SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – V – Internet of Things – SCSA5301

HANDS-ON PROJECTS

Industry 4.0 concepts - Sensors and sensor Node and interfacing using any Embedded target boards (Raspberry Pi / Intel Galileo/ARM Cortex/ Arduino) - DIY Kits – Soil moisture monitoring - Weather monitoring - Air quality Monitoring - Movement Detection.

1. Soil Moisture Monitoring

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture.

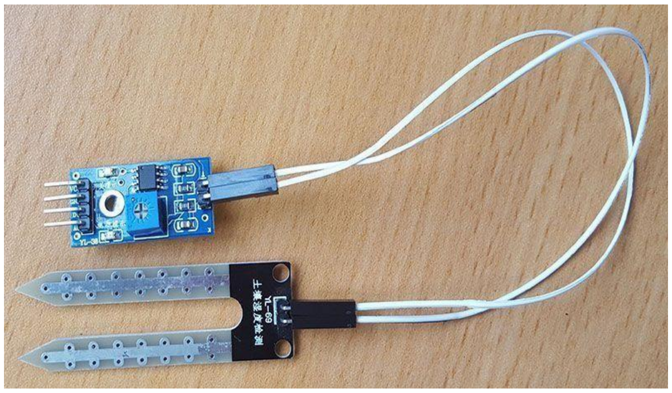


Figure 1: Soil Moisture Sensor

Now let‘s wire the sensor to the Raspberry Pi.

VCC-->3v3
GND -->
GND
D0 --> GPIO 17 (Pin 11)

With everything now wired up, we can turn on the Raspberry Pi. Without writing any code we can test to see our moisture sensor working. When the sensor detects moisture, a second led will illuminate .So as a quick test, grab a glass of water (be very careful not to spill water!!) then place the probes into the water and see the detection led shine. If the detection light doesn‘t illuminate, you can adjust the potentiometer on the sensor which allows you to change the detection threshold (this only applies to the digital output signal). In this example we want to monitor the moisture levels of our plant pot. So we want to set the detection point at a level so that if it drops below we get notified that our plant pot is too dry and needs watering. Our plant here, is a little on the dry side, but ok for now, if it gets any drier it‘ll need watering.

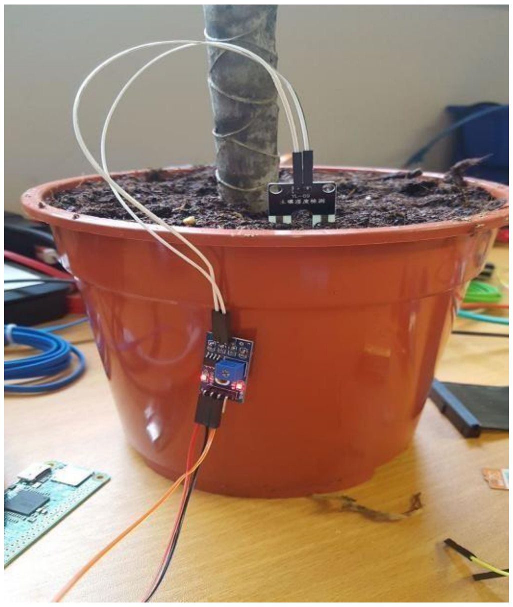


Figure 2: Experimental Setup

To run the script simply run the following command from the same directory as the script: sudo python moisture.py

1.1 Code

```
import RPi.GPIO as

GPIO import smtplib

import time

smtp_username = "enter_username_here" # This is the username used to login
to your SMTP provider

smtp_password = "enter_password_here" # This is the password used to login
to your SMTP provider

smtp_host = "enter_host_here" # This is the host of the SMTP

provider smtp_port = 25 # This is the port that your SMTP

provider uses smtp_sender = "sender@email.com" # This is the

FROM email address smtp_receivers = ['receiver@email.com']

# This is the TO email address message_dead = """From:

Sender Name <sender@email.com>

To: Receiver Name <receiver@email.com>

Subject: Moisture Sensor Notification
```

```
# This is the message that will be sent when moisture IS

detected again message_alive = """From: Sender Name

<sender@email.com>

To: Receiver Name <receiver@email.com>

Subject: Moisture Sensor

Notification # This is our

sendEmail function

def sendEmail(smtp_message):

    try:

        smtpObj = smtplib.SMTP(smtp_host, smtp_port)

        smtpObj.login(smtp_username, smtp_password) # If you don't
need to login to your smtp provider, simply remove this line

        smtpObj.sendmail(smtp_sender, smtp_receivers,

        smtp_message) print "Successfully sent email"

    except smtplib.SMTPException:

        print "Error: unable to send email"

# This is our callback function, this function will be called every time there is a
change on the specified GPIO channel, in this example we are using 17

def callback(channel):
        if GPIO.input(channel):
                print "LED off"
                sendEmail(message_d
                ead)
        else:
                print "LED on"
                sendEmail(message_al
                ive)
# Set our GPIO numbering to
BCM
GPIO.setmode(GPIO.BCM)
# Define the GPIO pin that we have our digital output from our sensor
connected to channel = 17
# Set the GPIO pin to an input
GPIO.setup(channel, GPIO.IN)
# This line tells our script to keep an eye on our gpio pin and let us know
when the pin goes HIGH or LOW
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
```

```
# This line assigns a function to the GPIO pin so that when the above line
tells us there is a change on the pin, run this function

GPIO.add_event_callback(channel, callback)

# This is an infinte loop to keep our script

running while True:

        # This line simply tells our script to wait 0.1 of a second, this is so the script
doesnt hog all of the CPU

        time.sleep(0.1)
```

2. Weather Monitoring

The DHT11 is a low-cost temperature and humidity sensor. It isn‘t the fastest sensor around but its cheap price makes it useful for experimenting or projects where you don‘t require new readings multiple times a second. The device only requires three connections to the Pi.
+3.3v, ground and one GPIO pin.

DHT11 Specifications

The device itself has four pins but one of these is not used. You can buy the 4-pin device on its own or as part of a 3-pin module. The modules have three pins and are easy to connect directly to the Pi‘s GPIO header.

