Objective 2

Aim: To Setup Weather Station using Raspberry-pi Sense Hat.

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.

Also, in addition the Sense HAT is an add-on board for Raspberry Pi, made especially for the Astro Pi mission – it launched to the International Space Station in December 2015 – and is now available to buy. The Sense HAT has an 8×8 RGB LED matrix, a five-button joystick and includes the sensors like Temperature, Humidity, Barometric pressure, Gyroscope, Accelerometer, and Magnetometer. As we are interested in Development of Weather station module, we will consider only sensors like: Temperature, Humidity, and Barometric Pressure to link it with Time-Series.

The Hardware Devices used are described as follows:

1. **Raspberry Pi 3 Model B.**

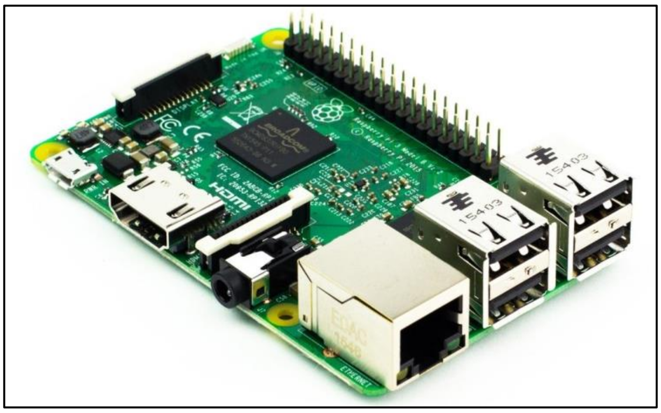


Fig.1 Raspberry pi 3 Model B

Specifications:

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port

- Ethernet port
- Combined 3.5mm audio jack and composite video
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core.

2. **Sense Hat**

Specifications:

The Raspberry Pi Sense HAT is attached on top of the Raspberry Pi via the 40 GPIO pins to create an 'Astro Pi'. The Sense HAT has several integrated circuit based sensors can be used for many different types of experiments, applications, and even games. And it's being used in conjunction with the Raspberry Pi Foundation to perform science experiments aboard the International Space Station (ISS).

The 8x8 LED Matrix enables you to display the data from the various sensors, it can show you which way is geomagnetic North by programming a compass using the magnetometer, or simply be used to play games like Tetris, Pong and Snake with the joystick. The joystick can also be used to enable a human user to interact with the programs running on the Raspberry Pi Sense HAT.

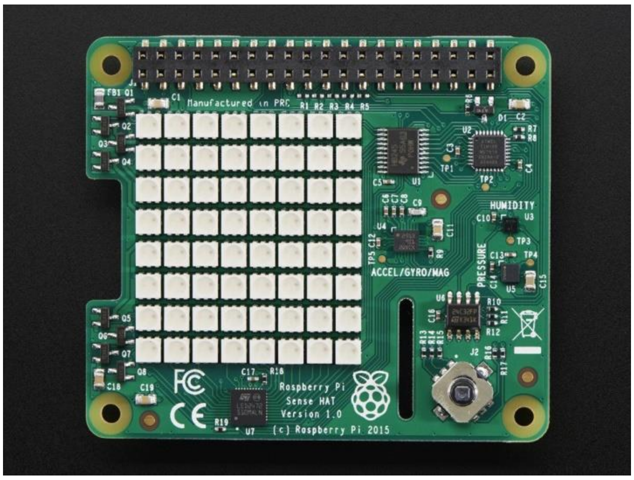


Fig.2 Sense Hat Module

TECHNICAL DETAILS

- Gyroscope.
- Angular rate sensor (dps): ~245/500/2000.
- Accelerometer.
- Linear acceleration sensor (g): ~2/4/8/16.
- Magnetometer.

- Magnetic sensor (gauss): ~4/8/12/16.
- Barometer: 260 - 1260 hPa absolute range (accuracy depends on the temperature and pressure, ~0.1 hPa under normal conditions).
- Temperature sensor: Accurate to ~2°C in the 0-65°C range.
- Relative humidity sensor: Accurate to ~4.5% in the 20-80%rH range, accurate to ~0.5°C in 15-40°C range.
- 8x8 LED matrix display.
- Small 5 button joystick.

Working:

First of all we have installed raspbian debian OS on raspberry pi which is a free operating system based on Debian optimized for the Raspberry Pi hardware.

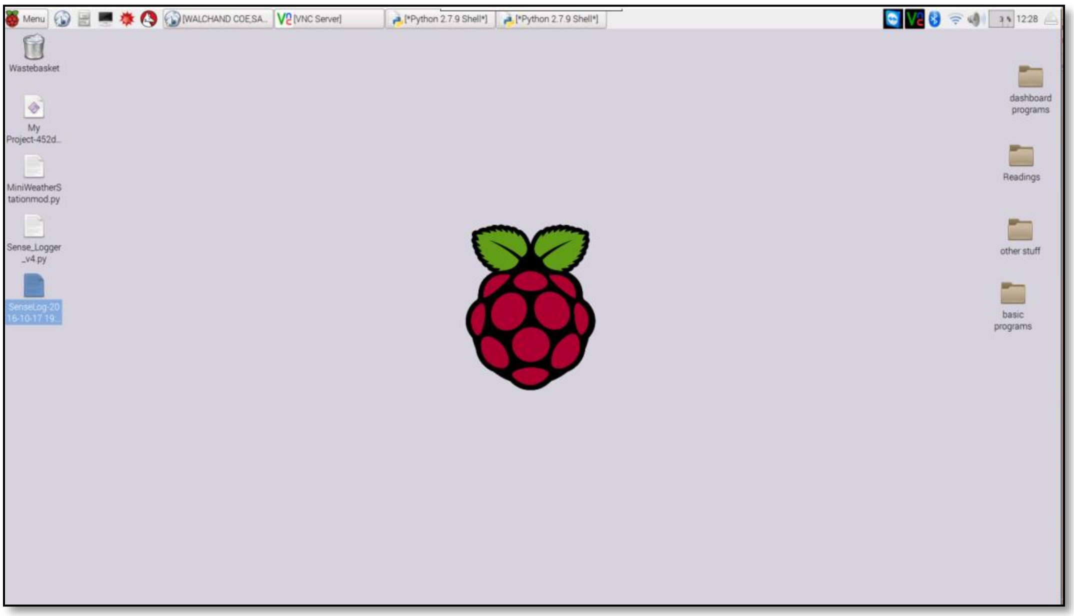


Fig. Raspbian Debian Os

An operating system is the set of basic programs and utilities that make your Raspberry Pi run. Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more. Then we install Sense Hat on top of Rapsberry pi board.

