Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 01 Date of Performance: ……

Title: Study of Raspberry-Pi, Beagle board, Arduino.

Objective: To study IoT platforms such as Raspberry-Pi/Beagle board/Arduino.

Theory:

Internet of Things: - IoT is short for **Internet of Things**. **The Internet of Things** refers to the ever-growing network of physical objects that feature an **IP** address for **internet** connectivity, and the communication that occurs between these objects and other **Internet**-enabled devices and systems. The Internet of Things (**IoT**) refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. In other **words**, the **IoT** (Internet of Things) can be called to any of the physical objects connected with network.

Examples of IoT: -

1) Apple Watch and Home Kit.
2) Smart Refrigerator.
3) Smart Refrigerator.
4) Smart cars.
5) Google Glass.
6) Smart thermostats.

A) Raspberry-Pi:- The **Raspberry Pi** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. It does not include peripherals (such as keyboards and mice). The **Raspberry Pi** is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The **Raspberry Pi** is a credit-card-sized computer that costs between $5 and $35. It's available anywhere in the world, and can function as a proper desktop computer or be **used** to build smart devices. A Raspberry Pi is a general-purpose computer, usually with a Linux **operating system**, and the ability to run multiple programs. Raspberry Pi is like the brain. Its primary advantage comes in processing higher level processing capability. It's a single board computer.

Fig.A.1: - **Raspberry-Pi**

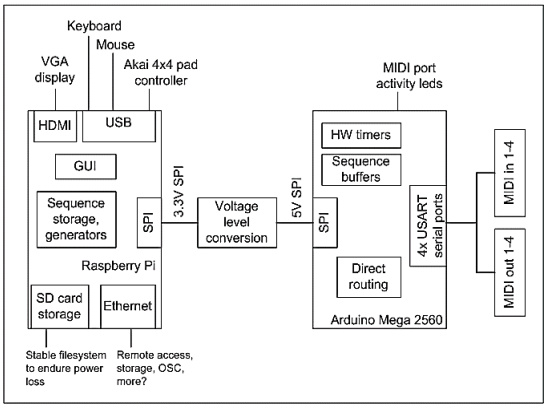


Fig. A.2: -**Raspberry-Pi Architecture**

Here are the various components on the Raspberry Pi board:

- **ARM CPU/GPU** -- This is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Video core 4 graphics processing unit (GPU). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.
- **GPIO** -- These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.
- **RCA** -- An RCA jack allows connection of analog TVs and other similar output devices.
- **Audio out** -- This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.
- **LEDs** -- Light-emitting diodes, for your entire indicator light needs.
- **USB** -- This is a common connection port for peripheral devices of all types (including your mouse and keyboard). Model A has one, and Model B has two. You can use a USB hub to expand the number of ports or plug your mouse into your keyboard if it has its own USB port.
- **HDMI** -- This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.
- **Power** -- This is a 5v Micro USB power connector into which you can plug your compatible power supply.
- **SD card slot** -- This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device. They are available for purchase from the manufacturers, but you can also download an OS and save it to the card yourself if you have a Linux machine and the wherewithal.
- **Ethernet** -- This connector allows for wired network access and is only available on the Model B.

B) **Beagle board:-** The **Beagle Board** is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The Beagle Board was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.[The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used. **Beagle Bone** Black is a low-cost,

open source, community-supported development platform for ARM® Cortex™-A8 processor developers and hobbyists. Boot Linux in under 10-seconds and get started on Sitara™ AM335x ARM Cortex-A8 processor development in less than 5 minutes with just a single USB cable.

Fig.B.1: -Beagle Board Black

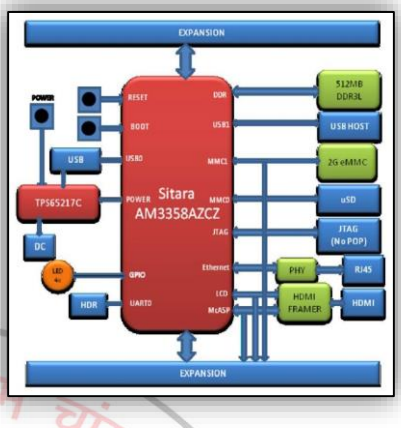


Fig.B.1: - Beagle Board Black architecture

Here are the various components on the Beagle board:

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

Software Compatibility

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ BoneScript library
- plus, much more

C) **Arduino:- Arduino** is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (*shields*) and other circuits. The boards feature serial

communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler tool chains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project. Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.

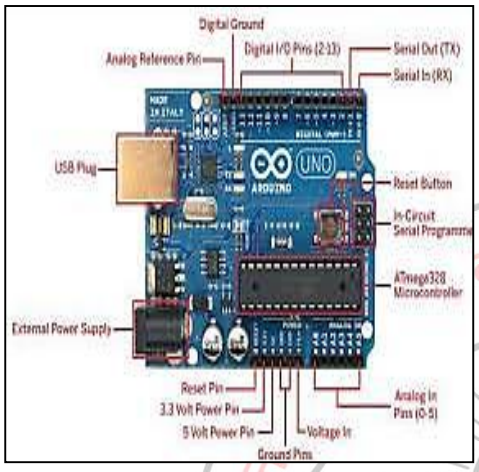


Fig.C.1: - Arduino Board

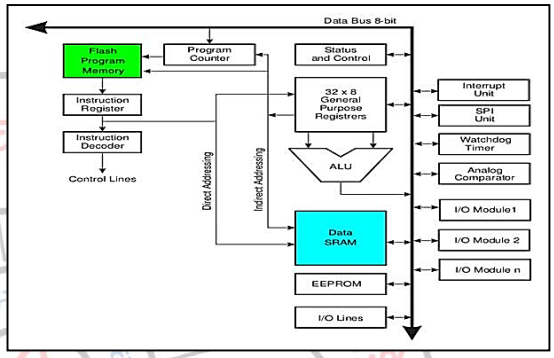


Fig.C.1: - Arduino Board Architecture.

Here are the various components on the Arduino board:

Microcontrollers

ATmega328P (used on most recent boards)

ATmega168 (used on most Arduino Diecimila and early Duemilanove)
ATmega8 (used on some older board)

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

Power Pins

- **VIN** (sometimes labelled "9V"). The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated

power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.

Other Pins

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - *These pins cannot be used for digital i/o (**digitalRead** and **digitalWrite**) if you are also using serial communication (e.g. **Serial.begin**).*
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

Conclusion: - Thus, we have studied of IoT platforms such as Raspberry-Pi/Beagle board/Arduino.

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 02 Date of Performance: ……

Title: Study of different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation on Raspberry-Pi/Beagle board/Arduino.

Objective: To study operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

Theory:

1) **Raspberry-Pi**: - The Pi can run the official Raspbian OS, Ubuntu Mate, Snappy Ubuntu Core, the Kodi-based media centers OSMC and LibreElec, the non-Linux based Risc OS (one for fans of 1990s Acorn computers). It can also run Windows 10 IoT Core, which is very different to the desktop version of Windows, as mentioned below.

- **OS which install on Raspberry-Pi**: Raspbian, Ubuntu MATE, Snappy Ubuntu, Pidora, Linutop, SARPi, Arch Linux ARM, Gentoo Linux, etc.