- Humidity : 20-80% (5% accuracy)
- Temperature : 0-50°C (±2°C accuracy)

Hardware Setup

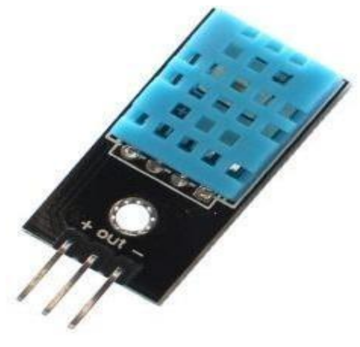


Figure 3: Humidity and Temperature Sensor

The 4-pin device will require a resistor (4.7K-10K) to be placed between Pin 1 (3.3V) and Pin 2 (Data). The 3-pin modules will usually have this resistor included which makes the wiring a bit easier. The 3 pins should be connected to the Pi as shown in the table below :

DHT Pin	Signal	Pi Pin
1	3.3V	1
2	Data/Out	11 (GPIO17)
3	not used	–
4	Ground	6 or 9

Your data pin can be attached to any GPIO pin you prefer. In my example I am using physical pin 11 which is GPIO 17. Here is a 4-pin sensor connected to the Pi‘s GPIO header. It has a 10K resistor between pin 1 (3.3V) and 2 (Data/Out).

Python Library

The DHT11 requires a specific protocol to be applied to the data pin. In order to save time trying to implement this yourself it‘s far easier to use the Adafruit DHT library. The library deals with the data that needs to be exchanged with the sensor but it is sensitive to timing issues. The Pi‘s operating system may get in the way while performing other tasks so to compensate for this the library requests a number of readings from the device until it gets one that is valid. To start with update your package lists and install a few Python libraries :

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

Then clone the Adafruit library from their repository :

```
Git clone
https://github.com/adafruit Cd
Adafruit_Python_DHT
```

Then install the library for Python 2 and

```
Python 3 sudo python setup.py install
sudo python3 setup.py

install python

AdafruitDHT.py 11 17
```

The example script takes two parameters. The first is the sensor type so is set to ―11‖ to represent the DHT11. The second is the GPIO number so for my example I am using ―17‖ for GPIO17. You can change this if you are using a different GPIO pin for your data/out wire. You should see an output similar to this :

```
Temp=22.0*Humidity=6

      8.0% import

      Adafruit_DHT
```

```
 # Set sensor type : Options are DHT11,DHT22 or
AM2302 sensor=Adafruit_DHT.DHT11
 # Set GPIO sensor is
connected to gpio=17
 # Use read_retry method. This will retry up to 15 times to
# get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)
 # Reading the DHT11 is very sensitive to timings and occasionally
# the Pi might fail to get a valid reading. So check if readings
are valid. if humidity is not None and temperature is not
None:
  print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature,
humidity)) else:
  print('Failed to get reading. Try again!')
```

3. Air Pollution Monitoring

Air pollution is a major problem in urban centers as well as rural set-up. The major pollutants of concern are primarily carbon monoxide, nitrogen oxides, hydrocarbons and particulate matter (PM10, PM2.5). Ozone, PAN and PBN are other secondary pollutants generated as a result of the photochemical reactions of the primary pollutants. These pollutants affect human health as well as environment. Therefore, air pollution monitoring is necessary to keep a check on the concentration of these pollutants in ambient air. The grove sensors, grove DHT (for temperature and humidity), grove gas sensor modules like dust, MQ-5 (for smoke), MQ-7 (for CO) and MQ-135 (for CO_2) are interfaced to this shield for monitoring in our proposed.

Adafruit CCS811 is a gas sensor that can detect a wide range of Volatile Organic Compounds (VOCs) and is intended for indoor air quality monitoring. When connected to your microcontroller (running our library code) it will return a Total Volatile Organic Compound (TVOC) reading and an equivalent carbon dioxide reading (eCO2) over I2C. There is also an on- board thermistor that can be used to calculate the approximate local ambient temperature.

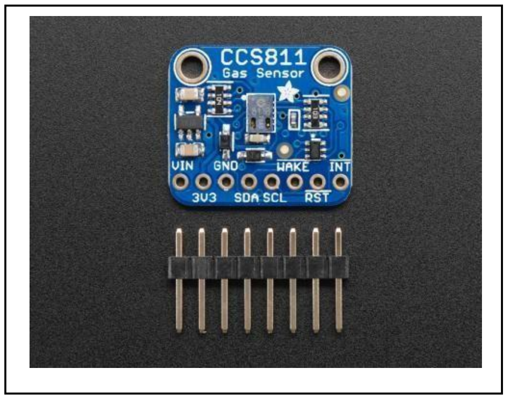


Figure 4: Gas Sensor

Power Pins

- **Vin** - this is the power pin. Since the sensor uses 3.3V, we have included an onboard voltage regulator that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

Logic pins

- **SCL** - this is the I2C clock pin, connect to your microcontrollers I2C clock line. There is a 10K pullup on this pin and it is level shifted so you can use 3 - 5VDC.
- **SDA** - this is the I2C data pin, connect to your microcontrollers I2C data line. There is a 10K pullup on this pin and it is level shifted so you can use 3 - 5VDC.
- **INT** - this is the interrupt-output pin. It is 3V logic and you can use it to detect when a new reading is ready or when a reading gets too high or too low.
- **WAKE** - this is the wakeup pin for the sensor. It needs to be pulled to ground in order to communicate with the sensor. This pin is level shifted so you can use 3- 5VDC logic.
- **RST** - this is the reset pin. When it is pulled to ground the sensor resets itself. This pin is level shifted so you can use 3-5VDC logic.

3.3 Raspberry Pi Wiring & Test

The Raspberry Pi also has an I2C interface that can be used to communicate with this sensor. Once your Pi is all set up, and you have internet access set up, lets install the software we will need. First make sure your Pi package manager is up to date

```
sudo apt-get update
```

Next, we will install the Raspberry Pi library and Adafruit_GPIO which is our hardware interfacing layer

```
1.   sudo apt-get install -y build-essential python-pip python-dev python-smbus  git
2.   git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
3.   cd Adafruit_Python_GPIO
4.   sudo python setup.py  install
```

Next install the adafruit CCS811 python library.

```
1. sudo pip install Adafruit_CCS811
```

Enable I2C

We need to enable the I2C bus so we can communicate with the sensor.

```
1. sudo raspi-config
```

Once I2C is enabled, we need to slow the speed way down due to constraints of this particular sensor.

```
1. sudo nano /boot/config.txt
```

add this line to the file

```
1.  dtparam=i2c_baudrate=10000
```

press **Ctrl+X**, then **Y**, then **enter** to save and exit. Then run **sudo shutdown -h now** to turn off the Pi and prepare for wiring.

