communication of data.

Combined UART is a serial communication protocol in which data is transferred serially i.e. bit by bit.

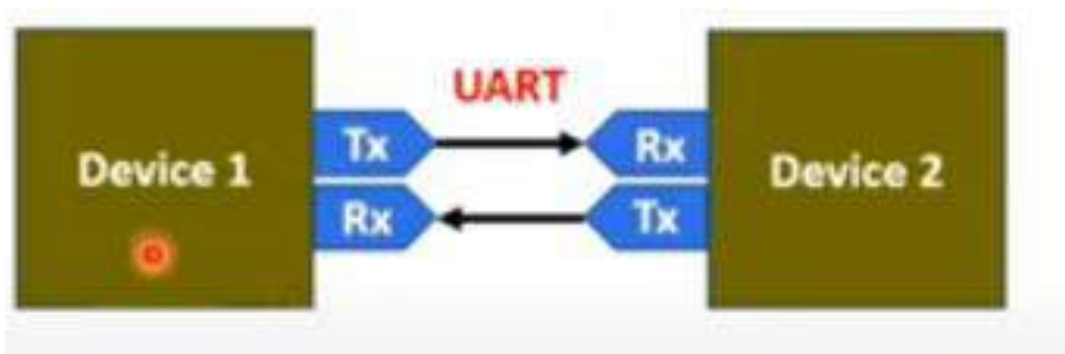
Asynchronous serial communication is widely used for byte oriented transmission. In Asynchronous serial communication, a byte of data is transferred at a time.

UART is two wire communication protocols (one for transmission and one for reception)

Data format and transmission speeds are configurable means

Before starting communication you have to define data format and transmission feed.

Here clock is not given for synchronization



Comparing serial UART communication with parallel



When we compare serial communication of UART with parallel. In parallel we need to have many buses, based on number of lines in terms of bus complexity UART is better but parallel communication is good in terms of speed.

Configuration of UART

- Both devices should be configured with same transmission speed (Baud rate)
- Fir data length (5 to 9 bits)

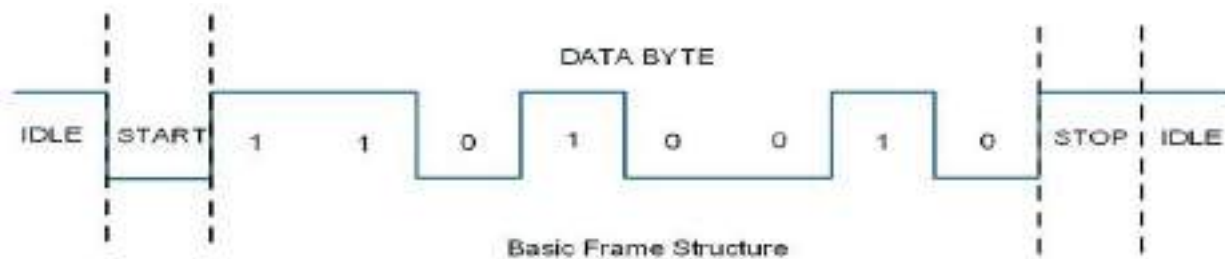
It requires start bit and stop bit

UART serial communication protocol uses a defined frame structure for their data bytes. Frame structure in Asynchronous communication consists:

Start bit: It is a bit with which indicates that serial communication has started and it is always low.

Data bits packet: Data bits can be packets of 5 to 9 bits. Normally we use 8-bit data packet, which is always sent after the start bit.

Stop bit: This usually is one or two bits in length. It is sent after data bits packet to indicate the end of frame. Stop bit is always logic high.



Usually, an asynchronous serial communication frame consists of a START bit (1 bit) followed by a data byte (8 bits) and then a STOP bit (1 bit), which forms a 10-bit frame as shown in the figure, above. The frame can also consist of 2 STOP bits instead of a single bit, and there can also be a PARITY bit after the STOP bit.

Advantages of UART

Less physical interface
Simple configuration
Data size is configurable
Speed is configurable
Full duplex can be configured by two wires

Disadvantages of UART

Less speed of communication
Start and stop bits are required
Asynchronous communication

Raspberry Pi UART

Raspberry Pi has two in-built UART which are as follows:

- **PL011 UART**
- **mini UART**

PL011 UART is an ARM based UART. This UART has better throughput than **mini** UART.

In Raspberry Pi 3, mini UART is used for Linux console output whereas PL011 is connected to the On-board Bluetooth module.

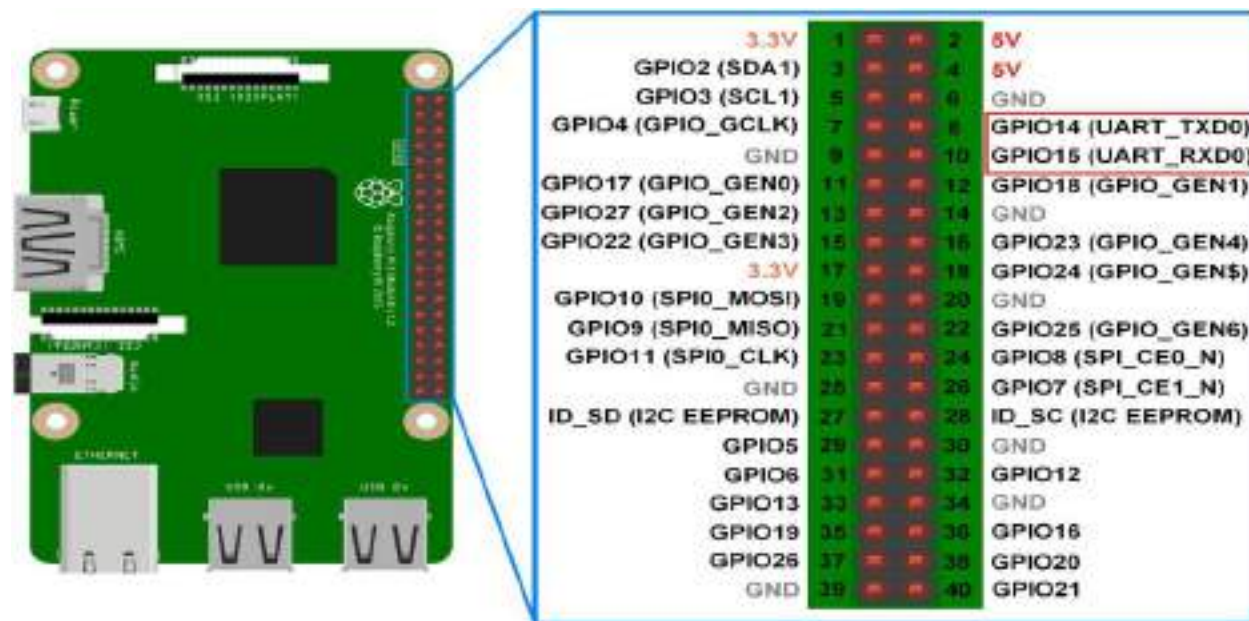
And in the other versions of Raspberry Pi, PL011 is used for Linux console output.

Mini UART uses the frequency which is linked to the core frequency of GPU.

So as the GPU core frequency changes, the frequency of UART will also change which in turn will change the baud rate for UART.

This makes the mini UART unstable which may lead to data loss or corruption. To make mini UART stable, fix the core frequency. Mini UART doesn't have parity support.

The PL011 is a stable and high performance UART. For better and effective communication use PL011 UART instead of mini UART.