➢ **How to install Raspbian on Raspberry-Pi:**

Step 1: Download Raspbian

Step 2: Unzip the file. The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it.

Step 3: Write the disc image to your microSD card. Next, pop your microSD card into your computer and write the disc image to it. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these programs will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up. Once the disc image has been written to the microSD card, you're ready to go! Put that sucker into your Raspberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry.**

2) **BeagleBone Black**: - The **BeagleBone Black** includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others.

1. **Os which install on BeagleBone Black:** Angstrom, Android, Debian, Fedora, Buildroot, Gentoo, Nerves Erlang/OTP, Sabayon, Ubuntu, Yocto, MINIX 3

- **How to install Debian on BeagleBone Black:**

Step 1: Download Debian img.xz file.

Step 2: Unzip the file.

Step 3: Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.

Step 4: Now open Win32 Disk imager, click the blue folder icon, navigate to the debian img location, and double click the file. Now click Write and let the process complete. Depending on your processor and available RAM it should be done in around 5 minutes.

Step 5: Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insert the uSD into the BBB.

Step 6: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes, I just did it again in around 5 minutes. Your mileage will vary. Go back and surf reddit some more.

Step 7: If you are not seeing the leds swing back and forth you will need to unplug the USB cable, press and hold down the user button above the uSD card slot (next to the 2 little 10 pin ICs) then plug in the USB cable. Release the button and wait. You should see the LEDs swinging back and forth after a few seconds. Once this happens it's waiting time. When all 4 LEDs next to the USB slot stay lit at the same time the flash process has been completed.

Step 8: Remove the uSD card and reboot your BBB. You can reboot the BBB by removing and reconnecting the USB cable, or hitting the reset button above the USB cable near the edge of the board.

Step 9: Now using putty, or your SSH flavor of choice, connect to the BBB using the IP address 192.168.7.2. You'll be prompted for a username. Type root and press Enter. By default, there is no root password. I recommend changing this ASAP if you plan on putting your BBB on the network. To do this type password, hit enter, then enter your desired password. You will be prompted to enter it again to verify.

3) **Arduino**: - The Arduino itself has no real operating system. You develop code for the Arduino using the Arduino IDE which you can download from Arduino - Home. Versions are available for **Windows**, **Mac** and **Linux**. The Arduino is a constrained microcontroller.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, **used** to write and upload computer code to the physical board. You are literally writing the "firmware" when you write the code and upload it. It's both good and its bad.

Conclusion: - Thus, we have studied of how to install operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 03 Date of Performance: ……

Title: Open source prototype platform- Raspberry-Pi/Beagle board/Arduino - Simple program digital read/write using LED and Switch - Analog read/write using sensor and actuators.

Objective: To get knowledge for communicating with objects using sensors and actuators.

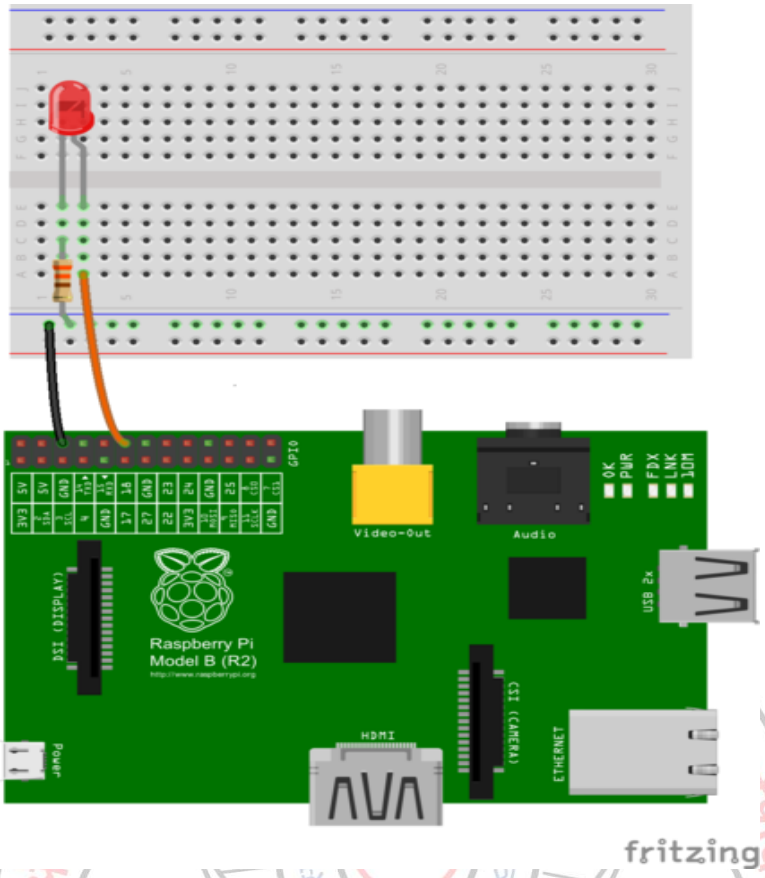
Theory:

An application of blinking LEDs using Raspberry Pi

In this assignment, we will learn how to interface an LED with Raspberry Pi. To implement this, we will gather the requirements and follow the instructions given below.

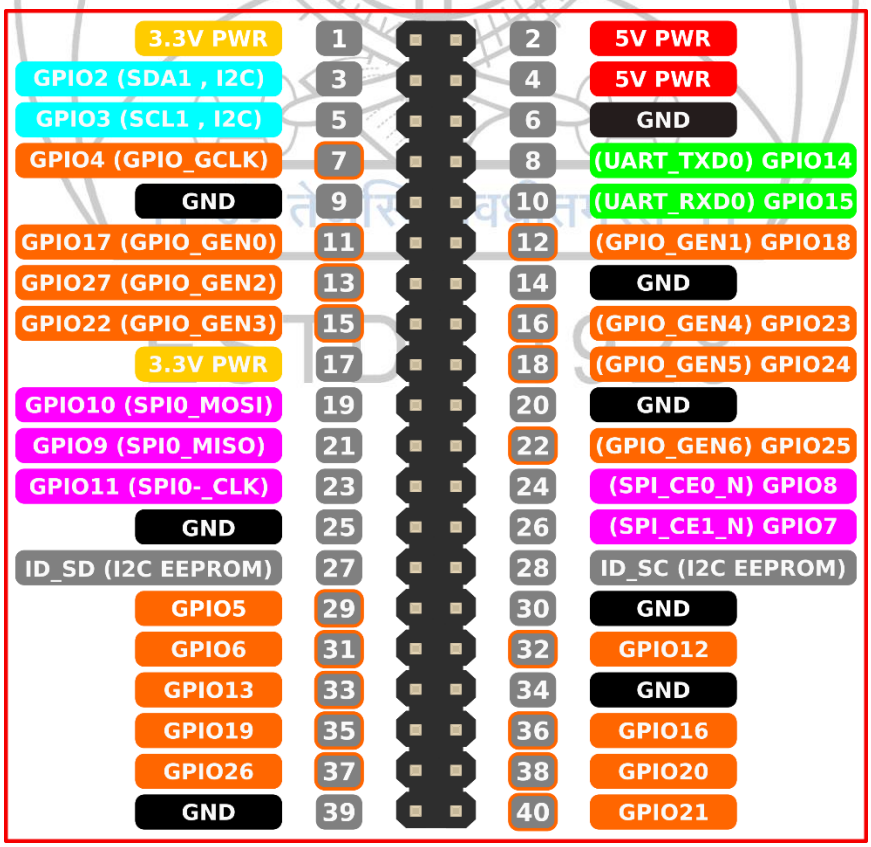
Requirements:

1. Raspberry Pi3 Kit
2. Breadboards
3. LEDs
4. Jumper wires



Instructions to Glow LED:

Use GPIO diagram of Raspberry Pi, which will give you understanding of GPIO structure.