Wiring Up Sensor

With the Pi powered off, we can wire up the sensor to the Pi Cobbler like this:

- Connect **Vin** to the 3V or 5V power supply (either is fine)
- Connect **GND** to the ground pin on the Cobbler
- Connect **SDA** to **SDA** on the Cobbler
- Connect **SCL** to **SCL** on the Cobbler
- Connect **Wake** to the ground pin on the Cobbler

Now you should be able to verify that the sensor is wired up correctly by asking the Pi to detect what addresses it can see on the I2C bus:

```
sudo i2cdetect -y 1
```

Run example code

At long last, we are finally ready to run our example code

```
cd ~/
git clone https://github.com/adafruit/Adafruit_CCS811_python.git
cd Adafruit_CCS811_python/examples
sudo python CCS811_example.py
```

```
from time import sleep

from Adafruit_CCS811 import

Adafruit_CCS811 ccs =

Adafruit_CCS811()
```

```
while not

      ccs.available():

      pass

temp =

ccs.calculateTemperature()

ccs.tempOffset = temp - 25.0

while(1):

      if ccs.available():

        temp =

        ccs.calculateTemperature() if

        not ccs.readData():

         print "CO2: ", ccs.geteCO2(), "ppm, TVOC: ", ccs.getTVOC(), "

        temp: ", temp else:

         print

         "ERROR!"

         while(1):

            pas

      s sleep(2)
```

4. Movement Detection

PIR stands for passive infrared. This motion sensor consists of a fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it toward the infrared detector. Our bodies generate infrared heat, and as a result, this heat is picked up by the motion sensor. The sensor outputs a 5V signal for a period of one minute as soon as it detects the presence of a person. It offers a tentative range of detection of about 6–7 meters and is highly sensitive. When the PIR motion sensor detects a person, it outputs a 5V signal to the Raspberry Pi through its GPIO and we define what the Raspberry Pi should do as it detects an intruder through the Python coding. Here we are just printing "Intruder detected".

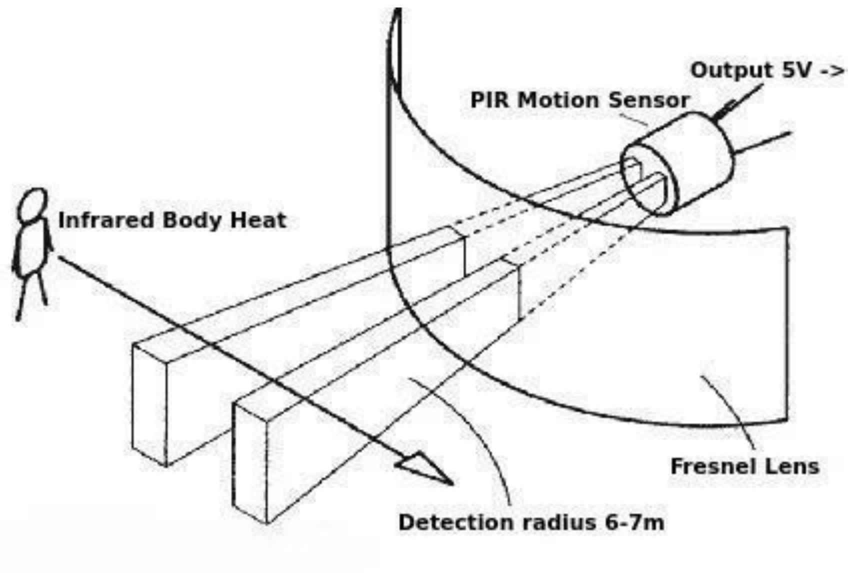


Figure 5: Working of PIR Sensor

4.1 Working Mechanism

All living beings radiate energy to the surroundings in the form of infrared radiations which are invisible to human eyes. A PIR (Passive infrared) sensor can be used to detect these passive radiations. When an object (human or animal) emitting infrared radiations passes through the field of view of the sensor, it detects the change in temperature and therefore can be used to detect motion.HC-SR501 uses differential detection with two pyroelectric infrared sensors. By taking a difference of the values, the average temperature from the field of view of a sensor is removed and thereby reducing false positives.

```
import RPi.GPIO as GPIO

import time #Import time

library

GPIO.setmode(GPIO.BOARD) #Set GPIO pin

numbering pir = 26 #Associate pin 26 to pir

GPIO.setup(pir, GPIO.IN) #Set pin as GPIO in print "Waiting for

sensor to settle" time.sleep(2) #Waiting 2 seconds for the sensor to

initiate print "Detecting motion"

while True:

if GPIO.input(pir): #Check whether pir is HIGH print "Motion

Detected!" time.sleep(2) #D1- Delay to avoid multiple

detection

time.sleep(0.1) #While loop delay should be less than detection(hardware) delay
```

