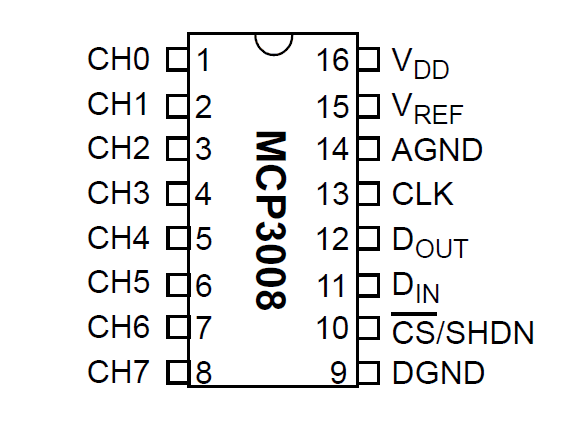
Documentary:

**Step1:** Connect the terminals of MCP 30008 ADC, 10 bit, 8 channel IC with Raspberry Pi as stated below



MCP3008 VDD to Raspberry Pi 3.3V(pin 1)

MCP3008 VREF to Raspberry Pi 3.3V(pin 17)

MCP3008 AGND to Raspberry Pi GND(pin 9)

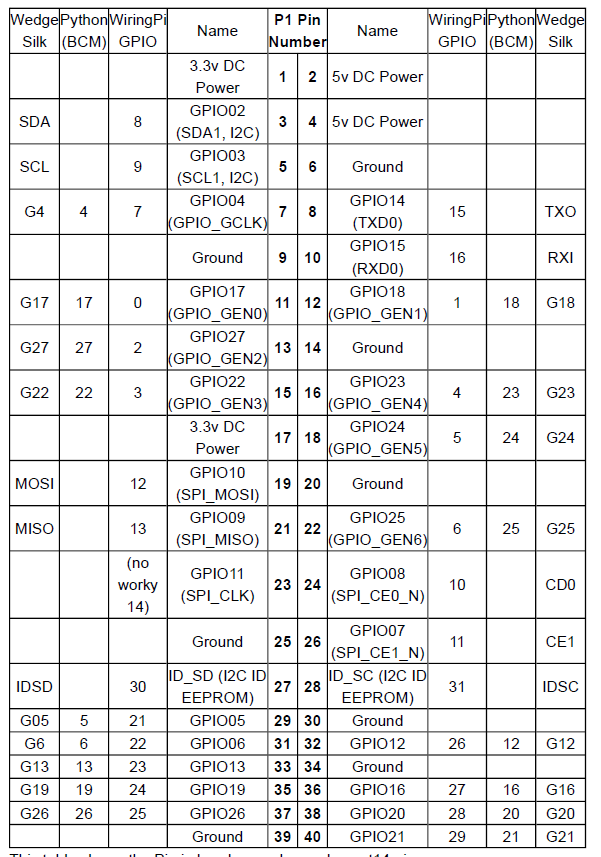
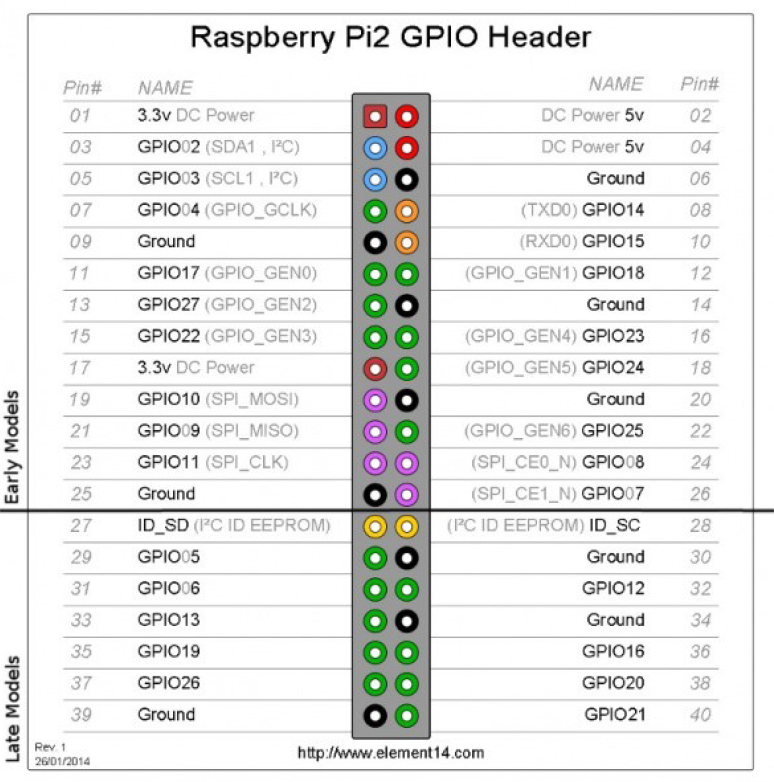
MCP3008 DGND to Raspberry Pi GND(pin 25)