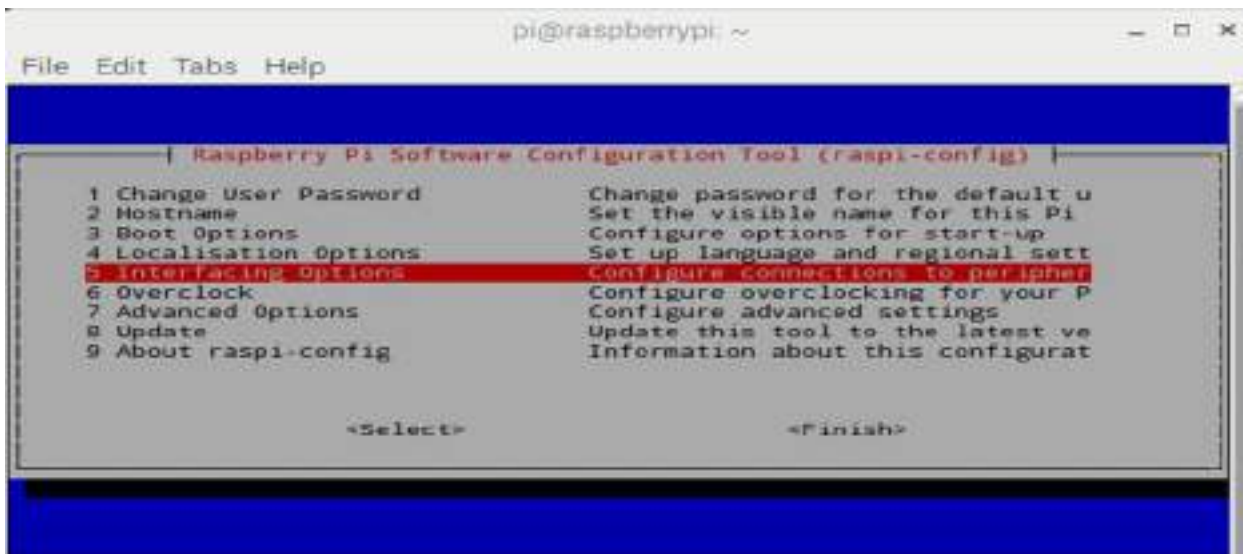


Configure UART on Raspberry Pi

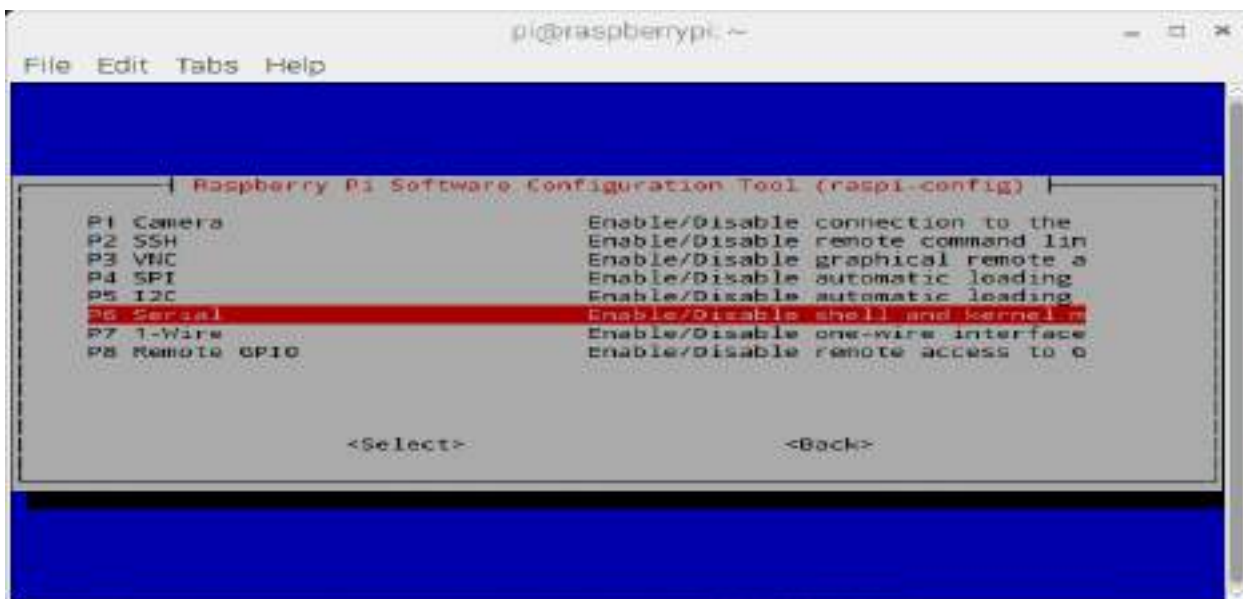
In Raspberry Pi, enter following command in Terminal window to enable UART,

```
sudo raspi-config
```

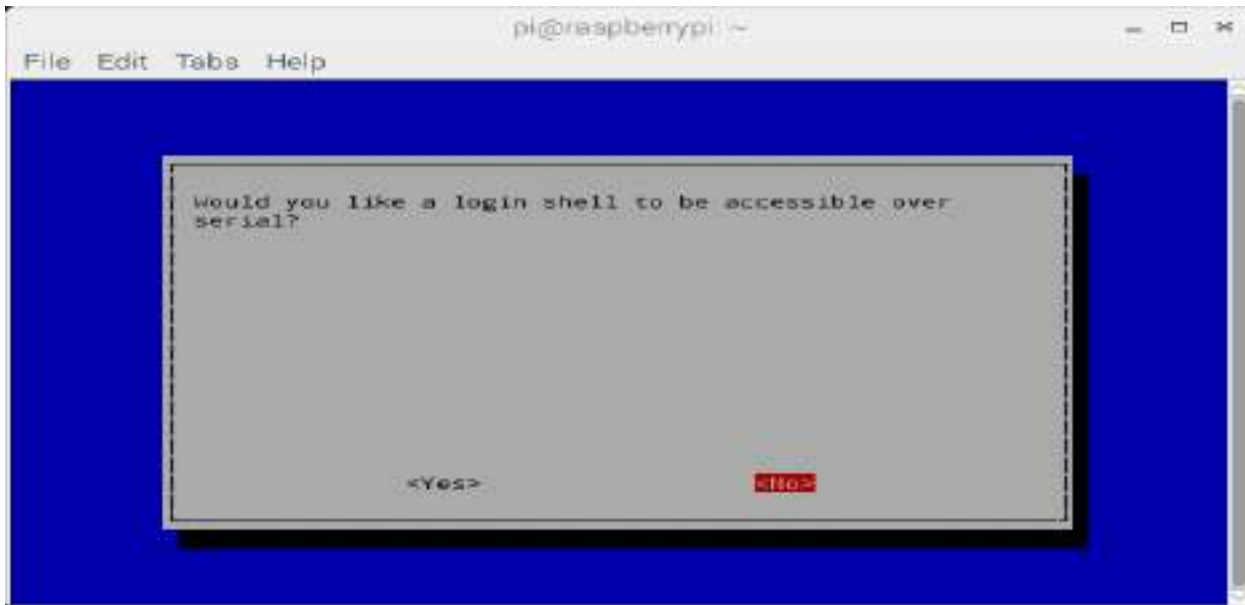
Select -> **Interfacing Options**



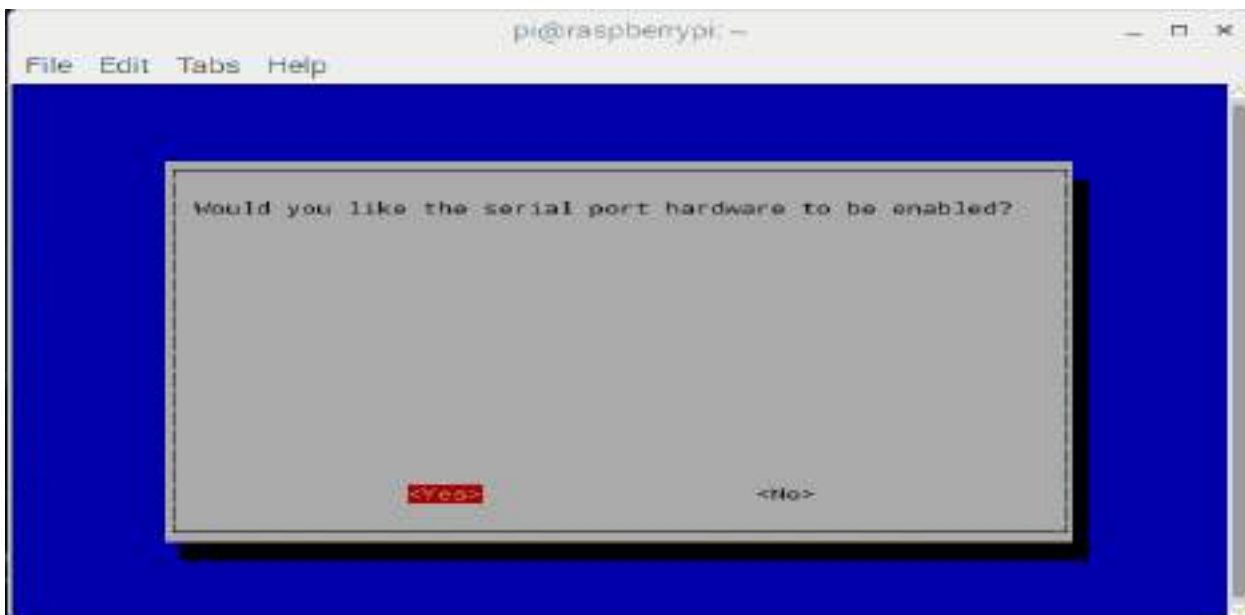
After selecting Interfacing option, select **Serial** option to enable UART