5. **Upload Your Raspberry Pi Sensor Data to**

   Thingspeak Website Things needed

1. Raspbeery Pi
2. Power Cable
3. Wifi adapter or LAN connection to Raspbeery Pi

Step 1: Signup for

Thingspeak Go to

www.thinspeak.com

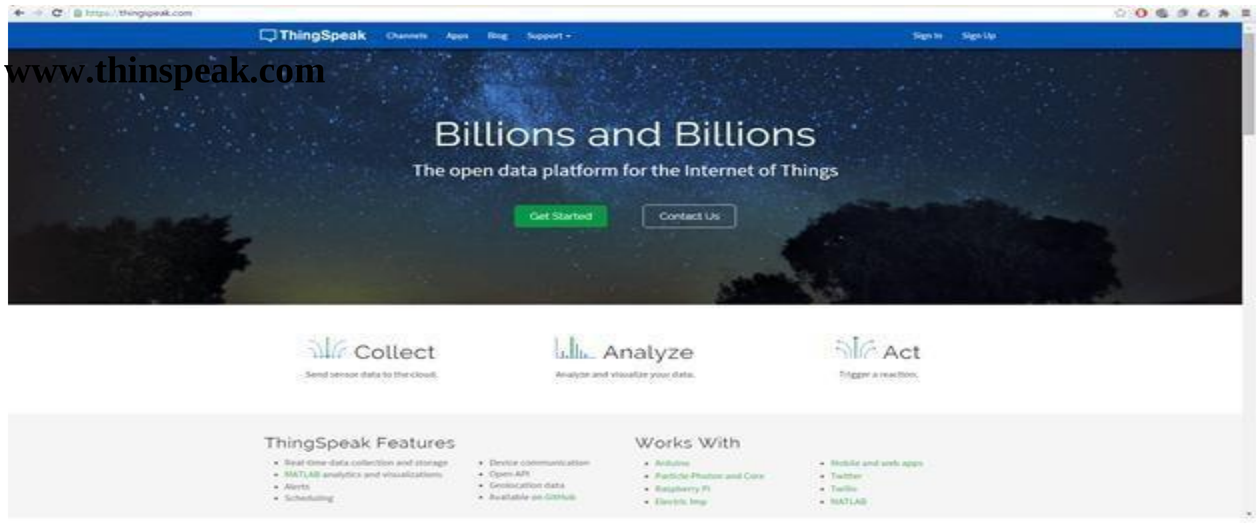


Figure 6: Thingspeak Website

Click on ―Sign Up‖ option and complete the details

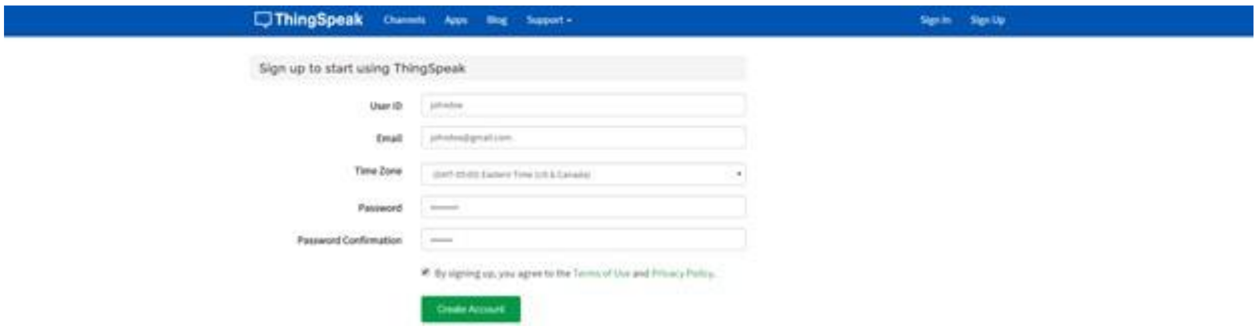
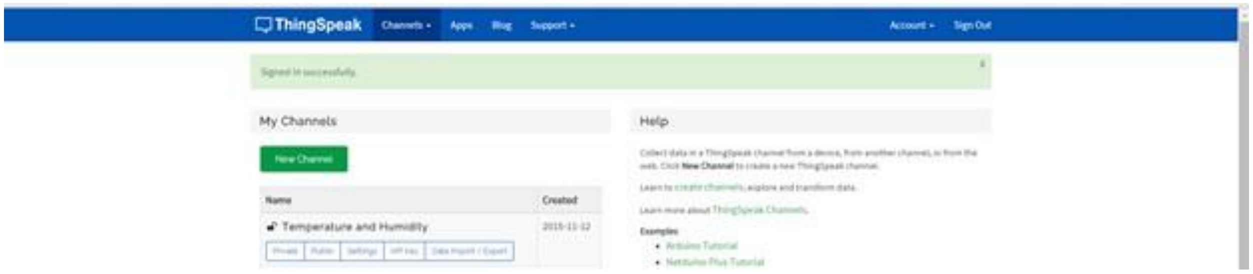


Figure7: Create user

account Step 2: Create a Channel for Your Data

Once you Sign in after your account activation, Create a new channel by clicking –New Channel‖ button



ThingSpeak Channels ▾ Apps Blog Support ▾ Account ▾ Sign Out

Signed in successfully.

My Channels

New Channel

Name	Created
Temperature and Humidity	2015-11-12

Private Public Settings API Key Data Import / Export

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web. Click **New Channel** to create a new ThingSpeak channel.

Learn to create channels, explore and transform data.

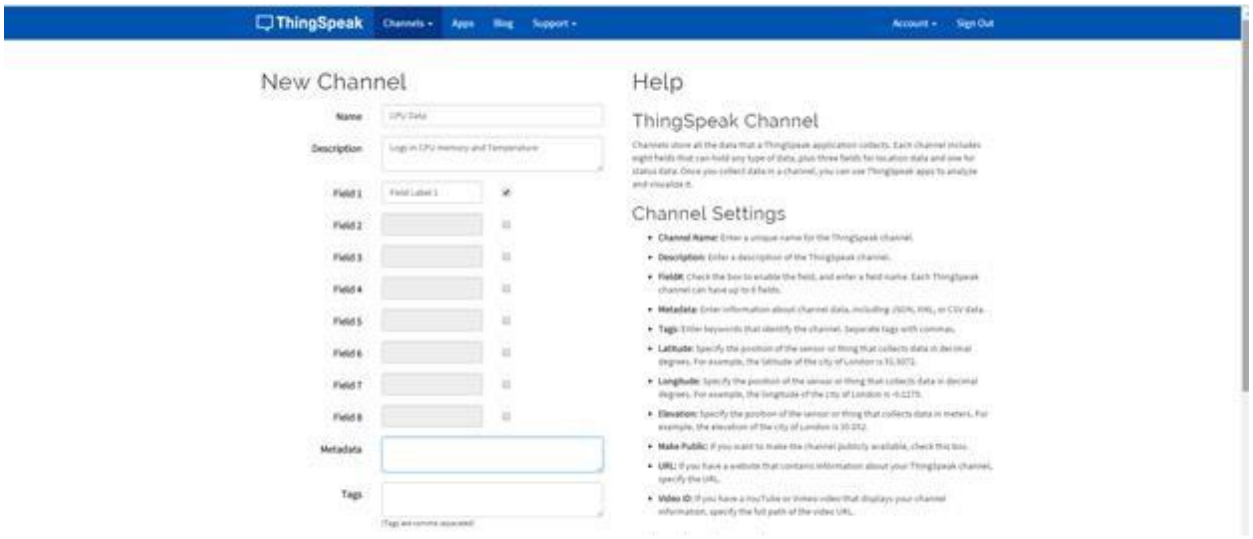
Learn more about ThingSpeak Channels.

Examples:

- Arduino Tutorial
- Netduino Plus Tutorial

Figure 8: Creating New Channel

After the –New Channel‖ page loads, enter the Name and Description of the data you want to upload. You can enter the name of your data (ex: Temperature) in Field1. If you want more Fields you can check the box next to Field option and enter the corresponding name of your data.