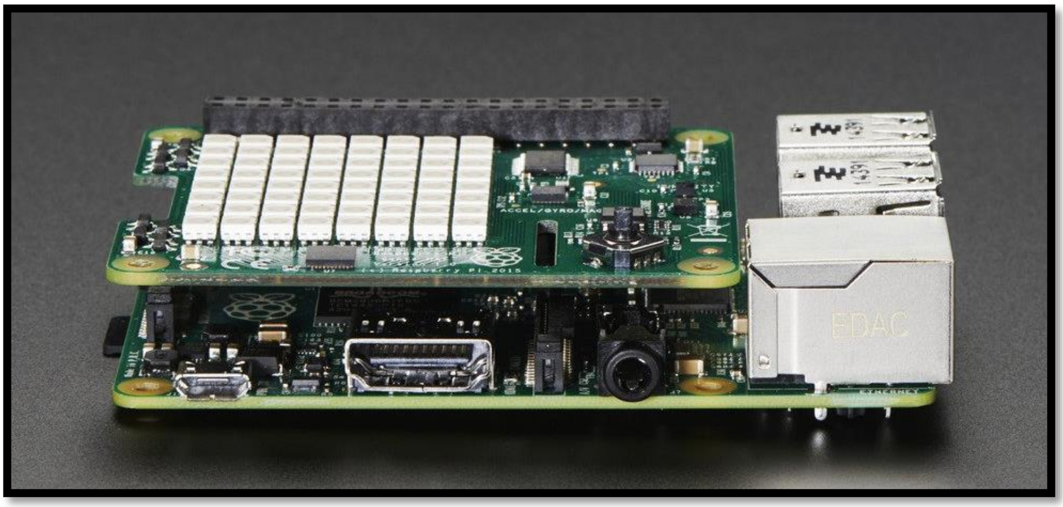


Fig. Sense Hat Setup.

After that the software configuration steps over raspbian are as mentioned below:

In order to work correctly, the Sense HAT requires an up-to-date kernel, I2C to be enabled, and a few libraries to get started.

1. Ensure your APT package list is up-to-date:

```
sudo apt-get update
```

2. Next, install the sense-hat package which will ensure the kernel is up-to-date, enable I2C, and install the necessary libraries and programs:

```
sudo apt-get install sense-hat
```

3. Finally, a reboot may be required if I2C was disabled or the kernel was not up-to-date prior to the install:

```
sudo reboot
```

4. RTIMULIB

RTIMULib is a C++ and Python library that makes it easy to use 9-dof and 10-dof IMUs with embedded Linux systems. A pre-calibrated settings file is provided in /etc/RTIMULib.ini, which is also copied and used by sense-hat. The included examples look for RTIMULib.ini in the current working directory, so you may wish to copy the file there to get more accurate data.

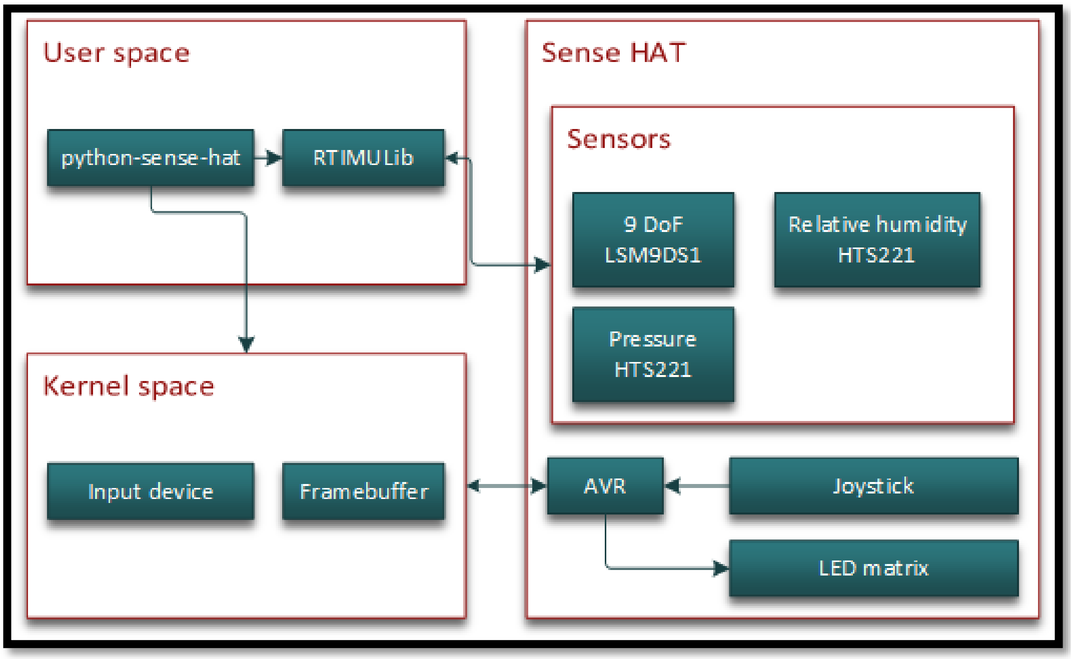


Fig. Sense Hat Architecture.

Coding Approach

For gathering Time series weather data, we need only 3 parameters from Sense Hat:

1. Temperature (°Celsius)
2. Relative Humidity (percentage) and
3. Barometric pressure (millibars).

For Gathering this data, two approaches are being used:

1. Gathering the data and sending the results to: **Google Spreadsheet on your Google Drive**.

2. Storing the sensed data on raspbian -OS as csv **(Comma Separated Values)** file.

- **Approach 1:** Gathering the data and sending the results to: **Google Spreadsheet on your Google Drive**.

A core part of the "**Internet of Things**" movement is the idea of devices that gather data and send it to the Internet. That data is then acted on or observed for later. It's a simple concept and has been going on for a while but lately it's been getting cheaper and easier to do.

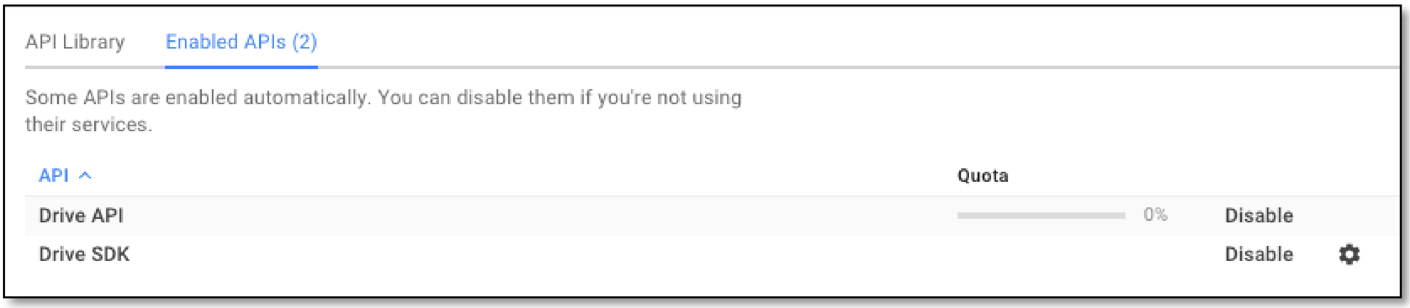
Google Spreadsheets

You can output data to a Google Spreadsheet application. You will need to setup OAuth with Google, and create a JSON file. The steps are as mentioned below:

Using OAuth2 for Authorization (OAuth Credentials)