Algorithm:

1. Connect GPIO 18 (Pin No. 12) of Raspberry Pi to the Anode of the LED through connecting jumper wires and Breadboard.
2. Connect Ground of Raspberry Pi to Cathode of the LED through connecting wires and breadboard.
3. Now power up your Raspberry Pi and boot.
4. Open terminal and type *nano blinkled.py*
5. It will open the nano editor. Use following pseudo code in the python to blink LED and save

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
while True:
    GPIO.output(18, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    time.sleep(1)
```

6. Now run the code. Type *python blinkled.py*
7. Now LED will be blinking at an interval of 1s. You can also change the interval by modifying time.sleep in the file.
8. Press Ctrl+C to stop LED from blinking.

An application to read the environment temperature and humidity & display it.

In this assignment, we will learn how to find environment temperature and humidity using DHT11 sensor and Raspberry Pi.

Temperature and Humidity Sensor- DHT11

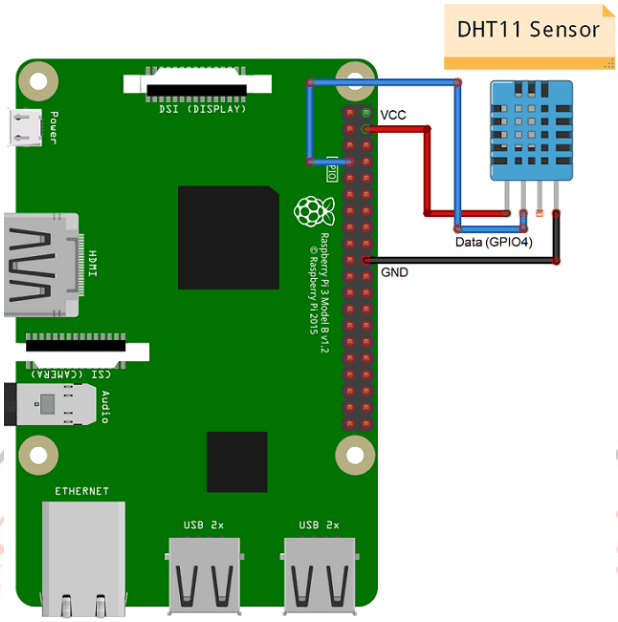
- It's a very basic and slow sensor, easy to use for small scale.
- The DHT sensors are made of two parts, a capacitive humidity sensor and thermistor.
- There is also a very basic chip inside that does some analog to digital conversion and spits out a digital signal with the temperature and humidity.
- The digital signal is fairly easy to read using any microcontroller.
- These sensors are frequently used in remote weather stations, soil monitors, and home environment control systems.

Features of DHT11 Sensor: -

- Ultra-low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20–80% humidity readings with 5% accuracy
- Good for 0–50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

Requirements:

1. Raspberry Pi3 Kit
2. Breadboards
3. DHT11 Sensor
4. Jumper wires



Algorithm:

1. Connect GPIO 18 (Pin No. 12) of Raspberry Pi to the Data pin of DHT11 sensor through connecting jumper wires and Breadboard.
2. Connect Power (5V) and Ground of Raspberry Pi to DHT11 sensor through connecting wires and breadboard.
3. Now power up your Raspberry Pi and boot.
4. Download Adafruit_Python_DHT.tar.gz package
5. Open terminal and type *tar -xvzf Adafruit_Python_DHT.tar.gz* command to extract Adafruit package.
6. In the same terminal, type *nano temp.py*
7. It will open the nano editor. Use following pseudo code in the python to blink LED and save