ThingSpeak Channels ▾ Apps Blog Support ▾ Account ▾ Sign Out

New Channel

Name	CPU Data	
Description	Logs in CPU memory and Temperature	
Field 1	Field Label 1	✓
Field 2		☐
Field 3		☐
Field 4		☐
Field 5		☐
Field 6		☐
Field 7		☐
Field 8		☐
Metadata		
Tags		

(Tags are comma separated)

Help

ThingSpeak Channel

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

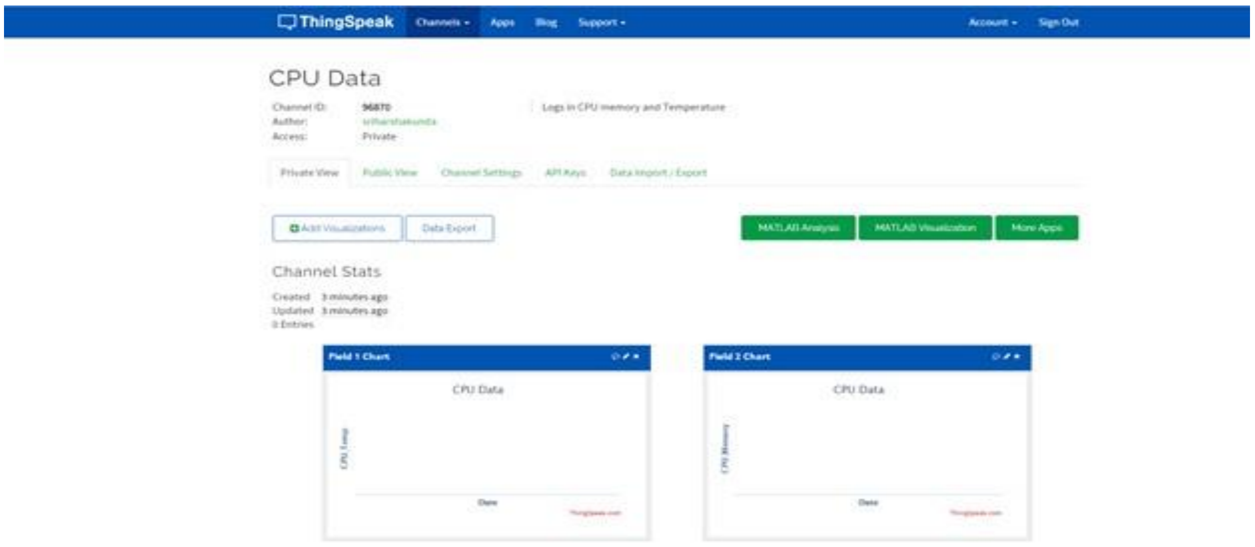
Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Latitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
- **Longitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the longitude of the city of London is -0.1275.
- **Elevation:** Specify the position of the sensor or thing that collects data in meters. For example, the elevation of the city of London is 35.052.
- **Make Public:** If you want to make the channel publicly available, check this box.
- **URL:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Video ID:** If you have a YouTube or Vimeo video that displays your channel information, specify the full path of the video URL.

Figure 9: New Channel settings

Click on ―Save Channel‖ button to save all of your settings.

We created two Fields, one is CPU Memory and one for CPU Temperature



CPU Data

Channel ID: 96870

Author: [illegible]

Access: Private

Logs in CPU memory and Temperature

Channel Stats

Created 3 minutes ago

Updated 3 minutes ago

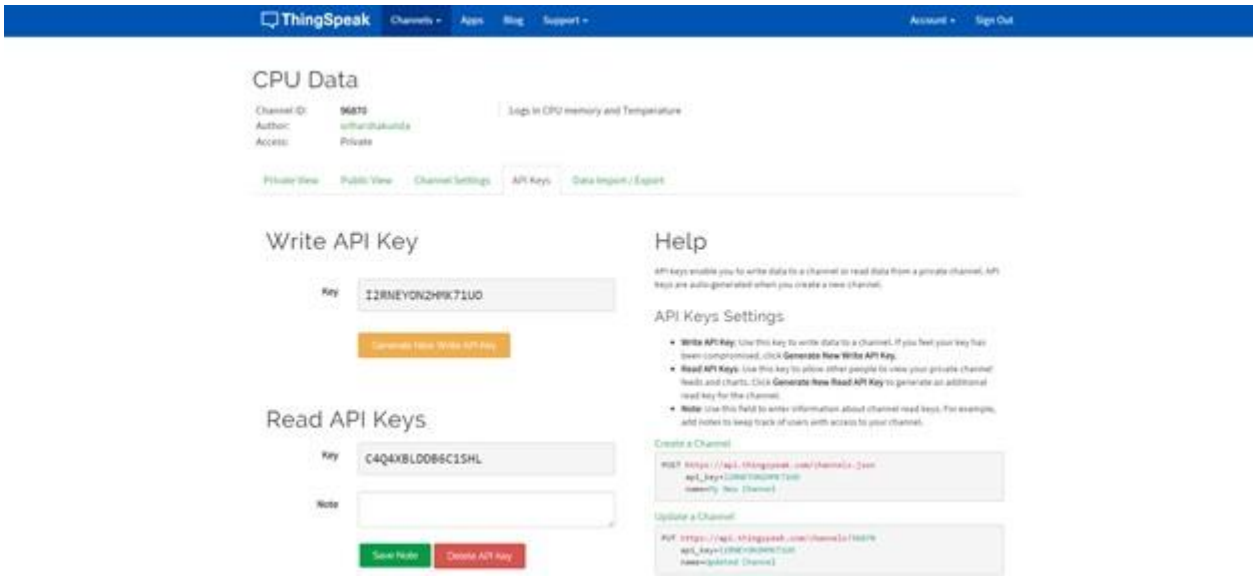
0 Entries

Figure 10: Creating field charts to display data

Step 3: Get an API Key

To upload our data, we need an API key, which we will later include in a piece of python code to upload our sensor data to Thingspeak Website.

Click on ―API Keys‖ tab to get the key for uploading your sensor data.



CPU Data

Channel ID: 96870
Author: sriharshakunda
Access: Private

Logs in CPU memory and Temperature

Write API Key

Key I2RNEYON2HMK71UO

Read API Keys

Key C4Q4XBLDDB6C1SHL

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

Create a Channel

```
POST https://api.thingspeak.com/channels.json
     api_key=[illegible]
     name=My New Channel
```

Update a Channel

```
PUT https://api.thingspeak.com/channels/96870
    api_key=[illegible]
    name=Updated Channel
```

Figure 11: Copy the Write API Key of the channel

The advantage of using Thingspeak compared to Xively or any other websites is that the convenience of using Matlab Analysis and Matlab Visualizations. Once you have the ―Write API Key‖. We are almost ready to upload our data, except for the python code.

Step 4: Modifying the Python Code

Go to

https://github.com/sriharshakunda/Thingspeak_CPU_Python-

Code Download the code into your Raspberry Pi Home folder.