1. Head to Google Developers Console and create a new project (or select the one you have.)

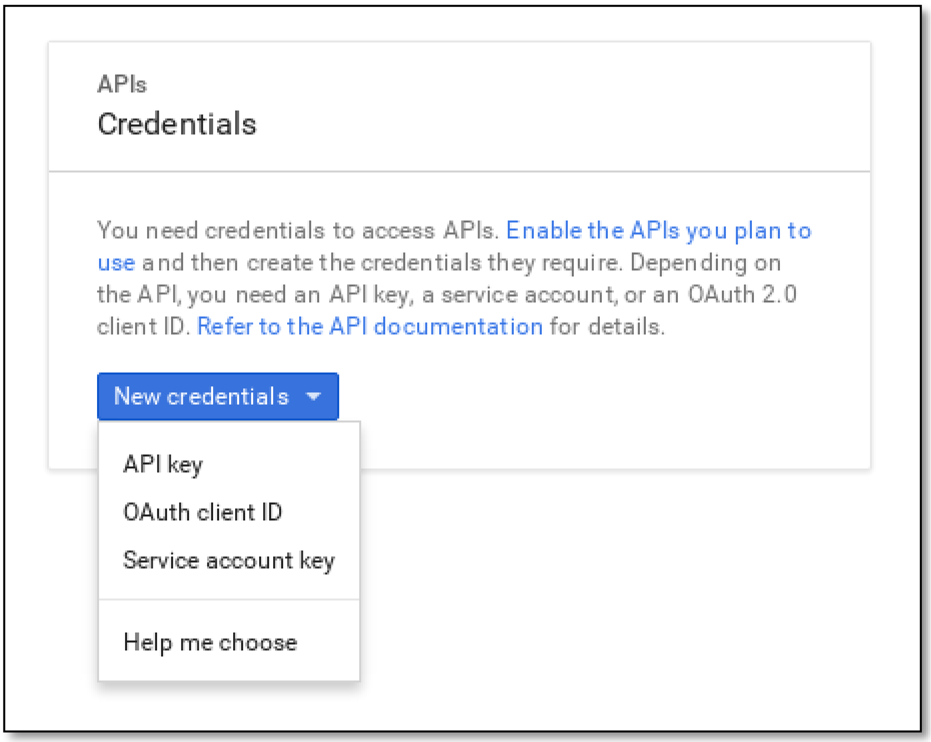
2. Under "API & auth", in the API enable "Drive API".



Some APIs are enabled automatically. You can disable them if you're not using their services.

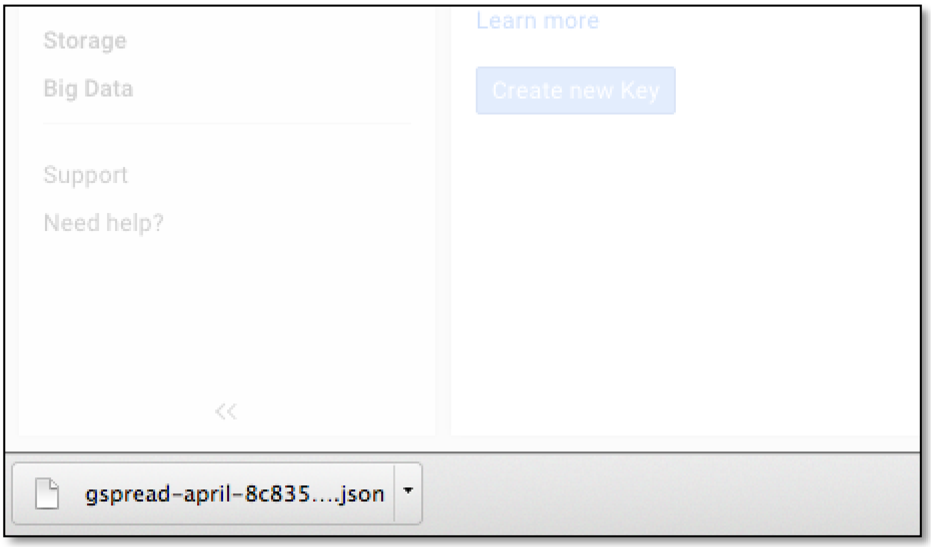
API	Quota		
Drive API	0%	Disable	
Drive SDK		Disable	⚙

3. Go to "Credentials" and choose "New Credentials > Service Account Key".



You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. Refer to the API documentation for details.

4. You will automatically download a JSON file with this data.



5. This is how this file may look like:

```
{
    "private_key_id": "2cd … ba4",
    "private_key": "-----BEGIN PRIVATE KEY-----\nNrDyLw … jINQh/9\n-----END PRIVATE
KEY-----\n",
    "client_email": "473 … hd@developer.gserviceaccount.com",
    "client_id": "473 … hd.apps.googleusercontent.com",
    "type": "service_account"
}
```

You'll need *client_email* and *private_key*.

6. Install oauth2client:

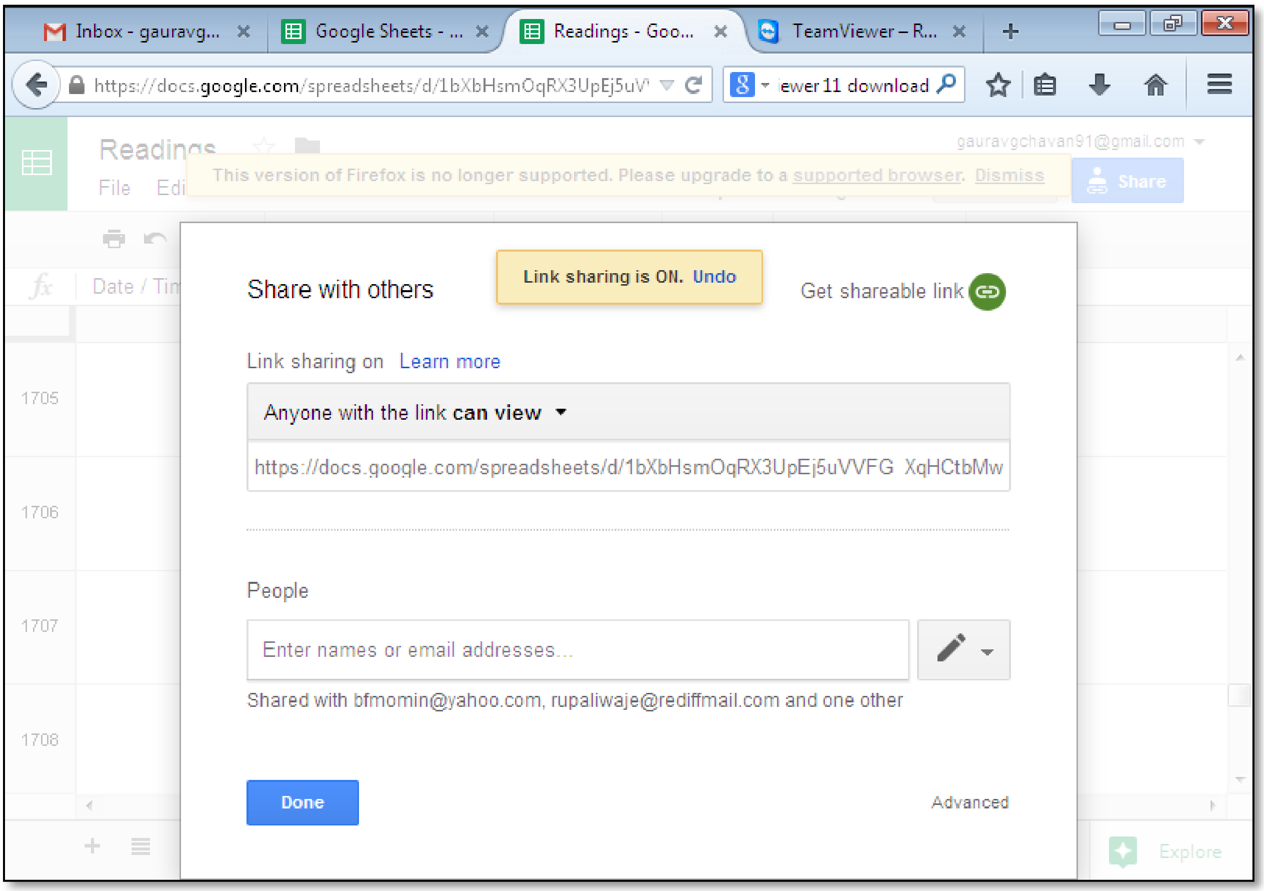
```
pip install --upgrade oauth2client
```

