IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Implementation of IoT with Raspberry Pi: Part 2

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

IOT: Remote Data Logging

- Collect data from the devices in the network
- Send the data to a server/remote machine
- Control the network remotely

IOT: Remote Data Logging

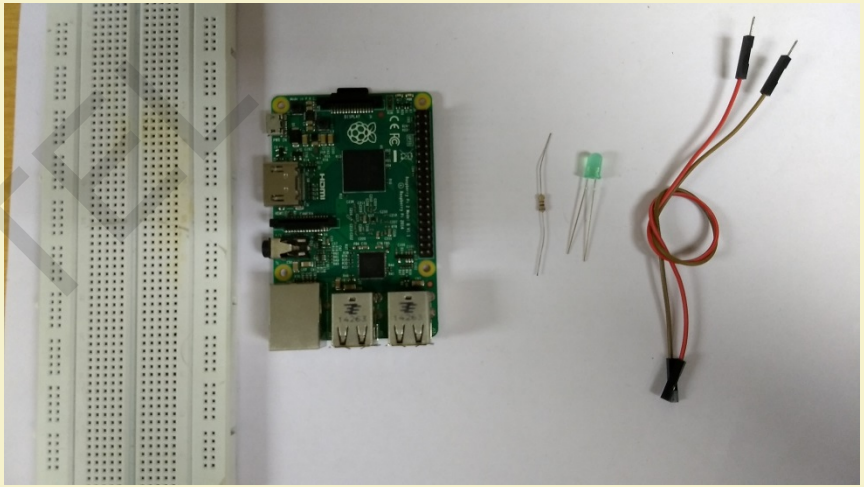
System Overview:

- A network of Temperature and humidity sensor connected with Raspberry Pi
- Read data from the sensor
- Send it to a Server
- Save the data in the server

IOT: Remote Data Logging (contd..)

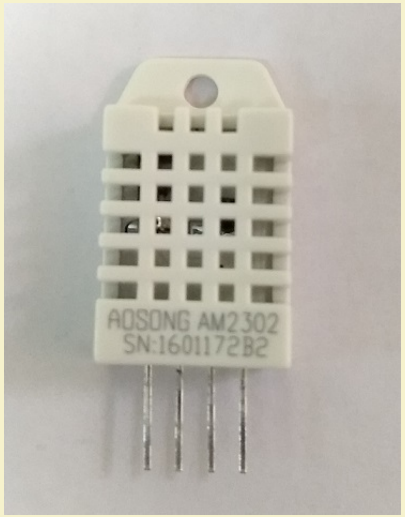
Requirements

- DHT Sensor
- 4.7K ohm resistor
- Jumper wires
- Raspberry Pi



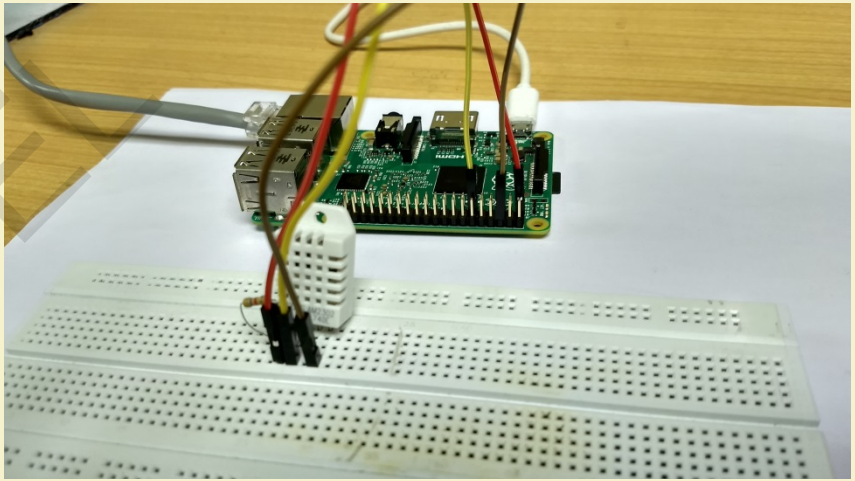
DHT Sensor

- 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



Sensor- Raspberry Pi Interface

- 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



Read Data from the Sensor

Adafruit provides a library to work with the DHT22 sensor

Install the library in Raspberry Pi

Use the function Adafruit_DHT.read_retry() to read data from the sensor

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

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:~ $
```

Sending Data to a Server

Sending data to Server using network protocols

- Create a server and client
- Establish connection between the server and the client
- Send data from the client to the server

Sending Data to a Server (contd..)

Socket Programming:

- Creates a two-way communication between two nodes in a network
- The nodes are termed as Server and Client
- Server performs the task/service requested by the client

Sending Data to a Server (contd..)

Creating a socket:

s = socket.socket (SocketFamily, SocketType, Protocol=0)

- ✓ SocketFamily can be AF_UNIX or AF_INET
- ✓ SocketType can be SOCK_STREAM or SOCK_DGRAM
- ✓ Protocol is set default to 0

Source: PYTHON NETWORK PROGRAMMING, TutorialsPoint

Sending Data to a Server (contd..)

Server:

