IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Python Programming – Part I

Dr. Sudip Misra
Associate Professor
Department of Computer Science and Engineering
IIT KHARAGPUR
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/

Why Python?

- Python is a versatile language which is easy to script and easy to read.
- It doesn’t support strict rules for syntax.
- Its installation comes with integrated development environment for programming.
- It supports interfacing with wide ranging hardware platforms.
- With open-source nature, it forms a strong backbone to build large applications.

Python IDE

- Python IDE is a free and open source software that is used to write codes, integrate several modules and libraries.
- It is available for installation into PC with Windows, Linux and Mac.
- Examples: Spyder, PyCharm, etc.

Starting with Python

- Simple printing statement at the python interpreter prompt,

  >>> print “Hi, Welcome to python!”

  Output: Hi, Welcome to python!

- To indicate different blocks of code, it follows rigid indentation.

```
if True:
        print “Correct"
else:
        print “Error"
```

Data-types in Python

- There are 5 data types in Python:
  - ✓ Numbers

    x, y, z = 10, 10.2, " Python "

  - ✓ String

    x = ‘This is Python’

    print x >>This is Python

    print x[0] >>T

    print x[2:4] >>is

Data-types in Python (contd..)

- ✓ List

  x = [10, 10.2, 'python']

- ✓ Tuple

- ✓ Dictionary

  d = {1:‘item','k':2}

Controlling Statements

- if (cond.):
  - statement 1
  - statement 2

  elif (cond.):
  - statement 1
  - statement 2

  else:
  - statement 1
  - statement 2

- while (cond.):
  - statement 1
  - statement 2
- x = [1,2,3,4]

  for i in x:
  - statement 1
  - statement 2

Controlling Statements (contd..)

- Break

```
for s in "string":
    if s == ‘n':
        break
    print (s)
print “End”
```

- Continue

```
for s in "string":
    if s == ‘y':
        continue
    print (s)
print “End”
```

Functions in Python

- Defining a function
  - ✓ Without return value

```
def funct_name(arg1, arg2, arg3):      # Defining the function
        statement 1
        statement 2
```

  - ✓ With return value

```
def funct_name(arg1, arg2, arg3):      # Defining the function

        statement 1
        statement 2
        return x                       # Returning the value
```

Functions in Python

- Calling a function

```
def example (str):
    print (str + "!")


example ("Hi")                    # Calling the function
```

Output:: Hi!

Functions in Python (contd..)

- Example showing function returning multiple values

```
def greater(x, y):
    if x > y:
        return x, y
    else:
        return y, x

val = greater(10, 100)
print(val)
```

Output:: (100,10)

Functions as Objects

- Functions can also be assigned and reassigned to the variables.
- Example:

```
def add (a,b)
    return a+b

print (add(4,6))
c = add(4,6)
print c
```

Output:: 10 10

Variable Scope in Python

Global variables:
These are the variables declared out of any function , but can be accessed inside as well as outside the function.

Local variables:
These are the ones that are declared inside a function.

Example showing Global Variable

```
g_var = 10

def example():
    l_var = 100
    print(g_var)

example()              # calling the function
```

Output:: 10

Example showing Variable Scope

```
var = 10

def example():
    var = 100
    print(var)

example()              # calling the function
print(var)
```

Output:: 100
10

Modules in Python

- Any segment of code fulfilling a particular task that can be used commonly by everyone is termed as a module.

- Syntax:

```
import module_name          #At the top of the code

using module_name.var       #To access functions and values
                                with 'var' in the module
```

Modules in Python (contd..)

- Example:

```
import random

for i in range(1,10):
        val = random.randint(1,10)
        print (val)
```

Output:: varies with each execution

Modules in Python (contd..)

- We can also access only a particular function from a module.

- Example:

```
from math import pi

print (pi)
```

Output:: 3.14159

Exception Handling in Python

- An error that is generated during execution of a program, is termed as exception.
- Syntax:

```
try:
    statements
except _Exception_:
    statements
else:
  statements
```

Exception Handling in Python (contd..)

- Example:

```
while True:
    try:
        n = input ("Please enter an integer: ")
        n = int (n)
        break
    except ValueError:
        print "No valid integer! "
print “It is an integer!"
```

Example Code: to check number is prime or not

```
x = int (input("Enter a number: "))
def prime (num):
        if num > 1:
           for i in range(2,num):
                if (num % i) == 0:
                        print (num,"is not a prime number")
                        print (i,“is a factor of”,num)
                        break
           else:
                print(num,"is a prime number")
        else:
            print(num,"is not a prime number")
prime (x)
```

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Python Programming – Part II

Dr. Sudip Misra
Associate Professor
Department of Computer Science and Engineering
IIT KHARAGPUR
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/

File Read Write Operations

- Python allows you to read and write files
- No separate module or library required
- Three basic steps
  - Open a file
  - Read/Write
  - Close the file

File Read Write Operations (contd..)

Opening a File:

- Open() function is used to open a file, returns a file object

  open(file_name, mode)

- Mode: Four basic modes to open a file

  r: read mode

  w: write mode

  a: append mode

  r+: both read and write mode

File Read Write Operations (contd..)

Read from a file:

- read(): Reads from a file

```
file=open('data.txt', 'r')
file.read()
```

Write to a file:

- Write(): Writes to a file

```
file=open('data.txt', 'w')
file.write('writing to the file')
```

File Read Write Operations (contd..)

Closing a file:

- Close(): This is done to ensure that the file is free to use for other resources

```
file.close()
```

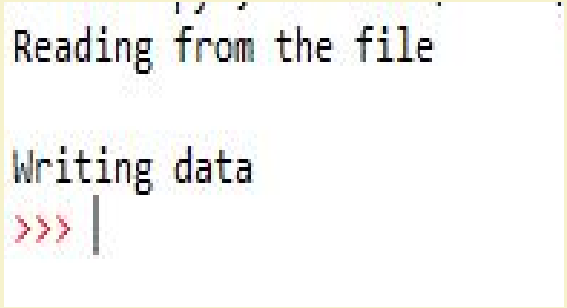
Using WITH to open a file:

- Good practice to handle exception while file read/write operation
- Ensures the file is closed after the operation is completed, even if an exception is encountered

```
with open(“data.txt","w") as file:
file.write(“writing to the text file”)
file.close()
```

File Read Write Operations code + image

```
with open("PythonProgram.txt","w") as file:
    file.write("Writing data")
file.close()

with open("PythonProgram.txt","r") as file:
    f=file.read()
    print('Reading from the file\n')
    print (f)
file.close()
```



File Read Write Operations (contd..)

