IEEE Internet of Things

IoT-A
Internet of Things - Architecture

# Internet of Things Technology(15CS81)
# Module 5- Chapter 7

By,
Manoj T
Department of CSE
SMVITM, Bantakal, Udupi

# Chapter 7 : IoT Physical Devices and Endpoints
## Arduino Uno

This chapter explores the following topics

- **Introduction to Arduino**
- **Arduino UNO**
- **Installing the Software**
- **Fundamentals of Arduino Programming**

# Introduction to Arduino

- Arduino is an open-source advancement prototyping platform which depends on simple to-utilize equipment and programming.

- 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 is a small computer that can be programmed to read information from the world around us and to send commands to the outside world.

- Arduino is a tiny computer that can be connected to electrical circuits.

- This makes it easy to read inputs- read data from outside and control outputs- send a command to the outside.

- The brain of this board(Arduino Uno) is an ATmega328 chip where the program can be stored and it will instruct Arduino what to do.

**Why Arduino?**

- Arduino is an open source product, software/hardware which is accessible and flexible customers.

- Arduino is flexible because of offering variety of digital and analog pins, SPI and PWM outputs.

- Arduino is easy to use, connected to a computer via a USB and communicates using serial protocol.
- Inexpensive, around 500 rupees per board with free authoring software.
- Arduino has growing strong online community where lots of source code is available for use, share and post examples for others to use too.
- Arduino is cross-platform, which can work on Windows, Mac or Linux platforms.
- Arduino follows simple, clear programming environment as C language.

## Family of Arduino Boards

- Some of the boards from Arduino family are:

**i. Arduino Mega**

- It is a big sister to the UNO with more memory and pins with a different chip the ATmega2560, which is useful when the project doesn't fits in an UNO.

**ii. Arduino Micro**

- It is bit smaller with a chip Atmega32u4 that can act like a keyboard or mouse which does its task with native USB.

- Its slim with downward pins which can be plugged into breadboard.

### iii. Arduino MKR1000

- It is like an Arduino Micro but has a more powerful 32-bit ATSAM ARM chip and built-in WiFi.

### iv. Flora

- Flora is an Arduino compatible from Adafruit which is a round wearable which can sewed into clothing.

# Exploring Arduino Uno Learning Board

- The figure 7.1 shows an labelled Arduino board

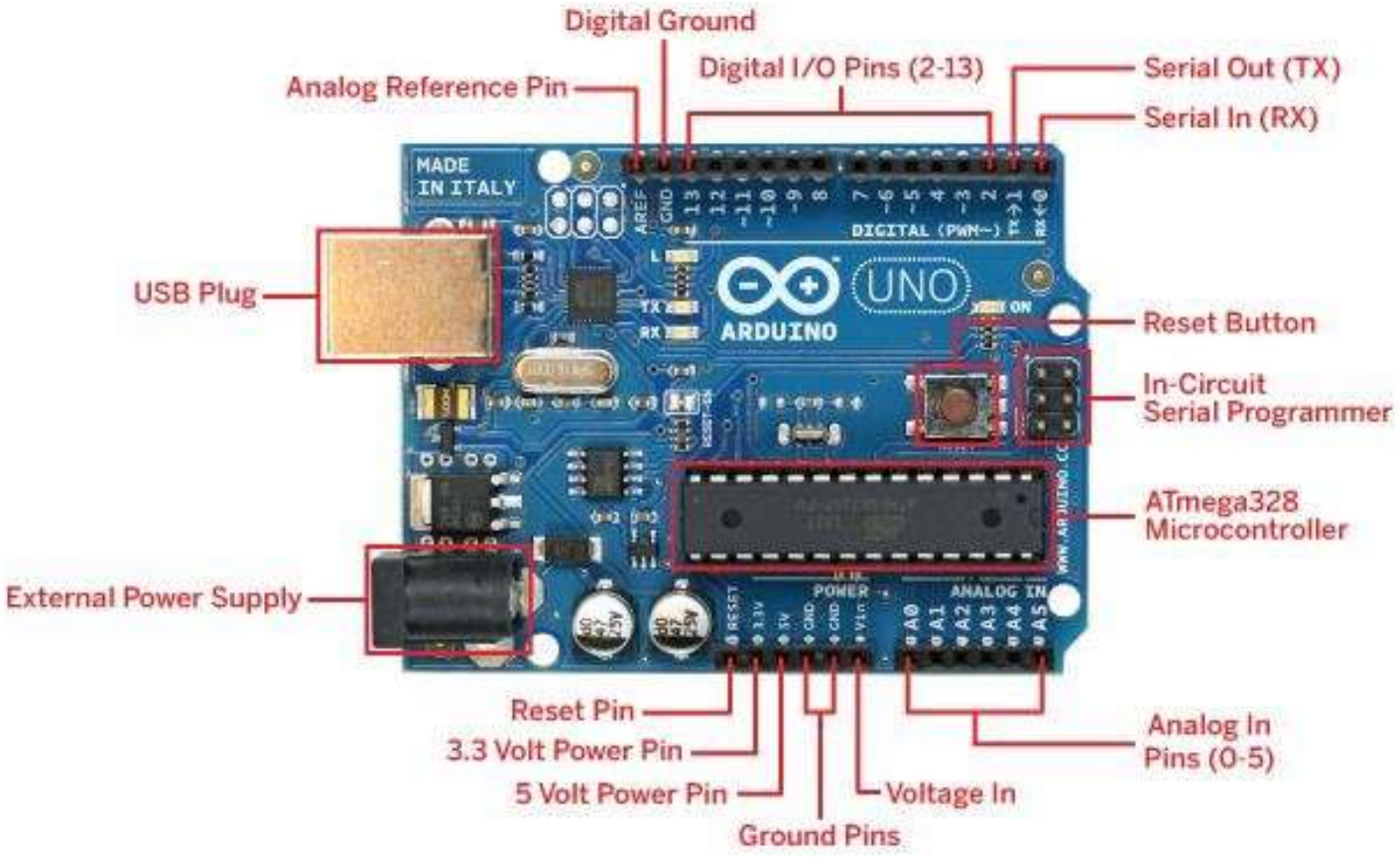


**Figure 7.1** : Arduino Uno Learning Board

- The following are components of a Arduino Uno board
  - **Mirocontroller** : The ATmega328p is the Arduino brain. Everything on the Arduino board is meant to support this microcontroller.

  - **Digital Pins**:
    - Arduino has 14 digital pins, labelled from 0 to 13 that can act as inputs or outputs.

    - When set as inputs, these pins can read voltage. They can read two different states **HIGH** or **LOW**.

    - When set as outputs, these pins can apply voltage. They can only apply **5V(HIGH)** or **0V(LOW)**.

- **PWM pins**
  - These are digital pins marked with a pins 11,10, 9, 6, 5 and 3). PWM stands for **"Pulse Width Modulation"** and allows to make digital pins output **"fake"** by varying amounts of voltage.