Open the CPU_Python.py file in a

notepad. Code:

```
import httplib, urllib
import time
sleep = 60 # how many seconds to sleep between posts to the channel
key = 'Put your Thingspeak Channel Key here' # Thingspeak channel to update

#Report Raspberry Pi internal temperature to Thingspeak
Channel def thermometer():
   while True:
      #Calculate CPU temperature of Raspberry Pi in Degrees C
      temp = int(open('/sys/class/thermal/thermal_zone0/temp').read()) / 1e3 # Get
Raspberry Pi CPU temp
      params = urllib.urlencode({'field1': temp, 'key':key })
      headers    =    {"Content-typZZe":  "application/x-www-form-
urlencoded","Accept": "text/plain"}
      conn =
      httplib.HTTPConnection("api.thingspeak.com:80"
      ) try:
         conn.request("POST", "/update", params,
         headers) response = conn.getresponse()
         print temp
         print response.status,
         response.reason data =
         response.read()
         conn.close
      () except:
         print "connection
      failed" break
#sleep for desired amount of
time if____name___== "
main_":
      while True:
```

```
thermometer()
time.sleep(sle
ep)
```

Edit the line 19 by using **CPU_Temp** instead of **temp**.

Use your Write API Key to replace the **key** with your **API Key**

Use your Write API Key to replace the **key** with your **API Key**

Save the file to overwrite changes

Step 5: Assuming you have python 2.7 and proper python libraries, go to the folder where you copied the CPU_Python.py file

Type python2.7 CPU_Python.py file

In case if there are any errors uploading the data, you will receive –connection failed‖ message

Step 6: Check Thinspeak API and Confirm data transfer

Open your channel and you should see the temperature uploading into thinspeak website.

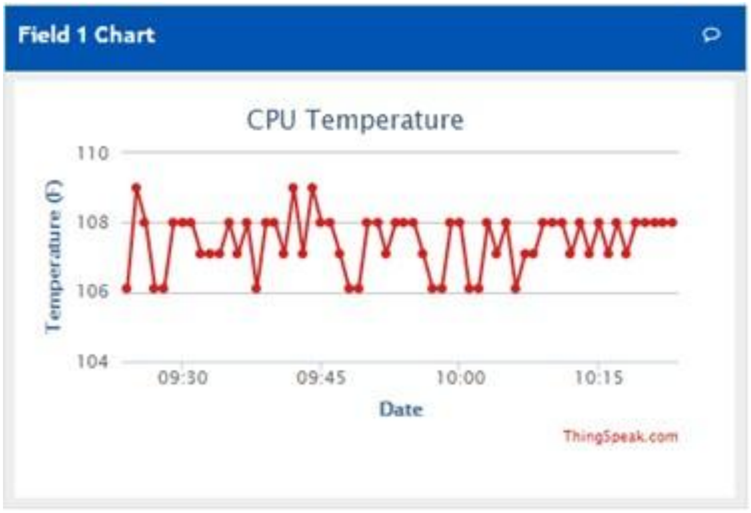


Figure: 12 CPU Temperature data displayed in Field Chart

6. IFTTT

IFTTT – short for _If This Then That‘ – is a free online service that lets you automate specific tasks. It lets you trigger actions on other apps, web services and devices automatically every time certain conditions are met. These trigger > action relationships used to be called recipes. but will now be known as Applets. The trigger and action relationships have always been known as recipes‘, but that‘s all changing. From now on, they will be dubbed Applets. There are no major changes to the way things work, but there are a few differences worth being aware of. Each Applet you enable will still trigger ‘if this then that‘ action when the relevant conditions are met, but now Applets can trigger multiple actions, instead of just one. Once the first action is completed, a second, third, fourth, etc, can also be triggered.

"Channel" is IFTTT parlance for a Web service or other action. IFTTT currently supports 67 different channels spanning a wide range of popular services, and it can perform basic actions such as calling or texting a phone, or sending you an email. Here's a list of all the available IFTTT channels. You may recognize some of them: Craigslist, Dropbox, Evernote, Facebook, Flickr, Foursquare, Instagram, SkyDrive, Twitter, YouTube, and a wide range of Google services are just the tip of the iceberg. A newly released iPhone IFTTT app even adds channels for your phone's Reminders, Contacts, and Photos apps. Once you've gone through and activated some channels—basically, granting IFTTT access to your various services or providing it with personal details—you're ready to start crafting.

Services

A service is just what it sounds like, a tool, application, or facility that works with IFTTT. The brilliant thing about IFTTT is that its variety of channels allows it to offer something to everybody. **The list of available services** is enormous and more are added all the time. Some of the most popular services include Facebook, Twitter, Instagram, YouTube, SoundCloud, **Dropbox**, Evernote, and Pocket.

Applets

Applets are what make IFTTT worth your time. Basically, they are the combination of services that use a trigger and an action. When something happens on one service, it triggers an action on another.

Create an Applet

The first step is to click **My Applets** and then **New Applet**. Next, click the word **This**.

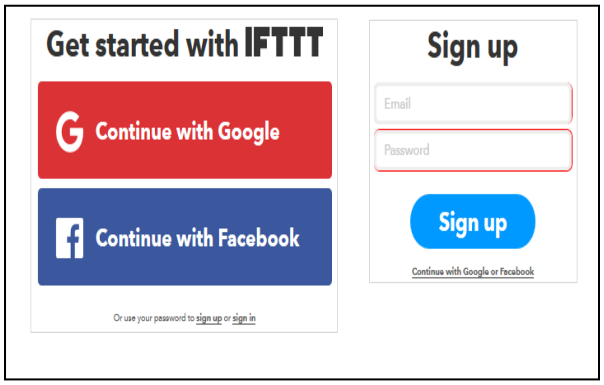


Figure13: Login Page

For this example, we will select the Instagram trigger, which will then ask us to activate Instagram just this once. Having done that, we will choose a trigger action:

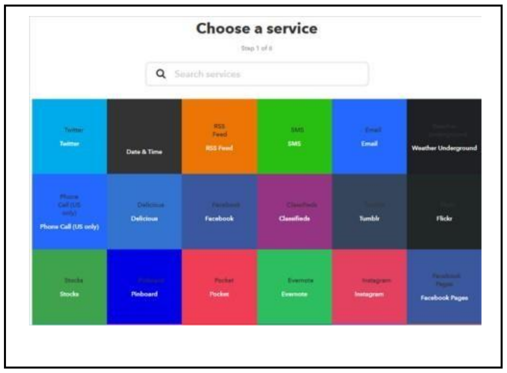
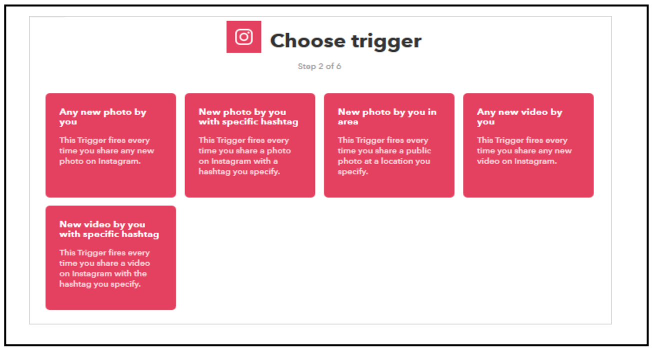


Figure 14: Service


Choose trigger

Step 2 of 6

Any new photo by you
This Trigger fires every time you share any new photo on Instagram.