Comma Separated Values Files

- CSV module supported for CSV files

Read:

```
with open(file, "r") as csv_file:
   reader = csv.reader(csv_file)
   print("Reading from the CSV File\n")
   for row in reader:
      print(" ".join(row))
csv_file.close()
```

Write:

```
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file = "output.csv"
with open(file, "w") as csv_file:
   writer = csv.writer(csv_file, delimiter=',')
   print("Writing CSV")
   for line in data:
      writer.writerow(line)
csv_file.close()
```

File Read Write Operations (contd..)

```
import csv

#writing a csv file
data = ["1,2,3,4,5,6,7,8,9".split(",")]
file = "output.csv"
with open(file, "w") as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    print("Writing CSV")
    for line in data:
        writer.writerow(line)
csv_file.close()

#reading from a csv file
with open(file, "r") as csv_file:
    reader = csv.reader(csv_file)
    print("Reading from the CSV File\n")
    for row in reader:
        print(" ".join(row))
csv_file.close()
```

```
Writing CSV
Reading from the CSV File

1 2 3 4 5 6 7 8 9

>>>
```

Image Read/Write Operations

- Python supports PIL library for image related operations
- Install PIL through PIP

sudo pip install pillow

PIL is supported till python version 2.7. Pillow supports the 3x version of python.

Image Read/Write Operations

Reading Image in Python:

- PIL: Python Image Library is used to work with image files

  from PIL import Image

- Open an image file

  image=Image.open(image_name)

- Display the image

  image.show()

Image Read/Write Operations (contd..)

Resize(): Resizes the image to the specified size

image.resize(255,255)

Rotate(): Rotates the image to the specified degrees, counter clockwise

image.rotate(90)

Format: Gives the format of the image
Size: Gives a tuple with 2 values as width and height of the image, in pixels
Mode: Gives the band of the image, 'L' for grey scale, 'RGB' for true colour image

print(image.format, image.size, image.mode)

Image Read/Write Operations (contd..)

Convert image to different mode:

- Any image can be converted from one mode to ‘L’ or ‘RGB’ mode

conv_image=image.convert(‘L’)

- Conversion between modes other that ‘L’ and ‘RGB’ needs conversion into any of these 2 intermediate mode

Output

Converting a sample image to Grey Scale

```
from PIL import Image

im = Image.open('/home/saswati/VRP_Linux/Images/i3.jpg')
im.show()
grey_image=im.convert('L')
grey_image.show()
grey_image.save('GreyScaleImage.jpg')
```

Output

Networking in Python

- Python provides network services for client server model.
- Socket support in the operating system allows to implement clients and servers for both connection-oriented and connectionless protocols.
- Python has libraries that provide higher-level access to specific application-level network protocols.

Networking in Python (contd..)

- Syntax for creating a socket:
  s = socket.socket (socket_family, socket_type, protocol=0)

socket_family – AF_UNIX or AF_INET

socket_type – SOCK_STREAM or SOCK_DGRAM

protocol – default '0'.

Example - simple server

- The socket waits until a client connects to the port, and then returns a connection object that represents the connection to that client.

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('10.14.88.82', 2017)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)
```

Example - simple server (contd..)

```
# Listen for incoming connections
sock.listen(1)

connection, client_address = sock.accept()

#Receive command
data = connection.recv(1024)
print(data)
sock.close()
```

Example - simple client

```
import socket
import sys

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect to Listener socket
client_socket.connect(("10.14.88.82", 2017))
print>>sys.stderr,'Connection Established'

#Send command
client_socket.send('Message to the server')
print('Data sent successfully')
```

Code Snapshot

```python
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('10.14.88.82', 2017)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

connection, client_address = sock.accept()

#Receive command
data = connection.recv(1024)
print(data)
sock.close()
```

```python
import socket
import sys

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Connect to Listener socket
client_socket.connect(("10.14.88.82", 2017))
print>>sys.stderr,'Connection Established'

#Send command
client_socket.send('Message to the server')
print('Data sent successfully')
```

Output

```
starting up on 10.14.88.82 port 2017
Message to the server
saswati@saswati-BK361AA-ACJ-CQ3236IX:~/Desktop$
```