7. Depending on your system setup you may need to install PyOpenSSL:

```
pip install PyOpenSSL
```

You will want to store the generated **JSON** file in the **MiniWeatherStation.py** folder. One thing you will need to is open up that **OAuth JSON** file and look for "**client_email**". It should look like this:

"client_email": "1985453359310-asdlkjried8ss98eeEic@developer.gserviceaccount.com", Take note of that email address value and go to your Google spreadsheet in a web browser. Using the **File -> Share...** menu item share the spreadsheet with **read access** to the email address found above.



This version of Firefox is no longer supported. Please upgrade to a supported browser. Dismiss

Share with others

Link sharing is ON. Undo

Get shareable link

Link sharing on Learn more

Anyone with the link **can view**

https://docs.google.com/spreadsheets/d/1bXbHsmOqRX3UpEj5uVVFG XqHCtbMw

People

Enter names or email addresses...

Shared with bfmomin@yahoo.com, rupaliwaje@rediffmail.com and one other

Done

Advanced

8. Next, open up the **WeatherStationmod.py** file and edit:

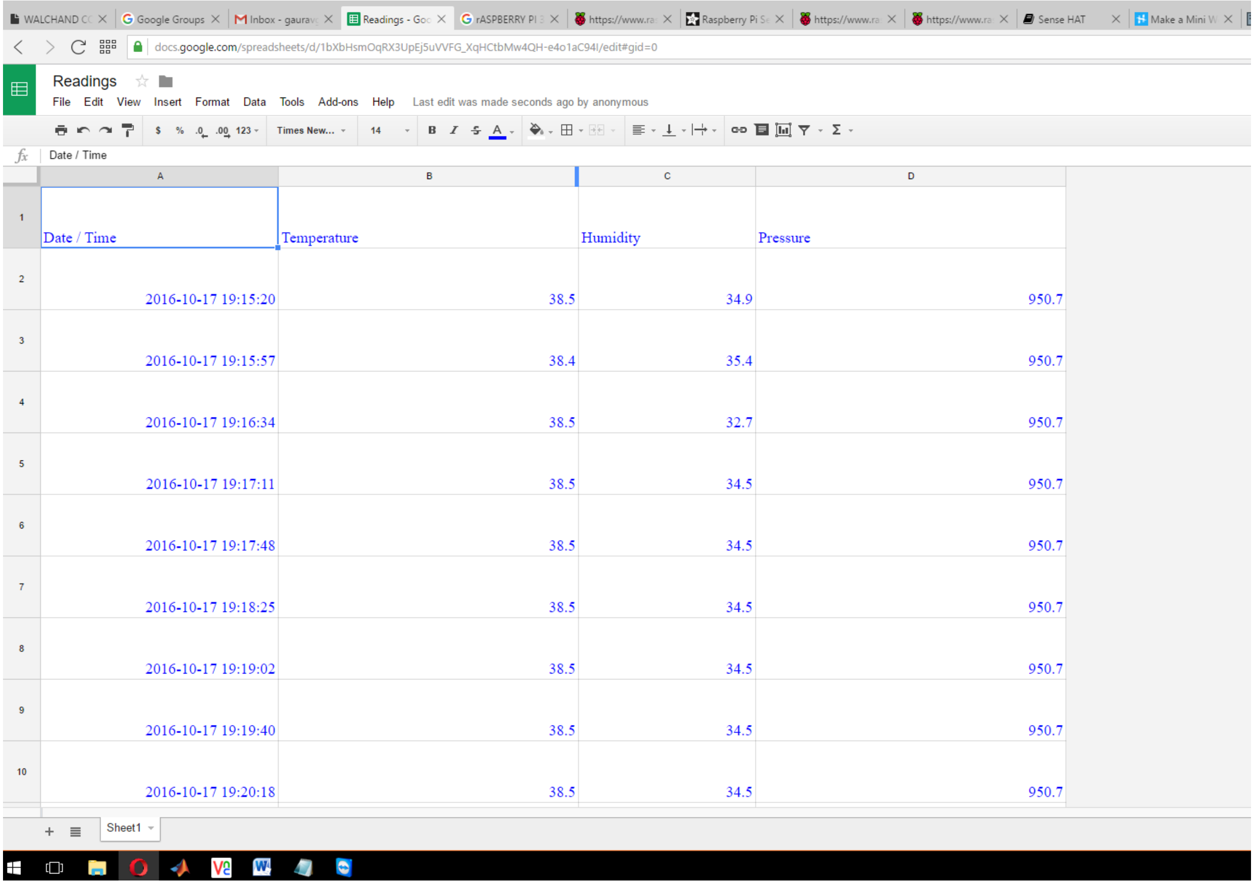
```
sudo nano WeatherStationmod.py
```

Replace the **GDOCS_OOAUTH_JSON** value with the name of your **JSON** file you downloaded. Set the **GDOCS_SPREADSHEET_NAME** with the name of your sheet. Save it.

Type in :

```
sudo python WeatherStationmod.py
```

If all your information is correct, it will start running and adding rows to your spreadsheet every 10 seconds.