```
import Adafruit_DHT as dht
import time

while True:
        hum, temp = dht.read_retry(11,18)
        print "Temp: ", temp
        print "Hum: ", hum
        time.sleep(1)
```

8. Now run the code. Type *python temp.py*
9. Now temperature and humidity will be displayed at an interval of 1s. You can also change the interval by modifying time.sleep in the file.
10. Press Ctrl+C to stop displaying temperature and humidity.

Conclusion: - Thus, we have studied how to monitor LEDs and measure temperature and humidity in the environment using DHT11 sensor and Raspberry Pi 3.

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 04

Date of Performance: ……

Title: Upload data from environmental sensor to cloud server (You can use any public cloud IBM Watson IoT cloud or Google or AWS etc.)

Objective: To explore cloud environment for IoT.

Theory:

- ThingSpeak is an open data platform for monitoring your data online. You can set the data as private or public depending on your choice. ThingSpeak takes minimum of 15 seconds to update your readings. It's a great platform for building your IOT projects.
- We will read the temperature and humidity from the DHT22 and then we will send it to the API of the ThingSpeak channel. We will get the API after creating the channel.
- **Temperature sensor:** It is a device, a thermocouple or RTD, that provides temperature measurement through an electrical signal.
- **Thermocouple:** It is made from two dissimilar metals that generate electrical voltage in direct proportion to changes in temperature. The wires are joined together to form measuring junction and reference junction.
- **RTD:** Resistor temperature detection is variable resistor that will change its electrical resistance in direct proportion to changes in temperature in precise, repeatable & linear manner.

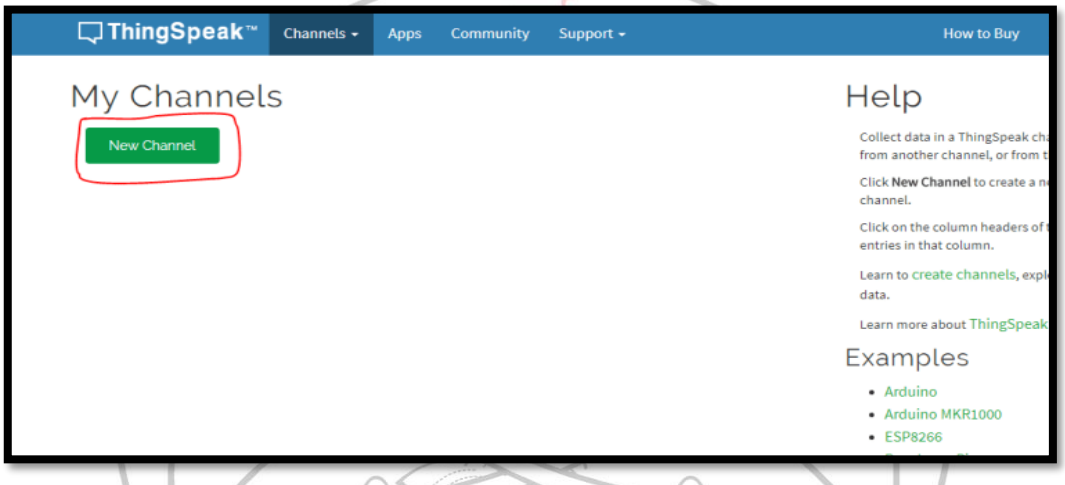
Components Required:

- Raspberry Pi 3
- DHT22
- 10k Resistor
- Jumper cables

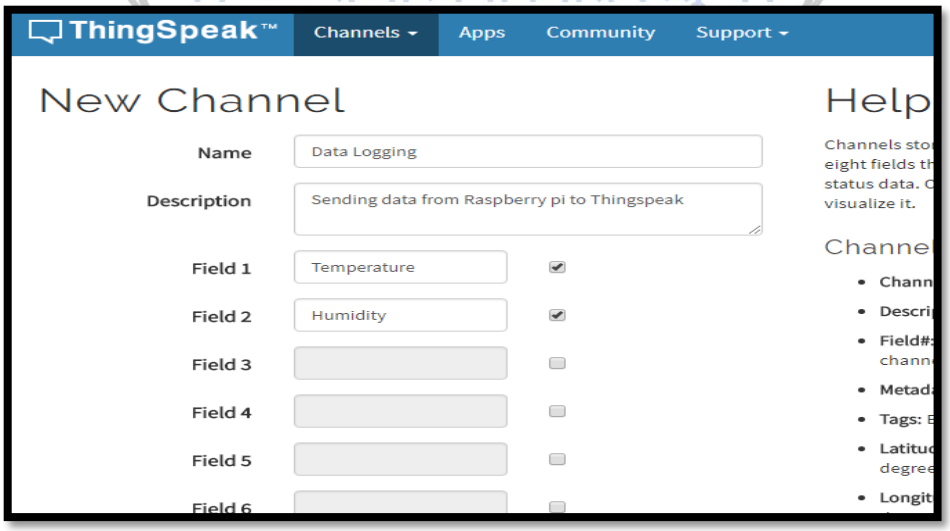
Setting up the ThingSpeak Account:

1. First of all, go to the following link and sign up to ThingSpeak. If you already have an account, then sign in. https://thingspeak.com/

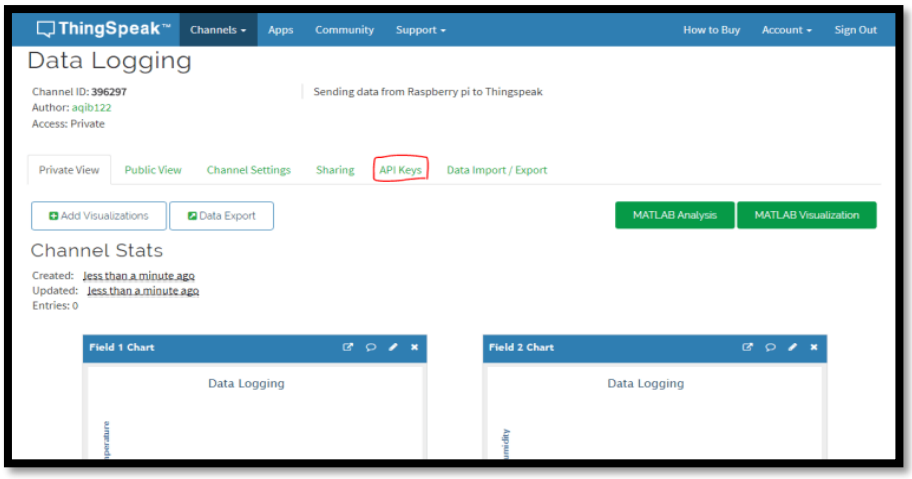
2. After creating the account or logging in, click on new channel.



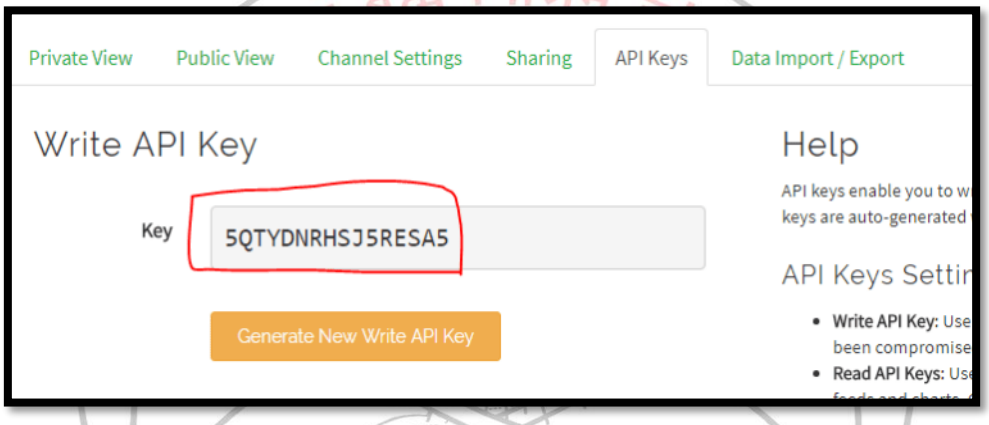
3. Fill the information about the channel. Select two fields because we will be sending the data for the two fields from the raspberry pi. Leave the other information as it is and save the channel.



4. Go to the API keys tab.



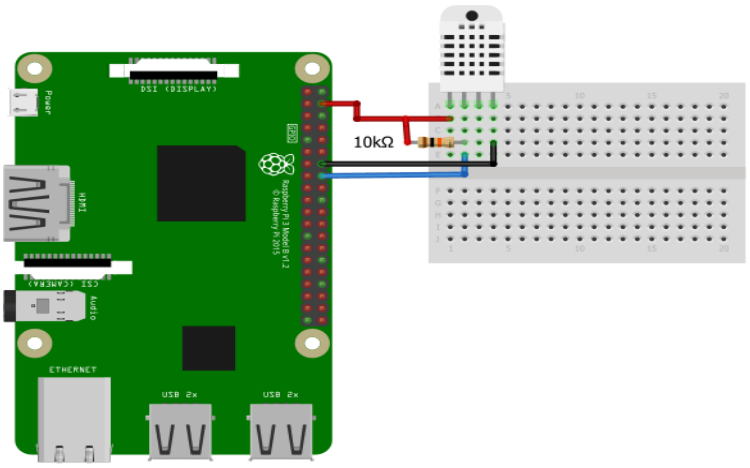
5. In the API keys tab, copy the write API key. This is the API key at which we will send the data from the Raspberry Pi.



Circuit Diagram and Explanation:

Connect DHT22 with the Raspberry pi as described below

DHT22 Pin	Raspberry pi
VCC	5V
Data Pin	Connect to GPIO 23 and also connect to 5V through 10K resistor
GND	GND



Installing the DHT22 Library:

1. Enter the below command to clone the library
   git clone https://github.com/adafruit/Adafruit_Python_DHT.git
2. Then enter in to the installed directory using the below command
   cd Adafruit_Python_DHT
3. Now download the required modules using the below command
   sudo apt-get install build-essential python-dev
4. Then install the library using the below command
   sudo python setup.py install

Raspberry Pi Pseudo Code