```
Connection Established
Data sent successfully
saswati@saswati-BK361AA-ACJ-CQ3236IX:~/Desktop$
```

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Raspberry Pi – Part I

Dr. Sudip Misra
Associate Professor
Department of Computer Science and Engineering
IIT KHARAGPUR
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/

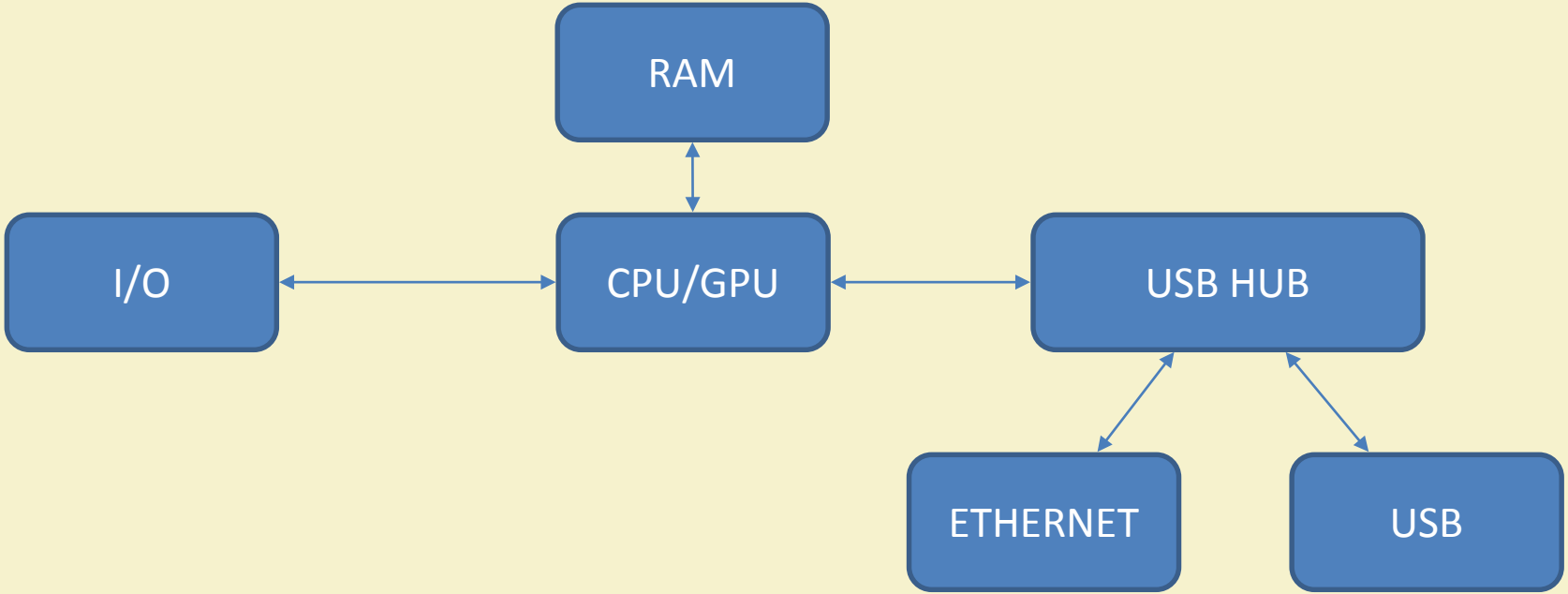
What is Raspberry Pi?

- Computer in your palm.
- Single-board computer.
- Low cost.
- Easy to access.

Specifications

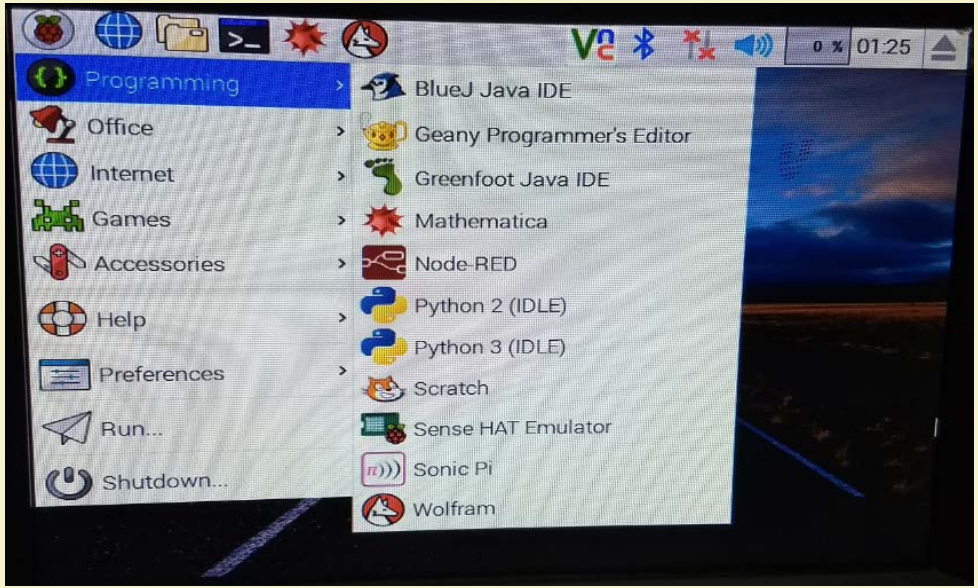
Key features	Raspberry pi 3 model B	Raspberry pi 2 model B	Raspberry Pi zero
RAM	*1GB SDRAM*	*1GB SDRAM*	*512 MB SDRAM*
CPU	*Quad cortex A53@1.2GHz*	*Quad cortex A53@900MHz*	*ARM 11@ 1GHz*
GPU	*400 MHz video core IV*	*250 MHz video core IV*	*250 MHz video core IV*
Ethernet	*10/100*	*10/100*	*None*
Wireless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

Basic Architecture



Raspberry Pi

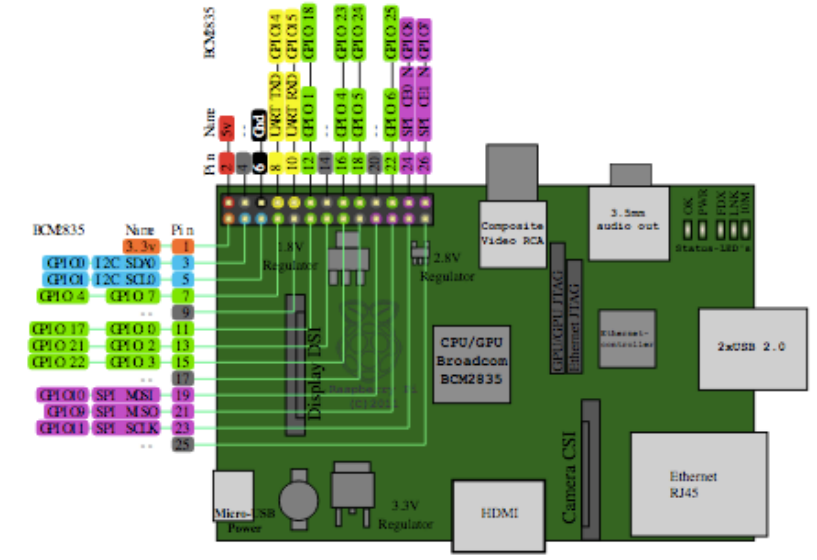
Start up raspberry pi



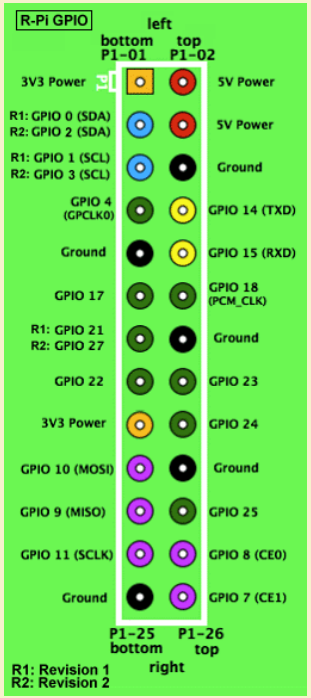
Raspberry Pi GPIO

- Act as both digital output and digital input.
- **Output**: turn a GPIO pin high or low.
- **Input**: detect a GPIO pin high or low.

Raspberry Pi pin configuration



Source: Raspberry Pi PCB Pin Overview, Wikimedia Commons (Online)



Source: Raspberry Pi GPIO, Wikimedia Commons (Online)

Basic Set up for Raspberry Pi

- HDMI cable.
- Monitor.
- Key board.
- Mouse.
- 5volt power adapter for raspberry pi.
- LAN cable .
- Min- 2GB micro sd card

Basic Set up for Raspberry Pi

Operating System

Official Supported OS :

- Raspbian
- NOOBS