	A	B	C	D
1	Date / Time	Temperature	Humidity	Pressure
2	2016-10-17 19:15:20	38.5	34.9	950.7
3	2016-10-17 19:15:57	38.4	35.4	950.7
4	2016-10-17 19:16:34	38.5	32.7	950.7
5	2016-10-17 19:17:11	38.5	34.5	950.7
6	2016-10-17 19:17:48	38.5	34.5	950.7
7	2016-10-17 19:18:25	38.5	34.5	950.7
8	2016-10-17 19:19:02	38.5	34.5	950.7
9	2016-10-17 19:19:40	38.5	34.5	950.7
10	2016-10-17 19:20:18	38.5	34.5	950.7

Code:

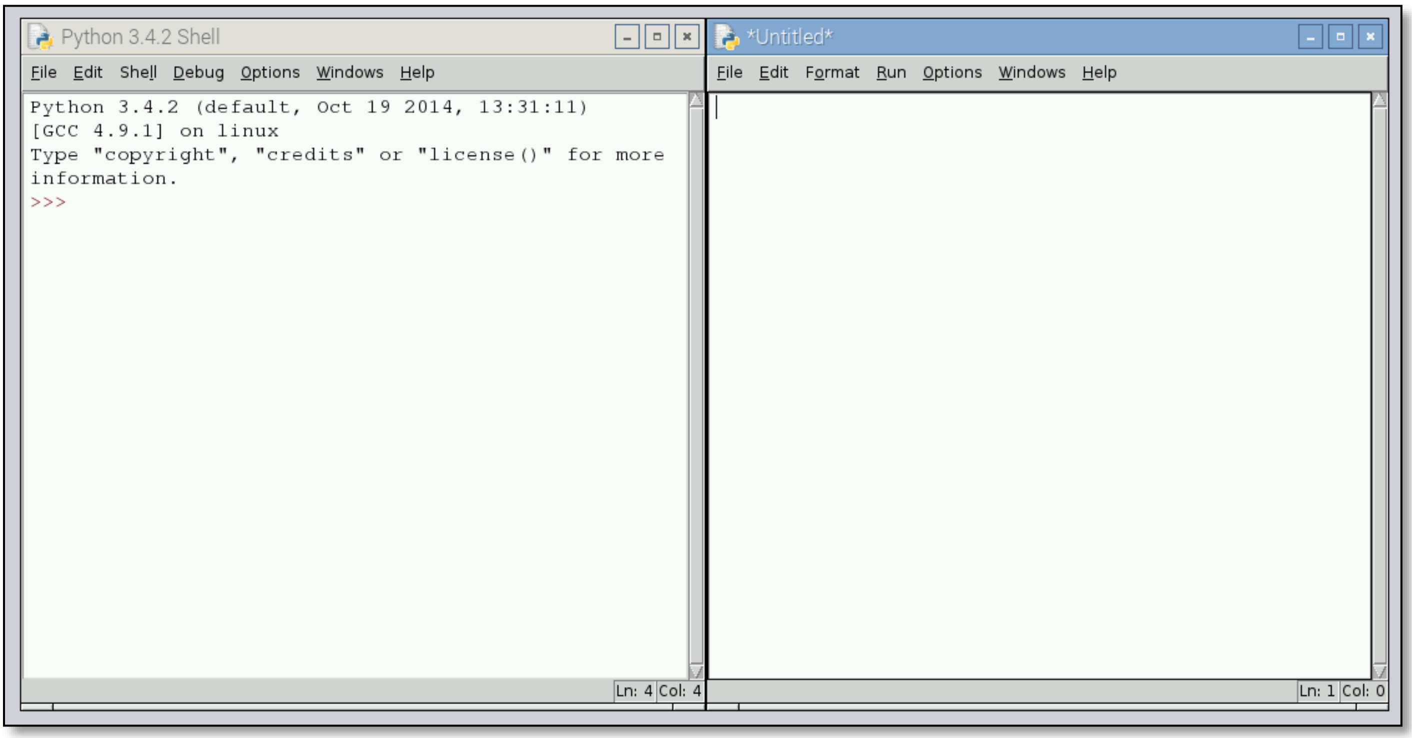
```
#!/usr/bin/python
import json
import sys
import time
import datetime
# libraries
import sys
import urllib2
import json
import gspread
from oauth2client.client  import SignedJwtAssertionCredentials
from sense_hat import SenseHat

# Oauth JSON File
GDOCS_OAUTH_JSON       = 'My Project-452d7668e39b.json'

# Google Docs spreadsheet name.
GDOCS_SPREADSHEET_NAME = 'Readings'

# How long to wait (in seconds) between measurements.
FREQUENCY_SECONDS=5

def login_open_sheet(oauth_key_file, spreadsheet): """Connect to Google Docs spreadsheet and return the first worksheet."""
```

```
try:
json_key = json.load(open(oauth_key_file))
credentials=SignedJwtAssertionCredentials(json_key['client_email'],json_key['private_key'], ['https://spreadsheets.google.com/feeds'])
gc = gspread.authorize(credentials)
worksheet = gc.open(spreadsheet).sheet1
return worksheet
except Exception as ex:
print 'Unable to login and get spreadsheet.  Check OAuth credentials, spreadsheet name, and make sure spreadsheet is shared to the client_email address in the OAuth .json file!'
print 'Google sheet login failed with error:', ex
sys.exit(1)


sense = SenseHat()
sense.clear()
print 'Logging sensor measurements to {0} every {1} seconds.'.format(GDOCS_SPREADSHEET_NAME, FREQUENCY_SECONDS)
print 'Press Ctrl-C to quit.'
worksheet = None


while True:
# Login if necessary.
if worksheet is None:
worksheet = login_open_sheet(GDOCS_OAUTH_JSON, GDOCS_SPREADSHEET_NAME)
```

```
    # Attempt to get sensor reading.
    temp = sense.get_temperature()
    temp = round(temp, 1)
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)
    pressure = sense.get_pressure()
    pressure = round(pressure, 1)

    # 8x8 RGB
    # sense.clear()
    info = 'Temperature (C): ' + str(temp) + 'Humidity: ' + str(humidity) + 'Pressure: ' + str(pressure)
sense.show_message(info, text_colour=[255, 0, 0])

    # Append the data in the spreadsheet, including a timestamp
    try:
        worksheet.append_row((datetime.datetime.now(), temp,humidity,pressure))
    except:
        # Error appending data, most likely because credentials are stale.
        # Null out the worksheet so a login is performed at the top of the loop.
        print 'Append error, logging in again'
        worksheet = None
        time.sleep(FREQUENCY_SECONDS)
        continue
    # Wait 30 seconds before continuing
    print 'Wrote a row to {0}'.format(GDOCS_SPREADSHEET_NAME)
    time.sleep(FREQUENCY_SECONDS)
```

