

IoT with python and Raspberry Pi

PyDelhi 2016

Dr. Sandeep Nagar

(Assistant Professor, Department of Electrical and Electronics Engineering,
School of Engineering, G D Goenka University, Sohna, Haryana, INDIA)

Instructor



Dr. Sandeep Nagar

M.Sc. Physics (MSU, Vadodara) & PhD in Material Science
(Department of Material Science and Engineering, KTH, Sweden)

contact e-mail: sandeep.nagar@gmail.com

Why would I do IoT?

- Its for everybody!
- Started just for fun
- Some serious experimentation
- Making scientific instruments
- Internet control gives multi-functionality to experiments

Outline of workshop

- ① Introduction to IoT and Raspberry Pi (30 Minutes)
 - Intro to IoT
 - Various parts
 - Installing OS
- ② Running python (30 minutes)
 - Writing simple programs
 - Installing packages
 - Running Numpy and Scipy
- ③ IoT with RPi (30 minutes)
 - Running RPi headless
 - Blinking LED
 - Some serious examples

Introduction to IoT

- IoT is a connecting **things** to internet
 - Example: Mobile phone
 - It was initially used only for two-way vice communication
 - Now it connects to internet too!
 - Its a **thing**, which is connected to **internet** \Rightarrow IoT
- Two types:
 - Device computes locally and interacts on internet
 - This is simpler task which we will learn in present workshop
 - Device does not compute locally but interacts on internet
 - This is complex task outside the scope of present workshop

- Open Source hardwares
- Two examples:
 - Arduino
 - RPi
- We shall use RPi in this workshop
 - It has lot more features!
- Lets see how RPi is different than Arduino

Arduino vs RPi

<i>Feature</i>	Ardunio	RPi
Processor	16MHz	900 MHz
Resolution	8 bit	32 bit
Memory	32 K Flash, 2K SRAM	4GB flash, 512K/1GB SRAM
Voltage level	5V	3.3 V
Interface	No OS	Has OS

OS

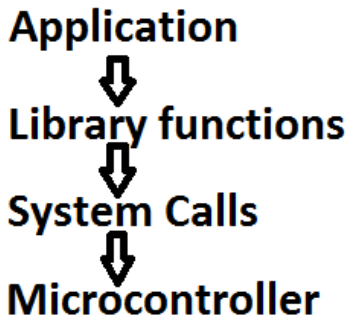


Figure: Working of OS

Arduino vs RPi

- Arduino has only an Application and a Micro-controller
 - Application directly controls the pins
- RPi has two extra layers i.e library functions and systems calls
 - Application does not directly control the pins
 - Application needs to call a library function, which in turn calls systems calls to control pins
 - **Not real time**
 - *Consumes a lot of memory for OS*

Advantages of having OS on board

User Interface

- Can use small programs provided by OS without writing code
 - Text based
 - Write commands to perform tasks
 - GUI-based
 - Point-click actions to perform tasks
- **Why use text based interface if you have GUI**
 - GUI provides only a small feature of control over OS
 - Command line provides full feature control
 - But one needs to memorize a **large** set of commands

Advantages of having OS on board

Multiple processes

- Can execute many processes concurrently
 - In Arduino, only one program runs at a time
 - One needs to put all the features in a single program
 - In RPi, one can make a lot of small programs and call them from a master program at will
 - When one has a single core, multi-processing does not mean all processes running **at same time**
 - Time given to all processes is swapped so fast that we don't feel the difference
 - Advantage
 - Can run some processes in background while doing important tasks

Advantages of having OS on board

Easier changing hardware connected

- User Application \Rightarrow `/dev/xxx` \Rightarrow Device driver \Rightarrow Hardware device
- `xxx` is a file associated with a hardware device
- User simply interacts with all devices by accessing the file
- Every device is accessed in a uniform way
 - User simply interacts with file for device
 - File contains code to interact with device driver
 - The burden to working with device rests with the file
- Device driver is the code for accessing physical features of devices

Intro to RPi

- Microcomputer
 - Credit card sized (20×10 cm)
 - Weight = 68 g
- Very cost effective
 - Presently available for INR 2,875 at amazon
- Low power consumption
 - We will use mobile phone charger
- Remote access over internet
 - We will use a LAN cable for connectivity
- Runs Linux
 - Raspbian is a version of Debian optimized for RPi

Versions of RPi boards

Name	Release date
RPi 1 Model A	February 2012
RPi 1 Model A+	November 2014
RPi 1 Model B	April-June 2012
RPi 1 Model B+	July 2014
RPi 2 Model B	February 2015
RPi zero	November 2015