Some of the third party OS :

- UBUNTU mate
- Snappy Ubuntu core
- Windows 10 core
- Pinet
- Risc OS

Source: Downloads, Raspberry Pi Foundation

Raspberry Pi Setup

Download Raspbian:

- Download latest Raspbian image from raspberry pi official site: https://www.raspberrypi.org/downloads/

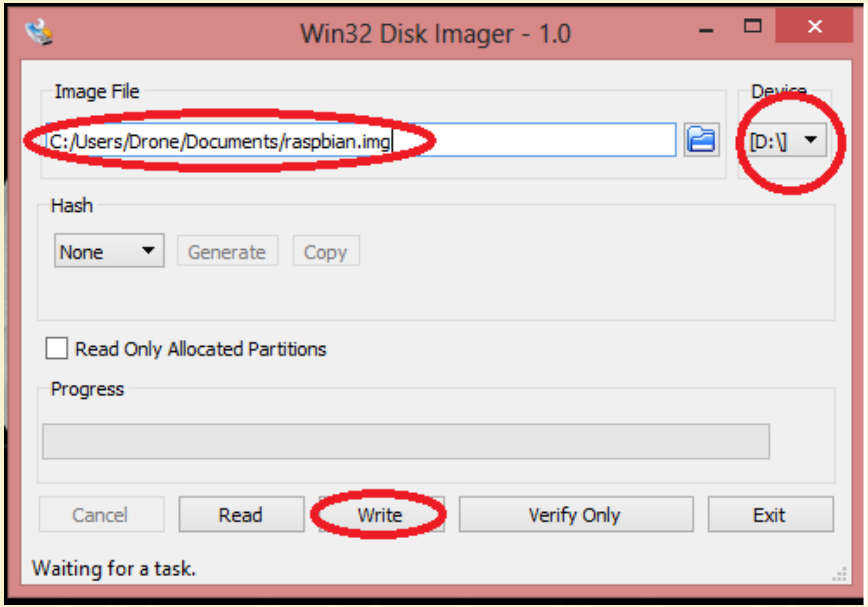
- Unzip the file and end up with an .img file.

Raspberry Pi OS Setup

Write Raspbian in SD card :

- Install “Win32 Disk Imager” software in windows machine .
- Run Win32 Disk Imager
- Plug SD card into your PC
- Select the “Device”
- Browse the “Image File”(Raspbian image)
- Write

Raspberry Pi OS Setup



Basic Initial Configuration

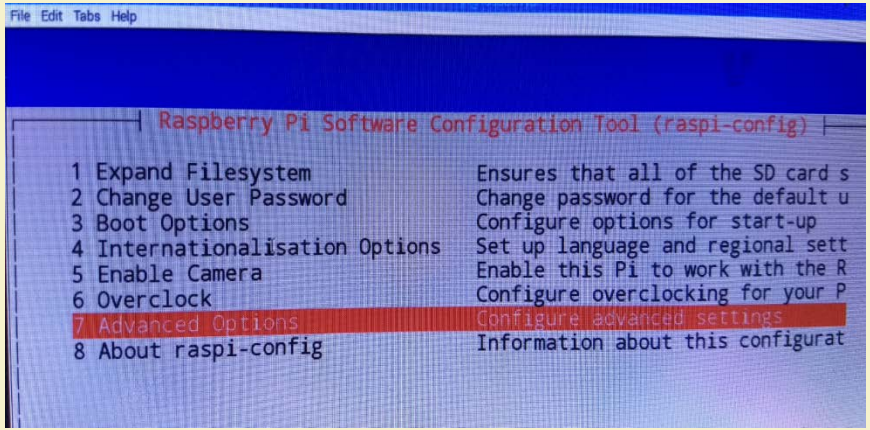
Enable SSH

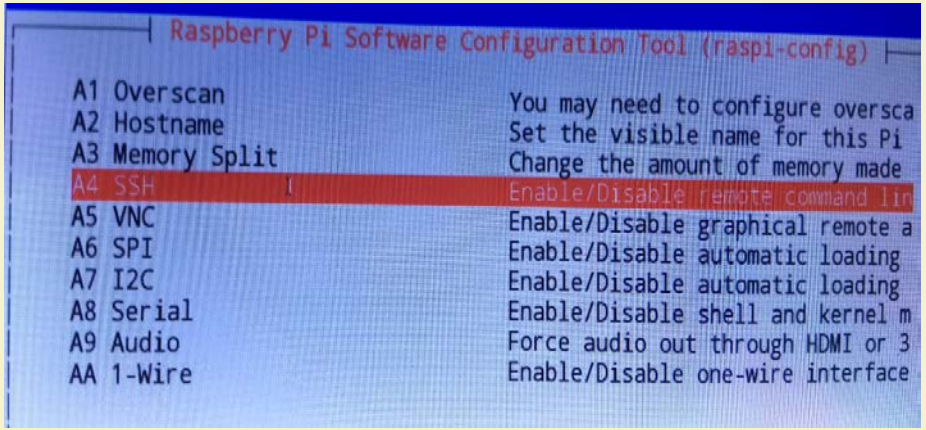
Step1 : Open command prompt and type **sudo raspi-config** and press enter.

Step2: Navigate to SSH in the Advance option.

Step3: Enable SSH

Basic Initial Configuration





Basic Initial Configuration contd.

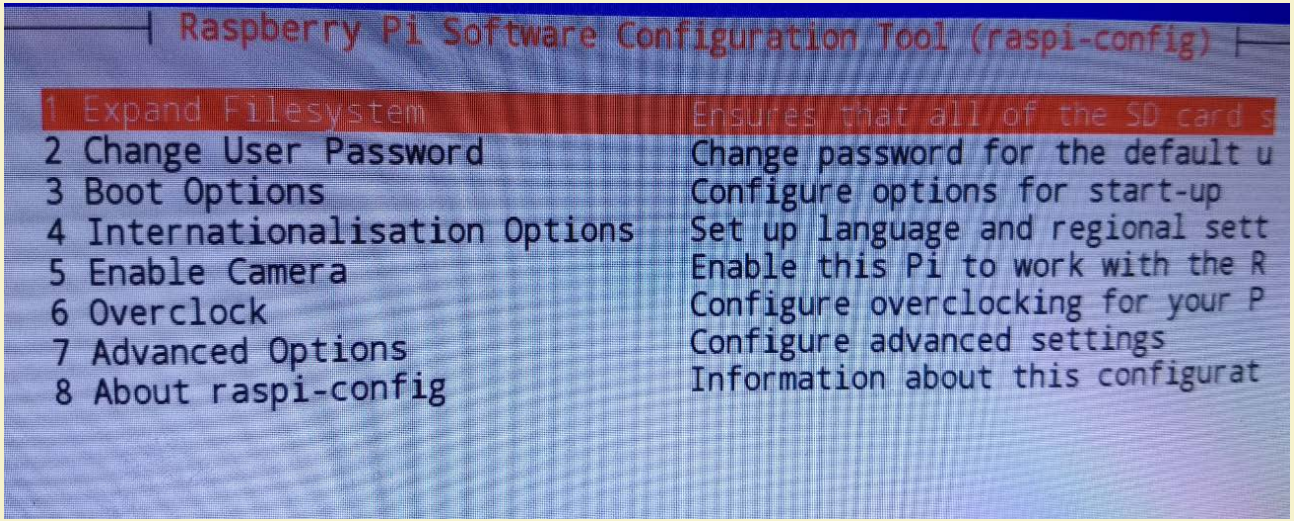
Expand file system :

Step 1: Open command prompt and type **sudo raspi-config** and press enter.

Step 2: Navigate to Expand Filesystem

Step 3: Press enter to expand it.

Basic Initial Configuration contd.



Programming

Default installed :

- Python
- C
- C++
- Java
- Scratch
- Ruby

Note : Any language that will compile for ARMv6 can be used with raspberry pi.

Source: Programming languages for Raspberry Pi, eProseed, Lonneke Dikmans, August 07, 2015

Popular Applications

- Media streamer
- Home automation
- Controlling BOT
- VPN
- Light weight web server for IOT
- Tablet computer