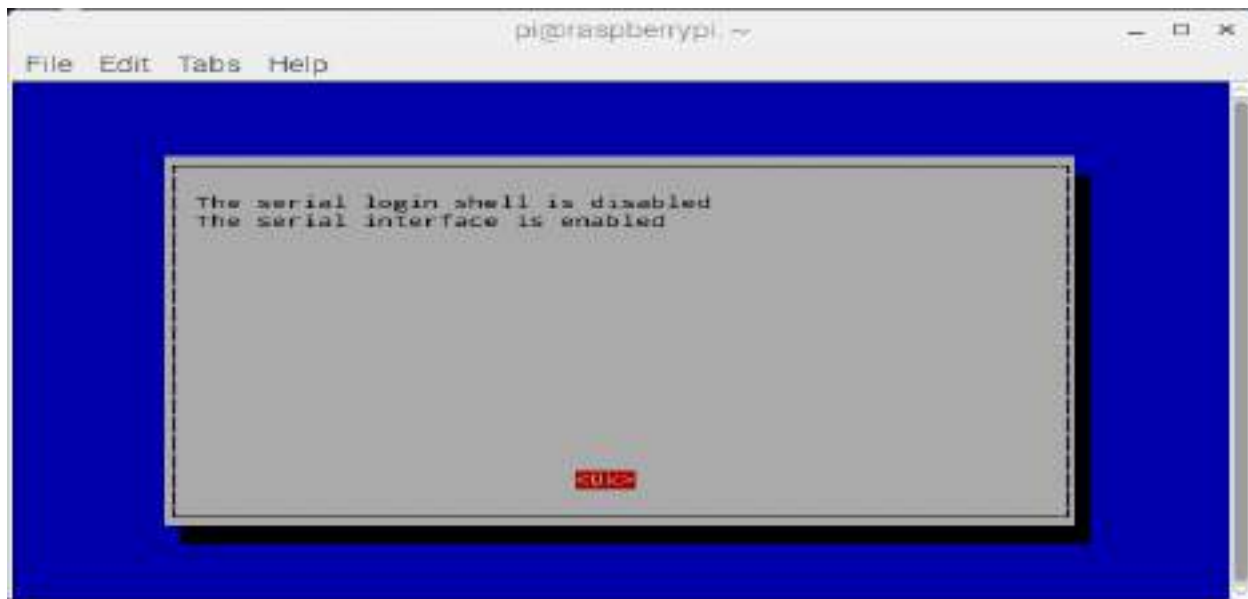
Then it will ask for login shell to be accessible over Serial, select **No** shown as follows.



At the end, it will ask for enabling Hardware Serial port, select **Yes**,



Finally, our UART is enabled for Serial Communication on RX and TX pin of Raspberry Pi 3.



INTERNET OF THINGS

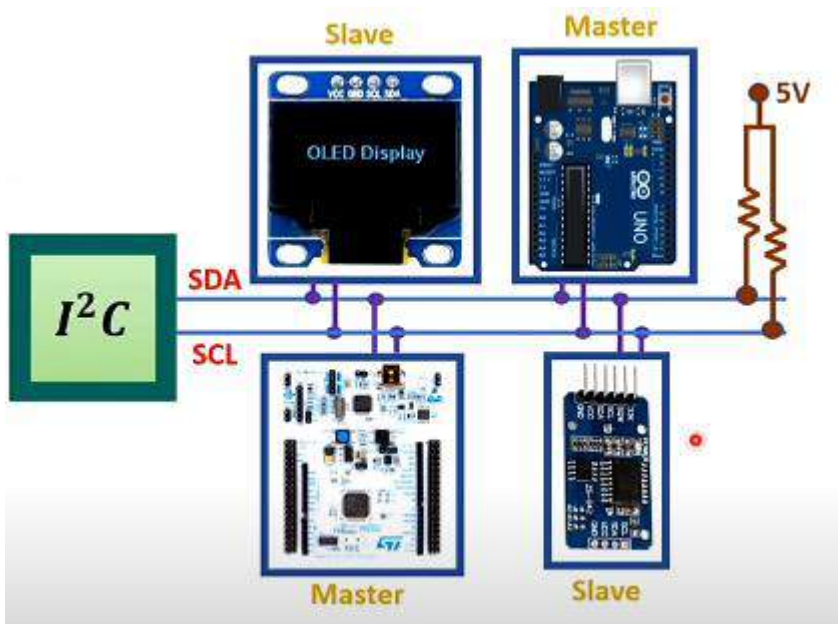
UNIT – 4

Raspberry Pi I2C Communication

I2C (Inter Integrated Circuit)

Introduction

- I2C is a synchronous serial protocol that communicates data between two devices.
- It is a master-slave protocol which may have one master or many master and many slaves whereas SPI has only one master.
- I2C has two lines SDA (serial data line) and SCL (serial clock line), these should be pulled up by 5V connection Via resistor (it is not compulsory because many of the Arduino or any master board has internal pull up available so, by configuring we can make these two lines active high).
- It is generally used for communication over short distance.
- Used for low bandwidth communication but effective communication.



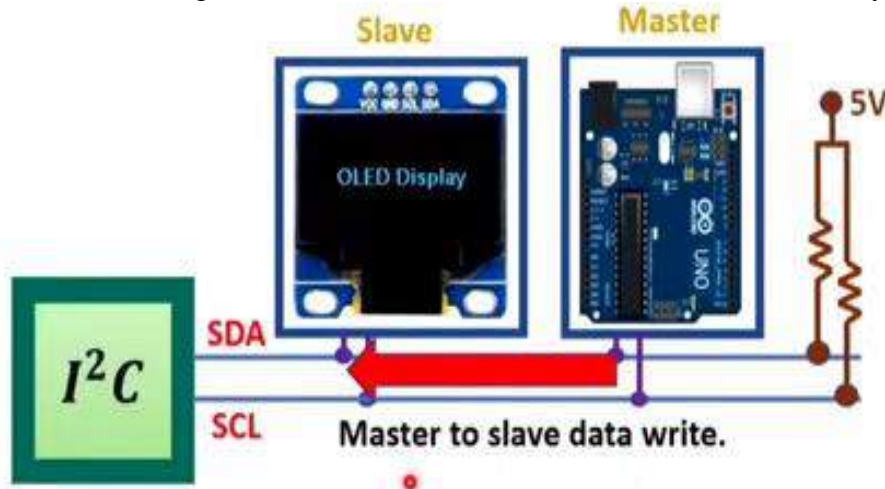
- With just two lines we can interface 128 devices and also called as Two Wire Interface (TWI) protocol.
- The I2C device has 7-bit or 10-bit unique address. So, to access these devices, master should address them by the 7-bit or 10-bit unique address
- I2C is used in many applications like reading RTC (Real time clock), accessing external EEPROM memory. It is also used in sensor modules like gyro, magnetometer etc.
- Master is initiating data transfer and controls the clock manually and the device being addressed is slave.

How master can send data to slave

Master initiates data transfer by sending start bit (0)

There can be many devices on the line so master should send the address of slave on the line with respect to clock

Master sends write bit (0) to slave and slave gives acknowledgement to master (Ack bit = 0) once master receives acknowledgement write the data and terminate communication by sending stop bit (0).

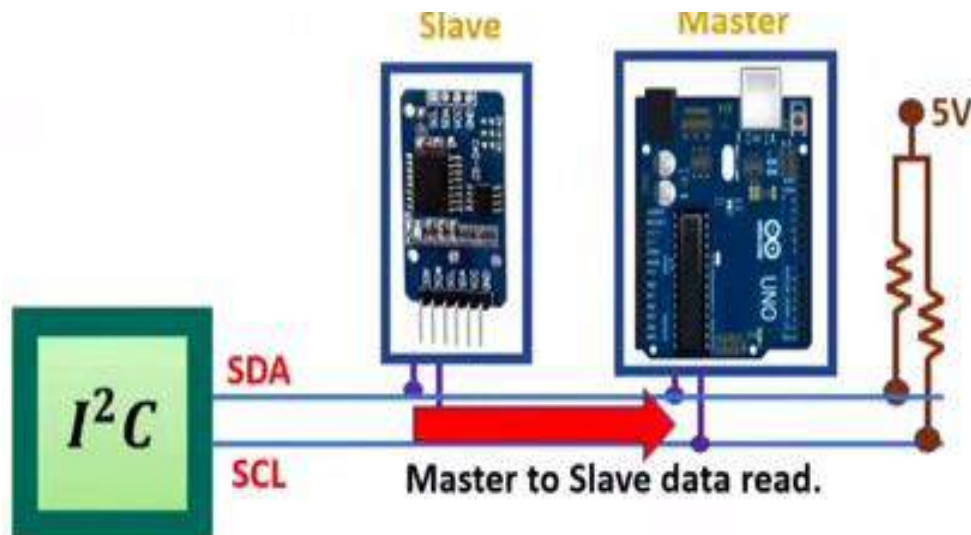


Master reads data from slave

Master initiates signaling by start bit (0) , then address of slave is forwarded by master.