Approach 2: Storing readings in Device itself as csv file

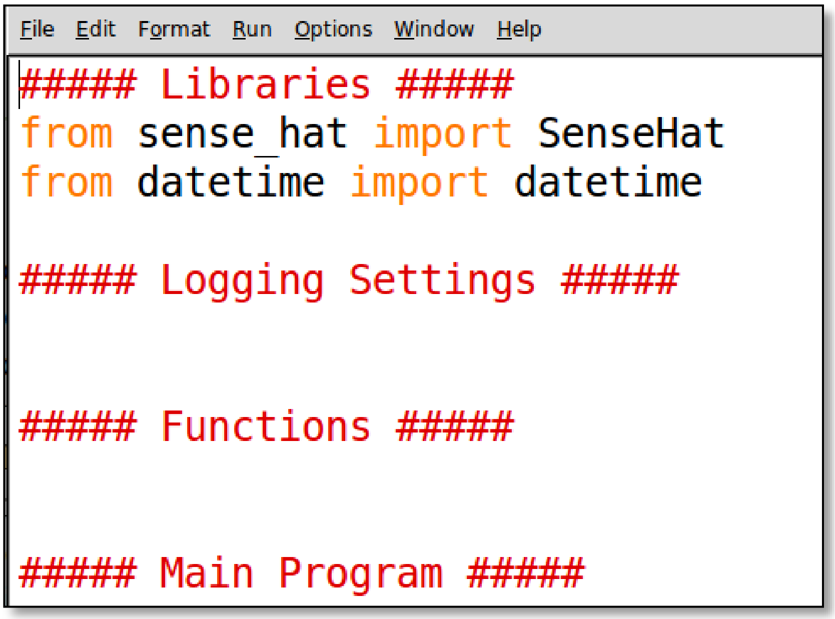
In this approach, python script is used which takes the readings from device and stores it in comma separated values file. To begin your script you need to boot your Raspberry Pi into desktop mode and run Idle for Python 3 from the programming section of the menu. Once Idle has loaded you will need to select **File** and then **New File** which will load a separate window in which you can write your code.



Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Ln: 4 Col: 4
Untitled
File Edit Format Run Options Windows Help
Ln: 1 Col: 0

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more
information.
>>>
```

In your right hand window, add the following lines of python code. The lines starting with a # symbol are **comments** and are ignored by the computer. We have used comments here to break the code into four sections, which will make it easier to build your program as it gets more complex.



```
##### Libraries #####
from sense_hat import SenseHat
from datetime import datetime

##### Logging Settings #####


##### Functions #####


##### Main Program #####
```

- The first section, **Libraries**, is where you will import code that will give your program extra abilities. The line from sense_hat import SenseHat allows your program to use the Sense-HAT hardware. The line from datetime import datetime allows your program to use the time module.
- The section headed **Logging Settings** is where you will be able to control different features of your logger program.
- The third section, **Functions**, will contain short "chunks" of reusable code which do a specific job, such as writing the current data to a file.
- The final section, **Main Program**, is the part of your code which uses each of the functions in the right sequence to run the whole program.

In order to get data from the Sense HAT you will need to write a function called **get_sense_data** which will check each sensor in turn and store the sensor data in a list. The function should be added to the **Functions** section.

```
def get_sense_data():

 sense_data=[]



 sense_data.append(sense.get_temperature_from_humidity())

 sense_data.append(sense.get_temperature_from_pressure())

 sense_data.append(sense.get_humidity())

 sense_data.append(sense.get_pressure())
```

The first line defines your function name, and the second sets up an empty **list** structure into which you will add your collected data.

The next four lines get data from some of the sensors and adds (or appends) them to the `sense_data` list.

Next we'll need to add some lines to your **Main Program** Section, this will need to do two things:

- create a sense object, which represents the Sense HAT
- repeatedly **get_sense_data** from the sensors and display it

Add the following code to the **Main Program** section:

```
sense = SenseHat()

while True:

  sense_data = get_sense_data()

  print(sense_data)
```

3. Another function you will need is the file_setup function which will create a list of headings that will be written to the file before any data. The function is shown below and needs to be added to your Functions section.

```
def file_setup(filename):

  header =["temp_h","temp_p","humidity","pressure",

  "pitch","roll","yaw",

  "mag_x","mag_y","mag_z",

  "accel_x","accel_y","accel_z",

  "gyro_x","gyro_y","gyro_z",

  "timestamp"]

  with open(filename,"w") as f:

   f.write(",".join(str(value) for value in header)+ "\n")
```

This function is slightly different to the previous as it needs an input (or parameter) in order to work; in this case the input has been called filename. When the main program calls this function it must also give the function the name of the file to write to. If it were called like this file_setup("output.csv") the function would create output.csv

The function itself creates a list of header names called header. It then opens a file in write mode (which will overwrites any previous data) and refers to that file as f. whilst the file is open it joins all the list headings together using commas and writes that line to the file.

4. The two functions and the settings you added now need to be used in the main program.

Straight after the lines that read:

```
##### Main Program #####

sense = SenseHat()
```

add the following:

```
batch_data= []

if FILENAME == "":

  filename = "SenseLog-"+str(datetime.now())+".csv"

else:

  filename = FILENAME+"-"+str(datetime.now())+".csv"

file_setup(filename)
```

The first line here creates an empty list that the program will keep adding sense_data lines to until it reaches 50 (or whatever value is set by WRITE_FREQUENCY). The if/else block checks whether a FILENAME has been set, if it hasn't then the default of "SenseLog" is used. The current date and time is also added to the filename.Finally the file_setup functions is called and given the filename that was decided upon in the previous if / else block.

5. The last step is to change some of the logic inside the while True: loop.

You need to make it collect sense_data

Then use the log_data function to convert the sense_data into csv form and add the the current batch_data. Once the data is logged, the program checks whether the size of batch_data exceeds the WRITE_FREQUENCY setting, if so the data is written to a file and batch_data is reset.

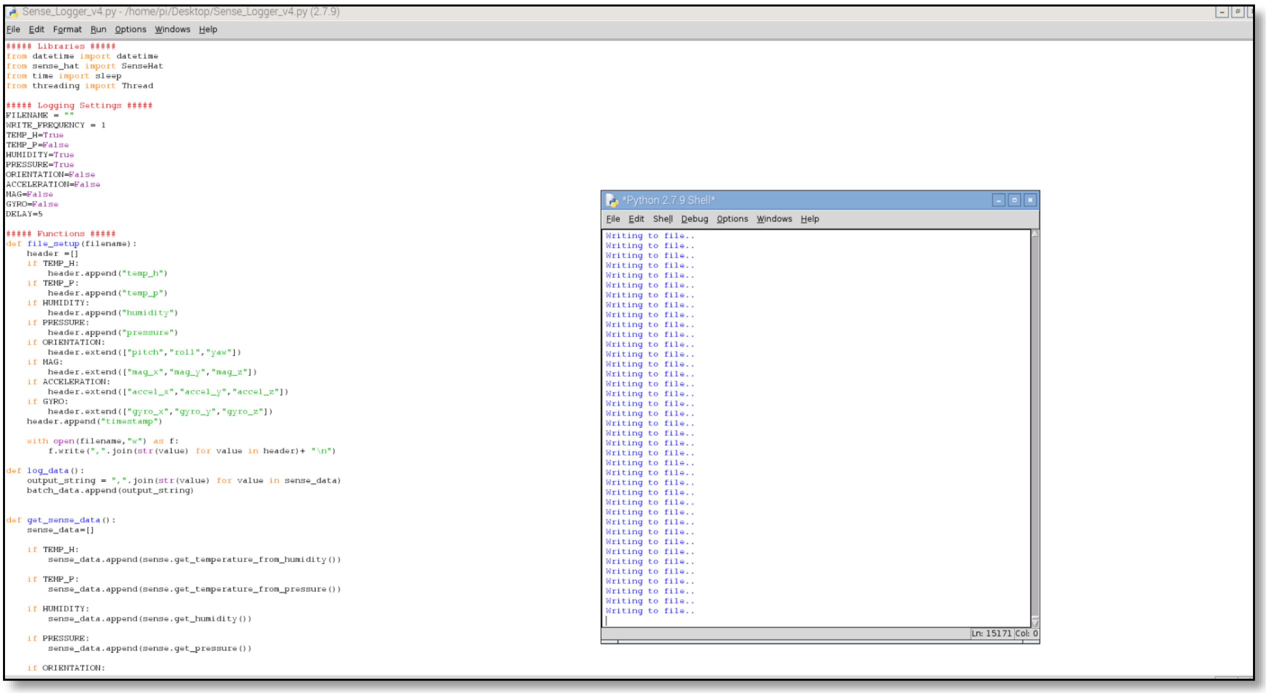
Your while True: loop should be updated to look like this:

```
while True:

  sense_data = get_sense_data()

  log_data()



  if len(batch_data) >= WRITE_FREQUENCY:

   print("Writing to file..")

    with open(filename,"a") as f:

    for line in batch_data:

              f.write(line + "\n")

          batch_data = []
```

The line print("Writing to file..") is optional, but it will show whenever data is being written.The line with open(filename,"a") as f: opens the file in append mode which adds the data at the end point of the file rather than overwriting. You can check your code against a full code listing here. When you running the program you should simply see the messages saying Writing to file.. every so often.

```
Writing to a file
```

You can stop logging by pressing Ctrl+C


Sense_Logger_v4.py - /home/pi/Desktop/Sense_Logger_v4.py (2.7.9)
File Edit Format Run Options Windows Help
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Ln: 15171 Col: 0

```
##### Libraries #####
from datetime import datetime
from sense_hat import SenseHat
from time import sleep
from threading import Thread

##### Logging Settings #####
FILENAME = ""
WRITE_FREQUENCY = 1
TEMP_H=True
TEMP_P=False
HUMIDITY=True
PRESSURE=True
ORIENTATION=False
ACCELERATION=False
MAG=False
GYRO=False
DELAY=5

##### Functions #####
def file_setup(filename):
    header =[]
    if TEMP_H:
        header.append("temp_h")
    if TEMP_P:
        header.append("temp_p")
    if HUMIDITY:
        header.append("humidity")
    if PRESSURE:
        header.append("pressure")
    if ORIENTATION:
        header.extend(["pitch","roll","yaw"])
    if MAG:
        header.extend(["mag_x","mag_y","mag_z"])
    if ACCELERATION:
        header.extend(["accel_x","accel_y","accel_z"])
    if GYRO:
        header.extend(["gyro_x","gyro_y","gyro_z"])
    header.append("timestamp")

    with open(filename,"w") as f:
        f.write(",".join(str(value) for value in header)+ "\n")

def log_data():
    output_string = ",".join(str(value) for value in sense_data)
    batch_data.append(output_string)


def get_sense_data():
    sense_data=[]

    if TEMP_H:
        sense_data.append(sense.get_temperature_from_humidity())

    if TEMP_P:
        sense_data.append(sense.get_temperature_from_pressure())

    if HUMIDITY:
        sense_data.append(sense.get_humidity())

    if PRESSURE:
        sense_data.append(sense.get_pressure())

    if ORIENTATION:
```

```
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
Writing to file..
```

Code:

Libraries

from datetime import datetime

from sense_hat import SenseHat

from time import sleep

from threading import Thread

Logging Settings

FILENAME = ""

WRITE_FREQUENCY = 1

TEMP_H=True

TEMP_P=False

HUMIDITY=True

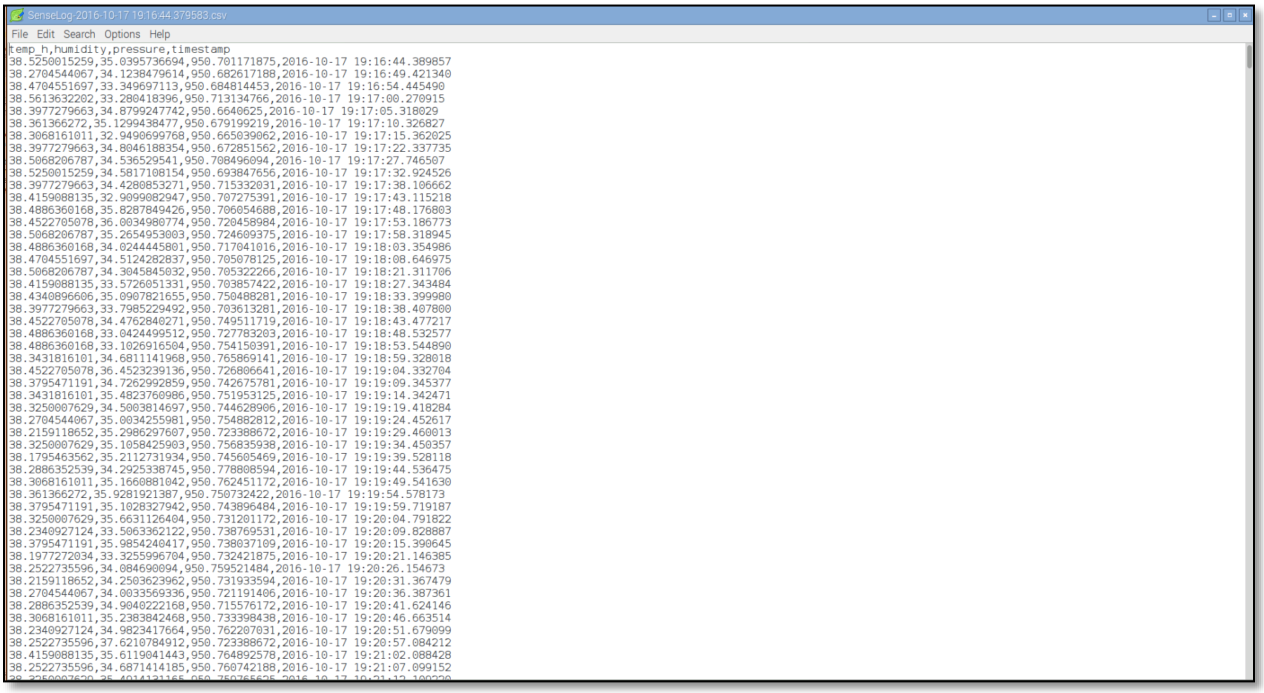
PRESSURE=True

ORIENTATION=False

ACCELERATION=False