- **TX and RX pins**
  - Digital pins 0 and 1. The T stands for "transmit" and the R for "receive".

  - Arduino uses these pins to communicate with the computer.

- **LED attached to digital pin 13**
  - This is useful for an easy debugging of the Arduino sketches.

- **TX and RX pins**
  - These pins blink when there are information being sent between the computer and the Arduino.

- **Analog pins**
  - The analog pins are labelled from A0 to A5 and are most often used to read analog sensors.
  - They can read different amounts of voltage between 0 and 5V.
  - They can also be used as digital output/input pins.
- **Power Pins**
  - The Arduino has 3.3V or 5V supply, which is really useful since most components require 3.3V or 5V. The pins labelled as **"GND"** are the ground pins.

- **Reset button**
  - When this button is pressed, the program that is currently being run in the Arduino will start from the beginning.
  - The Reset pin next to the power pins also acts as reset button. When a small voltage is applied to that pin, it will reset the Arduino.
- **Power ON LED**
  - It will be **"on"** since power is applied to the Arduino.
- **USB jack**
  - The programs are uploaded from computer to Arduino board by connecting a male USB A to male USB B cable.

- **Power jack**
  - The Arduino board is powered using the power jack. There are several ways to power up the Arduino: rechargeable batteries, disposable batteries, wall-warts and solar panel.

# Installing the Software(Arduino IDE)

- The **Arduino IDE**(Integrated Development Environment) is the software used for the developing the programs which will tell Arduino what to do.

- The new programs are uploaded onto the main chip (ATmega328p) via USB using the Arduino IDE.

- To download the Arduino IDE, browse on the following link: https://www.arduino.cc/en/Main/Software

- The Arduino IDE can be downloaded based on the underlying operating system using which the programmer is working.

- The operating systems can be Windows, Mac and Linux.

- When the Arduino IDE is opened for the first time, a window will appear as shown in the figure 7.2

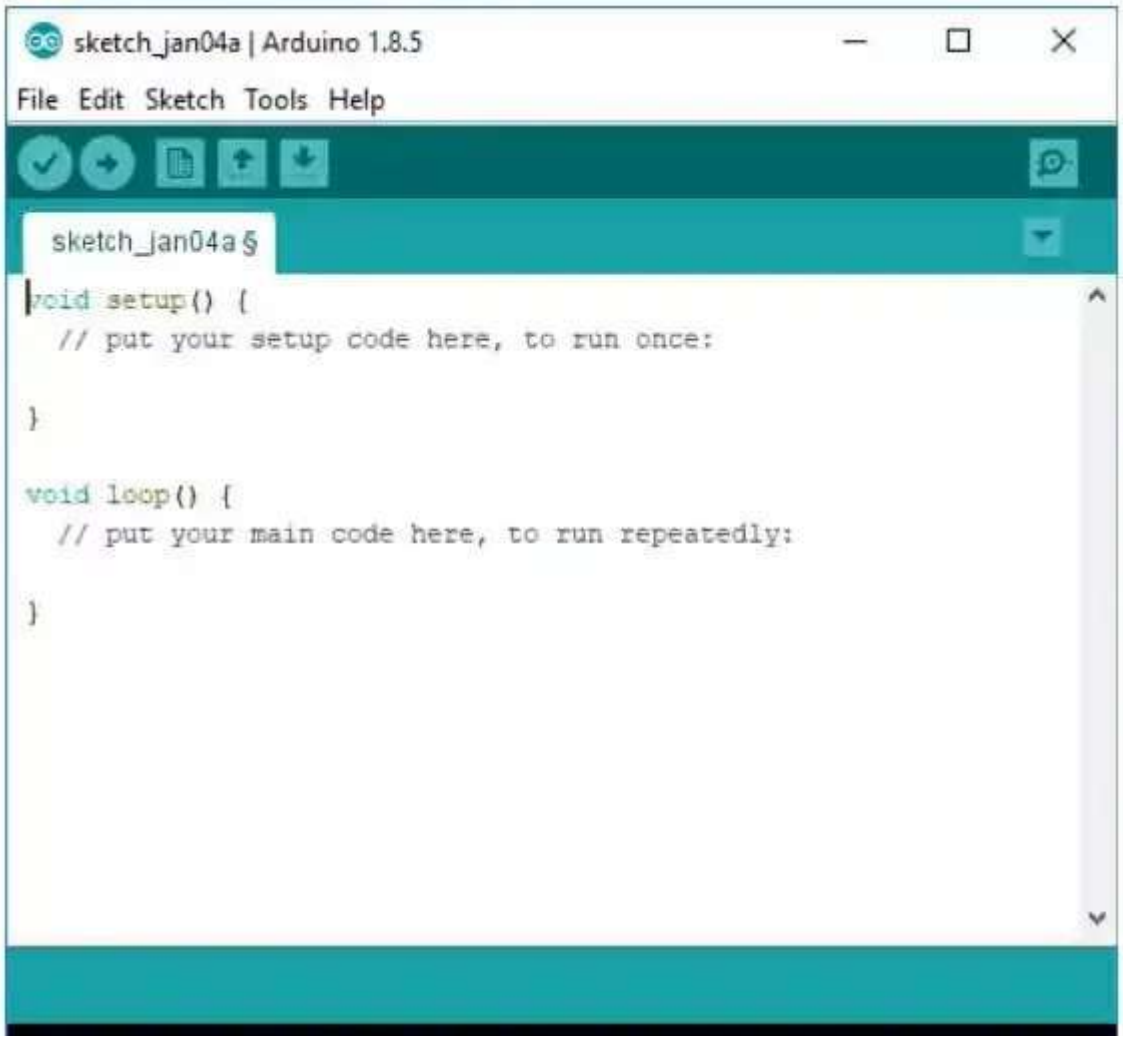


**Figure 7.2** : Arduino IDE

## Connecting Arduino Uno Learning Board

- After the Arduino board is connected to the computer using USB cable, one has to ensure that Arduino IDE has selected the right board being used.
- In this case Arduino Uno is being used, so go and click on Tools->Board: Arduino/Genuino Uno as shown in the figure 7.3.
- Then a right serial port needs to selected to which an Arduino board is connected.
- Go to Tools->Port and select the right port as shown in the figure 7.4.
- The figure 7.5 shows the toolbar options for Arduino IDE.