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Raspberry Pi – Part II

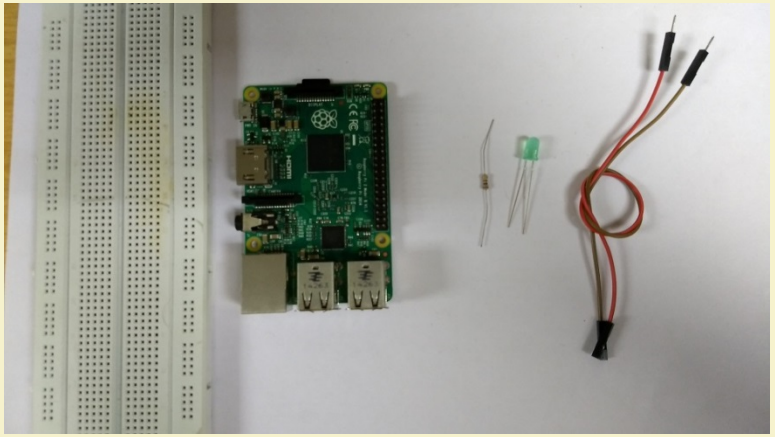
Dr. Sudip Misra
Associate Professor
Department of Computer Science and Engineering
IIT KHARAGPUR
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/

Topics Covered

- Using GPIO pins
- Taking pictures using PiCam

Blinking LED

- Requirement:
- Raspberry pi
- LED
- 100 ohm resistor
- Bread board
- Jumper cables



Blinking LED (contd..)

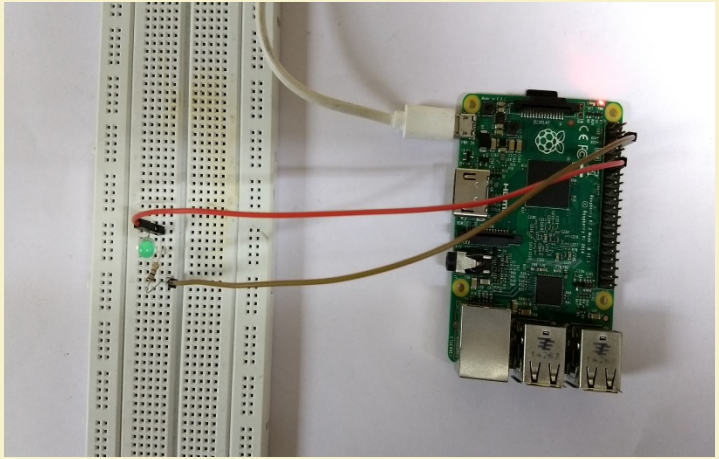
Installing GPIO library:

- Open terminal
- Enter the command “sudo apt-get install python-dev” to install python development
- Enter the command “sudo apt-get install python-rpi.gpio” to install GPIO library.

Blinking LED (contd..)

Connection:

- Connect the negative terminal of the LED to the ground pin of Pi
- Connect the positive terminal of the LED to the output pin of Pi



Blinking LED (contd..)

Basic python coding:

- Open terminal enter the command

  sudo nano filename.py

- This will open the nano editor where you can write your code
- Ctrl+O : Writes the code to the file
- Ctrl+X : Exits the editor

Blinking LED (contd..)

Code:

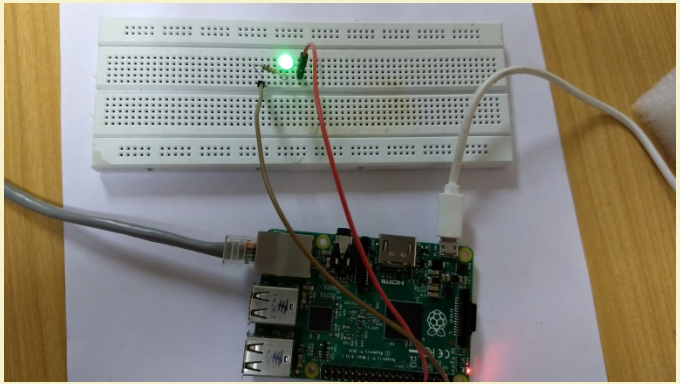
```
import RPi.GPIO as GPIO          #GPIO library
import time
GPIO.setmode(GPIO.BOARD)         # Set the type of board for pin numbering
GPIO.setup(11, GPIO.OUT)         # Set GPIO pin 11as output pin
for i in range (0,5):
   GPIO.output(11,True)          # Turn on GPIO pin 11
   time.sleep(1)
   GPIO.output(11,False)
   time.sleep(2)
   GPIO.output(11,True)
GPIO.cleanup()
```

Blinking LED (contd..)