MCP3008 CLK to Raspberry Pi pin GPIO18(pin 12)

MCP3008 DOUT to Raspberry Pi pin GPIO23(pin 16)

MCP3008 DIN to Raspberry Pi pin GPIO24(pin 18)

MCP3008 CS/SHDN to Raspberry Pi pin GPIO25(pin 22)



Connect indoor Dust sensor (aout)analog to CH0

Connect indoor MQ7sensor(aout) analog to CH1

Connect indoor MQ135 sensor (aout)analog to CH2

Connect outdoor MQ7 sensor(aout) analog to CH4

Connect outdoor MQ135 sensor(aout) analog to CH5

**Step 2 :**After wiring the MCP3008 to the Raspberry Pi we install the Adafruit MCP3008 Python library from<https://github.com/adafruit/Adafruit_Python_MCP3008>

OR

We install the library to our Raspberry Pi by connecting to the internet through a wired or wireless network connection

The codes are :

->sudo apt-get update

->sudo apt-get install build-essential python-dev python-smbusgit

->cd ~ git clone https://github.com/adafruit/Adafruit\_Python\_MCP3008.git

->cd Adafruit\_Python\_MCP3008

->sudo python setup.py install

To install from the **Python package index** connect to a terminal on the Raspberry Pi and execute the following commands:

->sudo apt-get update

->sudo apt-get install build-essential python-dev python-smbus python-pip

->sudo pip install adafruit-mcp3008

if installed from the Python package index it wouldnt have the example code for the library. We would need to download these MCP3008 examples to the Pi manually and run them in the next section.

**Step 3 :** The python code is implemented as:  
(22/12/2016)

importRPi.GPIO as GPIO

import time

importdatetime

importAdafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

GPIO.setmode(GPIO.BCM)

GPIO.setup(14,GPIO.IN)

CLK = 18

MISO = 23

MOSI = 24

CS = 25

mcp = Adafruit\_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

dt=datetime.datetime.now()

y=int(dt.year)

m=int(dt.month)

d=int(dt.day)

ymd=y\*10000+m\*100+d

i=0

fo=open("%d.csv"%ymd,"wb")

fo.write('Time,Dust,CO2,CO');

fo.write("\n")

for i in range(0,):

GPIO.setmode(GPIO.BCM)

GPIO.setup(14,GPIO.IN)

importdatetime

dtime=datetime.datetime.now()

h=int(dtime.hour)

mi=int(dtime.minute)

sec=int(dtime.second)

fo.write("%d."%h);

fo.write("%d."%mi);

fo.write("%d,"%sec);

digital= [0]\*3

analog= [0]\*3

for i in range(3):

digital[i]= mcp.read\_adc(i)

analog[i]= (3.3\*digital[i])/1023

fo.write(' {0:>4},{1:>4},{2:>4}'.format(\*analog))

fo.write("\n");

time.sleep(1)

fo.close()

(7/1/2017)

importRPi.GPIO as GPIO

import time

importdatetime

importAdafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

import sys

importos

importAdafruit\_DHT

GPIO.setmode(GPIO.BCM)

DHTsensor = Adafruit\_DHT.DHT11

DHT = 21

PWM=13

CLK = 18

MISO = 23

MOSI = 24

CS = 25

mcp = Adafruit\_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

dt=datetime.datetime.now()

y=int(dt.year)

m=int(dt.month)

d=int(dt.day)

ymd=y\*10000+m\*100+d

h=int(dt.hour)

mi=int(dt.minute)

se=int(dt.second)

t=3600\*h+60\*mi+se

ymd=ymd\*100000+t

i=0

fo=open("%d.csv"%ymd,"wb")

fo.write('Time(h.m.s),Dust(mg/m3),CO(ppm),CO2,temp(C)');

fo.write("\n")

for i in range(0,200):

GPIO.setmode(GPIO.BCM)

GPIO.setup(DHT,GPIO.IN)