```
MAG=False
GYRO=False
DELAY=5

##### Functions #####
def file_setup(filename):
    header =[]
    if TEMP_H:
        header.append("temp_h")
    if TEMP_P:
        header.append("temp_p")
    if HUMIDITY:
        header.append("humidity")
    if PRESSURE:
        header.append("pressure")
    if ORIENTATION:
        header.extend(["pitch","roll","yaw"])
    if MAG:

        header.extend(["mag_x","mag_y","mag_z"])
    if ACCELERATION:
        header.extend(["accel_x","accel_y","accel_z"])
    if GYRO:
        header.extend(["gyro_x","gyro_y","gyro_z"])
    header.append("timestamp")
    with open(filename,"w") as f:
        f.write(",".join(str(value) for value in header)+ "\n")
def log_data():
    output_string = ",".join(str(value) for value in sense_data)
    batch_data.append(output_string)
def get_sense_data():
    sense_data=[]

    if TEMP_H:
        sense_data.append(sense.get_temperature_from_humidity())

    if TEMP_P:
        sense_data.append(sense.get_temperature_from_pressure())

    if HUMIDITY:
        sense_data.append(sense.get_humidity())

    if PRESSURE:
        sense_data.append(sense.get_pressure())

    if ORIENTATION:
        o = sense.get_orientation()
        yaw = o["yaw"]
        pitch = o["pitch"]
        roll = o["roll"]
        sense_data.extend([pitch,roll,yaw])
    if MAG:
        mag = sense.get_compass_raw()
        mag_x = mag["x"]
        mag_y = mag["y"]
        mag_z = mag["z"]
        sense_data.extend([mag_x,mag_y,mag_z])

    if ACCELERATION:
        acc = sense.get_accelerometer_raw()
        x = acc["x"]
        y = acc["y"]
        z = acc["z"]
        sense_data.extend([x,y,z])

    if GYRO:
        gyro = sense.get_gyroscope_raw()
        gyro_x = ["x"]
        gyro_y = ["y"]
        gyro_z = ["z"]
        sense_data.extend([gyro_x,gyro_y,gyro_z])
    sense_data.append(datetime.now())
    return sense_data
def timed_log():
    while True:
        log_data()
        sleep(DELAY)


##### Main Program #####

sense = SenseHat()

batch_data= []

if FILENAME == "":

    filename = "SenseLog-"+str(datetime.now())+".csv"

else:

    filename = FILENAME+"-"+str(datetime.now())+".csv"

file_setup(filename)

if DELAY > 0:

    sense_data = get_sense_data()

    Thread(target= timed_log).start()

while True:

    sense_data = get_sense_data()

    if DELAY == 0:

        log_data()

    if len(batch_data) >= WRITE_FREQUENCY:

        print("Writing to file..")

        with open(filename,"a") as f:

            for line in batch_data:

                f.write(line + "\n")

            batch_data = []
```

OUTPUT CSV FORMAT:



```
temp_h,humidity,pressure,timestamp
38.5250015259,35.0395736694,950.701171875,2016-10-17 19:16:44.389857
38.2704544067,34.1238479614,950.682617188,2016-10-17 19:16:49.421340
38.4704551697,33.349697113,950.684814453,2016-10-17 19:16:54.445490
38.5613632202,33.280418396,950.713134766,2016-10-17 19:17:00.270915
38.3977279663,34.8799247742,950.6640625,2016-10-17 19:17:05.318029
38.361366272,35.1299438477,950.679199219,2016-10-17 19:17:10.326827
38.3068161011,32.9490699768,950.665039062,2016-10-17 19:17:15.362025
38.3977279663,34.8046188354,950.672851562,2016-10-17 19:17:22.337735
38.5068206787,34.536529541,950.708496094,2016-10-17 19:17:27.746507
38.5250015259,34.5817108154,950.693847656,2016-10-17 19:17:32.924526
38.3977279663,34.4280853271,950.715332031,2016-10-17 19:17:38.106662
38.4159088135,32.9099082947,950.707275391,2016-10-17 19:17:43.115218
38.4886360168,35.8287849426,950.706054688,2016-10-17 19:17:48.176803
38.4522705078,36.0034980774,950.720458984,2016-10-17 19:17:53.186773
38.5068206787,35.2654953003,950.724609375,2016-10-17 19:17:58.318945
38.4886360168,34.0244445801,950.717041016,2016-10-17 19:18:03.354986
38.4704551697,34.5124282837,950.705078125,2016-10-17 19:18:08.646975
38.5068206787,34.3045845032,950.705322266,2016-10-17 19:18:21.311706
38.4159088135,33.5726051331,950.703857422,2016-10-17 19:18:27.343484
38.4340896606,35.0907821655,950.750488281,2016-10-17 19:18:33.399980
38.3977279663,33.7985229492,950.703613281,2016-10-17 19:18:38.407800
38.4522705078,34.4762840271,950.749511719,2016-10-17 19:18:43.477217
38.4886360168,33.0424499512,950.727783203,2016-10-17 19:18:48.532577
38.4886360168,33.1026916504,950.754150391,2016-10-17 19:18:53.544890
38.3431816101,34.6811141968,950.765869141,2016-10-17 19:18:59.328018
38.4522705078,36.4523239136,950.726806641,2016-10-17 19:19:04.332704
38.3795471191,34.7262992859,950.742675781,2016-10-17 19:19:09.345377
38.3431816101,35.4823760986,950.751953125,2016-10-17 19:19:14.342471
38.3250007629,34.5003814697,950.744628906,2016-10-17 19:19:19.418284
38.2704544067,35.0034255981,950.754882812,2016-10-17 19:19:24.452617
38.2159118652,35.2986297607,950.723388672,2016-10-17 19:19:29.460013
38.3250007629,35.1058425903,950.756835938,2016-10-17 19:19:34.450357
38.1795463562,35.2112731934,950.745605469,2016-10-17 19:19:39.528118
38.2886352539,34.2925338745,950.778808594,2016-10-17 19:19:44.536475
38.3068161011,35.1660881042,950.762451172,2016-10-17 19:19:49.541630
38.361366272,35.9281921387,950.750732422,2016-10-17 19:19:54.578173
38.3795471191,35.1028327942,950.743896484,2016-10-17 19:19:59.719187
38.3250007629,35.6631126404,950.731201172,2016-10-17 19:20:04.791822
38.2340927124,33.5063362122,950.738769531,2016-10-17 19:20:09.828887
38.3795471191,35.9854240417,950.738037109,2016-10-17 19:20:15.390645
38.1977272034,33.3255996704,950.732421875,2016-10-17 19:20:21.146385
38.2522735596,34.084690094,950.759521484,2016-10-17 19:20:26.154673
38.2159118652,34.2503623962,950.731933594,2016-10-17 19:20:31.367479
38.2704544067,34.0033569336,950.721191406,2016-10-17 19:20:36.387361
38.2886352539,34.9040222168,950.715576172,2016-10-17 19:20:41.624146
38.3068161011,35.2383842468,950.733398438,2016-10-17 19:20:46.663514
38.2340927124,34.9823417664,950.762207031,2016-10-17 19:20:51.679099
38.2522735596,37.6210784912,950.723388672,2016-10-17 19:20:57.084212
38.4159088135,35.6119041443,950.764892578,2016-10-17 19:21:02.088428
38.2522735596,34.6871414185,950.760742188,2016-10-17 19:21:07.099152
38.3250007629,35.4014131165,950.750765625,2016-10-17 19:21:12.109220
```