New photo by you with specific hashtag
This Trigger fires every time you share a photo on Instagram with a hashtag you specify.

New photo by you in area
This Trigger fires every time you share a public photo at a location you specify.

Any new video by you
This Trigger fires every time you share any new video on Instagram.

New video by you with specific hashtag
This Trigger fires every time you share a video on Instagram with the hashtag you specify.

Figure 15: Trigger

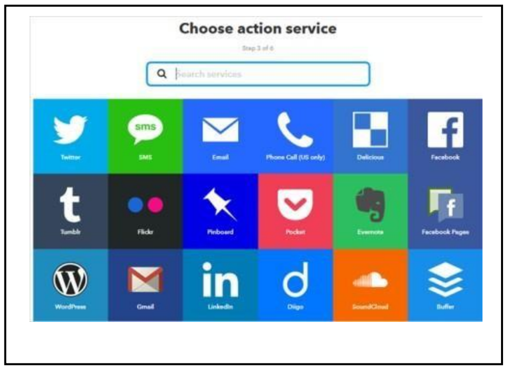
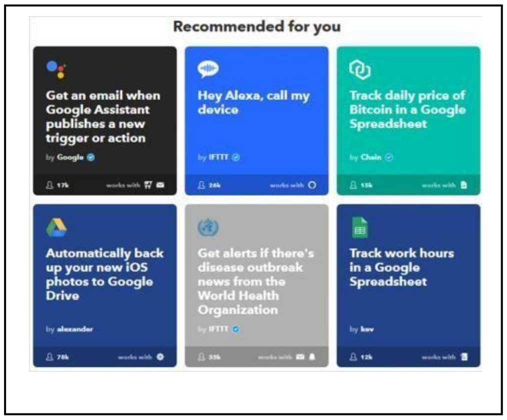


Figure 16: Action Service

After doing this, we will be greeted by the second batch of actions. We‘ll select the first one and be asked to complete the fields. In this case, it‘s asking us where to grab the photos, how to name them and where it should put them. All you have to do is click on the **Add ingredient**, make your selection from the drop-down box, and hit the **Create Action** button. Finally, you will be asked to review your Applet. You can optionally enable notifications when the Applet runs. Then, click **Finish**.

Pre-Made Applets

We can browse other people‘s Applets, view options by category, check out collections, look at recommendations, or do a search if you are looking for something specific. And, using existing Applets is easier than creating your own.


Recommended for you

Get an email when Google Assistant publishes a new trigger or action
by Google
17k works with

Hey Alexa, call my device
by IFTTT
26k works with

Track daily price of Bitcoin in a Google Spreadsheet
by Chain
15k works with

Automatically back up your new iOS photos to Google Drive
by alexander
78k works with

Get alerts if there's disease outbreak news from the World Health Organization
by IFTTT
33k works with

Track work hours in a Google Spreadsheet
by kev
12k works with

Figure 17: Pre Made Applets

Just click on an Applet to review the details, and move the slider to turn it on. Depending on the Applet you choose, you may be asked to connect an account like Facebook or configure pieces of the Applet like date and time. But, this is all very simple and self-explanatory as you move through the process.

Top 7 Applets

Applet #1 – Daily SMS Weather Forecast

You get IFTTT to send an SMS each morning telling you what the weather conditions are going to be for the day.

Applet #2 – Wake Up Call

You get a call at a time of your preference with an automated message.

Applet #3 – Starred Emails in Gmail to Evernote

When you mark an email with a star on Gmail, a copy of it is sent to your Evernote account.

Applet #4 – NASA's Image of the Day
NASA is well-known for many things, not the least of which is their stunning photographs of our galaxy. Set this up and you‘ll get an amazing photo in your email every day.

Applet #6 – Email For a Call to Find a Lost Phone
We‘ve all lost our phone before. With this Applet you get a call when you send an email to the specified address, helping you hear where it is.

Applet #7 – Timed Daily Tweet
Your account sends a tweet every day at a time you choose.

7. Other Apps and Services

Cayenne IoT Builder

Cayenne is an app for smartphones and computers that allows you to control the Raspberry Pi and soon also the Arduino through the use of an elegant graphical interface and a solid nice communication protocol.

The features are:

- Add and remotely control sensors, motors, actuators, GPIO boards, and more
- Customizable dashboards with drag-and-drop widgets for connection devices
- Create triggers and threshold alerts for devices, events, and actions
- Schedule one-time or multi-device events for easy automation
- Quick and easy setup - connect your Pi in minutes

Step 1: Go to Cayenne site and Sign Up. After download the file and install the Cayenne system on your Raspberry Pi. Download the **app** on your Smartphone or tablet by using the follows link

Cayenne on Apple Store or Cayenne on Play Store

Install on your Raspberry Pi the Raspbian system. For this step download NOOBS from Raspberripi.org: https://www.raspberrypi.org/downloads/

Copy the package on your SD, and start the Raspbian installation. For the raspbian installation I recommend to use a HDMI screen, a USB mouse and a USB keyboard.After this, connect your Raspberry at your LAN by cable. Then open your **Cayenne app** and install the library on your device. The next step.

Step 2: Download the App and Install Cayenne

Download the app on your Smartphone or tablet by using the

follows link Cayenne on Apple Store or Cayenne on Play Store

Install on your Raspberry Pi the Raspbian system. After this, connect your Raspberry at your LAN by cable. Then open your Cayenne app and install the library on your device (or)

Install **manually** Cayenne on your Raspberry Pi by using commands in **Terminal** of Raspberry Pi:

wget

https://cayenne.mydevices.com/dl/rpi_b8w8pn82i

9.sh sudo bash rpi_b8w8pn82i9.sh -v

After this **reboot** your Raspberry.