Powerful IoT platform

- Broadcom 900 MHz BCM2836 ARMv7 Quad Core Processor SoC
- Broadcom VideoCore IV GPU
- 1 GB RAM
- Expanded 40-pin GPIO Header
- 4 x USB2.0 Ports with up to 1.2A output
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

Ref: [https:](https://www.raspberrypi.org/products/raspberry-pi-2-model-b/)

[//www.raspberrypi.org/products/raspberry-pi-2-model-b/](https://www.raspberrypi.org/products/raspberry-pi-2-model-b/)

RPi

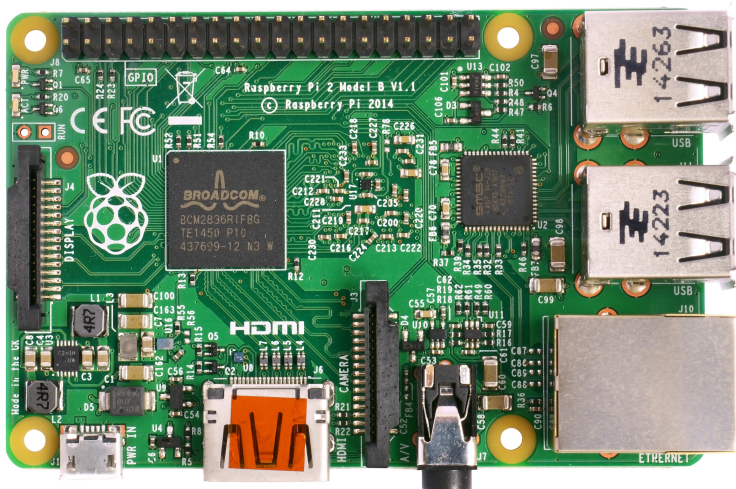


Figure: Top portion of Raspberry Pi 2 Model B

OS

- OS is installed on a micro SD card
- Raspbian is optimized OS for RPi
- Available at <https://www.raspbian.org/>
- Micro SD cards with pre-installed OS are also available
- Installation
 - Format the card
 - Install NOOBS
 - Choose Raspbian
 - Install

Hardware Connections

- Monitor
 - Using HDMI cable, connect a monitor/projector
 - If monitor has VGA port, use a VGA-HDMI adapter
- Keyboard and mouse
 - If you have wired keyboard and mouse, connect them individually to two USB ports
 - If you have set of wireless keyboard and mouse, use one USB port for adapter
- SD card
 - Use SD card slot on back side
 - Use 4GB + card for better performance and enough storage
- Power
 - Use a 5 V 2 A charger

Setup

- ❶ Plug in monitor(HDMI), keyboard (USB), mouse (USB)
- ❷ Get OS on a formatted microSD card
 - Format micro SD card using a SD reader
 - Use NOOBS (New Out-Of-Box Software)
 - Download from <https://www.raspbian.org/downloads>
 - Extract NOOBS download
 - Put it in micro SD card
- ❸ Plug Micro SD card in RPi
- ❹ Power ON the RPi
- ❺ NOOBS GUI starts running on screen
- ❻ NOOBS will install an OS on your micro SD card
 - Will offer a list of options
 - If you are connected to internet, it will offer a longer list
 - Click **Raspbian** and install

Configuring RPi

- First time RPi boots up, it runs a tool called **raspi-config**
 - Used to configure RPi
 - Defines username, password etc.
 - Can be invoked at any stage by writing `raspi-config` on terminal
 - Expand file system
 - Reformats micro SD card file system to access all the memory
 - Change User account
 - Default username is **pi** and password is **raspberry**
 - Enable boot to Console/Desktop/Scratch
 - Console is default boot option
 - Choose Desktop
 - Scratch is a programming language for kids
 - Internationalization and Rastrack
 - Change Locale Change timezone Change keyboard Layout
 - Add to Rastrack
 - Service which allows RPi users to find one another based on IP location
 - Optional

Over Clocking

- Increasing clock frequency of device beyond recommendation
- There are a lot of clocks in RPi
- Example:
 - ARM frequency
 - SDRAM frequency
- Impact
 - Instructions are executed faster
 - About one instruction is executed per clock time period
- Risk
 - Signals might not reach destination on time
 - Whole machine fails
 - Heating
 - Lifetime of device gets shortens

Raspbian-Info

- Version of Debian(Linux) optimized for RPi
- Linux commands run
- Shell
 - Command line interface between OS and user
 - Many shells exist
 - We will use BASH (Bourne Again SHell)
 - BASH is default shell for Raspbian
- Console/Terminal
 - Text entry and display device
 - can be physical device
 - Now one mostly uses virtual terminal
 - A software emulator of terminal
 - LXTerminal is default terminal in Raspbian
 - Prompt is \$