```
  GNU nano 2.2.6                                    File: BLINK_LED.py

import RPi.GPIO as GPIO ## GPIO library
import time
GPIO.setmode(GPIO.BOARD) ## Set the type of board for pin numbering
GPIO.setup(11, GPIO.OUT) ## Set GPIO pin 11 as output pin
for i in range (0,5):
    GPIO.output(11,True) ## Turn on GPIO pin 11
    time.sleep(1)
    GPIO.output(11,False)
    time.sleep(2)
    GPIO.output(11,True)
GPIO.cleanup()
```

Blinking LED (contd..)

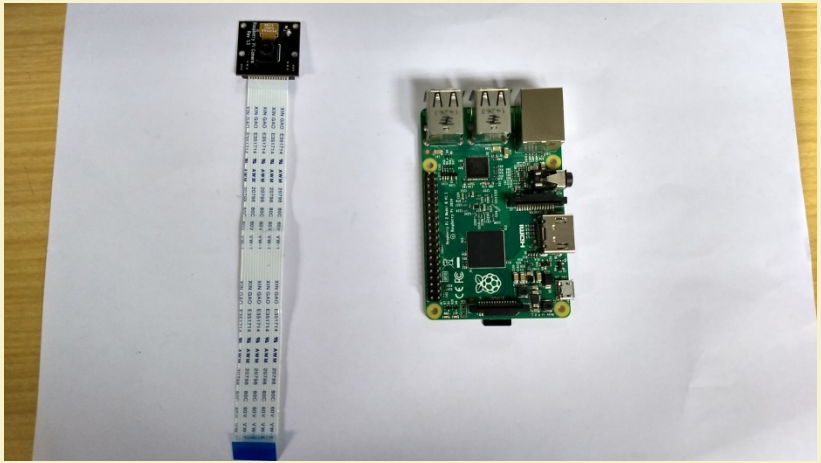
The LED blinks in a loop with delay of 1 and 2 seconds.



Capture Image using Raspberry Pi

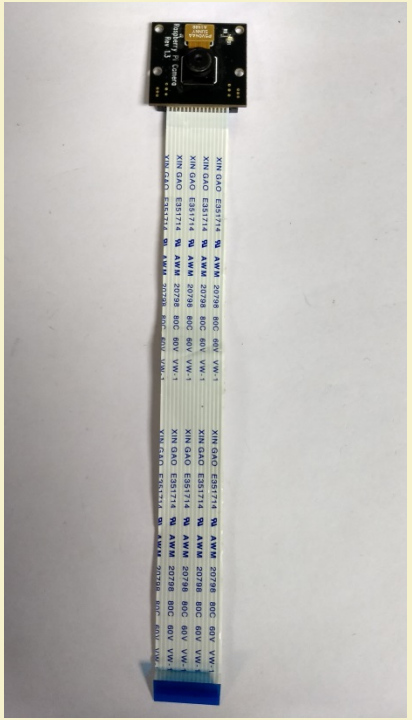
Requirement

- Raspberry Pi
- Raspberry Pi Camera



Raspberry Pi Camera

- Raspberry Pi specific camera module
- Dedicated CSI slot in Pi for connection
- The cable slot is placed between Ethernet port and HDMI port



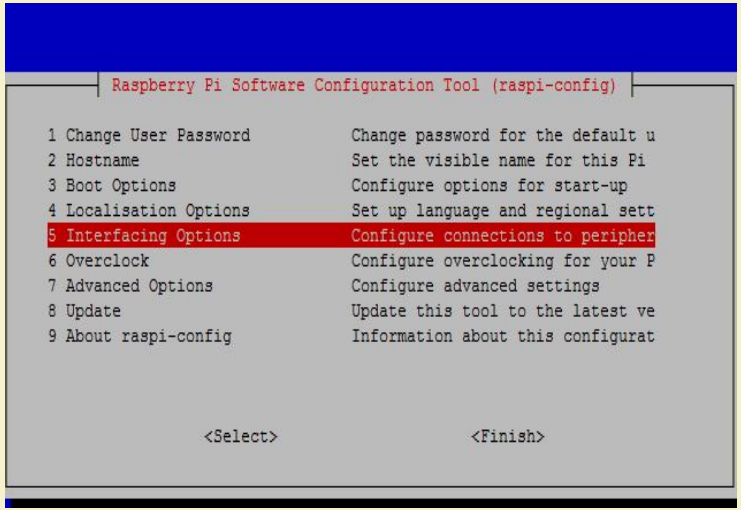
Connection

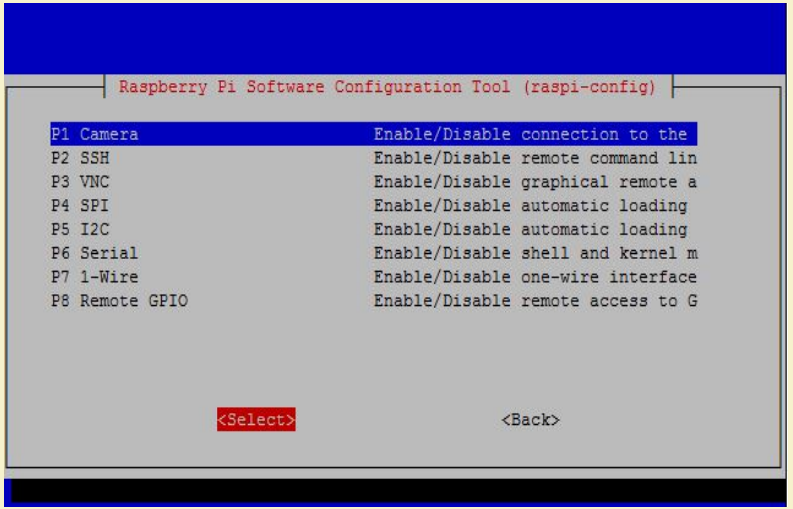
Boot the Pi once the camera is connected to Pi

Configuring Pi for Camera

- In the terminal run the command “sudo raspi-config” and press enter.
- Navigate to “Interfacing Options” option and press enter.
- Navigate to “Camera” option.
- Enable the camera.
- Reboot Raspberry pi.

Configuring Pi for Camera (contd..)





Capture Image

- Open terminal and enter the command-

  raspistill -o image.jpg

- This will store the image as ‘image.jpg’

Capture Image (contd..)

PiCam can also be processed using Python camera module python-picamera

sudo apt-get install python-picamera

Python Code:

```
Import picamera
camera = picamera.PiCamera()
camera.capture('image.jpg')
```

Source: PYTHON PICAMERA, Raspberry Pi Foundation

Capture Image (contd..)