```
import sys
import urllib2
from time import sleep
import Adafruit_DHT as dht
# Enter Your API key here
myAPI = '5QTYDNRHSJ5RESA5'
# URL where we will send the data, Don't change it
baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI
def DHT22_data():
# Reading from DHT22 and storing the temperature and humidity
humi, temp = dht.read_retry(dht.DHT22, 23)
return humi, temp
while True:
try:
humi, temp = DHT22_data()
# If Reading is valid
ifisinstance(humi, float)andisinstance(temp, float):
# Formatting to two decimal places
humi = '%.2f' % humi
temp = '%.2f' % temp
# Sending the data to thingspeak
conn = urllib2.urlopen(baseURL + '&field1=%s&field2=%s' % (temp, humi))
print conn.read()
# Closing the connection
conn.close()
else:
print 'Error'
```

```
28. # DHT22 requires 2 seconds to give a reading, so make sure to add delay of
    above 2 seconds.
29. sleep(20)
30. except:
31. break
```

After running it, output looks like below

Outcome: - Thus, we have studied how to interface cloud for uploading temperature and humidity in the environment measured using DHT22 sensor and Raspberry Pi 3.

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 05 Date of Performance: ……

Title: Introduction to MQTT/ CoAP and sending sensor data to cloud using Raspberry-Pi/Beagle board/Arduino.

Objective: To provide knowledge for IoT related protocols such as MQTT / CoAP etc.

Theory:

- MQTT (Message Queue Telemetry Transport) is a lightweight messaging protocol which is ideal for communication of IoT connected devices.
- MQTT has three components: broker, publisher, and subscriber. A broker is an intermediary entity that handles the communication going on between devices. A publisher is a device that sends messages. A subscriber listens to the messages sent by the publisher.

Client –

- A program or device that uses MQTT.
- A Client always establishes the Network Connection to the Server.
- It can Publish Application Messages that other Clients might be interested in.
- Subscribe to request Application Messages that it is interested in receiving.

Server –

- A program or device that acts as an intermediary between Clients which publish Application Messages and Clients which have made Subscriptions. A Server
- Accepts Network Connections from Clients.
- Accepts Application Messages published by Clients.

Publish/Subscribe - The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. Many clients may subscribe to the same topics and do with the information as they please. The broker and MQTT act as a simple, common interface for everything to connect to. This means that you if you have clients that dump subscribed messages to a database, to Twitter, Cosm or even a simple text file, then it becomes very simple to add new sensors or other data input to a database, Twitter or so on. Temperature upload over MQTT using Raspberry Pi and DHT22 sensor and sending data to ThingsBoard platform.

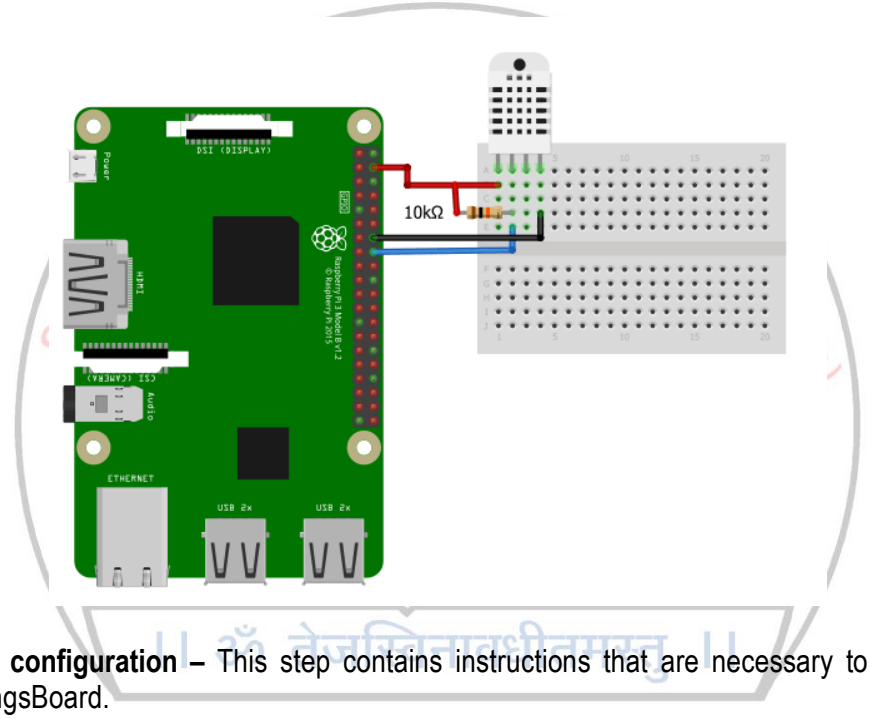
ThingsBoard –

- It is an open-source server-side platform that allows you to monitor and control IoT devices. It is free for both personal and commercial usage and you can deploy it anywhere.
- This sample application performs collection of temperature and humidity values produced by DHT22 sensor and further visualization on the real-time web dashboard. Collected data is pushed via MQTT to ThingsBoard server for storage and visualization.
- The DHT22 sensor is connected to Raspberry Pi. Raspberry Pi offers a complete and self-contained Wi-Fi networking solution. Raspberry Pi push data to ThingsBoard server via MQTT

protocol by using paho mqtt python library. Data is visualized using built-in customizable dashboard. The application that is running on Raspberry Pi is written in Python.

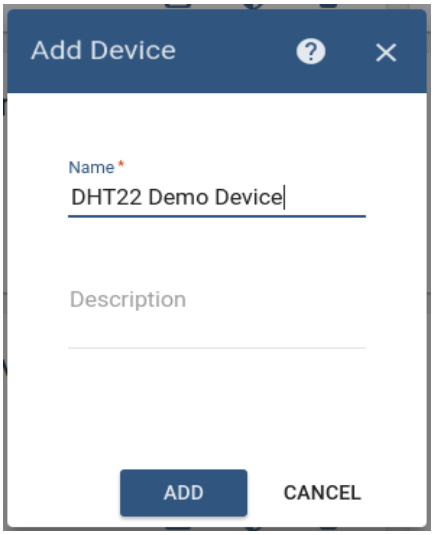
Circuit Diagram and Explanation – Make the connections of the DHT22 with the Raspberry pi as described below:

DHT22	**Raspberry pi**
VCC	5v
Data pin	Connect to GPIO 23 and also connect to 5V through 10K resistor
GND	GND

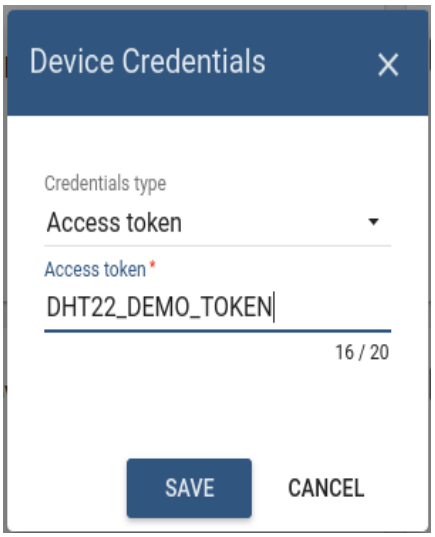


ThingsBoard configuration – This step contains instructions that are necessary to connect your device to ThingsBoard.

- Open ThingsBoard Web UI (http://localhost:8080) in browser and login as tenant administrator
- login: tenant@thingsboard.org
- password: tenant
- Goto “Devices” section. Click “+” button and create device with name “DHT22 Demo Device”.



Once device created, open its details and click “Manage credentials”. Copy auto-generated access token from the “Access token” field. Please save this device token. It will be referred to later as **$ACCESS_TOKEN**.



Click “Copy Device ID” in device details to copy your device id to the clipboard. Paste your device id to some place, this value will be used in further steps.

Download the dashboard file using this link. Use import/export instructions to import the dashboard to your ThingsBoard instance.

MQTT library installation – The following command will install MQTT Python library:

```
sudo pip install paho-mqtt
```

Adafruit DHT library installation –

Install python-dev package:

```
sudo apt-get install python-dev
```

Downloading and install the Adafruit DHT library:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install
```

Pseudo Code –

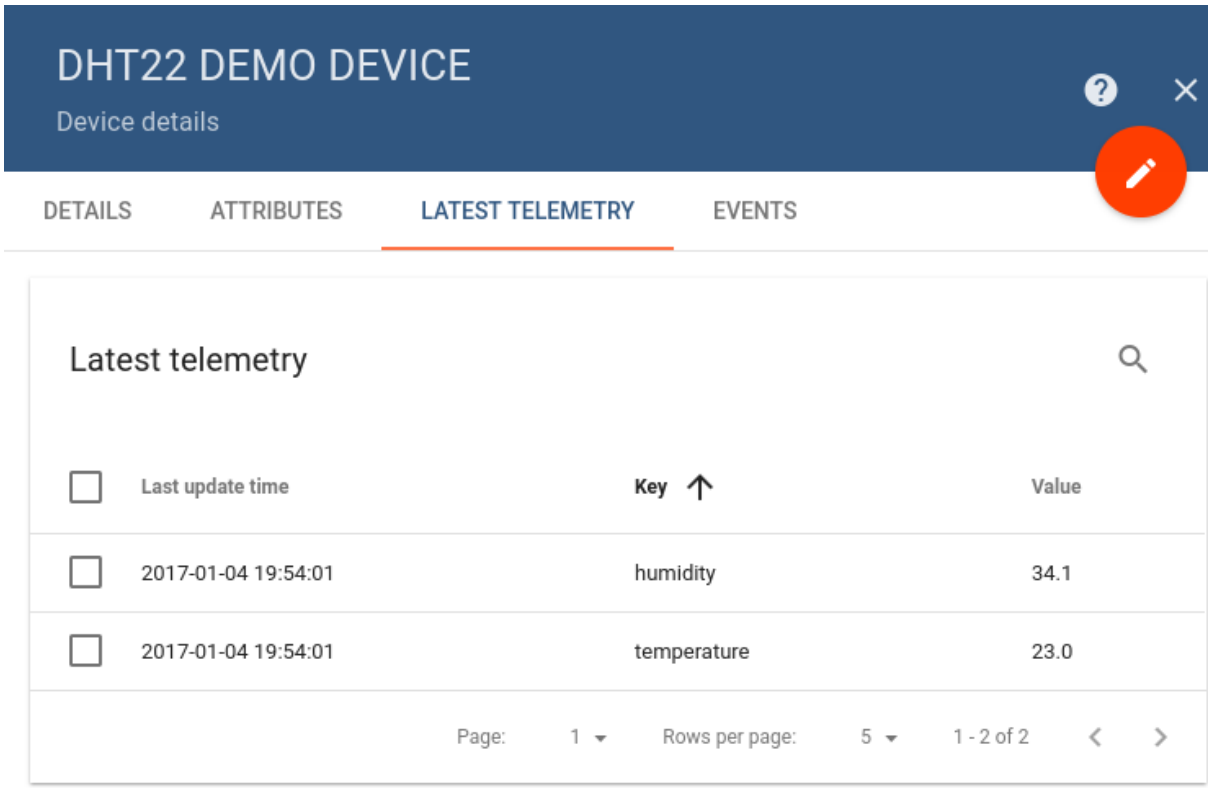
```
import os
import time
import sys
import Adafruit_DHT as dht
import paho.mqtt.client as mqtt
import json

THINGSBOARD_HOST='demo.thingsboard.io'
ACCESS_TOKEN='DHT22_DEMO_TOKEN'

# Data capture and upload interval in seconds. Less interval will eventually hang the DHT22.
INTERVAL=2

sensor_data={'temperature':0,'humidity':0}

next_reading=time.time()

client=mqtt.Client()

# Set access token
client.username_pw_set(ACCESS_TOKEN)

# Connect to ThingsBoard using default MQTT port and 60 seconds keepalive interval
client.connect(THINGSBOARD_HOST,1883,60)

client.loop_start()

try:
while True:
humidity,temperature=dht.read_retry(dht.DHT22,4)
humidity=round(humidity,2)
temperature=round(temperature,2)
print(u"Temperature: {:g}\u00b0C, Humidity: {:g}%".format(temperature,humidity))
sensor_data['temperature']=temperature
sensor_data['humidity']=humidity

# Sending humidity and temperature data to ThingsBoard
client.publish('v1/devices/me/telemetry',json.dumps(sensor_data),1)

next_reading+=INTERVAL
sleep_time=next_reading-time.time()
ifsleep_time>0:
time.sleep(sleep_time)
except KeyboardInterrupt:
pass

client.loop_stop()
client.disconnect()
```

Running the application – This simple command will launch the application:

```
python mqtt-dht22.py
```

Data visualization –

- Finally, open ThingsBoard Web UI. You can access this dashboard by logging in as a tenant administrator.
- Go to **"Devices"** section and locate **"DHT22 Demo Device"**, open device details and switch to **"Latest telemetry"** tab. If all is configured correctly you should be able to see latest values of *"temperature"* and *"humidity"* in the table.



Outcome: - Thus, we have studied how to interface cloud for uploading the environment properties measured with the help of DHT22 sensor and Raspberry Pi 3 using MQTT/CoAP.

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 06 Date of Performance: ……

Title: Design a web interface to control connected LEDs remotely using Raspberry-Pi/Beagle board/Arduino.

Objective: To design the web interface for IoT.

Theory: One of the function of IOT is remote control of devices, being able to trigger action from a remote location. This is otherwise known as remote configuration. Whether it's a home, an office, or an industrial site, connected devices rely on some form of automation through sensors or machines and need a mechanism to control their operation remotely. Here we will build a small IOT simulation using RPi and showcase the most common operation switching on and off a remote device.