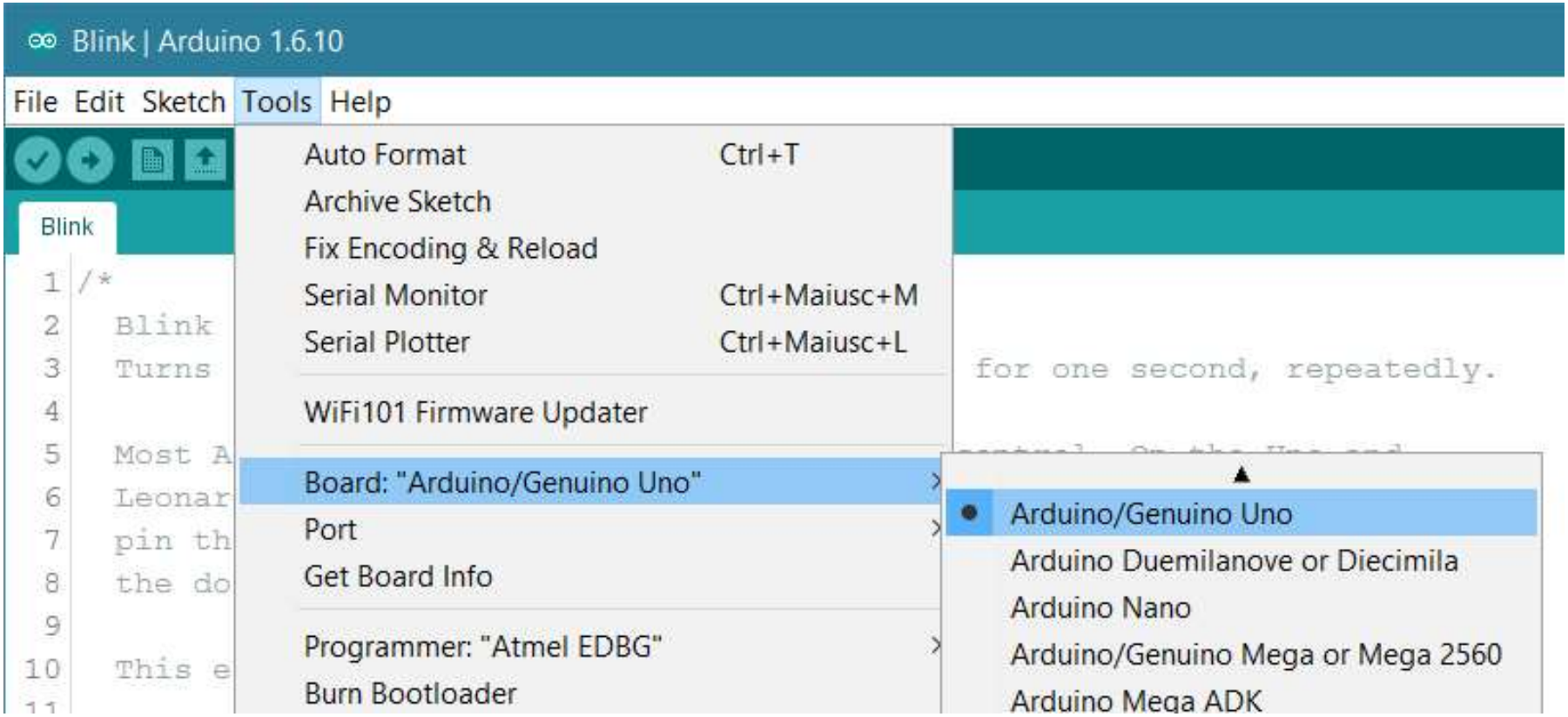


**Figure 7.3** : Selecting the Right Board

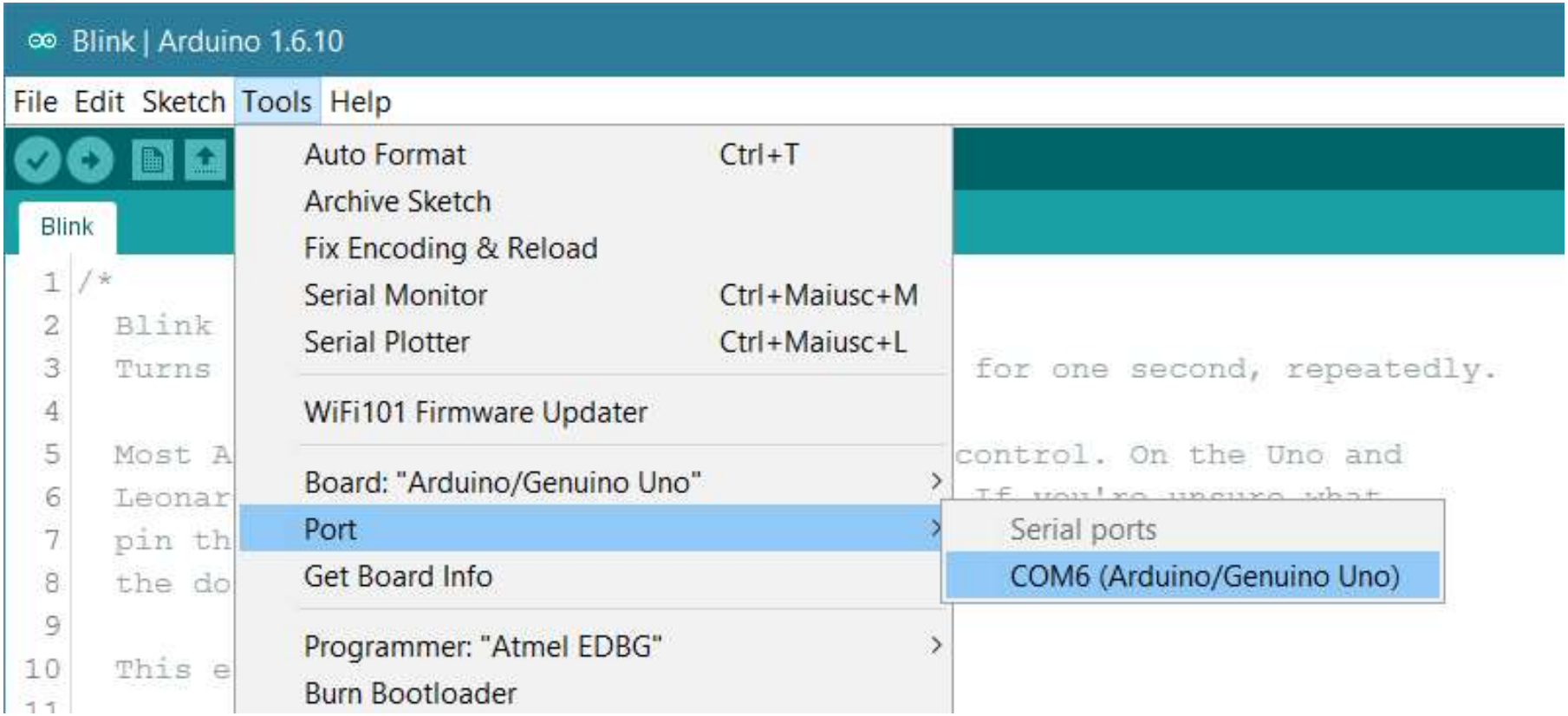


**Figure 7.4** : Selecting the Right Port

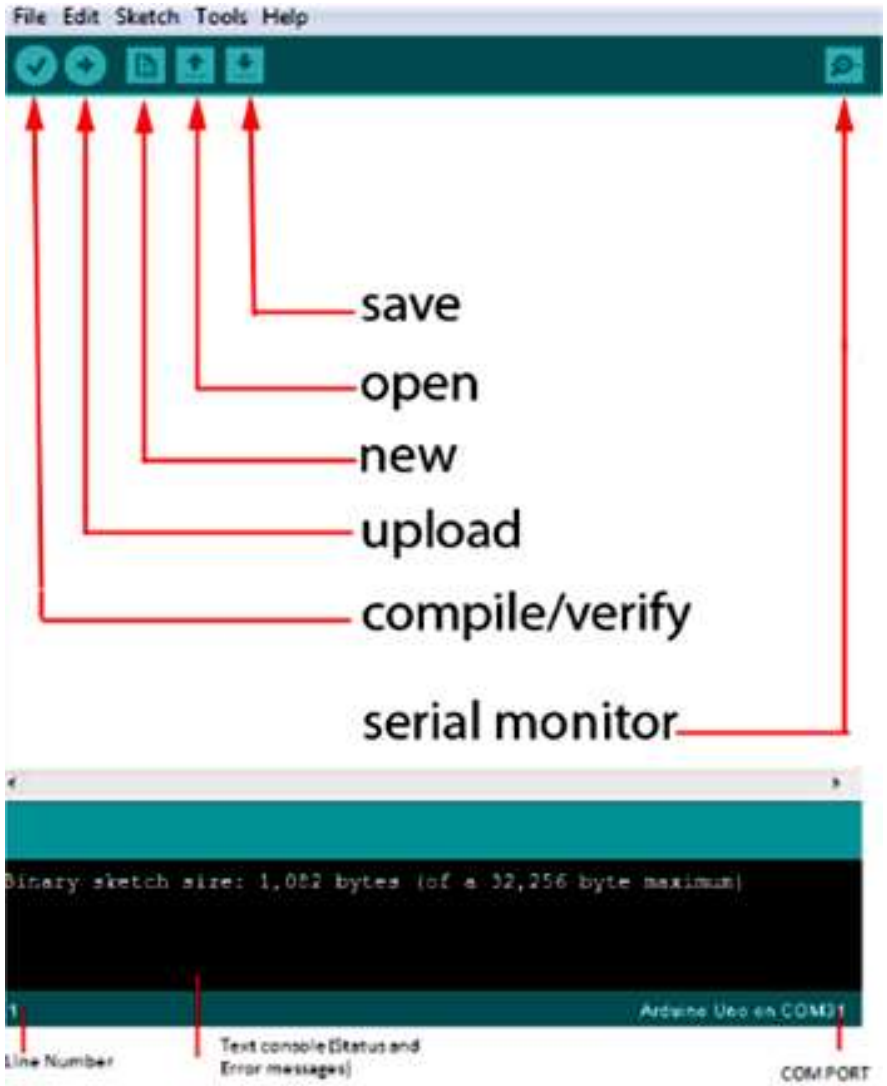


**Figure 7.5** : Toolbar Buttons in Arduino IDE

- The toolbar buttons and functions of each button are shown in Table 7.1 as follows

| | |
|---|---|
| **Verify/Compile** | Checks the code for errors |
| **Stop** | Stop the serial monitor or un-highlight other buttons |
| **New** | Creates a new blank sketch. Enter a name and location of the sketch. |
| **Open** | Shows a list of sketches in the sketchbook |
| **Upload** | Uploads the current sketch to the Arduino. |
| **Serial Monitor** | Display serial data being sent from the Arduino |

**Table 7.1** : Toolbar options in Arduino IDE

# Fundamentals of Arduino Programming

| | |
|---|---|
| **Structure** | Software structure consist of two main functions −<br>i. Setup( ) function<br>ii. Loop( ) function<br><br>void setup() // Preparation function used to declare variables<br>{<br>// First function runs only once in the program<br>Statements(s); // used to set pins for serial communication<br>}<br><br>void loop() // Execution block where instructions are executed repeatedly<br>{<br>Statements(s); // Functionalities involve reading inputs, triggering outputs etc.<br>} |

| | |
|---|---|
| **void setup()** | void setup()<br>{<br>pinMode(pin,INPUT); // 'pin' configured as input<br>} |
| **void loop()** | void loop() // After calling setup(), loop() function does its task<br>{<br>digitalWrite(pin,HIGH) // sets pin ON<br>delay(10000); // pauses for ten thousand milliseconds<br><br>digitalWrite(pin,LOW) // sets pin OFF<br>delay(10000); // pauses for ten thousand milliseconds<br>} |