```
pi@raspberrypi:~ $ raspistill -o image.jpg
pi@raspberrypi:~ $
```

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Implementation of IoT with Raspberry Pi: Part 1

Dr. Sudip Misra
Associate Professor
Department of Computer Science and Engineering
IIT KHARAGPUR
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/

IOT

Internet Of Things

- Creating an interactive environment
- Network of devices connected together

Sensor

- Electronic element
- Converts physical quantity into electrical signals
- Can be analog or digital

Actuator

- Mechanical/Electro-mechanical device
- Converts energy into motion
- Mainly used to provide controlled motion to other components

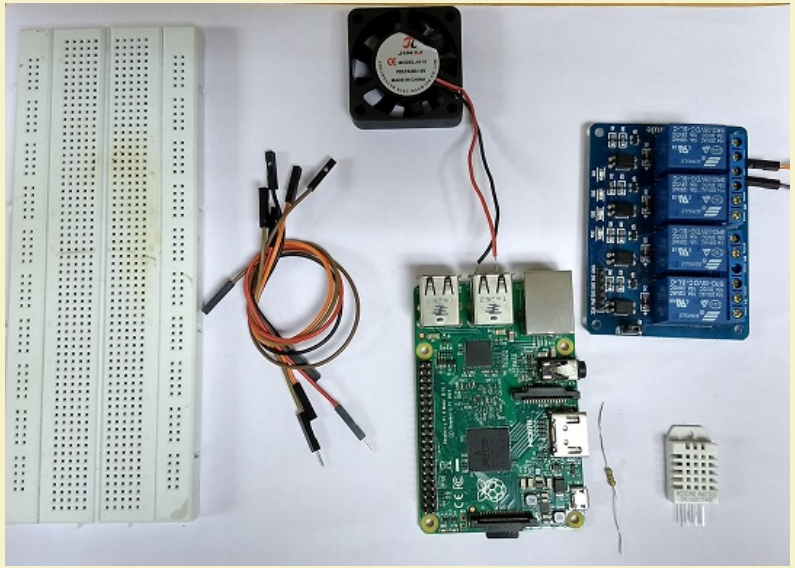
System Overview

- Sensor and actuator interfaced with Raspberry Pi
- Read data from the sensor
- Control the actuator according to the reading from the sensor
- Connect the actuator to a device

System Overview (contd..)

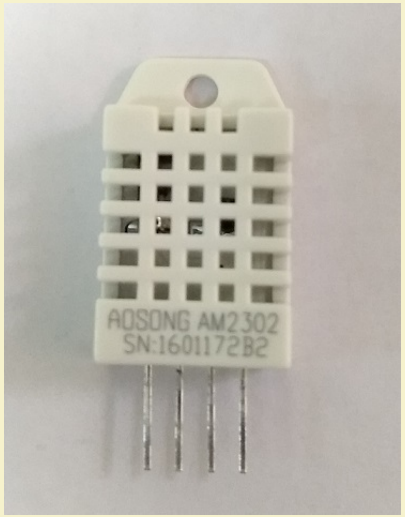
Requirements

- DHT Sensor
- 4.7K ohm resistor
- Relay
- Jumper wires
- Raspberry Pi
- Mini fan



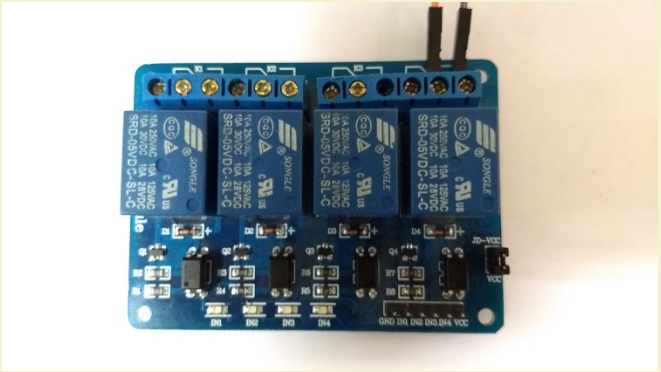
DHT Sensor

- Digital Humidity and Temperature Sensor (DHT)
- PIN 1, 2, 3, 4 (from left to right)
  - PIN 1- 3.3V-5V Power supply
  - PIN 2- Data
  - PIN 3- Null
  - PIN 4- Ground



Relay

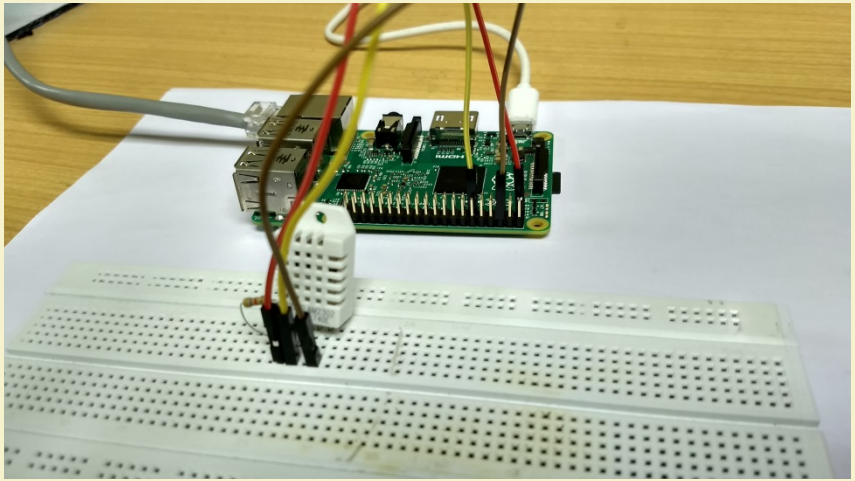
- Mechanical/electromechanical switch
- 3 output terminals (left to right)
  - NO (normal open):
  - Common
  - NC (normal close)



Temperature Dependent Auto Cooling System

Sensor interface with Raspberry Pi

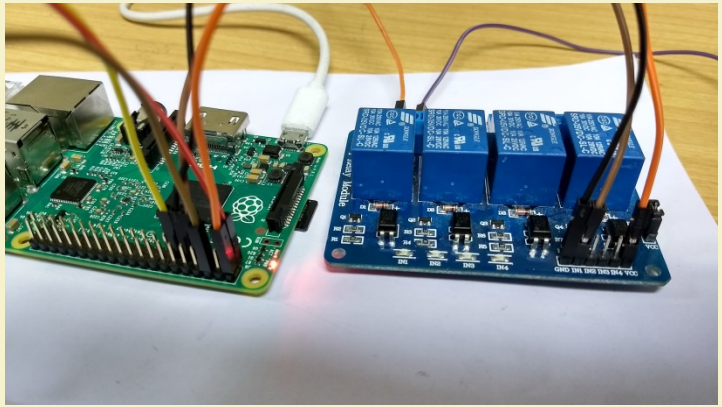
- Connect pin 1 of DHT sensor to the 3.3V pin of Raspberry Pi
- Connect pin 2 of DHT sensor to any input pins of Raspberry Pi, here we have used pin 11
- Connect pin 4 of DHT sensor to the ground pin of the Raspberry Pi



Temperature Dependent Auto Cooling System (contd..)

Relay interface with Raspberry Pi

- Connect the VCC pin of relay to the 5V supply pin of Raspberry Pi
- Connect the GND (ground) pin of relay to the ground pin of Raspberry Pi
- Connect the input/signal pin of Relay to the assigned output pin of Raspberry Pi (Here we have used pin 7)



Temperature Dependent Auto Cooling System (contd..)

Adafruit provides a library to work with the DHT22 sensor

- Install the library in your Pi-
  - Get the clone from GIT
    git clone https://github.com/adafruit/Adafruit_Python_DHT.g...
  - Go to folder Adafruit_Python_DHT
    cd Adafruit_Python_DHT
  - Install the library
    sudo python setup.py install

Source: ADAFRUIT DHTXX SENSORS, Lady Ada, 2012-07-29

Program: DHT22 with Pi

```
import RPi.GPIO as GPIO
from time import sleep
import Adafruit_DHT                                  #importing the Adafruit library

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
sensor = Adafruit_DHT.AM2302                         # create an instance of the sensor type
print ('Getting data from the sensor')
#humidity and temperature are 2 variables that store the values received from the sensor

humidity, temperature = Adafruit_DHT.read_retry(sensor,17)
print ('Temp={0:0.1f}*C humidity={1:0.1f}%'.format(temperature, humidity))
```

Program: DHT22 interfaced with Raspberry Pi

Code

```
 GNU nano 2.2.6                File: IOTSR.py

import RPi.GPIO as GPIO
from time import sleep

import Adafruit_DHT

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

sensor = Adafruit_DHT.AM2302  # create an instance of the sensor type

print ('Getting data from the sensor')

#humidity and temperature are 2 variables that store the values received from the sensor
humidity, temperature = Adafruit_DHT.read_retry(sensor,17)

print ('Temp={0:0.1f}*C humidity={1:0.1f}%'.format(temperature, humidity))
```

Output