```
s = socket.socket()             # creating a socket object
host = socket.gethostname()             # local machine name/address
port = 12321                    # port number for the server
s.bind((host, port))            # bind to the port
s.listen(5)                     # waiting for the client to connect
while True:
        c, addr = s.accept()                # accept the connection request from the client
        print ‘Connected to', addr
        c.send(‘Connection Successful')
c.close()                                   #close the socket
```

Source: PYTHON NETWORK PROGRAMMING, TutorialsPoint

Sending Data to a Server (contd..)

Client:

```
s = socket.socket()                  # creating a socket object
host = socket.gethostname()          # getting local machine name
port = 12345                         # assigning a port
s.connect((host, port))
print s.recv(1024)
s.close
```


Source: PYTHON NETWORK PROGRAMMING, TutorialsPoint

Sending Data to a Server (contd..)

Client Code: Obtain readings from the sensor

```
def sensordata():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)
    sensor = Adafruit_DHT.AM2302
    humidity, temperature = Adafruit_DHT.read_retry(sensor,17)
    return(humidity, temperature)
```

This function returns the values from the DHT sensor

Sending Data to a Server (contd..)

Client Code: Connecting to the server and sending the data

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)                    #create UDP socket
server_address = ('10.14.3.194', 10001)
try:
   while (1):
            h,t = sensordata()
            message = str(h)+','+str(t)
            #Send data
            print >>sys.stderr, 'sending "%s"' % message

            sent = sock.sendto(message, server_address)
finally:
   print >>sys.stderr, 'closing socket'
   sock.close()
```

Sending Data to a Server (contd..)

Server Code: Receive data from client and save it

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('10.14.3.194', 10001)
sock.bind(server_address)
while True:
   data, address = sock.recvfrom(4096)
   with open("Datalog.txt","a") as f:
      mess=str(data)
      f.write(mess)
      print mess
   f.close()
```

```
# Create a UDP socket

# Bind the socket to the port
```

Result

- The client takes reading from the sensor and sends it to the server
- The server receives the data from the client and saves it in a text file DataLog.txt

```
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
```

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Implementation of IoT with Raspberry Pi: Part 3

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

IOT: Remote Data Logging

- Collect data from the devices in the network
- Send the data to a server/remote machine
- Processing the data
- Respond to the network

IOT: Remote Data Logging

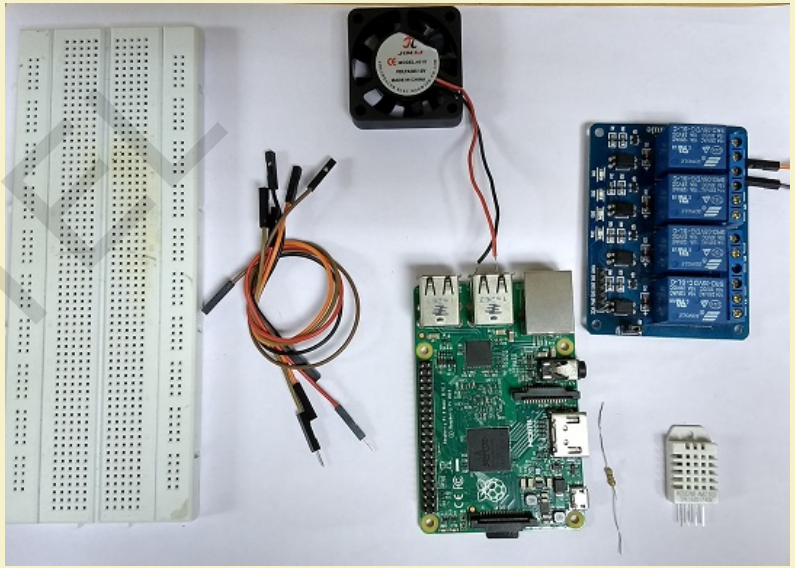
System Overview:

- A network of Temperature and humidity sensor connected with Raspberry Pi
- Read data from the sensor
- Send it to a Server
- Save the data in the server
- **Data Splitting**
- **Plot the data**

IOT: Remote Data Logging (contd..)

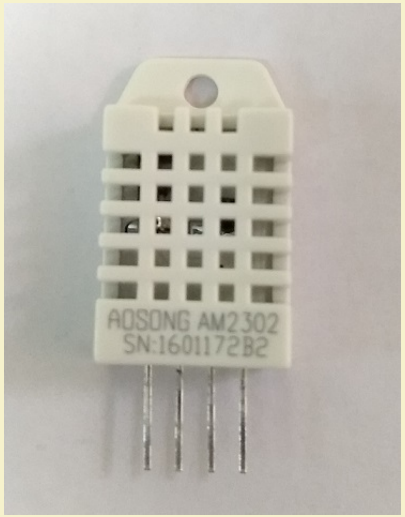
Requirements

- DHT Sensor
- 4.7K ohm resistor
- Jumper wires
- Raspberry Pi



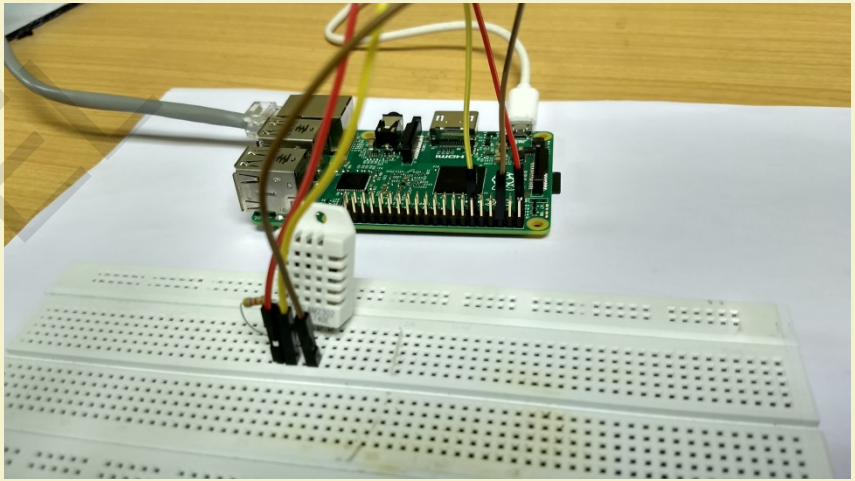
DHT Sensor

- 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



Sensor- Raspberry Pi Interface

- 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



Read Data from the Sensor

Use the Adafruit library for DHT22 sensor to read the sensor data