| **Functions** | A function is a piece of code that has a name and set of statements executed when function is called. Functions are declared by its type followed with name of a function.<br>**Syntax** : type functionName(parameters)<br>{<br>Statement(s);<br>} |

| | |
|---|---|
| **Functions** | **Example**:<br>int delayvar()<br>{<br>int var; // create temporary variable<br>var = analogRead(potent) ; // read from potentiometer<br>var = var/4; // convert the value of variable var<br>return var;<br>} |
| **{} curly braces** | They define beginning and end of function blocks, unbalanced braces may lead to compilation errors. |
| **semicolon** | It is used to end a statement and separate elements of a program<br>**Syntax** : int x = 14; |
| **/* … */ block comments** | Multiline comments begin with /* with a description of the block and ends with */.<br>**Syntax** :<br>/* This is an enclosed block of comments<br>Use the closing comment to avoid errors */ |

| | |
|---|---|
| **// line comments** | Single line comment begins with // and ends with next instruction followed.<br>**Syntax**:<br>// This is a single line comment |
| **Variables** | A variable is a way of storing value for later use in the program. A variable is defined by its value type as an **int**, **long**, **float** etc by setting a specified name and optionally assigning an initial value.<br><br>A global variable can be seen in every part of the program which is declared at the beginning of program before setup() function.<br><br>A local variable is defined inside a function in which it is declared.<br><br>**Example**:<br>int var;<br>void setup()<br>{<br>// nothing is required<br>} |

**Variables**

**Example contd..**

```
void loop()
{
  for(int local=0;local<=5;)
  {
    local++;
  }
  float local_f ;  // variable local_f is only visible inside the loop
}
```