Hardware Requirements:

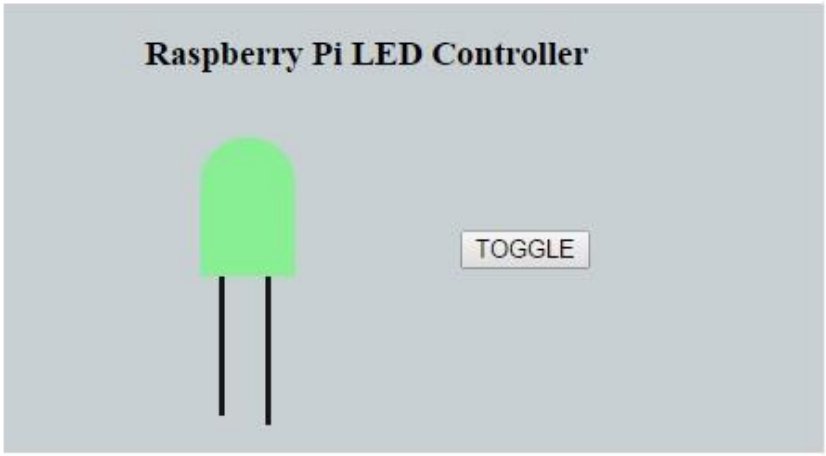
- RPi model with Raspbian OS
- 1 LED AND 1490OHM Resistor

Software Requirements:

- RPi GPIO, GPIO Python library for Raspbian OS.
- Pubnub Python SDK
- Pubnub JavaScript SDK

We will create 2 subsystems. One will be a controller, basic web application in the form of a webpage which can display the current states of device and send control messages to it and second one is the actual device simulated as on LED and controlled via Raspberry Pi.

Web application: The web interface looks like this:



This is a very simple webpage with a visual indicator for the device and a button to toggle ON/OFF state of LED. In the background, we have pubnub JavaScript API that performs 2 operations upon receiving certain events.

1. Sends a request to toggle the state of device.
2. Receives response with current state of the device.

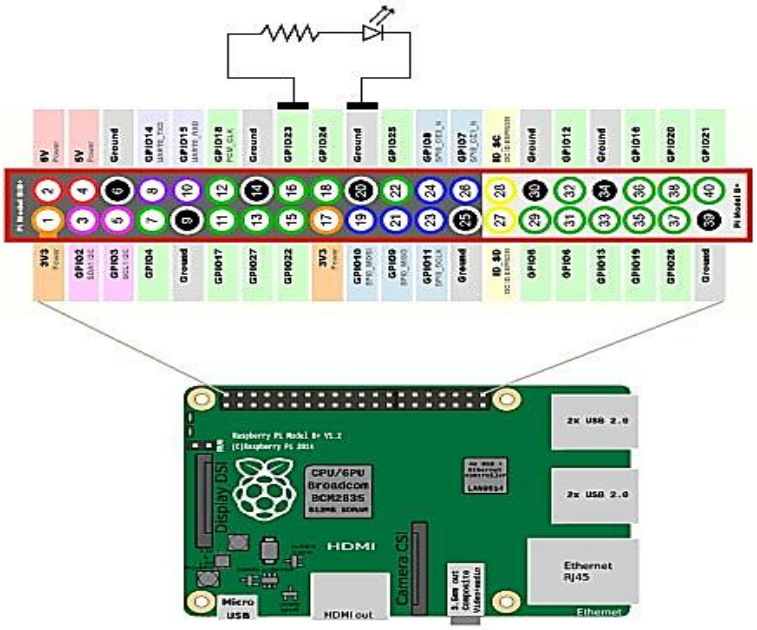
Button click event: When the toggle button is clicked, webpage sends a TOGGLE request message to the device via 'gpio-raspberry-control' channel.

```
$('#toggle').click(function(e){
        pubmsg = {"req" : "toggle"};
        pubnub.publish(
                {
                        channel: 'gpio-raspberry-control',
                        message:  pubmsg
                }
        );
});
```

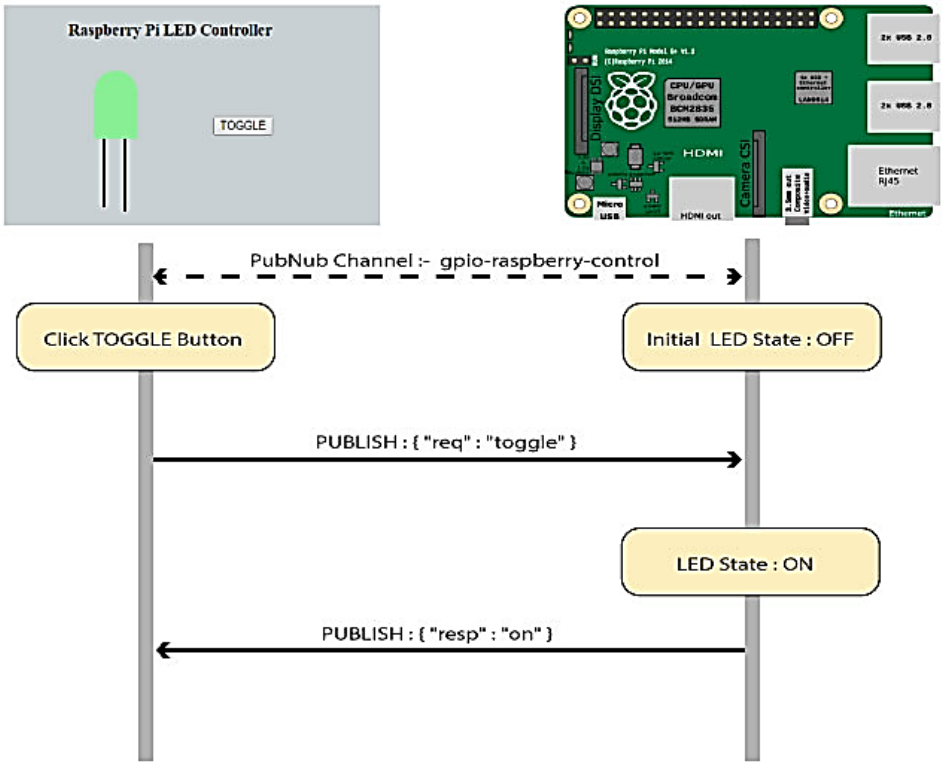
PubNub Channel Subscribe Callback Event: On receiving the toggle request, the Raspberry Pi toggles the state of the LED and sends a response back to the web page with the current state. Web page updates the visual indicator for the LED based on the received state information.

```
pubnub.subscribe({
        channel: 'gpio-raspberry-control',
        message: function(m){
                console.log(m)
                if('resp' in m) {
                        if('on' ==  m['resp']){
                                $('#led').removeClass('dim');
                                $('#led').addClass('glow');
                        } else {
                                $('#led').removeClass('glow');
                                $('#led').addClass('dim');
                        }
                }
        }
});
```

RPi and LED: Here is the schematic for the raspberry Pi connections to be used in this application.