GPIO.setup(PWM,GPIO.OUT)

importdatetime

dtime=datetime.datetime.now()

h=int(dtime.hour)

mi=int(dtime.minute)

sec=int(dtime.second)

fo.write("%d."%h);

fo.write("%d."%mi);

fo.write("%d,"%sec);

print("loop=%d"%i)

digital= [0]\*4

analog= [0]\*4

values= [0]\*4

summer= [0]\*4

GPIO.output(PWM,GPIO.LOW)

time.sleep(0.0003)

digital[0]=mcp.read\_adc(0)

time.sleep(0.0001)

GPIO.output(PWM,GPIO.HIGH)

time.sleep(0.0006)

analog[0]=(3.3\*digital[0])

analog[0]= analog[0]/1024

for i in range(1,4):

digital[i]= mcp.read\_adc(i)

analog[i]=(3.3\*digital[i])

analog[i]= analog[i]/1024

values[0]=((0.17\*analog[0])-0.1)

values[1]=(3.027\*(2.71828\*\*(1.0698\*analog[1])))

values[2]=((436.03\*analog[2]-34.34525)/analog[2])

values[3]=100\*analog[3]

for i in range(0,4):

a=(summer[i]+values[i])

summer[i]=a

for i in range(0,4):

summer[i]=(summer[i]/200)

fo.write('{0:0.5f},{1:0.4f},{2:0.4f},{3:0.4f}'.format(\*values))

fo.write("\n");

time.sleep(0.3)

fo.close()

The range of i in the for loop is implemented as required  
the code has 1 second delay.

**Step 4 :**

Connect the Dust sensor to A1 pin  
Connect the MQ7 to A2 pin  
Connect the MQ135 to A3 pin

The arduino code is given as:

int dust = A1;

int mq7 = A2;

int mq135 = A3;

int val,val2,val4;

float val1,val3,val5;

void setup()

{

pinMode(dust, INPUT);

pinMode(mq7, INPUT);

pinMode(mq135, INPUT);

Serial.begin(9600);

}

void loop()

{

val=analogRead(dust);

val1=(val\*(5/1023));

Serial.print(val1);

Serial.print(",");

val2=analogRead(mq7);

val3=(val2\*(3.3/1023));

Serial.print(val3);

Serial.print(",");

val4=analogRead(dustPin);

val5=(val4\*(3.3/1023));

Serial.println(val5);

Serial.print(“,");

delay(1000);

}

**Step 5 :**The data is stored in a csv file of the current date

We install gnuplot by the command

->sudo apt-get installgnuplot

**Step 6:** the X11 applications are installed:

View this youtube video: https://www.youtube.com/watch?v=QRsma2vkEQE

Go to :https://sourceforge.net/projects/xming/files/?source=navbar

installxming setup

installxming fonts

**Step 7 :**X11 forwarding is enabled in raspberry pi by entering the command

->sudonano /etc/ssh/ssh\_config

Change x11 forwarding from no to yes (if already changed avoid the step)

**Step 8 :**we plot data by using commands :

->gnuplot

->set title “title”

->set ylabel “ylabel”

->set xlabel “xlabel”

->set datafile separator ","

->set autoscale

->set grid

->plot '20161220arduino.csv' using 1:2 with linespoints, '20161220arduino.csv' using 1:3 with linespoints, '20161220arduino.csv' using 1:4 with linespoints

->plot '20161220arduino.csv' using 1:2 with lines title "CO2", '20161220arduino.csv' using 1:3 with lines title "CO"

plot '20170130.csv' using 1:2 with lines title "Dust", '20170130.csv' using 1:3 with lines title "CO", '20170130.csv' using 1:4 with lines title "Co2", '20170130.csv' using 1:5 with lines title "Temp"

**Step 9 :**

Wifi Setup

https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md

Open the wpa-supplicant configuration file in nano:

->sudonano /etc/wpa\_supplicant/wpa\_supplicant.conf

Go to the bottom of the file and add the following:

network={

ssid="The\_ESSID\_from\_earlier"

psk="Your\_wifi\_password"

}

Now save the file by pressing Ctrl+X then Y, then finally press Enter.

At this point, wpa-supplicant will normally notice a change has occurred

within a few seconds, and it will try and connect to the network.

If it does not, either manually restart the interface with

sudoifdown wlan0 and sudoifup wlan0, or reboot your Raspberry Pi with sudo reboot.

You can verify if it has successfully connected using ifconfig wlan0.

If the inetaddr field has an address beside it,

the Pi has connected to the network.

If not, check your password and ESSID are correct.