**Data Types**

| Data Type | Syntax | Range |
|---|---|---|
| Byte | byte x = 100; | 0-255 |
| Int | int y = 10; | 32767 to -32768 |
| Long | long var = 8000; | 2147483647 to -2147483648 |
| Float | float x = 3.14 | 3.4028235E+38 to -3.4028235E+38 |
| Arrays | int myarray [] = {10,20,30,40} | Size depends upon the data type associated with declaration |

| | |
|---|---|
| **Operators** | Arithmetic operators(+,-,*,/)<br>Assignment operators(=,++,--,+=,*=,/=)<br>Comparison operators(==,!=,<,>,<=,>=)<br>Logical operators(&&,\|\|,!) |
| **Constants** | TRUE/FALSE<br>INPUT/OUTPUT<br>HIGH/LOW |
| **Flow control Statements** | |
| **if** | if(some_variable == value)<br>{<br>Statement(s); // Evaluated only if comparison results in a true value<br>} |
| **if….else** | if(input == HIGH)<br>{<br>Statement(s); // Evaluated only if comparison results in a true value<br>}<br>else<br>{<br>Statement(s); // Evaluated only if comparison results in a false value<br>} |

| | |
|---|---|
| **for** | for(initialization;condition;expression)<br>{<br>Dosomething; // Evaluated till condition becomes false<br>} |
| **while** | While loop executes until expression inside parenthesis becomes false<br><br>while(some_variable ?? value)<br>{<br>Statement(s); // Evaluated till comparison results in a false value<br>} |
| **do..while** | Bottom evaluated loop, works same way as while loop condition is tested at the end of loop.<br>do<br>{<br>Dosomething;<br>}while(somevalue); |

| Digital and Analog input output pins and their usage | | |
|---|---|---|
| **Digital i/o** | **Methods** | **Usage** |
| | pinMode(pin, mode) | Used in setup() method to configure pin to behave as INPUT/OUTPUT<br><br>pinMode(pin,INPUT) // pin set to INPUT<br>pinMode(pin,OUTPUT) // pin set to OUTPUT |
| | digitalRead(pin) | Read value from a specified pin with result being HIGH/LOW<br>val = digitalRead(pin) // val will be equal to input pin |
| | digitalWrite(pin) | Outputs to HIGH/LOW on a specified pin.<br>digitalWrite(pin,HIGH) ; // pin is set to HIGH |
| | Example | int x = 13; // connect ‘x’ to pin 13<br>int p = 7; // connect push button to pin 7<br>int val = 0; // variable to store the read value |

| | | |
|---|---|---|
| | Example contd… | void setup()<br>{<br>pinMODE(x,OUTPUT) // sets ‘x’ as OUTPUT<br>}<br>void loop()<br>{<br>val = digitalRead(p); // sets ‘value’ to 0<br>digitalWrite(x,val); // sets ‘x’ to button value<br>} |
| **Analog i/o** | **Methods** | **Usage** |
| | analogRead(pin) | Reads value from a specified analog pin works on pins 0-5<br>val = analogRead(pin); // ‘val’ equal to pin |
| | analogWrite(pin,value) | Writes an analog value using pulse width modulation(PWM) to a pin marked PWM works on pins 3,5,6,9,10. |
| | Example | int x = 10; // connect ‘x’ to pin 13<br>int p = 0; // connect potentiometer to analog pin 7<br>int val; // variable for reading |

| | | |
|---|---|---|
| | Example contd… | void setup()<br>{ } // No setup is required<br>void loop()<br>{<br>val = analogRead(p); // sets “value” to 0<br>val = val/4;<br>analogWrite(x,val); //outputs PWM signal to ‘x’<br>} |
| **time** | **Methods** | **Usage** |
| | delay(ms) | Pauses for amount of time specified in milliseconds<br>delay(1000); // waits for one second |
| | millis() | Returns the number of milliseconds since Arduino is running<br>val = millis(); // ‘val’ is equal to millis() |
| **math** | **Methods** | **Usage** |
| | min(x,y) | Calculates minimum of two numbers<br>val = min(val,10); // sets ‘val’ smaller than 10 or equal to but never gets above 10 |

| **math** | **Methods** | **Usage** |
|---|---|---|
| | max(x,y) | val = max(val,10) ; // sets ‘val’ to larger than 10 or 10 |
| **random** | **Methods** | **Usage** |
| | randomSeed(value) | Sets a value/seed as starting point for randomization |
| | random(min,max) | Allows to return numbers within range specified by min and max values.<br>val = random(100,200); // sets ‘val’ to random number between 100-200 |
| | Example | int number; // variable to store random value<br>int x = 10;<br>void setup()<br>{<br>randomseed(millis()) ; // set millis() as seed<br>number = random(200); // random number from 0-200<br>analogWrite(x,number); // outputs PWM signal<br>delay(500);<br>} |

| Serial | Methods | Usage |
|---|---|---|
| | Serial.begin(rate) | Opens serial port and sets the baud rate for serial data transmission<br><br>void setup()<br>{<br>Serial.begin(9600); // Sets default rate to 9600 bps<br>} |
| | Serial.println(data) | Prints data to the serial port<br>Serial.println(value); // sends the 'value' to serial monitor |

IEEE Internet of Things

IoT-A
Internet of Things - Architecture

# Internet of Things Technology(15CS81)
# Module 5- Chapter 8

By,
Manoj T
Department of CSE
SMVITM, Bantakal, Udupi

# Chapter 8 : IoT Physical Devices and Endpoints
# Raspberry Pi

This chapter explores the following topics

- **Introduction to Raspberry Pi**
- **About the Raspberry Pi Board: Hardware Layout**
- **Operating Systems on Raspberry Pi**
- **Configuring Raspberry Pi**
- **Programming Raspberry Pi with Python**
- **Wireless Temperature Monitoring System Using Pi**

# Introduction to Raspberry Pi

- The Raspberry Pi is a series of credit card sized single-board computers developed in the United Kingdom by the Raspberry Pi foundation to promote the teaching of basic computer science in schools and developing countries.
- The original model became extremely popular than it was anticipated, outside of the target market.
- The Raspberry Pi was launched in the year 2012, and there have been several iterations and variations released since then.
- The original Pi had a single-core 700 MHz CPU and just 256MB RAM and the latest model has a quad-core 1.4 GHz CPU with 1GB RAM.

- The main price point for Raspberry Pi has always been $35 or less, including Pi Zero which costs just $5.

- All models of Rapsberry Pi feature a Broadcom system on a chip(SoC), which includes an ARM compatible central processing unit(CPU) and an on chip graphics processing unit(GPU, a Video Core IV).

- Secure Digital(SD) cards are used to store the operating system and program memory in either in SDHC or MicroSDHC sizes.

- Most boards have between one and four USB slots, HDMI and composite video output and a 3.5 mm phone jack for audio.

- The foundation provides Raspbian, a Debian-based Linux distribution for download as well as third party Ubuntu, Windows 10 IoT core, RISC OS and specialized media center distributions.

## Why Raspberry Pi?

- Inexpensive, cross-platform, simple, clear programming environment, Open source and extensible software and Open source and extensible hardware.