Master reads data by read bit (0) , then slave sends acknowledgement (Ack=0) i.e slave is ready to send data (011011) once communication is complete master will give acknowledgement to slave (Ack=1).

Last master will terminate communication by stop bit(0).



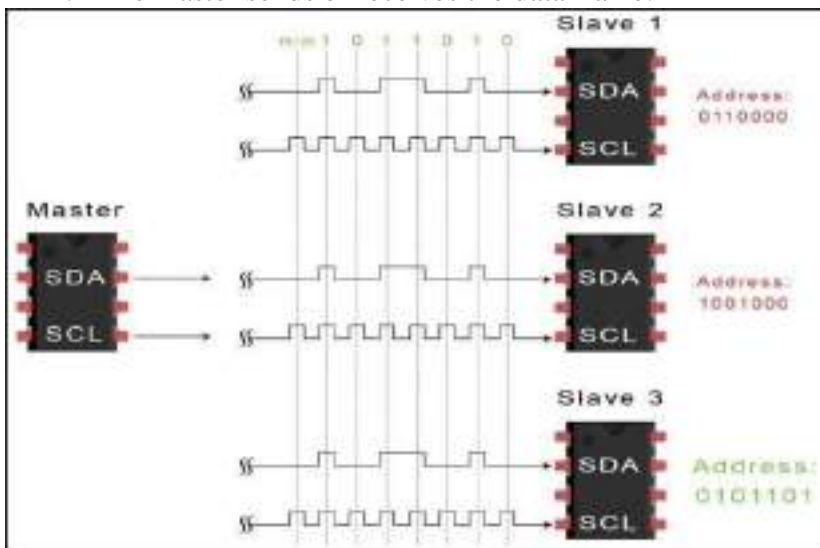
Raspberry Pi I2C

- Raspberry Pi has Broadcom processor having Broadcom Serial Controller (BSC) which is a master, fast-mode (400Kb/s) BSC controller. The BSC bus is compliant with the Philips I2C bus.

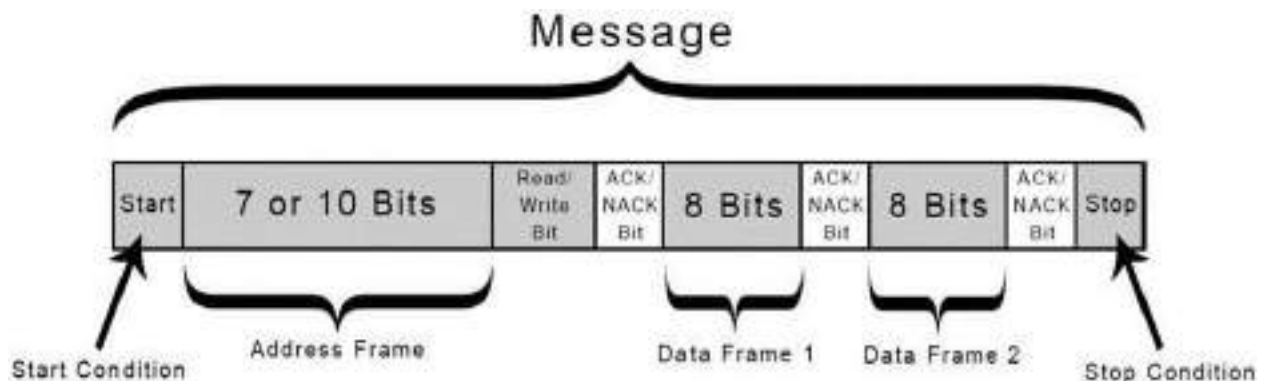
- It supports both 7-bit and 10-bit addressing.
- It also has BSC2 master which is dedicatedly used with HDMI interface and should not be accessed by user.
- I2C bus/interface is used to communicate with the external devices like RTC, MPU6050, Magnetometer, etc with only 2 lines. We can connect more devices using I2C interface if their addresses are different.

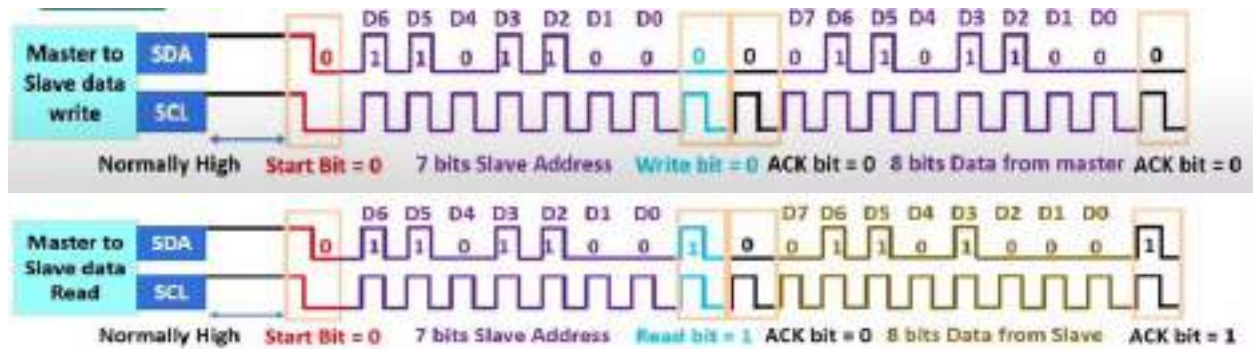
STEPS OF I2C DATA TRANSMISSION

1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level *before* switching the SCL line from high to low:
2. The master sends each slave the 7 or 10 bit address of the slave it wants to communicate with, along with the read/write bit:
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, SDA line high.
4. The master sends or receives the data frame:



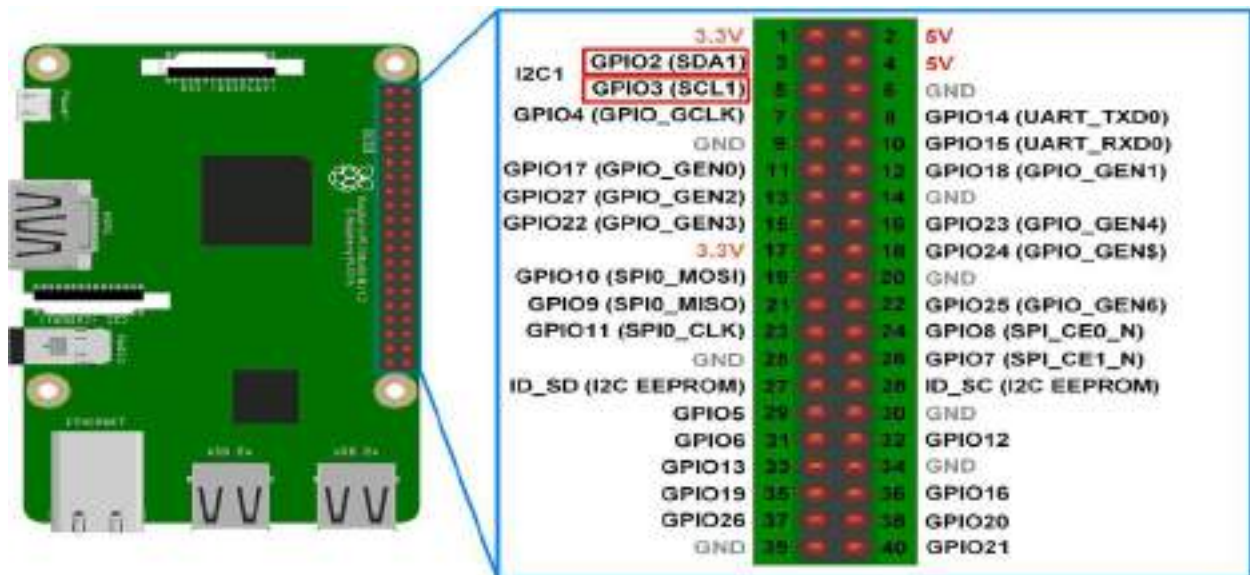
5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame:
6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high





To access I2C bus in Raspberry Pi, we should make some extra configuration. Raspberry Pi has I2C pins which are given as follows.