Step 3: See Your Device on Cayenne Dashboard

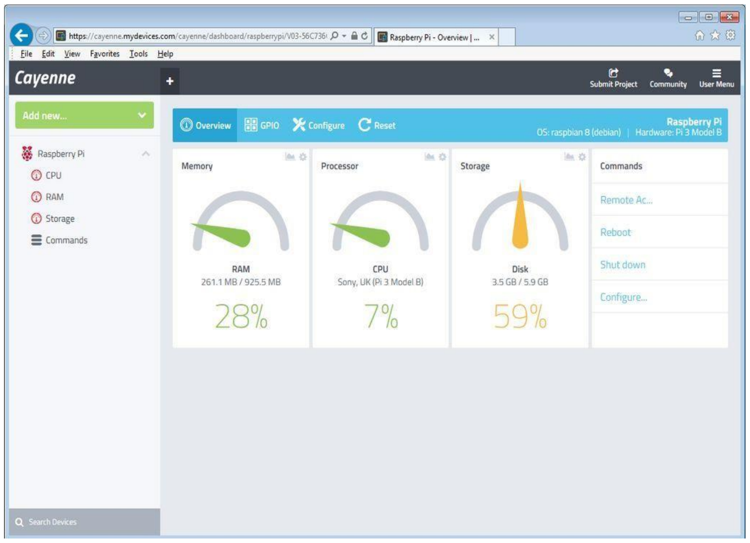


Figure 18: Device on Cayenne Dashboard

By using **Computer** you can see your device on: https://cayenne.mydevices.com/ like in photo. By using a **Smartphone** you can open the app and see your devices.You can personalize the Dashboard of Cayenne by using the widgets. The default Dashboard have CPU, Temp and RAM widget . These are the values of your Raspberry Pi. You can see the temp and the work flow of Raspberry pi. Then you can see the GPIO schedule. In the GPIO you can set every pin of GPIO of Raspberry. You can set the pin like Output or Input. Then you can activate the pin or read the value of the pin. The two values are **HIGH** or **LOW**. This is valid for Input and Output.

Step 4: Connect a Led to Your Raspberry Pi

Now you can connect a led to GPIO port of Raspberry Pi. Use ALWAYS a resistor in series to led. If you don't use a resistor, you can burn the led or the Raspberry Pi. See the photo and connect the led to pin number 11 or GPIO 17.

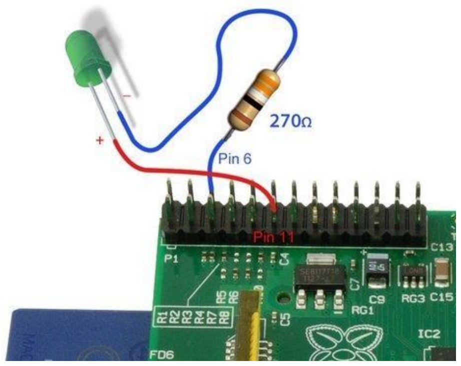


Figure 19: Connections with LED

Step 5: Switch ON the Led

Now go to the Dashboard of Cayenne, and open the GPIO schedule. Then Select the **Pin 17** and click on **Input**. Setup the pin like Output, and after click on LOW button. The button below Green and the word HIGH appear on it. Now your led in ON. You can switch on the led by using your Raspberry Pi by LAN, and also by a different Network Area. This is because Cayenne use a proprietary proxy.

Step 6: Create the Buttons on Dashboard

Go to Cayenne Dashboard on smartphone app, and click **+ on right up corner of screen.**

Now you can add a widget. Select Actuators device --> Select Generic --> Select

Digital Output. Fill the all the fields, and select the correct channel for RGB led.

The **Blue** pin in the **25**

The **Green** pin in the **24**

The **Red** pin in the **23**

Step 7: Now Switch on the Rainbow

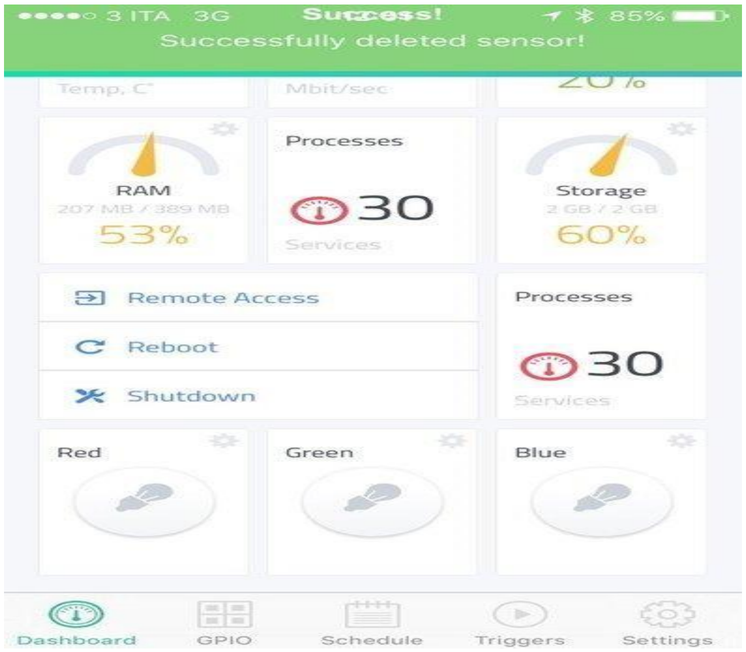


Figure 20: Create Buttons on Dashboard

You can use a normal breadboard and an RGB led common cathode. The RGB led is a led that have 3 leds inside. Why the pin are only 3? Because the 3 leds share a cathode, or anode pin. For this reason you must specify the kind of led you want. Now I use a common cathode led.

The GPIO port on Raspberry are:

The **Blue** pin in the

25 The **Green** pin in

the 24

The **Red** pin in the 23

Amazon Alexa

Alexa is Amazon’s cloud-based voice service available on tens of millions of devices from Amazon and third-party device manufacturers. With Alexa, you can build natural voice experiences that offer customers a more intuitive way to interact with the technology they use every day. Our collection of tools, APIs, reference solutions, and documentation make it easy for anyone to build with Alexa. Alexa Enabled is a category of products with built-in access to Alexa. You can talk to the device with the wake word “Alexa,” and receive voice responses and content instantly. Alexa- enabled products work with Alexa skills and Alexa compatible smart home devices and gadgets, bringing familiar capabilities from the Amazon Echo family of devices to a range of new form factors and use cases developed by leading brands.

The Alexa Voice Service (AVS) enables you to access cloud-based Alexa capabilities with the support of AVS APIs, hardware kits, software tools, and documentation. We simplify building voice-forward products by handling complex speech recognition and natural language understanding in the cloud, reducing your development costs and accelerating your time to market. Best of all, regular Alexa updates bring new features to your product and add support for a growing assortment of compatible smart home devices. Build with AVS, and become part of the Alexa family.