# Exploring The Raspberry Pi Learning Board

- The figure 8.1 shows a labelled Raspberry Pi board

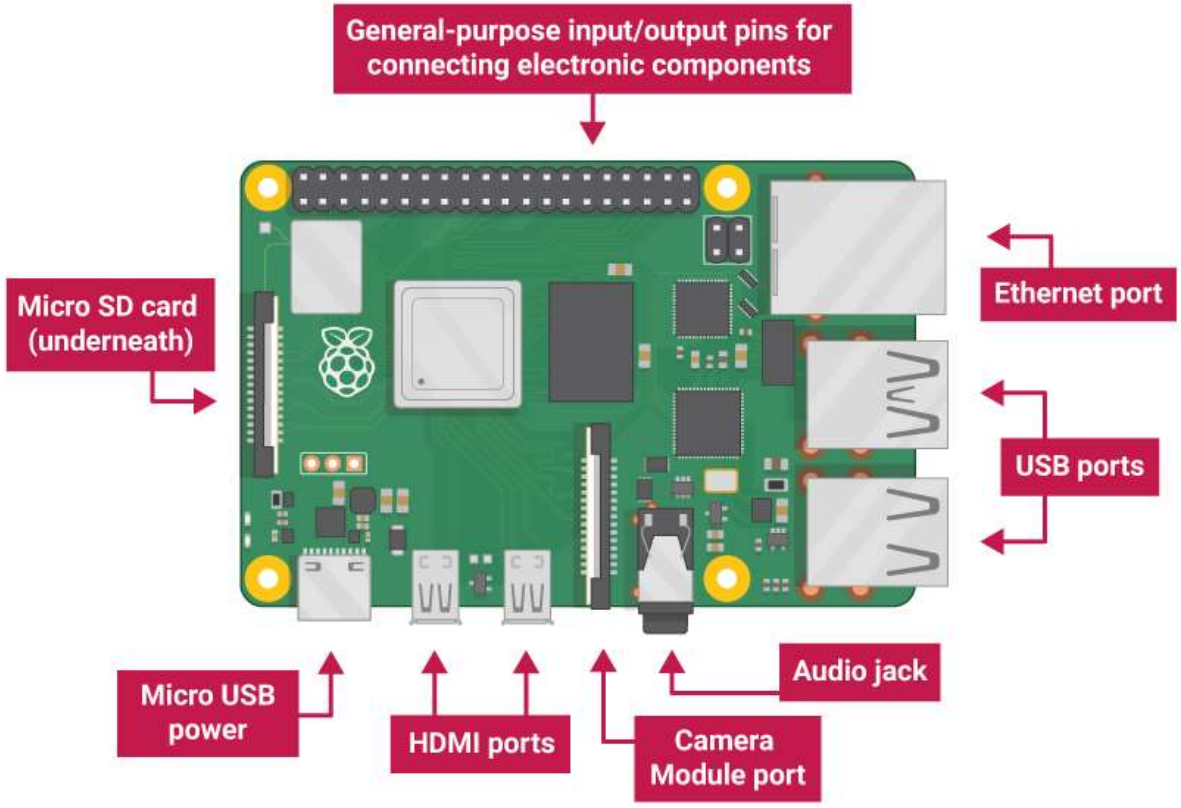


**Figure 8.1** : Raspberry Pi Learning Board

- Let us focus on each of the part in the learning board

➢ **Processor**

- The Broadcom BCM2835 SoC used in the first generation Raspberry Pi is somewhat similar to the chip used in first generation smart phones, which includes a 700 MHz ARM processor, Video Core IV graphics processing unit(GPU) and RAM.

- This has a level 1(L1) cache of 16KB and a level 2(L2) cache of 128 KB.

- The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

- The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM cortex A7 processor with 256 KB shared L2 cache.

- The Raspberry Pi 3 uses a Broadcom BCM2837 SoC 2 with a 1.2 GHz 64-bit quad core ARM cortex A53 processor, with a 512 KB shared L2 cache.

➢ **Power Source**

- The recommended and easiest way to power the Raspberry Pi is via the MicroUSB port on the side of the unit.

- The recommended input voltage is 5V, and the recommended input current is 2A.

- The Raspberry Pi can function on lower current power supplies Ex : 5V @ 1A.

## ➢ SD Card

- The Raspberry Pi does not have any locally available storage accessible.

- The working framework is stacked on a SD card which is embedded on the SD card space on the Raspberry Pi.

## ➢ GPIO(General Purpose Input Output)

- General-purpose input/output(GPIO) is a non-specific pins on a coordinated circuit to know is an input or output pin which can be controlled by the client at run time.

- GPIO capabilities may include:
  - ❑ GPIO pins can be designed to be input or output
  - ❑ GPIO pins can be empowered/crippled

- ❑ Input values are meaningful(normally high=1, low=0)
- ❑ Yield values are writable/meaningful
- ❑ Input values can be frequently be utilized as IRQs(regularly for wakeup connections)

➢ **DSI Display X**

- The Raspberry Pi connector S2 is a display serial interface(DSI) for connecting a liquid crystal display(LCD) panel using a 15-pin ribbon cable.

- The mobile industry processor interface(MIPI) inside the Broadcom BCM2835 IC feeds graphics data directly to the display panel through this connrector.

➢ **Audio Jack**

- A standard 3.5 mm TRS connector is accessible on the Rpi for stereo sound yield.

- Any earphone or 3.5 mm sound link can be associated straightforwardly.

## ➢ Status LEDs

- There are 5 status LEDs on the Rpi that demonstrate the status of different exercises as takes after:
  - ❑ OK – SD card access(by means of GPIO16)
  - ❑ POWER-3.3 V Power-named as “PWR” on all boards
  - ❑ FDX – Full Duplex(LAN) - marked as “FDX” on all boards
  - ❑ LNK – Link/Activity(LAN) – marked as “LNK” on all boards
  - ❑ 10M/100 – 10/100Mbit(LAN) – named(errorneously) as “10M” on Model B Rev 1.0 boards and “100” on Model B Rev 2.0 and Model A boards USB Ports.

- **Ethernet Port**
  - Ethernet port is accessible on Model B and B+.
  - It can be associated with a system or web utilizing a standard LAN link on the Ethernet port.

- **CSI Connector(CSI)**
  - Camera Serial Interface is a serial interface outlined by MIPI(Mobile Industry Processor Interface) organization which can be used for interfacing the computerized cameras with a portable processor.

- **JTAG headers**
  - JTAG is a standardised interface for debugging integrated circuits which can be used to debug the Raspberry Pi.

- **HDMI**
  - High Definition Multimedia Interface to give both video and sound yield.

## Description of System on Chip(SoC)

- A System on a chip(SoC) is an integrated circuit(IC) that coordinates all parts of a PC or other electronic framework into a solitary chip.

- SoCs are exceptionally regular in the portable gadgets that advertise in view of their low power utilization. A run of the mill application is in the range of implemented frameworks.

- An SoC comprises of:
  - A microcontroller, chip or DSP core(s). Some SoCs- called multiprocessor framework on chip(MPSoC)- incorporate more than one processor center.

  - Memory pieces including a choice of RAM, ROM, EEPROM and streak memory.

- Timing sources including oscillators and stage bolted circles.
- Peripherals including counter-clocks, ongoing clocks and power-on reset generators
- Outer interfaces, including industry norms For ex : USB, FireWire, Ethernet, USART, SPI.
- Simple interfaces including ADCs and DACs
- Voltage controllers and power administration circuits.