Raspberry Pi I2C Pins

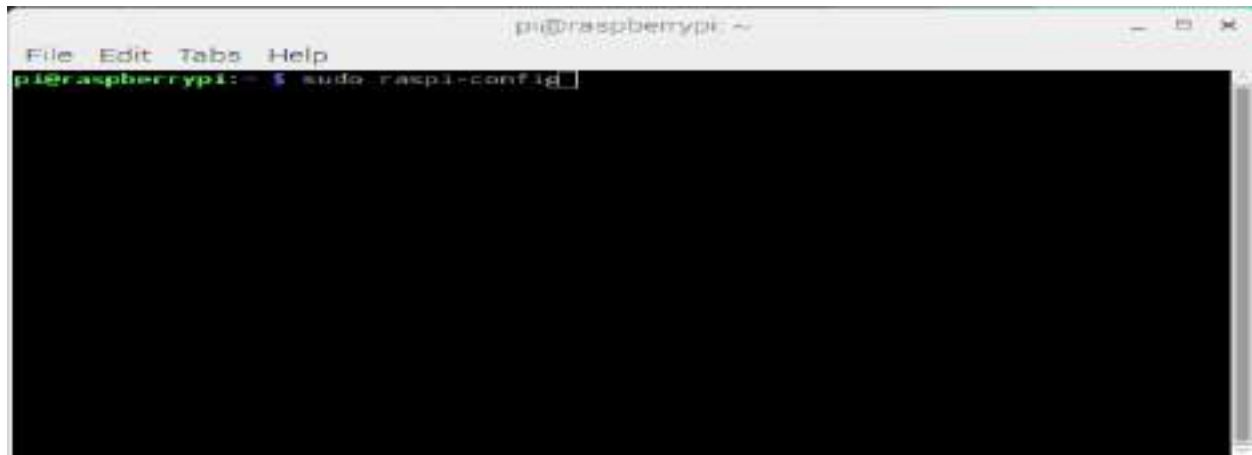


Raspberry Pi I2C Configurations

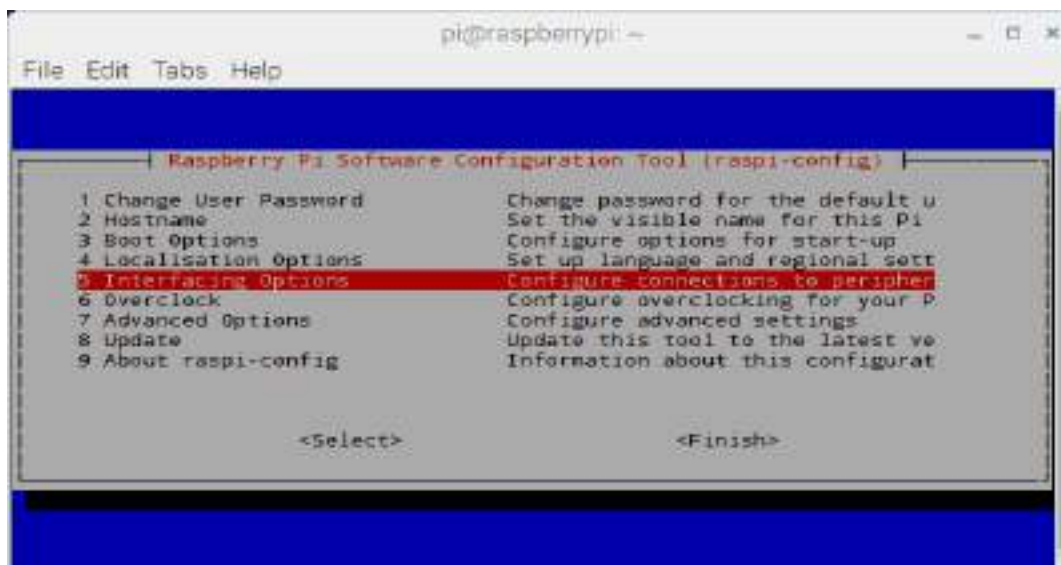
Before start interfacing I2C devices with Raspberry some prior configurations need to be done. These Configurations are given as follows:

First, we should enable I2C in Raspberry Pi. We can enable it through terminal which is given below:

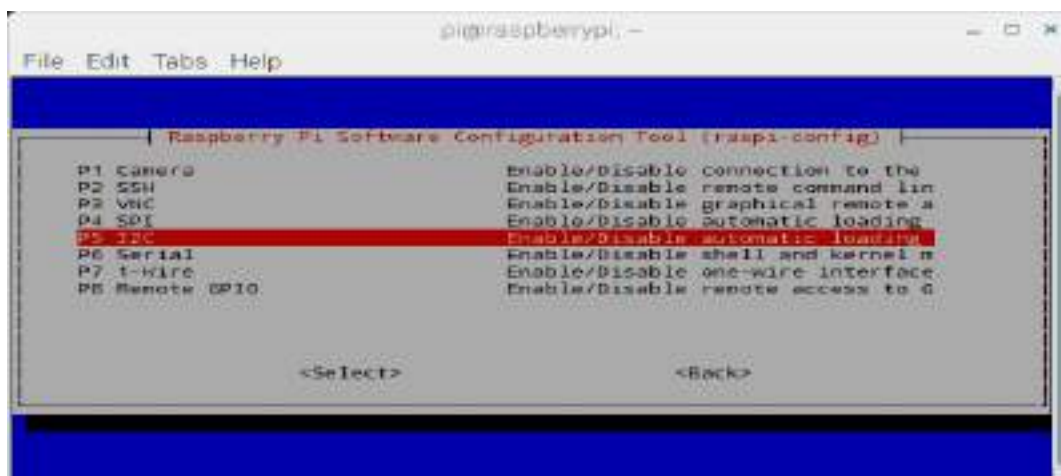
```
sudo raspi-config
```



Select **Interfacing Configurations**



In Interfacing option, Select -> **I2C**



Enable I2C configuration

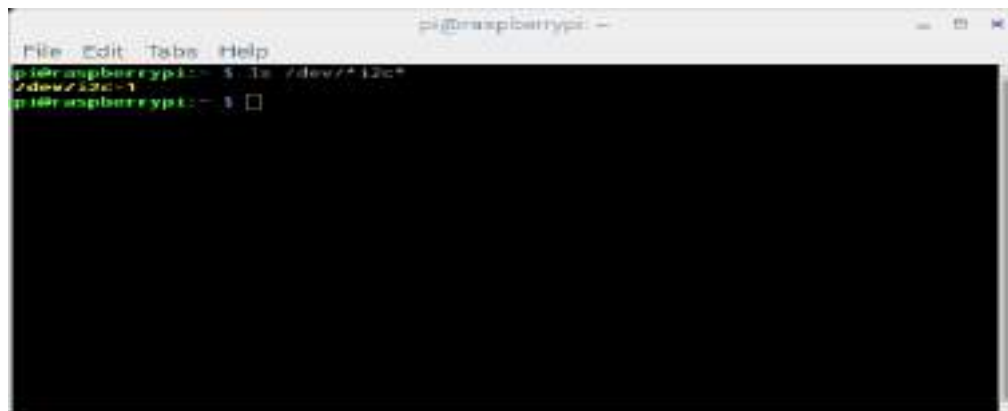


Select **Yes** when it asks to **Reboot**.

Now, after booting raspberry Pi, we can check user-mode I2C port by entering following command.

```
ls /dev/*i2c*
```