**Step 10 :**

auto run

http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/?ALLSTEPS

http://www.raspberrypi-spy.co.uk/2013/07/running-a-python-script-at-boot-using-cron/#prettyPhoto

https://www.dexterindustries.com/howto/auto-run-python-programs-on-the-raspberry-pi/

https://www.raspberrypi.org/forums/viewtopic.php?t=46125&f=63

launcher create

->nano launcher.sh

#!/bin/sh

#launcher.sh

#navigate to home directory, then to this directory,then execute python script,

# then back home

cd /

cd /home/pi/fawkes

sudo python xsensors.py

cd /

We need to make the launcher script an executable, which we do with this command

->chmod 755 launcher.sh

Now test it, by typing in:

->sh launcher.sh

This should run your Python code.

We create the logs directory by

logs directory

->mkdir logs

we open the crontab menu

crontab setup

->sudocrontab -e

# every 5 minutes

#\*/5 \* \* \* \* /home/pi/fawkes/launcher.sh >/home/pi/fawkes/logs/cronlog.log 2>&1

#@reboot sh /home/pi/fawkes/launcher.sh >/home/pi/fawkes/logs/cronlog.log 2>&1

#What this does is rather than executing the launcher script at a specific time, it will execute it once upon startup.

# at 6:22 EST, 3:22 PST (we are on PST); 15:22

22 15 \* \* \* /home/pi/marktwainbot/launcher.sh >/home/pi/logs/cronlog.log 2>&1

Unplug the power or just type in:

->sudo reboot

Wait for startup and see if your script automatically launches.

If it doesn't work, check out the log file:

->cd logs

->cat cronlog

This will show you any errors that you might have.

Once setup your Python script will run whenever your reboot or start-up your Pi. There may be times when you reboot and don’t want the script running. To stop it you can find out its process number and “kill” it. To do this type :

->ps aux | grep /home/pi/MyScript.py

This should give you a line starting with “root” and ending in the path to your script. Immediately after the “root” should be a process number. For example :

root 1863 0.0 1.0 24908 4012 ? Sl 19:45 0:00 python /home/pi/MyScript.py

In this case we can stop the process using :

->sudo kill 1863

**Step 11:**Git Hub setup:

->git@github.com:sovan88/indoor.git

->gitconfig --global user.name "sovan88"

->gitconfig --global user.email "sovan.cse@gmail.com"

->ssh-keygen -C [sovan.cse@gmail.com](mailto:sovan.cse@gmail.com)

cat /home/pi/.ssh/id\_rsa.pub

local repository setup :

->gitinit

->gitadd .

->git commit -m 'initial repository added'

remote repository setup:

->git remote add origin git@github.com:sovan88/indoor.git

->git push origin master

->git pull origin master

cd ~/projects/your-raspberry-project-top-level-dir

makeclean # Or clean it manually using rm (rm ./\*.o./\*.pyc)

gitinit # Create new repo here

git add . # Add source files to the staging index

git status # Verify if it's OK

git commit -a -m "Initial import" # Fix application's source changes

git add remote https://github.com/user/your-raspberry-project.git

git push -u origin master # Sends the sources to your github repo

git pull &&git push &&git status # Now origin/master is your tracking branch

Auto run commands:

->nanogit.run

Enter the below code:

# Specify the files to be backed up.

# Below command will backup everything inside the project folder

git add .

# You can also use specific files using the command git add file1 file2 ..

# Committing to the local repository with a message containing the time details

curtime=`date`

git commit -m "Automatic Backup @ $curtime"

# Push the local snapshot to a remote destination

git push origin master

Make the file git.run executable

->chmod +x git.run

update the crontab menu:

->crontab -e

0 0 \* \* 0 cd /path/working\_dir&& ./git.run

The contents will be uploaded to the github repository

**23/05/2017:: DATAPLICITY**

1.installxrdp in raspberrypi using raspberrypi  ssh terminal.

2.Open Remote pc connection.

3.put the IP of raspberrypie in it.

4.connect.

[5.It](http://5.it/) will open the raspberrypi.

6.Register in dataplicity with [mail-incorrectsms@gmail.com](mailto:mail-incorrectsms@gmail.com) password- suraj9775

7.opendataplicity website or mobile app of dataplicity.

8.Sign in with above user id and password.

9.Raspberrypi can be accessed allover the world using remote connection through dataplicity when only remote pc is on.

[10.It](http://10.it/) will open only raspberrypissh terminal

192.168.195.104

Mail id 🡪suvo.cst05@gmail.com

Pass🡪 suvo1234

**08/06/2017 : UPLOADING DATA TO THINGS SPEAK WEBSITE**

<http://www.iotleague.com/d-i-y-how-to-upload-your-raspberry-pi-sensor-data-to-thingspeak-website/>

**Step 1:** Signup for Thingspeak

Go to [www.thingspeak.com](http://www.thingspeak.com)

**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.

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.