- **Accessories**

  Numerous embellishments and peripherals for the Raspberry Pi go from USB center points, engine controllers to temperature sensors. Some of the official embellishments for the Rpi are:

## ➢ Camera

- The Raspberry Pi camera board contains a 5 Mpixel sensor and interfaces through a strip link to the CSI connector on the Raspberry Pi.

- In Raspbian, support can be empowered by introducing or moving up to the most recent version of the OS and after that running Raspi-config.

- A bus is either exclusive or industry standard For ex : the AMBA bus from ARM holdings- interfaces these squares.

## ➢ Gertboard

- A Raspberry Pi foundation authorized gadget intended for instructive purposes and grows the Raspberry Pi's GPIO pins to permit interface with and control of LEDs, switches, simple signs, sensors and different gadgets.

## ➢ USB Hub

- In spite of the fact that not an official embellishment, it is an exceptionally suggested extra for Pi.

- A fueled USB Hub with 7 additional ports is accessible at all online stores.

- It is mandatory to utilize a USB hub to associate outer hard plates or different adornments that draw control from the USB ports, as the Pi can't offer energy to them.

## Raspberry Pi Interfaces

- Raspberry Pi has serial, SPI and I2C interfaces
  - **Serial**
    - The serial interface on Raspberry Pi has receive(rx) and transmit(Tx) pins for communication with serial peripherals.

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

    - In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:

- MISO(Master In Slave Out): Master line for sending data to the peripherals
- MOSI(Master Out Slave In): Slave line for sending data to the master.
- SCK(Serial Clock) : Clock generated by master to synchronize data transmission.
- CE0(Chip Enable 0): To enable or disable devices.
- CE1(Chip Enable 1): To enable or disable devices.

## I2C

- The I2C interface pins on Raspberry Pi allows to connect hardware modules.

- I2C interface allows synchronous data transfer with just two pins-SDA(data line) and SCL(clock line).

# Operating Systems on Raspberry Pi

- Various operating systems can be installed on Raspberry Pi through SD cards.
- Most use a MicroSD slot located on the bottom of the board. The Raspberry Pi primarily uses Raspbian, a Debian-based Linux operating system.
- Other third party operating systems available via the official website include Ubuntu MATE, Snappy Ubuntu Core, Windows 10 IoT Core, RISC OS and other specialized distributions.

## Operating Systems( not Linux based)

- RISC OS Pi
- Free BSD
- NetBSD
- Plan 9 from Bell Labs and Inferno
- Windows 10 IoT core – a no cost edition of Windows 10 offered by Microsoft that runs natively on the Raspberry Pi 2.
- Xv6- is a modern reimplementation of sixth edition Unix OS for teaching purposes, it is ported to RaspberryPi from MIT x86, which can boot NOOBs.

- Haiku- is an open source BeOS clone that has an Be compile for the Raspberry Pi and several other ARM boards.

**Operating Systems(Linux based)**

- Xbian- using Kodi open source digital media content
- openSUSE
- Raspberrry Pi Fedora remix
- Pidora, another Fedora Remix optimized for the Raspberry Pi
- Gentoo Linux
- Diet Pi

- CentOS/Open Wrt
- Kali Linux
- Ark OS
- Kano OS
- Nard SDK

**Media center operating systems**

- OSMC, OpenElec, LibreElec, Xbian, Rasplex

**Audio operating systems**

- Volumio, Pimusicbox, Runeaudio, moOdeaudio

## Configuring Operating System on Raspberry Pi

- To setup an operating we need a SD card with minimum capacity of 8 GB.

## Formatting SD card

- Format the SD card before copying NOOBS onto it. To do this
  i. Download SD formatter 4.0 from SD Association website for either Windows or Mac.

  ii. Follow the instructions to install the software.

  iii. Insert the SD card into the computer or laptops SD reader and make a note of the drive letter allocated to it.

iv. In SD formatter, select the drive letter of the SD card and format it.

**OS Installation**

- Follow the steps to install operating system in the SD card.

  i. Go to Raspberry Pi foundation website and click on DOWNLOAD section.

  ii. Click on NOOBS, then click on the “Download ZIP” button under NOOBS(offline and network install) and select a folder to save this ZIP file.

  iii. Extract all the files from ZIP.

iv. Once SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.

v. The necessary files will then then be transferred to the SD card.

vi. When this process has finished, safely remove the SD card and insert into the Raspberry Pi.

**First Boot**

i. Plug in the keyboard, mouse and monitor cables.

ii. Now plug the USB cable into the Raspberry Pi.

iii. Now Raspberry Pi will boot and a window will appear

with a list of different operating systems that can be installed. Since Raspbian is preferred here, tick the box next to Raspbian and click install.

iv. Raspbian will then run through its installation process. This process may take few minutes to finish.

v. When the install process has completed, the Raspberry Pi configuration menu(raspi-config) will load.

# Raspberry Pi Commands

| General commands for Raspberry Pi | |
|---|---|
| raspi-config | Configuration settings menu |
| clear | Clears data from terminal |
| date | Current date |
| reboot | Reboot immediately |
| shutdown –h now | Shut down immediately |
| nano example.txt | Opens the example.txt in the text editor nano |
| poweroff | To shut down immediately |
| Shutdown –h 01:22 | To shut down at 01:22 AM. |
| apt-get update | Synchronizes the list of packages on the system to the list of repositories. |
| apt-get upgrade | Upgrades all of the software packages that are installed |
| Startx | Opens the GUI |

| **Directory and File commands** | |
|---|---|
| mkdir new_directory | Creates a new directory named new_directory |
| mv new_folder | Moves the file or directory named “new_folder” to a specified location. |
| rm new_file.txt | Deletes the file new_file.txt. |
| rmdir new_directory | Deletes the directory “new_directory” only if it is empty. |
| touch new_file.txt | Creates a new, empty file named new_file.txt in the current directory. |
| cat new_file.txt | Displays the contents of the file new_file.txt |
| cd /xyz/abc | Changes the current directory to the /xyz/abc directory |
| ls –l | Lists files in the directory |
| **Networking and Internet Commands** | |
| iwconfig | To check which wireless adapter is currently active |
| ifconfig | Wireless connection status |
| ping | Tests connectivity between two devices connected on a network |

| **Networking and Internet Commands** | |
|---|---|
| wget http://www.website.com /new_file.txt | Downloads the file new_file.txt from the web and saves it to the current directory. |
| nmap | Scans the network and lists connected devices, protocol, port number and other information. |
| iwlist wlan0 scan | List of currently available wireless networks |
| **System Information Commands** | |
| cat /proc/meminfo | Shows details about memory |
| cat /proc/version | Shows which version of Raspberry Pi we are using |
| df –h | Shows information about available disk space |
| df / | Shows how much free disk space is available |
| free | Shows how much free memory is available |
| hostname –l | Shows the IP address of Raspberry Pi |
| lsusb | Lists the USB hardware connected to Raspberry Pi |
| vegencmd measure_temp | Shows the temperature of the CPU |

# Programming Raspberry Pi with Python

- Here the focus will be on how the python programs are developed on Raspberry Pi.
- Raspberry Pi runs Linux and supports python out of the box. Hence any python program can be run that normally runs on a computer.
- The general purpose input/output capability provided by the GPIO pins on Raspberry Pi makes it useful device for IoT.
- Raspberry Pi can be interfaced with variety of sensors, actuators using GPIO pins and also SPI, I2C and serial interfaces.

