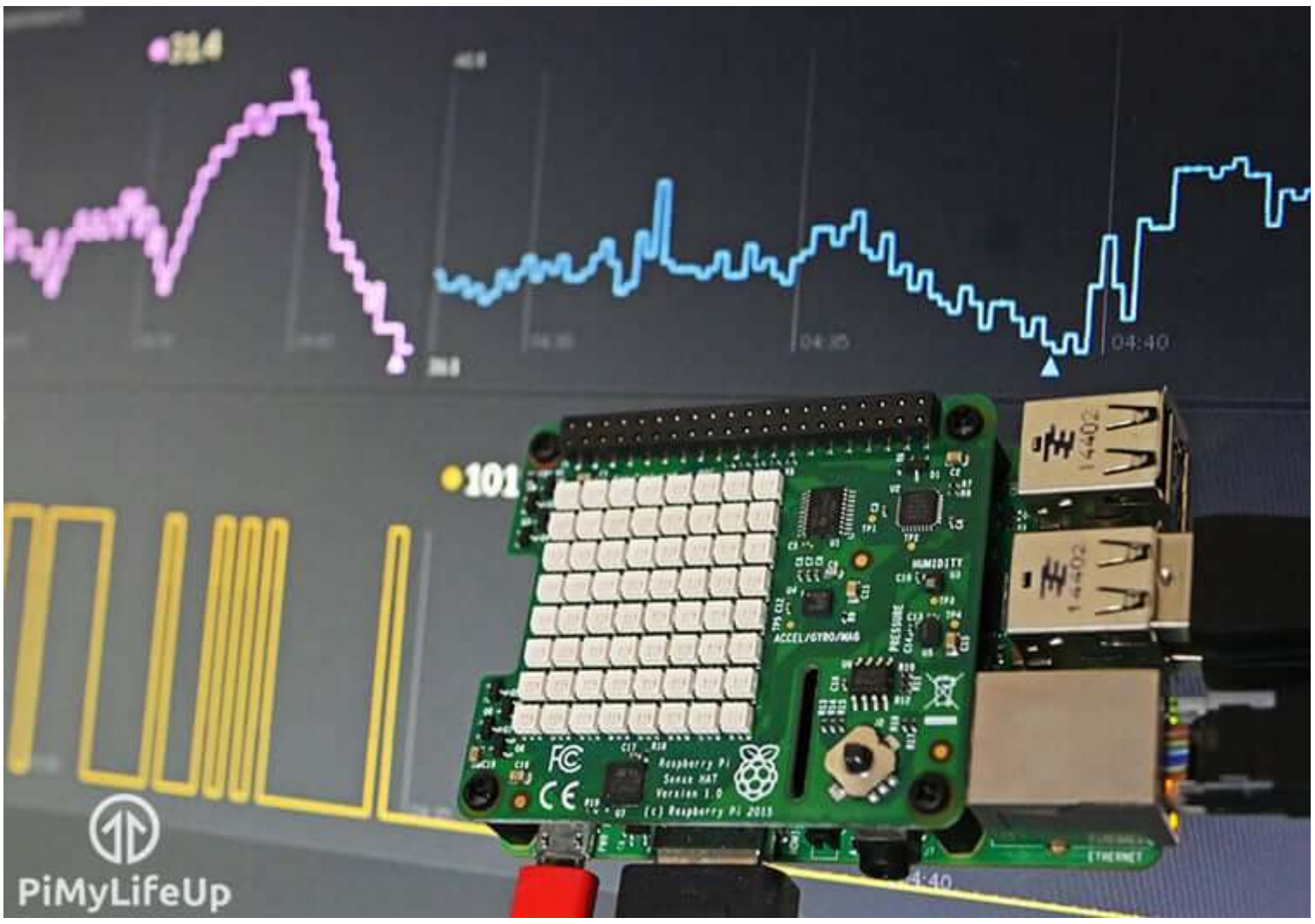
Raspberry Pi Weather Station using the Sense HAT

by Gus Apr 14, 2017 Updated Apr 12, 2019 Intermediate, IoT

In this tutorial, we will be showing you how to set up a basic Raspberry Pi weather station by utilizing the Sense HAT.