```
pi@raspberrypi:~ $ python IOTSR.py
Getting data from the sensor
Temp=26.1*C humidity=65.9%
pi@raspberrypi:~ $
```

Connection: Relay

- Connect the relay pins with the Raspberry Pi as mentioned in previous slides

- Set the GPIO pin connected with the relay's input pin as output in the sketch

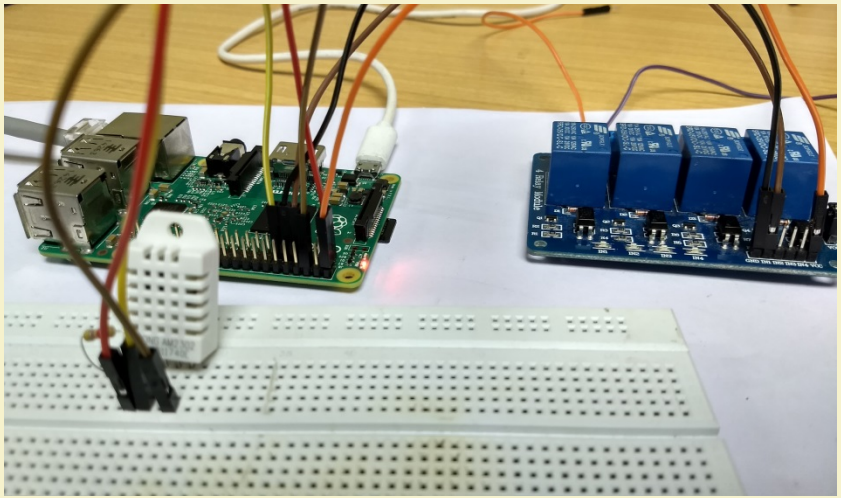
  GPIO.setup(13,GPIO.OUT)

- Set the relay pin high when the temperature is greater than 30

```
if temperature > 30:
    GPIO.output(13,0) # Relay is active low
    print('Relay is on')
    sleep(5)
    GPIO.output(13,1) # Relay is turned off after delay of 5 seconds
```

Connection: Relay (contd..)

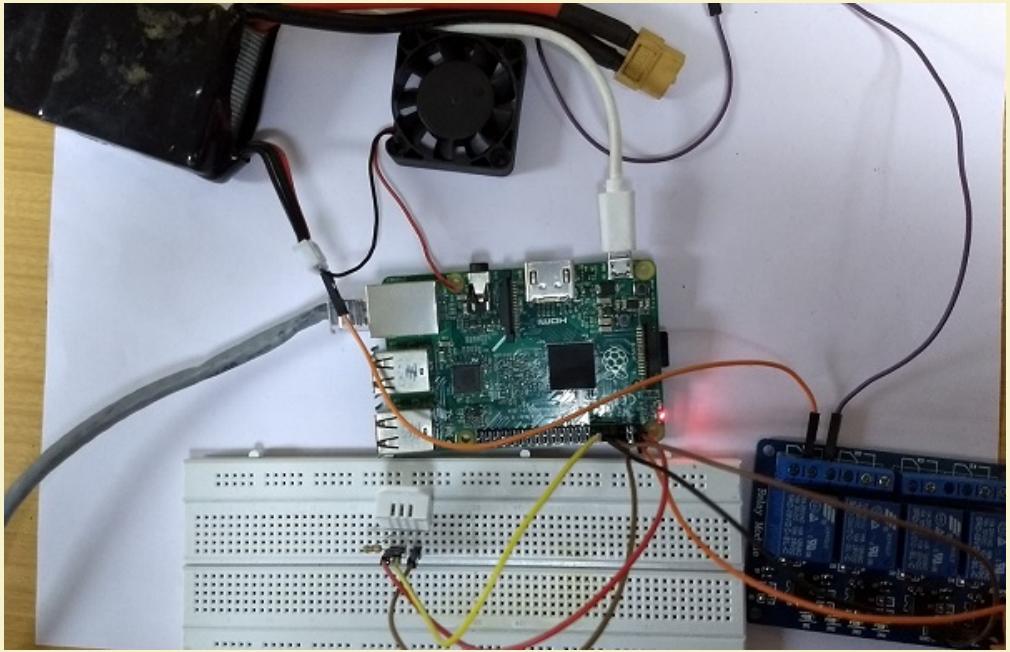
```
  GNU nano 2.2.6                                File: IOTSR.py

import RPi.GPIO as GPIO
from time import sleep

import Adafruit_DHT

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(7,GPIO.OUT)


sensor = Adafruit_DHT.AM2302  # create an instance of the sensor type

print ('Getting data from the sensor')

#humidity and temperature are 2 variables that store the values received from the sensor
humidity, temperature = Adafruit_DHT.read_retry(sensor,17)

print ('Temp={0:0.1f}*C humidity={1:0.1f}%'.format(temperature, humidity))
if temperature > 20:
    GPIO.output(7,0) # Relay is active low
    print('Relay is on')
    sleep(5)
    GPIO.output(7,1) # Relay is turned off after delay of 5 seconds
```



Connection: Fan

- Connect the Li-po battery in series with the fan
  - NO terminal of the relay -> positive terminal of the Fan.
  - Common terminal of the relay -> Positive terminal of the battery
  - Negative terminal of the battery -> Negative terminal of the fan.
- Run the existing code. The fan should operate when the surrounding temperature is greater than the threshold value in the sketch

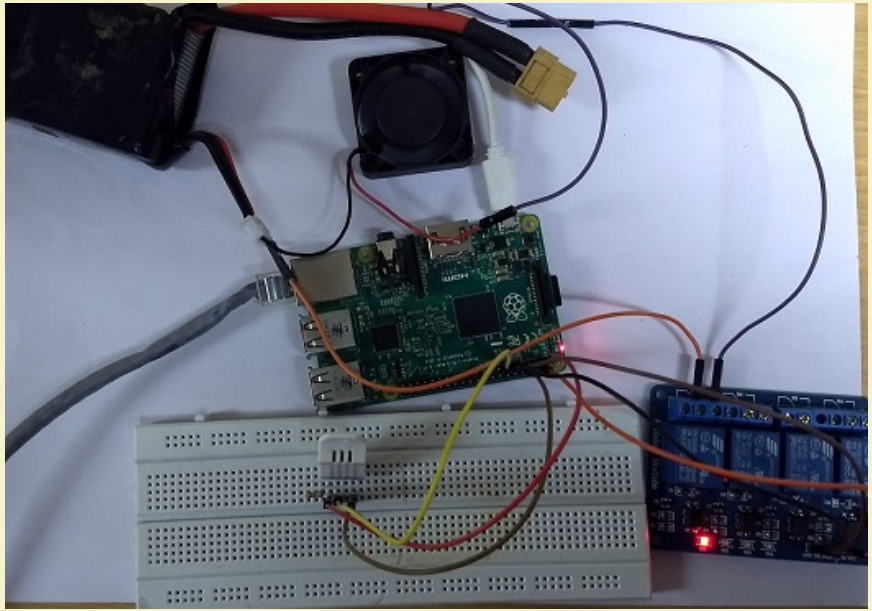
Connection: Fan (contd..)



Result

The fan is switched on whenever the temperature is above the threshold value set in the code.

Notice the relay indicator turned on.



Thank You!!