```
 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))
```

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

Sending Data to a Server

- Sending data to server using socket programming
  - Create a client and server
  - Establish connection between the two
  - Send data from the client to the server
  - Save the data in a file

Data Processing

Data from the client needs to be processed before it can be used further

- Data splitting/filtering
- Data plotting

Data Processing

Data splitting/filtering:

- Data from the client is saved in a text file
- The values are separated by a comma(‘ , ’)

message = str(h)+','+str(t)

- Split() function can be used to split a string into multiple strings depending on the type of separator/delimiter specified.

Example:

Data= ‘sunday,monday,tuesday’ #Data is a string with 3 words separated by a comma

Data.split(“,”) # split the data whenever a “,” is found

[‘sunday’,’monday’,’tuesday’] # Gives 3 different strings as output

Source: HOW TO USE SPLIT IN PYTHON, PythonForBeginners, Sep 26, 2012

Data Processing

Plotting the data:

- MATPLOTLIB is a python library used to plot in 2D
  - Plot(x,y): plots the values x and y
  - xlabel('X Axis'): Labels the x-axis
  - ylabel('Y Axis'): Labels the y-axis
  - title("Simple Plot"): Adds title to the plot

Source: MATPLOTLIB, John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team, 2012 - 2016

Data Processing (contd..)

Plotting the data:

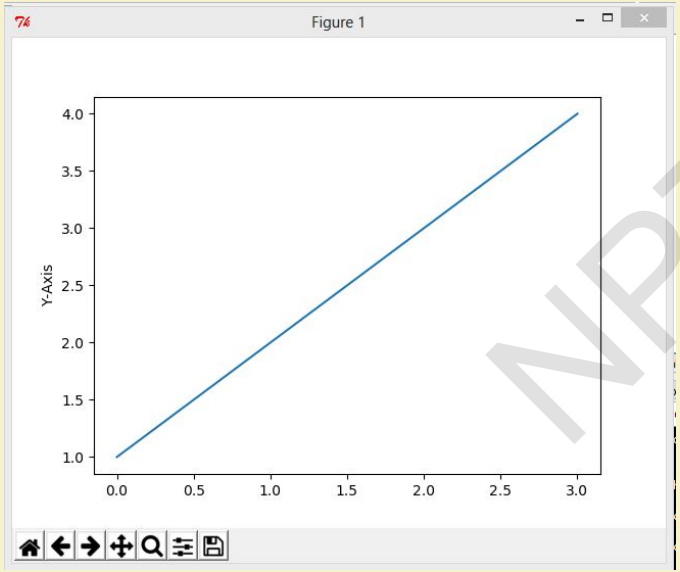
```
import matplotlib.pyplot as myplot
myplot.plot([1,2,3,4])
myplot.ylabel('Y-Axis')
myplot.show()
```

By default the values are taken for y-axis, values for x-axis are generated automatically starting from 0

Source: MATPLOTLIB, John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team, 2012 - 2016

Data Processing (contd..)

Basic Plot:




```
import matplotlib.pyplot as myplot
myplot.plot([1,2,3,4])
myplot.ylabel("Y-Axis")
myplot.show()
```

Data Proceessing (contd..)

Some other common functions used in plotting:

- figure(): Creates a new figure
- grid(): Enable or disable axis grids in the plot
- ion(): turns on the interactive mode
- subplot(): Adds subplot in a figure
- Close(): Close the current figure window
- Scatter(): make a scatter plot of the given points

Source: MATPLOTLIB, John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team, 2012 - 2016

Sending Data to a Server (contd..)

Client:

```
def sensordata():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)
    sensor = Adafruit_DHT.AM2302
    humidity, temperature =
Adafruit_DHT.read_retry(sensor,17)
    return(humidity, temperature)
```

```
sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)          #create UDP socket
server_address = ('10.14.3.194', 10001)
try:
  while (1):
      h,t = sensordata()
      message = str(h)+','+str(t)          #Send data
      print >>sys.stderr, 'sending "%s"' % message
      sent = sock.sendto(message, server_address)
finally:
  print >>sys.stderr, 'closing socket'
  sock.close()
```

Sending Data to a Server (contd..)

Server:

```
def coverage_plot(data,i):
    hum=data.split(",")[0]
    tem=data.split(",")[1]
    print 'temp='+(str(tem))+'iter='+str(i)
    plt.ion()
    fig=plt.figure(num=1,figsize=(6,6))
    plt.title(' IoT Temperature and Humidity Monitor')
    ax = fig.add_subplot(121)
    ax.plot(tem,i, c='r', marker=r'$\Theta$')
    plt.xlabel('Temp ($^0 C$)')
```

```
    ax.grid()
    ax = fig.add_subplot(122)
    ax.plot(hum,i, c='b', marker=r'$\Phi$')
    plt.xlabel('Humidity ($\%$)')
    ax.grid()
    fig.show()
    fig.canvas.draw()

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to the port
server_address = ('10.14.3.194', 10001)
sock.bind(server_address)
```

Sending Data to a Server (contd..)

Server:

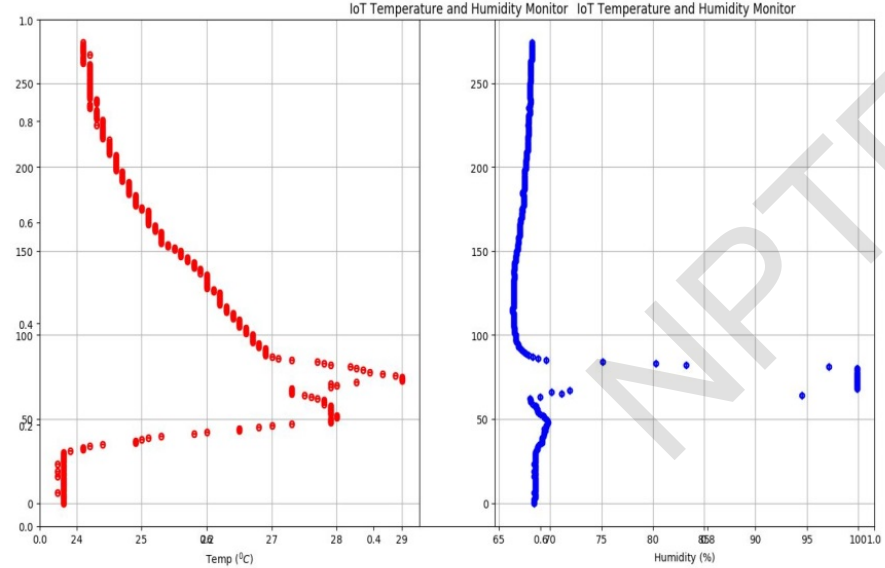
```
i=0
while True:
    data, address = sock.recvfrom(4096)
    with open("DataLog.txt","a") as f:
        mess=str(data)
        f.write(mess)
        coverage_plot(mess,i)
        print mess
        i+=1
    f.close()
```

Output

- The Reading from the sensor is sent to the Server and saved in a text file.
- Two different plots for temperature and humidity data

```
pi@raspberrypi:~ $ python client.py
sending "69.0,23.6000003815"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
sending "69.0,23.3999996185"
```

Output




```
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.8000030518,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.3999996185
68.9000015259,23.5
68.9000015259,23.5
68.9000015259,23.5
```

Thank You!!