then Pi will respond with name of i2c port

A terminal window titled 'pi@raspberrypi: ~' with a menu bar (File, Edit, Tabs, Help). The terminal shows a sequence of commands and their outputs: 'pi@raspberrypi:~\$ ls /dev/i2c*' returns '/dev/i2c-1', and 'pi@raspberrypi:~\$' is followed by a cursor and a small square icon.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ ls /dev/i2c*  
/dev/i2c-1  
pi@raspberrypi:~$ □
```

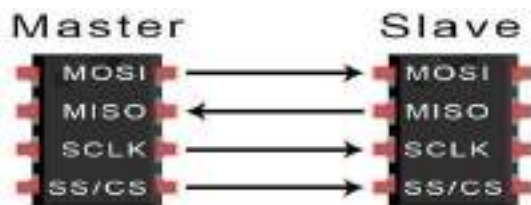
Above response represents the user-mode of I2C interface. Older versions of Raspberry pi may respond with **i2c-0** user-mode port.

INTERNET OF THINGS

UNIT – 4

SPI COMMUNICATION PROTOCOL

- SPI is a common communication protocol used by many different devices. Ex: SD card modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers.
- One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream.
- Devices communicating via SPI are in a master-slave relationship.
Master: controlling device (usually a microcontroller),
Slave: takes instruction from the master (usually a sensor, display, or memory chip).
- The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave (more on this below).



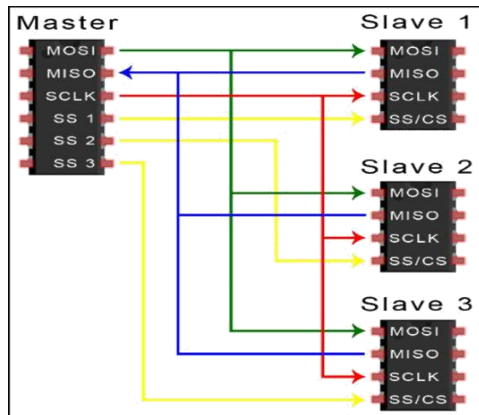
MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.

MISO (Master Input/Slave Output) – Line for the slave to send data to the master

SCLK (Clock) – Line for the clock signal

SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

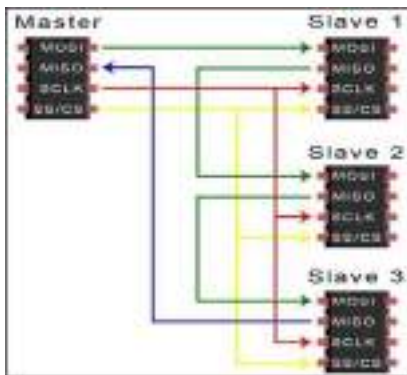
- The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave.
- One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal.
- SPI communication is always initiated by the master since the master configures and generates the clock signal.
- Any communication protocol where devices share a clock signal is known as Slave select
- The master can choose which slave it wants to talk to by setting a low voltage level.
- In the idle, non-transmitting state, the slave select line is kept at a high voltage level.
- Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel.
- MULTIPLE SLAVES



SPI can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master.

There are two ways to connect multiple slaves to the master. If the master has multiple slaves select pins, the slaves can be wired in parallel.

If only one slave select pin is available, the slaves can be daisy-chained like this:



MOSI AND MISO

The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first.

The slave can also send data back to the master through the MISO line in serial. The data sent from the slave back to the master is usually sent with the least significant bit first.

STEPS OF SPI DATA TRANSMISSION

1. The master outputs the clock signal:

2.



2. The master switches the SS/CS pin to a low voltage state, which activates the slave:



3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:



4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:



| Serial Protocol | Synchronous /Asynchronous | Type | Duplex | Data transfer rate (kbps) |
|-----------------|---------------------------|---------------------------|-------------|---------------------------|
| UART | Asynchronous | peer-to-peer | Full-duplex | 20 |
| I2C | Synchronous | master/slave multi-master | Half-duplex | 3400 |
| SPI | Synchronous | master/slave multi-master | Full-duplex | >1,000 |

INTERNET OF THINGS

UNIT – 4

Raspberry Pi Programming Basics

RPi.GPIO API Library

RPi.GPIO API is the library to launch input and output pins

This set of Python files and source is included with Raspbian OS --NOOBS, (Linux based)

Setup RPi.GPIO

RPi.GPIO library is available on Raspbian operating system by default and you don't need to install it.

In order to use RPi.GPIO in Python script, you need to put below statement at the top of your file:

```
import RPi.GPIO as GPIO
```

Pin Numbering Declaration

Two pin-numbering schemes

GPIO.BOARD: In Board numbering scheme. The pin numbers followed are the pin numbers on header P1.

GPIO.BCM: In Broadcom chip-specific pin numbers. The pin numbers followed are lower-level numbering system defined by the Raspberry Pi's Broadcom-chip brain.

To specify in your code which number-system is being used, use the GPIO.setmode() function.

Example:

```
GPIO.setmode(GPIO.BCM) #will activate the Broadcom-chip specific pin numbers.
```

Note: Both the import and setmode lines of code are required, if you want to use GPIO pins in Python.

Setting a Pin Mode:

In Raspberry Pi to set a pin mode, use the

```
GPIO.setup(pin number, pinmode) function.
```

Example: if you want to set pin 18 as an output, write:

```
GPIO.setup(18, GPIO.OUT)
```

Note: Remember that the pin number will change if you're using the board numbering system (instead of 18, it'd be 12).

Example:

```
GPIO.setmode(GPIO.BCM); Using pins BCM numbers
```

```
GPIO.setup(18, GPIO.OUT)
```

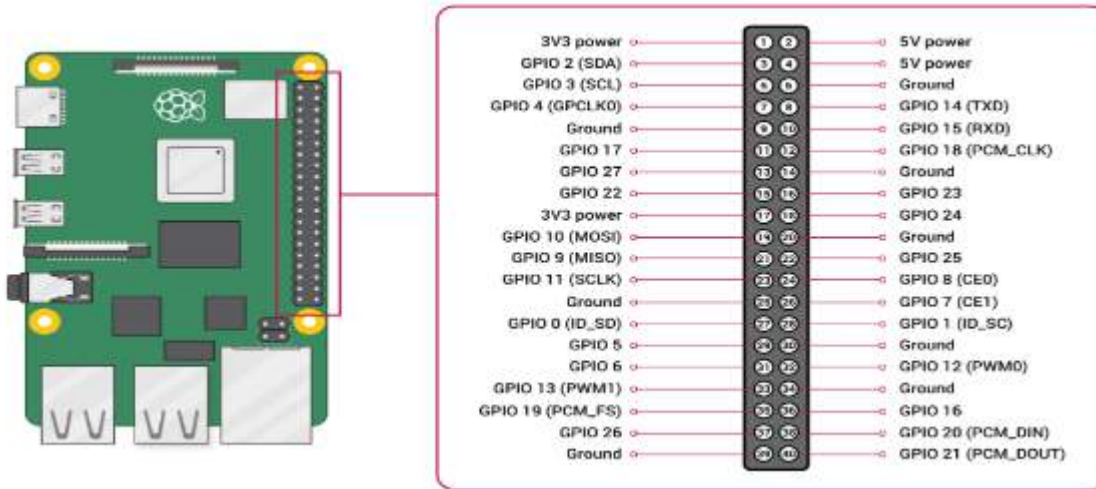
OR

```
GPIO.setmode(GPIO.BOARD); Using pins header P1 numbers
```

```
GPIO.setup(12, GPIO.OUT)
```

In above codes same physical Pin is used with different numbering Schemes

Raspberry Pi Pin diagram



Broadcom (BCM) pin names

| | | | |
|--|--------|--------|---|
| Available for GPIO if I2C is disabled using raspi-config | 3.3V | 5V | Available for GPIO if serial is disabled using raspi-config |
| | GPIO2 | 5V | |
| | GPIO3 | GND | |
| | GPIO4 | GPIO14 | |
| Used by DotStars GPIO unavailable | GND | GPIO15 | Pins above this line are present on all Raspberry Pi boards Modes A+, B+ and Pi 2 |
| | GPIO17 | GPIO18 | |
| | GPIO27 | GND | |
| | GPIO22 | GPIO23 | |
| GPIO Availability: No Maybe Yes | 3.3V | GPIO24 | |
| | GPIO10 | GND | |
| | GPIO9 | GPIO25 | |
| | GPIO11 | GPIO8 | |
| | GND | GPIO7 | |
| | DNC | DNC | |
| | GPIO5 | GND | |
| | GPIO6 | GPIO12 | |
| | GPIO13 | GND | |
| | GPIO19 | GPIO16 | |
| | GPIO26 | GPIO20 | |
| | GND | GPIO21 | |

Basic Commands

`GPIO.setup(pin, GPIO.IN)` # Determines the pin as input

`GPIO.setup(pin, GPIO.OUT)` # Determines the pin as an output

`GPIO.input(pin)` # Reading input pin

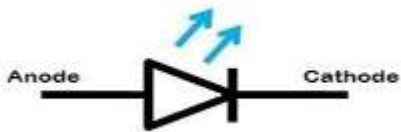
`GPIO.output(pin, state)` # Writing on the output pin.

Programming Raspberry Pi with Python

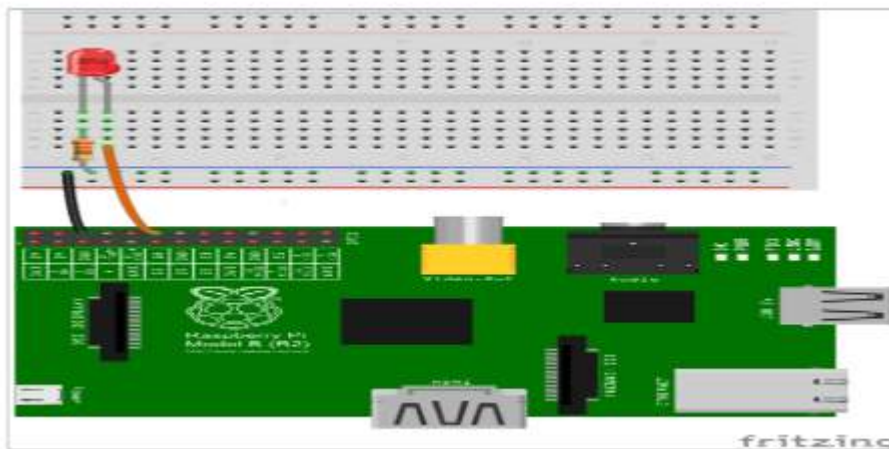
Controlling LED

LED allows the flow of current in the forward direction and blocks the current in the reverse direction.

The LED includes two terminals namely anode (+) and the cathode (-). The LED symbol is shown below



Interfacing LED with Raspberry Pi



Controlling LED ON/OFF PROGRAM