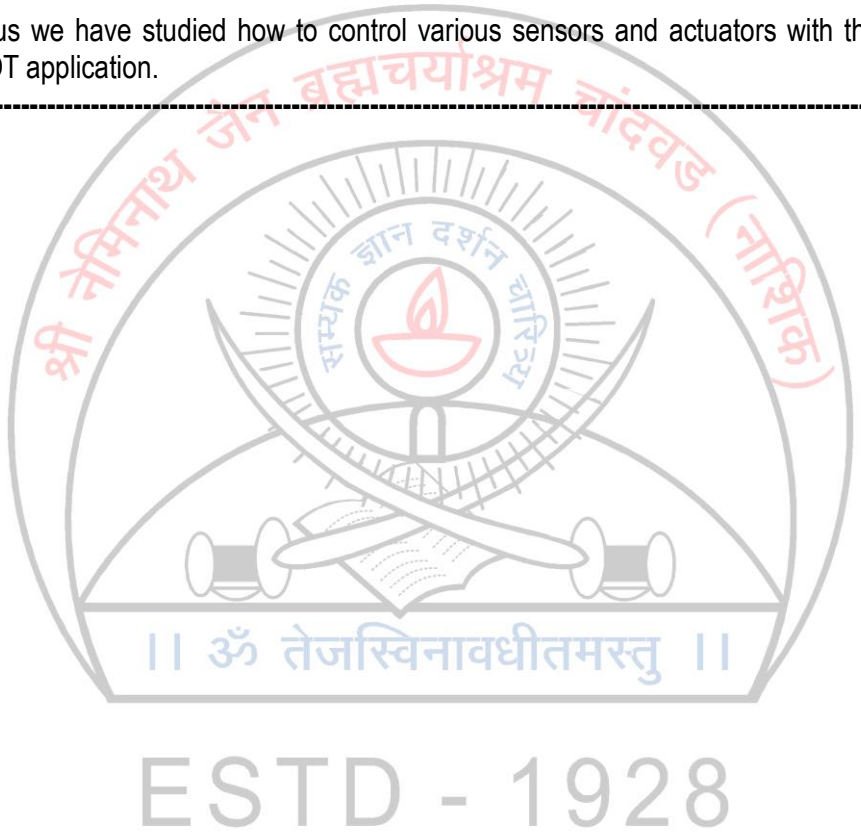
We are going to use the Raspberry Pi GPIO Python library to send the control messages to Raspberry Pi GPIO ports. This library works well with the python environment available by default with Raspbian OS. When a toggle request is received, the application checks the current state of the GPIO driving pin of the LED, toggles its state and then sends the new state back to the web application as a response message. The exchange of messages between the web application and Raspberry Pi can be visualized as follows.




```
glow = False
#PubNub Channel Subscribe Callback
def gpioCallback(msg,channel):
```

```
global glow
respstring = ''
command = msg
print "Command is: " + str(command)
if ('req' in command):
        if(command['req'] == 'toggle'):
                if(glow):
                        glow = False;
                        respstring = 'off'
                else:
                        glow = True
                        respstring = 'on'
                GPIO.output(16, glow)
                respmsg = {"resp": respstring}
                pubnub.publish(pubnubChannelName, respmsg)
```

Outcome: Thus we have studied how to control various sensors and actuators with the help of web interface for IOT application.



Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 07 Date of Performance: ……

Title: Install, configure XMPP server and deployed an application on Raspberry Pi/Beagle board/Arduino. Write client applications to get services from the server application.

Objective: To study installation & configuration of XMPP server for client-server application on Raspberry Pi/Beagle board/Arduino.

Theory:

Why a Raspberry Pi as a web server?

First, from an economic point of view, you should know that web hosting services are not free and that you have to pay every month / year. Unlike the Raspberry who just need to a connection. In addition, by choosing Raspberry, you have the possibility to modify your services like you want (examples: the size of the disk, the hosting of Database, etc.), which is generally not the case with specialized hosts, which often sell shared hosting with low configuration capacity. However, to support more users, you should use a Raspberry Pi 3 (the Pi 3 can be found here), the Raspberry Pi with 1 GB of RAM, rather than the Raspberry Type B + (512 MB of RAM).

What is Apache?

Apache is the most used web server, with about 60% market share. Apache has its own license, used by many other projects. In addition, the massive use of Apache (which has become the standard for web servers), coupled with its high popularity, has led to a tremendous abundance of documentation, courses, and other books dealing with its use, and his security.

Apache installation

Before installing the server, make sure we have an up-to-date machine. To do this we must have administrator rights, either because of the sudo command.

```
sudo apt update
sudo apt upgrade
sudo apt update
```

Once the Raspberry Pi is up to date, we will install the Apache server.

```
sudo apt install apache2
```

By the way, we'll take advantage of it to give rights to the apache file that you can easily manage your sites. To do this, run the following commands:

```
sudo chown -R pi:www-data /var/www/html/
sudo chmod -R 770 /var/www/html/
```

Check if Apache is working

Once the installation completed, we can test that Apache is working properly by going to the Raspberry address. To do this, it's necessary to try to access to the Raspberry from port 80 (this port not being opened from the outside, it will have to do since the Raspberry itself). Simply open the Raspberry web browser, and go to "http://127.0.0.1". You should then get a page with a message like "It works! "And plenty of other text. If you do not already have a GUI on your Raspbian, or you use SSH to connect to your Raspberry, you can use the following command:

```
wget -O check_apache.html http://127.0.0.1
```

This command will save the HTML code of the page in the file "check_apache.html" in the current directory. So you only have to read the file with the command

```
cat ./check_apache.html
```

If you see marked at a location in the code "It works!" is that Apache is working. Apache uses the directory "/var/www/html" as the root for your site. This means that when you call your Raspberry on port 80 (http), Apache looks for the file in "/var/www/html". For example, if you call the address "http://127.0.0.1/example", Apache will look for the "example" file in the "/var/www/html" directory. To add new files, sites, etc., you will need to add them to this directory. You can now use your Raspberry to make a site in HTML, CSS and JavaScript, internally. However, you may want to quickly allow interactions between the site and the user. For example, to allow the user to register, etc. For this, you are going to need PHP.

PHP installation on your Raspberry Pi

We will again use the administrator to install PHP with the command line.

```
sudo apt install php php-mbstring
```

Control if PHP is working

To know if PHP is working properly, it's not very complicated, and the method is quite similar to the one used for Apache. You will first delete the file "index.html" in the directory "/var/www/html".

```
sudo rm /var/www/html/index.html
```

Then create an "index.php" file in this directory, with this command line

```
echo "" > /var/www/html/index.php
```

From there, the operation is the same as for the Apache check.

A MySQL database for server

Now that we have set up PHP, you will probably want to store information for use in your sites. For this purpose, databases are most often used. We will therefore set up a DBMS (Database Management System), namely MySQL. MySQL is a free, powerful, massively used DBMS (about 56% market share of free DBMS). Here again, MySQL is so essential to development, whatever the language, that you must absolutely learn and master it.

How to install MySQL

To do this, we will install mysql-server **and** php-mysql (which will serve as link between php and mysql)

```
sudo apt install mysql-server php-mysql
```

Verify that MySQL is working correctly

To check the operation of MySQL, this time we will only use the command line. To do this, we will simply connect via the command:

```
sudo mysql --user=root
```

We will no delete the default mysql root user and create a new mysql root user, because the default one can only be used with Linux root account, and so not available for the webserver and php scripts. To do so, once you connect to MySQL, simply run those commands (replace password with the password you want):

```
DROP USER 'root'@'localhost';
CREATE USER 'root'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost'
```

So you now have a web server, connected to PHP and MySQL. That's all it takes.

Client Application:

The client application will be measuring environmental condition like humidity and temperature using DHT22 sensor.

Server Application:

On server side, the data generated on client side will be stored into MySQL database using XMPP server. Here client will be going to use different services of server.

Outcome: Thus we have studied how to install and configure XMPP server for data exchange between client application and server application.