Raspbian-login

- User Accounts

- Linux Machine can entertain multiple user accounts at same time
- Each account is access using a **username** and a **password**
- Process of accessing a machine using an account is called **login**
- Default:
 - username = pi
 - password = raspberry

Raspbian-file system

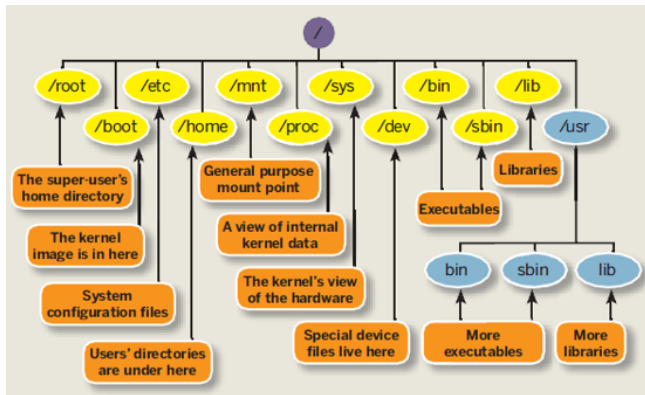


Figure: Linux file system

Ref: http://www.siongboon.com/projects/2013-07-08_raspberry_pi/images/linux_filesystem.png

Traversing raspbian file structure

- **pwd** = print working directory
- **mkdir** = make directory
- **rmdir** = remove directory
- **rmdir -r** = remove directory recursively
- **cd** = change directory
- **cd ..** = step back in directory by one step
- **/** = step back to home
- **ls** = lists the contents of directory
- **ls -l** = long list of contents with great details

Action

- Lets make a directory names "IoT"
- Lets make a subdirectory within "IoT" named "PyDelhi"
- Lets go back to directory "IoT"
- See the LIST of content of this directory
- Print the present working directory
- remove the subdirectory named "PyDelhi" and make it again

So we are ready to work on Raspbian!

Text Editors

- For creating and modifying a file
- A word processors but simpler
- Two types:
 - Command line based
 - emacs, vi, vim, nano
 - GUI based
 - gedit

Making and Viewing Files

- Write the name of text editor succeeded by file
- If a file does not exist, it is created and opened for editing
 - **nano test.py**
- **cat test** prints the file to the terminal
- **head test** prints the first 10 lines
- **last test** or **tail test** prints last 10 lines
- **cp test new-test** copies file **test** to a new file **new-test**
- **mv test a/test1** moves **test** file to a new destination (/a/) by *renaming* it **test1**
- create a file, view and edit the contents, make a copy and delete the copy and then rename it!

Making files and writing codes

- Make a file using **nano**
- Make a file named "hello.py"
- Make the following python program:
 - **print "hello world"**
- Save this file
- Check, where is this program saved!
- run this python code by writing the command
 - **python hello.py**
- Output can be checked in next line

Permission

- Files has owner (user) and defined permission
 - read (r)
 - write (w)
 - execute (x)
- Permission are assigned according to type:
 - user : file owner
 - group : a permission group
 - other : all users
- Type **ls -l** to see these permissions
- Do this at home directory
- For **Desktop** directory it shows following permission
 - **drwxr-xr-x**
 - **d** shows that its a directory
 - Next three symbols define *user* permissions **rwr**
 - Next three symbols define *group* permissions **r-x**
 - Next three symbols define *other* permissions **r-x**
- create a file and check out its permissions

Root

- Root account has highest permission level
- Key files and directory is accessible only to root
- Sometimes you would need to be root
 - To install a program
 - Change OS as per requirements
- **su** : super-user
 - asks for root password
 - If password is correct, one login as root
- **sudo**: super-user do
 - Just applies root permission for single command
 - Ex: **sudo ls** will simply apply root permission for listing the files
 - safer way!
- We would need **sudo** command to access Raspberry Pi pins

Python on RPi

- Used to program to access pins and process
- Can use any language!
- C and C++ requires a compiler called **gcc** which is pre-installed
- Python interpreter is also pre-installed
- We shall use Python 3 instead of Python 2 here because most packages for RPi usage are written in Python 3

Other ways to work with python on RPi

- Python programming Environments
 - IDE
 - Combines the facilities of interpreter and text editor
 - Default IDE is IDLE
 - Invoke: Menu → Programming → Python
 - Select Python 3
 - Text-editor and interpreter separately
 - Use **nano** to write code (ex: hello.py)
 - Execute the program by typing **python3 hello.py**

GPIO configuration



www.raspberrypi.org

GPIO

- Dedicated power and ground pins
 - 3.3V(1, 17)
 - 5V(2,4)
 - GND (6, 9, 14, 20, 30, 39)
- GPIO = General Purpose Input Output
- Make pins *input* or *output* pins as per choice
- There are two numbering systems
 - pin number based on location
 - Pin number given as GPIO1, GPIO2 etc.