Click on “Save Channel” button to save all of your settings.

**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.

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.

Edit the line 19 by using **CPU\_Temp** instead of **temp**.

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

If the code runs properly you should see “200 ok” and something like “58.30” (CPU temperature value)

In case if there are any errors uploading the data, you will receive “connection failed” message.

**Step 6:** Check Thingspeak API and Confirm data transfer

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

**ANDROID APP TO CONNECT TO THINGSSPEAK:**

<http://noobtechiespeaks.blogspot.in/2014/10/android-app-to-connect-to-thingspeak.html>

### What is this APP thing?

In this tutorial we will make use of the Thingspeak Apps. You may ask what is this App. It is nothing but services provided by Thingspeak to communicate with your devices. These apps can be accessed by POST, GET, PUT and DELETE HTTP requests. You can check out my previous [post](http://noobtechiespeaks.blogspot.in/2014/10/iot-using-beaglebone-black-and.html) on how to make an HTTP request to upload data to your Thingspeak Channel. In this tutorial we will make use of the following Thingspeak Apps.

* Talkback - This App is used to control your device based on some decision made on the basis of your sensor data in your channel. Using this app you can store commands which your device may fetch at regular intervals or when an event occurs to execute.
* React - This is a decision making App. We can use this app to trigger an event based on your Thingspeak data channel. The field on which decision is to be made can be String, Location or Numeric data. There are a whole lot of logic that you can execute based on the data. In this project I will use my android app to get my current GPS location and upload it to my channel based on which an event will be triggered.
* ThingHTTP - This App is used to do an HTTP request from within Thingspeak. So we can use this App to trigger our Talkback App in turn to update the commands.

### Setting up Thingspeak

It is assumed that you already have a Thingspeak channel set up. If not follow my previous [post.](http://noobtechiespeaks.blogspot.in/2014/10/iot-using-beaglebone-black-and.html)

Before moving on please read the documentation each of this App so that you have a better understanding of how they work.

* Open your Thingspeak account and click on the Apps tab.
* Now click on "New Talkback".
* Now Fill in the Name and select your channel and set a default command by clicking add new command. For now we will add "CLOSE".
* Now again click on the Apps tab and go to the "ThingHTTP" App.
* Now click on "New ThingHTTP" and fill  up like this and save your ThingHTTP.
* Now create another ThingHTTP App for closing. Like this.
* Now click on App and now go to the React App.
* Click on :"New React" and fill up like this. Save your React App.
* Now create another React App in similar manner for closing like this.
* While doing all the above steps give the parameters carefully like latitude, longitude,api\_key etc. specific to your Apps.
* Now you are ready to go. Lets create the Android App.

### Setting up App Inventor 2

Since you are developing an app for android it is already assumed that you have a google account. Now go to the website [http://ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu/). Use your google account to log in and create a new project. App Inventor is an online development platform for creating android apps without any knowledge of programming. It is based on graphical block development. You only need to arranged the required blocks just like a puzzle. Follow the [getting started guide](http://appinventor.mit.edu/explore/get-started.html) to have a clear understanding.

* After you have opened [App Inventor 2](http://ai2.appinventor.mit.edu/) and created a project bring the following objects to the canvas. Check this image to get a list.

* Change the Button1 Text to Turn OFF and the Button2 Text to Turn ON
* Change the Web1 url to https://api.thingspeak.com/talkbacks/<your Talkback ID>/commands/<your command ID>
* Change the Web2 url to https://api.thingspeak.com/update
* Change the Clock1 Time interval to 10000.
* Change LocationSensor1 Time interval to 10000.
* Now click on "Blocks" on the top left corner of your screen.
* If you have gone through the Getting started portion then you will be familiar with the blocks. So arrange the blocks as shown below.
* Replace Web1.PutText api\_key to your Talkback API\_Key.
* Replace Web2.PostText api\_key to your Channel API\_Key.
* Now click on "Build"->"App(save .apk to my Computer)". It will download the file to your PC. Transfer the file to your android phone and install the app.
* Now turn ON you GPS and WiFi/Data connection.
* Now open the Talkback App on your Thingspeak account. You will find the default command as CLOSE.
* Now open the app in your android phone and wait for sometime untill the location is displayed.
* Now refresh the Talkback page. If you are at or near the location you specified in your React App, the command will automatically change to OPEN. You can also manually TURN ON/OFF using the buttons provided.