- Input from the Raspberry Pi can be processed and actions can be taken for instance, sending data to server, sending an email, triggering a relay switch.

## Simple Python Programs on Raspberry Pi

| **Program** | **Code** |
|---|---|
| Print hello world | print(“Hello world”) |
| Program to add two numbers | a = 1.2<br>b = 5.39<br>sum = float(a) + float(b)<br>print(“The sum of {0} and {1} is {2}”.format(a,b,sum)) |
| Program to roll a dice | import random<br>min = 1<br>max = 6<br>roll_again = “yes”<br>while roll_again==“yes” or roll_again==“y”<br>  print(“Rolling the dices”)<br>  print(“The values are:”) |

| Program | Code |
|---|---|
| Program to roll a dice | print(random.randomint(min,max))<br>print(random.randint(min,max)) |
| Program to find the ip address of raspberry pi | import urllib<br>import re<br>print("We will try to open this url, in order to get ip address")<br>url = http://checkip.dyndns.org<br>print(url) |
| Program to generate password | import string<br>from random import *<br>characters = string.ascii_letters + string.punctuation + string.digits<br>password = "".join(choice(characters) for x in range(randint(8,16)))<br>print(password) |
| Program to print fibonacci series | a,b =0,1<br>while b<200<br>print(b)<br>a,b = b,a+b |

| Program | Code |
|---|---|
| Program to check for armstrong number | num = int(input("Enter a number"))<br>initial_sum = 0<br>temp = num<br>while num > 0<br>digit = temp % 10<br>  initial_sum += digit**3<br>temp/= 10<br>if num == initial_sum<br>print("Num is an armstrong number")<br>else<br>print("Num is not an armstrong number") |
| Program to display calendar of given month of the year | import calendar<br>yy = 2017<br>mm = 11<br>print(calendar.month(yy,mm)) |

# Wireless Temperature Monitoring System Using Pi

- Raspberry Pi has an inbuilt Wi-Fi module, which makes it suitable for IoT applications.
- Temperature monitoring system works by measuring the temperature in real-time by using LM35 as a temperature sensor.
- The temperature values are pushed into the backend cloud server. The most preferred backend cloud server is the ThingsSpeak.
- ThingsSpeak cloud is suitable to view the sensor logs in the form of graph plots and in this monitoring system one field to monitor the temperature value is created.
- This can be reconfigured to monitor a number of sensor values in various fields.

**Hardware Required**

- Raspberry Pi
- SD card
- Power supply
- VGA to HDMI converter (Optional)
- MCP3008 (ADC IC)
- A temperature sensor(LM35)

**Software Required**

- RaspbianStretch OS
- SD card Formatter
- Win32DiskImager (or) Etcher

**Python Libraries Used**

- RPi.GPIOas GPIO (To access the GPIO Pins of Raspberry Pi)
- Urllib2 to handle URL using Python programming
- Time library (For Time delay)

## DS18B20 Temperature Sensor

- The DS18B20 is a 1-wire programmable temperature sensor from maxim integrated.
- It is widely used to measure temperature in hard environments like in chemical solutions, mines or soil etc.
- The constriction of the sensor is rugged and also can be purchased with a waterproof option making the mounting process easy.
- It can measure a wide range of temperature from-55°C to +125°with a decent accuracy of±5°C.

.

- Each sensor has a unique address and requires only one pin of the MCU to transfer data so it a very good choice for measuring temperature at multiple points without compromising much of your digital pins on the microcontroller.

**Applications**

- Measuring temperature in hard environments
- Liquid temperature measurement
- Applications where temperature has to be measured at multiple points

# Pin Configuration

| No | Pin Name | Description |
|---|---|---|
| 1 | Ground | Connect to the ground of the circuit |
| 2 | Vcc | Voltage supplied to the sensor ,canbe3.3V or 5V |
| 3 | Data | This pin gives output temperature value which can be read using 1-wire method |

**Figure 8.2** : DS18B20 Temperature Sensor

# Connecting Raspberry Pi via SSH

- The SSH(Secure Shell) can be used to access the command line of a Raspberry Pi remotely from another computer or device on the same network.

- The Raspberry Pi will act as a remote device – one can connect to it using a client on another machine.

- Following are the steps to do that:
  - **i. Set up the local network and wireless connectivity**
    - ➢ Make sure that Raspberry Pi is properly set up and connected.

    - ➢ If wireless networking is used, it can be enabled via the desktop's user interface, or using the command line.

- If wireless connectivity is not used then plug your Raspberry Pi directly into the router.

- Note down the IP address of the Pi in order to connect to it later using the **ifconfig** command.

**ii. Enable SSH**

- By default SSH server is disabled in Raspberry Pi. It can be enabled manually from the desktop:
  - Launch **Raspberry Pi Configuration** from the **Preferences** menu
  - Navigate to the **Interfaces** tab
  - Select **Enabled** next to **SSH** and click **OK**.

**iii. Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)**

- For headless setup, SSH can be enabled by placing a file named **ssh**, without any extension, onto the boot partition of the SD card from another computer.

- When the Pi boots, it looks for the ssh file. If it is found, SSH is enabled and the file is deleted.

- The content of the file does not matter; it could contain text, or nothing at all.

**iv. Set up the client**

- SSH is built into Linux distributions and Mac OS, and is an optional feature in Windows 10. For older Windows versions and mobile devices, third-party SSH clients are available.

# Accessing Temperature from DS18B20 sensors

- The DS18B20 is a digital thermometer that allows to get 9-bit to 12-bit Celsius temperature measurements (programmable resolution).

- The temperature conversion time depends on the resolution used. For a 9-bit resolution it takes at most 93.75 ms and for a 12-bit resolution it takes at most 750 ms.

- The device is able to measure temperatures from -55°C to +125°C and has a ±0.5°C accuracy in the range from -10°C to +85°C.

- Additionally, it has an alarm functionality with programmable upper and lower temperature trigger points.

- These thresholds are stored internally in non-volatile memory, which means they are kept even if the device is powered off .
- The sensor communicates using the **OneWire** protocol, which means it only requires a pin from a microcontroller to be connected to it.
- Furthermore, each sensor has a unique 64-bit serial code, allowing multiple DS18B20 devices to function on the same **OneWire** bus.
- In terms of power supply, the device can operate with a voltage between 3.0 V and 5.5 V, which means it can operate with the same voltage of the ESP32 without the need for level conversion.