Protocol pins

- **I2C**

- Pin no.3 (GPIO2) = SDA1 I2C
- Pin No.5 (GPIO3) = SCL1 I2C
- Serial communication protocol between two chips relatively closely placed and need to share a clock Two wire protocol (SDA = sends Data signal, SCL = sends Clock signal)
- If there are several I2C compatible devices, one can connect their SDA and SCL lines together for serial communication between them

Protocol Pins

• SPI

- 19 (GPIO10) = MOSI (Master Out Slave In)
- 21 (GPIO9) = MISO (Master In Slave Out)
- 23 (GPIO11) = SCLK (S Clock)
- 24 (GPIO8) = CE01 (Chip Enable)
- 26 (GPIO7) = CE02 (Chip Enable)

Lets start!

Now lets start doing some stuff

Connecting RPi to internet

- Two ways:
 - Wired connection at built-in RJ45 connector
 - Use a USB wireless adapter compatible with Linux OS
 - Start **wifi config** program from the desktop
 - Click *Scan* to find networks
 - Select a network and give a password
 - DHCP (Dynamic Host Connection Protocol) is needed to get an IP address
 - Check connection using *ping*
 - RPi default browser is called *Epiphany*

Firewall

- Firewall block certain application
- Check the firewall of your network
- Get help from network administrator to unblock the application if you wish to use the same

Headless running Pi

- Remember that usually we use BASH (Bourne Again SHell) on RPi
- SSH (Secure SHell):
 - Used for *secured remote* connection
 - SSH runs on local machine (**ssh client**) to connect to a remote machine (**ssh server**)
 - Both machines must be connected to internet
 - For present workshop:
 - ssh client = Your laptop
 - ssh server = RPi
- *telnet* is another program which can connect to remote computer but it is not secured
- We shall use SSH in following lecture
- **Commands typed at client are executed at server**
- **Text printed by server appears on client screen**
- SSH has a command line tool!

Headless running Pi

- We will connect to RPi ssh server using virtual terminal in laptop (client)
- SSH client program is installed by default in Linux and MAC OSX
- Windows users can use putty (<http://www.putty.org/>)
- SSH server on RPi
 - SSH server must be **started**
 - Raspbian runs SSH server as demon by default so it need not be started
 - User must have an account on server machine
 - Must know username and password
 - Firewall must allow SSH application
- Open a terminal and type
`ssh < username > @ < domainname >`
- If username is found on remote machine then *password* is needed to login

Headless running Pi

- You will get a message that address could not be authenticated
- Enter **yes** to continue
- It remembers the credentials for next time onwards
- Running RPi as SSH server removes the need for keyboard, mouse and monitor
- This is termed as **headless running**

Private host ID keys

- Should change private host ID keys
- Private host ID keys are provided during configuration
- They are same on all RPis so others can listen to your communication
- type the following command at the terminal as a single line
`sudo rm /etc/ssh/ssh_host_* && sudo dpkg-reconfigure openssh-server`

Blinking an LED

Python Program to blink an LED connected at pin 1

```
import Rpi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13,GPIO.OUT)
while True:
    GPIO.output(13,True)
    time.sleep(1)
    GPIO.output(13,False)
    time.sleep(1)
```

Program: blink.py

Reading a pin connected to a sensor

Python Program to read a sensor at pin 13

```
import Rpi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13,GPIO.IN)
while True:
    value = GPIO.input(13)
    print(value)
```

Program: sensor.py

Generating Analog like signal using PWM

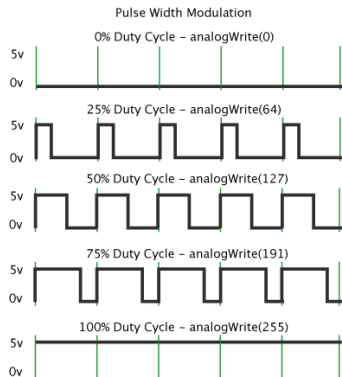


Figure: PWM signal showing different duty cycle

Ref: https://upload.wikimedia.org/wikipedia/commons/4/49/Pwm_5steps.gif

RPi Python commands for PWM

- **pwmObject = GPIO.PWM(18,500)**
 - A new objects named **pwmObject** is created at pin 18 which will be readied for generating signal at 400 Hz
 - Various methods can be assigned to this object
- **pwmObject.start(50)**
 - Starts generating a PWM signal at pin 18 with duty cycle of 50 %
- **pwmObject.changeDutyCycle(75)**
 - Changes duty cycle to 75 %

Because RPi has an OS which produces unpredictable time delays, the frequencies obtained by PWM may not be very accurate, especially at higher frequencies