```
import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.OUT)

while True:

    GPIO.output(18, True)
```

```
time.sleep(2)
```

```
GPIO.output(18, False)
```

```
time.sleep(2)
```

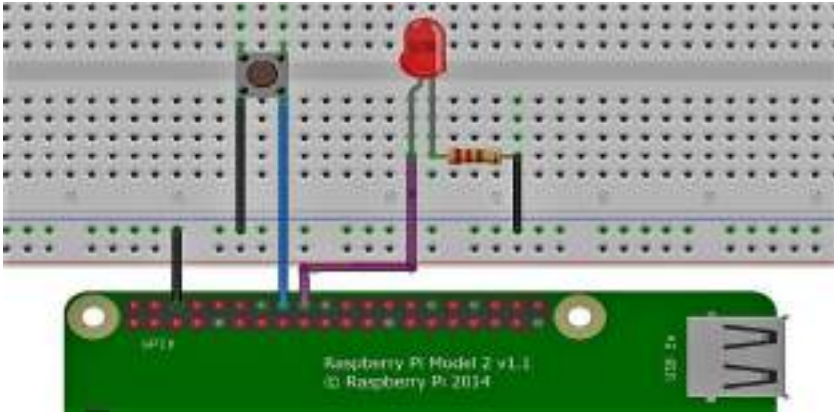
OUTPUT SNAPSHOTS:



INTERNET OF THINGS

UNIT – 4

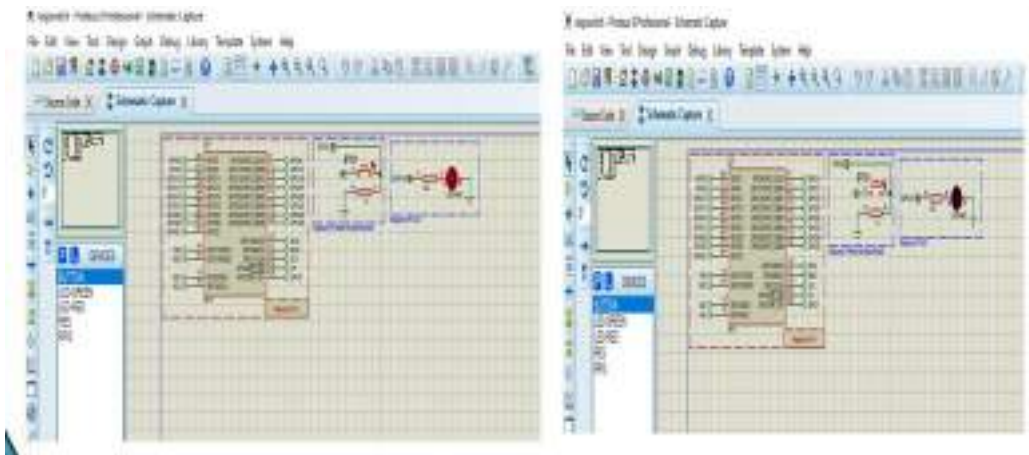
Integration of LED and Switch



Program:

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.OUT)
GPIO.setup(23,GPIO.IN)
# input of the switch will change the state of the LED
while True:
    GPIO.output(18,GPIO.input(4))
    time.sleep(0.05)
```

OUTPUT SNAPSHOTS:



PWM Illustration in Raspberry Pi

CODE:

```
import RPi.GPIO as GPIO
from time import sleep
ledpin = 12          # PWM pin connected to LED
GPIO.setwarnings(False)    #disable warnings
GPIO.setmode(GPIO.BOARD)   #set pin numbering system
GPIO.setup(ledpin,GPIO.OUT)
pi_pwm = GPIO.PWM(ledpin,1000)    #create PWM instance with frequency helps to
                                   generate PWM signal
pi_pwm.start(0)           #start PWM of required Duty Cycle

while True:
    for duty in range(0,101,1):
        pi_pwm.ChangeDutyCycle(duty) #provide duty cycle in the range 0-100
        sleep(0.01)
        sleep(0.5)

    for duty in range(100,-1,-1):
        pi_pwm.ChangeDutyCycle(duty)
        sleep(0.01)
        sleep(0.5)
```


OUTPUT:



FIGURE 2: LED IS ON WITH WHEN PWM VALUE IS MID RANGE (MEDIUM BRIGHTNESS)



FIGURE 3: LED IS ON WITH WHEN PWM VALUE IS HIGH RANGE (MORE BRIGHTNESS)

INTERNET OF THINGS

UNIT – 4

LAB EXPERIMENT 8: MEASURE THE DISTANCE USING ULTRASONIC SENSOR WITH RASPBERRY Pi

Interfacing Ultrasonic Sensor with Raspberry Pi

- An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.



HC-SR04 Ultrasonic Sensor Pin Configuration

- Includes four pins and the pin configuration of this sensor is discussed below.
- Pin1 (Vcc): This pin provides a +5V power supply to the sensor.
- Pin2 (Trigger): This is an input pin, used to initialize measurement by transmitting ultrasonic waves by keeping this pin high for 10us.
- Pin3 (Echo): This is an output pin, which goes high for a specific time period and it will be equivalent to the duration of the time for the wave to return back to the sensor.
- Pin4 (Ground): This is a GND pin used to connect to the GND of the system.

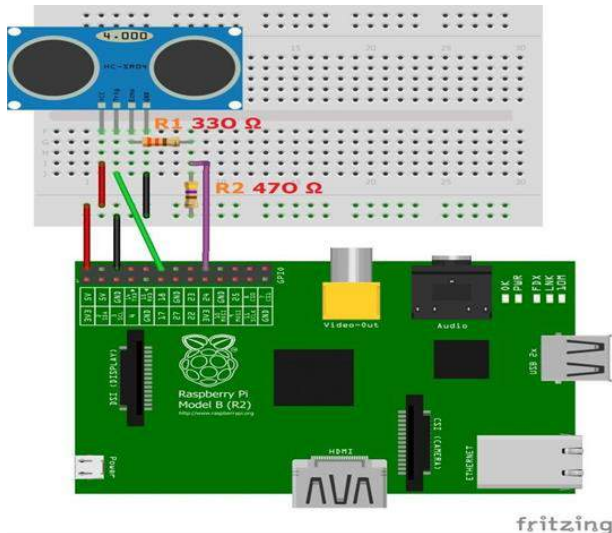
Features:

- The [power supply](#) is +5V DC
- Dimension is 45mm x 20mm x 15mm
- Quiescent current used for this sensor is <2mA
- The input pulse width of trigger is 10uS
- Operating current is 15mA
- Measuring angle is 30 degrees
- The distance range is 2cm to 800 cm
- Resolution is 0.3 cm
- Effectual Angle is <15°
- Operating frequency range is 40Hz
- Accuracy is 3mm

Connections

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2(VCC)
- GND to Pin 6(GND)
- TRIG to Pin12(GPIO18)
- Connect the 330Ω resistor to ECHO. On it send you connect it to Pin18 (GPIO24) and through a 470Ω resistor you connect it also to Pin6(GND).



