GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY
Faculty of Engineering
Department of Electrical, Electronics and Telecommunication Engineering

ET4242 – Internet of Things

Practical 03 – Designing a simple Wireless Sensor & Actuator Networks using Raspberry

Aim: To Design simple Wireless Sensor & Actuator Networks using

Objective: To familiar with WSN designing using RPi Platform and OS
To familiar with fundamentals of data reading, process and Tx/Rx in WSN

Outcome: After completing this experiment you would be able to,

a) Hands-on experience in DHT11 Temperature and Humidity Sensor Module
b) Understand the Structure of a wireless sensor network
c) Understand the Structure of a wireless sensor node
d) Understand the Communication structure of a wireless sensor network

Apparatus:

Equipment Required:

- Breadboard
- DC Power Supply (Micro USB Port (5V @ 2A) - Raspberry Pi)
- Router with an internet connection
- Personal Computer / Smartphone

Components Required:

- Raspberry Pi 3 or better
- DHT11 Temperature and Humidity Sensor Module
- Jumper Cables

Theory:

- **Introduction**

Wireless Sensor Networks (WSNs) can be defined as a self-configured and infrastructure-less wireless network to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to the main location or sink where the data can be observed and analyzed. A sink or base station acts as an interface between users and the network. One can retrieve required information from the network by injecting queries and gathering results from the sink. Typically, a wireless sensor network contains hundreds of thousands of sensor nodes. The sensor nodes can communicate among themselves using radio signals. A wireless sensor node is equipped with sensing and computing devices, radio transceivers and power components. The individual nodes in a wireless sensor network (WSN) are inherently resource-constrained: they have limited processing speed, storage capacity, and communication bandwidth. After the sensor nodes are deployed, they are responsible for self-organizing an appropriate network infrastructure often with multi-hop communication with them. Then the onboard sensors start collecting information of interest. Wireless sensor devices also respond to queries sent from a "control site" to perform specific instructions or provide sensing samples

- **Structure of a wireless sensor network**
  - Star network
  - Mesh network
  - Hybrid star
- **Structure of a wireless sensor node**
  - A sensor node is made up of four basic components such as sensing unit, processing unit, transceiver unit and power unit. It also has application dependent additional components such as a location finding system, a power generator and a mobilizer. Sensing units are usually composed of two subunits: sensors and analogue to digital converters (ADCs)

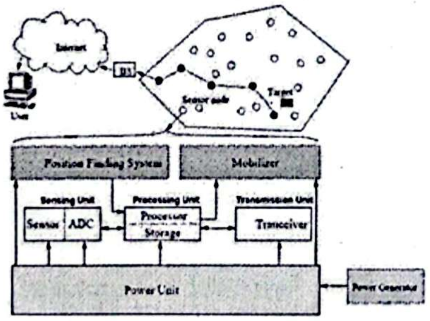


Figure 1

- **Communication structure of a wireless sensor network**

  - Scattered sensor nodes have the capabilities to collect data and route data back to the sink and the end-users. Data are routed back to the end-user by a multi-hop infrastructure-less architecture through the sink as shown in Figure 2 The sink may communicate with the task manager node via Internet or Satellite.

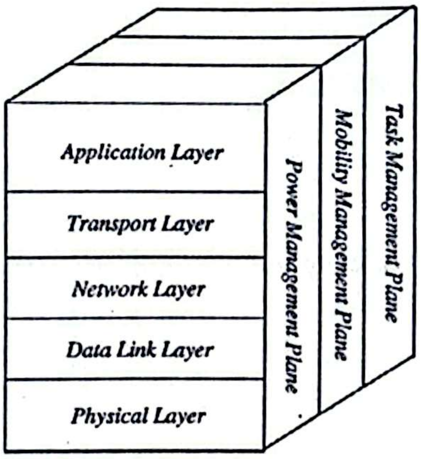
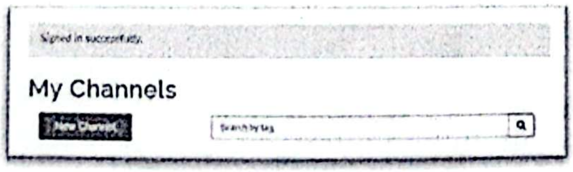


Figure 2

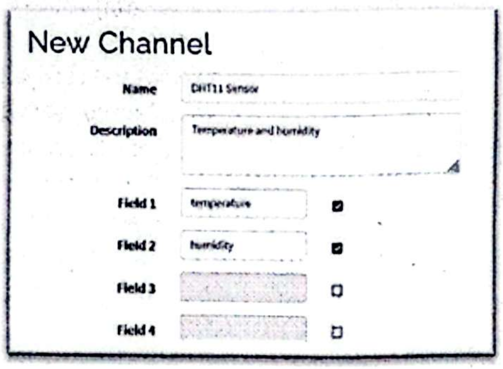
Procedure:

1. **Steps for building Raspberry Pi Data Logger on Cloud-** ThingSpeak is an open IoT platform for monitoring your data online. In the ThingSpeak channel, you can set the data as private or public according to your choice. ThingSpeak takes a minimum of 15 seconds to update your readings. It's a great and very easy to use platform for **building IoT projects.**

   a. Create an account by clicking the link. (https://thingspeak.com/) After creating the account, log in and click on New Channel to create a channel


Signed in successfully.

b. Define the Channel Name, Description and Field 1 as Temperature, Field 2 as Humidity and Save Channel



New Channel

Name: DHT11 Sensor

Description: Temperature and humidity

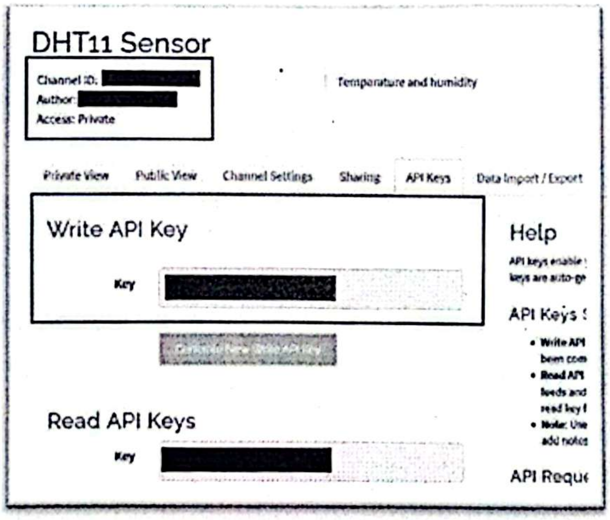
Field 1: temperature ☑

Field 2: humidity ☑

Field 3: □

Field 4: □

c. Go to the API Keys and Note down the ‘Channel ID’ and ‘Write API Keys’ which will use later in the main.py code



DHT11 Sensor

Channel ID: [redacted]
Author: [redacted]
Access: Private

Temperature and humidity

Write API Key

Key [redacted]

Read API Keys

Key [redacted]

2. Connection Diagram

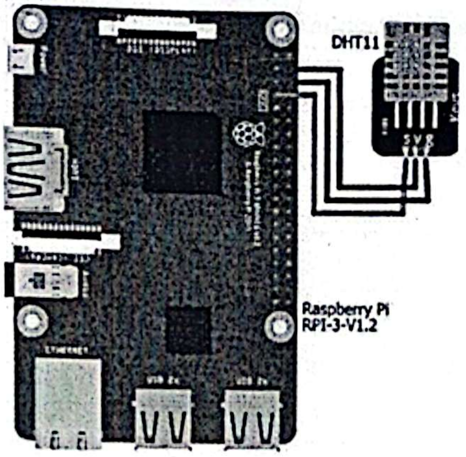


Figure 3

- Data pin of DHT11 sensor is connected with GPIO 4 with Raspberry Pi

The **DHT11** is a commonly used **Temperature and humidity sensor that comes with a dedicated NTC (Negative Temperature Coefficient) to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.**

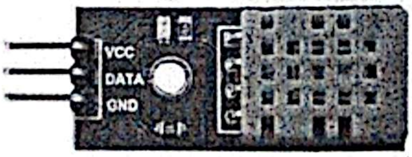


Figure 4

DHT11 Data Sheet
https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-

3. Installing Required Libraries

For installing the basic updates run these commands in a terminal window on your Raspberry Pi

I. Update the packages installed in Raspberry Pi.

```
sudo apt - get update
sudo apt - get install build - essential python - dev python - openssl git

sudo apt-get install python3-pip
sudo python3 -m pip install --upgrade pip setuptools wheel
```

II. Install the library to read the DHT11 sensor.

```
cd Adafruit_Python_DHT
sudo python3 setup.py install
```

```
Get the Library for the DHT11 Sensor
(https://github.com/adafruit/Adafruit_Python_DHT)
```

III. Installing Raspberry Pi Thingspeak Library

```
sudo pip install thingspeak
```

Code

```
import thingspeak

import time

import Adafruit_DHT


channel_id = (______________) # put here the ID of the channel you created before

write_key =  (______________) # update the "WRITE KEY"


pin = 4

sensor = Adafruit_DHT.DHT11


def measure(channel):

  try:

    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

    if humidity is not None and temperature is not None:

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

    else:

      print('Did not receive any reading from sensor. Please check!')

    # update the value

    response = channel.update({'field1': temperature, 'field2': humidity})

  except:

     print("connection failure")


if __name__ == "__main__":

    channel = thingspeak.Channel(id=channel_id, write_key=write_key)

    while True:

      measure(channel)

    #free account has a limitation of 15sec between the updates

      time.sleep(15)
```