Program:

```
import RPi.GPIO as GPIO import time
GPIO.setmode(GPIO.BCM)                #GPIOMode(BOARD/ BCM)
GPIO_TRIGGER=18                        #set GPIO Pins
GPIO_ECHO=24
GPIO.setup(GPIO_TRIGGER, GPIO.OUT) #set GPIO direction (IN / OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
def distance ():                       # set Trigger to HIGH GPIO.output(GPIO_TRIGGER,True)
GPIO.output(GPIO_TRIGGER,True)        # set Trigger to HIGH
time.sleep(0.00001)
GPIO.output(GPIO_TRIGGER,False)       # set Trigger after 0.01ms to LOW
StartTime = time.time()
StopTime=time.time()

                                     # Save StartTime
while GPIO.input(GPIO_ECHO) == 0:
StartTime=time.time()
while GPIO.input(GPIO_ECHO) == 1: StopTime=time.time() # save time of arrival
TimeElapsed=StopTime-StartTime        # time difference between start and arrival
distance = (TimeElapsed * 34300) / 2   #multiply with the
                                     sonicspeed(34300cm/s) #and
                                     divide by2, accounts for the round
                                     trip of the sound wave.

return distance
```

```

if name== 'main':
try:
while True:
    dist=distance ()
    print ("Measured Distance = %.1f cm" % dist)
    time.sleep(1)          # Reset by pressing CTRL + C
except KeyboardInterrupt:
    print ("Measurement stopped by User")
Finally ():
GPIO.cleanup()

```

OUTPUT:



INTERNET OF THINGS

UNIT – 4

LAB EXPERIMENT 5:

Print temperature and humidity using DHT22 sensor with Raspberry Pi

Interfacing DHT22 Sensor with Raspberry Pi

DHT22: Digital Humidity Temperature sensor

The DHT-22 (also named as AM2302) is a digital-output relative humidity and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.

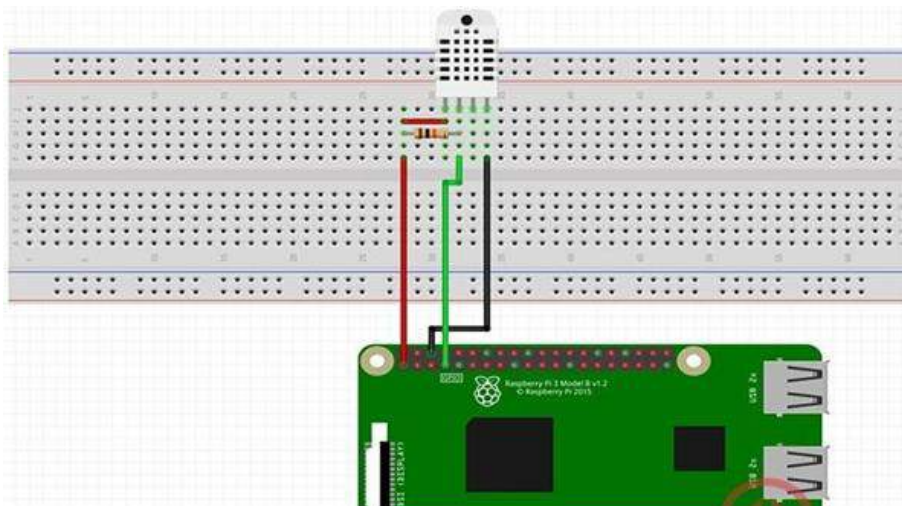
Pin Description:

It has only four pins.

- Vcc is the power pin. Apply voltage in a range of 3.5 V to 5.0 V at this pin.
- Data Out is the digital output pin. It sends out the value of measured temperature and humidity in the form of serial data.
- N/C is a not connect pin.
- GND: Connect the GND pin to main ground.



Connections



PRE REQUISTE library Installations:

1.update both the package list and installed packages.

```
Terminal $  
  
sudo apt-get update  
sudo apt-get upgrade
```

2.install [Adafruit's DHT library](#) to the Raspberry Pi.

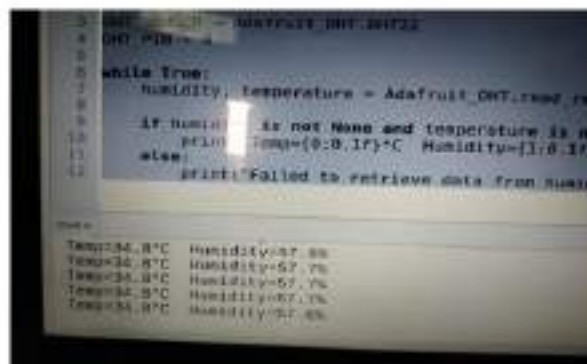
- Run the following command to install the DHT library to your Raspberry Pi.

```
Terminal $  
  
sudo pip3 install Adafruit_DHT
```

Program:

```
import Adafruit_DHT  
DHT_SENSOR =Adafruit_DHT.DHT22 //specifying that you are using DHT22 sensor.  
DHT_PIN = 4  
while True:  
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)  
  
    if humidity is not None and temperature is not None:  
        print("Temp={0:0.1f}*C    Humidity={1:0.1f}%".format(temperature, humidity))  
    else:  
        print("Failed to retrieve data from sensor")
```

OUTPUT



INTERNET OF THINGS

UNIT – 4

LAB EXPERIMENT 11: PERFORM IMAGE CAPTURE, READ, WRITE AND MODIFY THE IMAGE OPERATION USING RASPBERRY PI

Interfacing PI camera with Raspberry Pi

PI camera

- The Pi camera module is a portable light weight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol. It is normally used in image processing, machine learning or in surveillance projects.
- Pi Camera module can be used to take pictures and high-definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach Pi Camera module directly using 15-pin ribbon cable.

PI camera



PROGRAM 1

#capture & modify the image

```
from picamera import PiCamera
from time import sleep
from picamera import Color
camera = PiCamera()
camera.rotation = 180
camera.start_preview()          #what the camera is capturing and apply any effects or
                                #annotations.
camera.image_effect = 'watercolor' #to apply a watercolor effect. Adjust annotation
```

```

                                properties like background color
camera.annotate_background = Color('yellow')
camera.annotate_foreground = Color('blue')
camera.annotate_text_size = 80
camera.annotate_text = "Hello KMIT"
for i in range(5): # to capture multiple images.
    sleep(5)       # pause for every 5 seconds between captures
camera.capture('/home/pi/Desktop/Rasp1%s.jpg' % i) # save each captured image in
                                                    this file and location

camera.stop_preview()

```

PROGRAM 2

#video saving

```

from picamera import PiCamera
from time import sleep
from picamera import PiCamera, Color
camera = PiCamera()
camera.rotation = 180
camera.start_preview()
camera.annotate_background = Color('yellow')
camera.annotate_foreground = Color('blue')
camera.annotate_text_size = 80
camera.annotate_text = " KMIT IT " camera.start_recording('/home/pi/Desktop/video.h264')
sleep(5)                # wait for 5 sec to simulate a recording duration
camera.stop_recording()
camera.stop_preview()
camera.close()          # This ensures that the camera is released correctly, preventing
                        potential issues when running the script multiple times.

```

OUTPUT



INTERNET OF THINGS

UNIT – 4

LAB EXPERIMENT 6: CONTROL PIR BASED LED ON/OFF WITH RASPBERRY Pi

Interfacing PIR Sensor with Raspberry Pi

PIR SENSOR

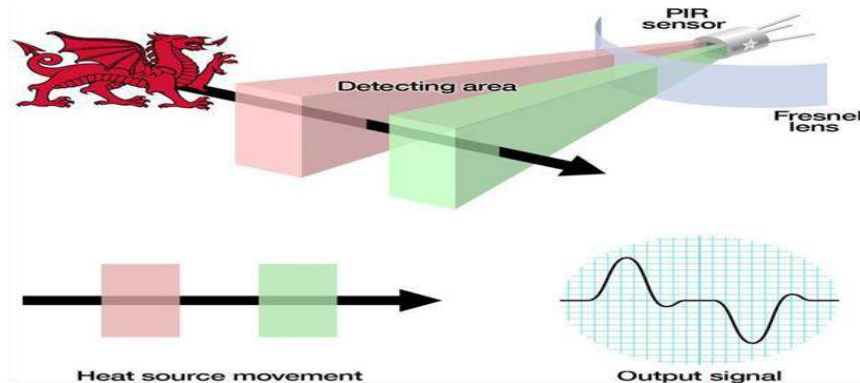
- A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. (range)
- PIR sensors are commonly used in security alarms and automatic lighting applications.
- PIR sensors detect general movement, but do not give information on who or what moved.

PIN DESCRIPTION

- Pin1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC.
- Pin2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor.
- Pin3 of the sensor connected to the ground.

PIR WORKING

- The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors.
- When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves.
- When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.



Program

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)    #Read output from PIR motion sensor
GPIO.setup(3, GPIO.OUT)    #LED output pin
While True:
    i=GPIO.input(11)
    if i == 0:             #When output from motion sensor is LOW print ("No Person detected",i)
        GPIO.output(3,0)
        time.sleep(0.1)
    elif i == 1:          #When output from motion sensor is HIGH print ("A Person is detected",i)
        GPIO.output(3,1)    #Turn ON LED
        time.sleep(0.1)
```

OUTPUT