This weather station is the perfect little hobby project for anyone wishing to start monitoring the weather conditions in a specific room, outside or anywhere you would like.

For anyone who doesn't know, the Sense HAT is a fantastic piece of equipment that comes with a large abundance of sensors all in a single package. It's the easiest way of adding a ton of sensors to the Pi without needing to do any extra circuitry.

This weather station tutorial will show you how to set up the sense HAT software itself and how to retrieve the data from its three primary sensors, those being the temperature, humidity and pressure sensors. We will also briefly touch on how to write text to the LED Matrix as we will use this as a way of displaying your sensor data.

We will also go into setting up your Raspberry Pi weather station so that the data is stored on Initial States Data analytics service, and this will allow us to stream the data directly to their web interface and view all the data in a pretty graph.

This tutorial will make use of a large amount of Python code, but luckily it's all pretty basic so you shouldn't have too many problems following it.

Equipment

You will need a sense HAT and a few other bits of equipment to be able to complete this tutorial.

Recommended

- Raspberry Pi 2 or 3
- Micro SD Card or a SD card if you're using an old version of the Pi.
- Power Supply
- Sense HAT
- Ethernet Cable or WiFi Dongle (The Pi 3 has WiFi inbuilt)

Optional

- Raspberry Pi Case

Getting started with the Sense HAT

Before you get started with this tutorial, make sure that you have correctly placed the Sense HAT on the GPIO pins on the Raspberry Pi. It's an incredibly easy installation and shouldn't require any extra fiddling once installed correctly.

1. Before we get started, we need to run the following commands to ensure that the Raspberry Pi is running the latest software.

```
sudo apt-get update
sudo apt-get upgrade
```

2. With the Raspberry Pi up to date, we now need to install the Sense Hat software package, and this provides all the libraries we need to interact with the Sense Hat.

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

3. Now we have the software installed we need to write a quick script to ensure the Sense HAT is working correctly.

We can start writing this script by entering the following command.

```
sudo nano ~/sensehat_test.py
```

4. Write the following lines into this file, and we will explain what each section of code does as we go.

```
from sense_hat import SenseHat
```

This line imports the Sense Hat module from the **sense_hat** library. This library allows us to interact with the Hat itself through python.

```
sense = SenseHat()
```

This line creates a link to the Sense Hat library and initializes itself so we can start making calls to it.

```
sense.show_message("Hello World")
```

This line writes a message to the Sense Hat, and you should see "Hello World" scroll across the RGB lights.

Press **Ctrl + X** then **Y** then press **enter** to save the file.

5. With the file now saved we can run it with the following command:

```
sudo python ~/sensehat_test.py
```

The text “Hello World” should now scroll across the RGB LEDs on top of the Sense HAT. If it doesn’t, it is likely that the HAT has not been properly pressed down on top of the GPIO pins.

If it is still not working, try restarting the Raspberry Pi by rerunning the following command.

```
sudo reboot
```

Setting up your Sense HAT as a weather station

1. Now that we have tested that the Sense HAT is working correctly we can now get writing our Python weather station script. We will start with a basic script that continually reads and displays all the data from the Sense HAT’s sensors.

To begin writing our new Python script run the following command in terminal.

```
sudo nano ~/weather_script.py
```

2. Now enter the following lines of code into the script, we will explain each block of code as we go.

```
#!/usr/bin/python
from sense_hat import SenseHat
import time
import sys
```

First, we need to import all the libraries that we plan to utilize in our script. In our case, we will be using the following libraries.

sense_hat Library
This import is the library that we utilize to interact with the Sense Hat itself, without this we wouldn’t be able to read any of the sensor data or interact with the LED matrix.

time Library
This import allows us to do a large variety of different time stuff, but for our simple script, we will be just using its sleep functionality. This functionality will enable us to suspend the current thread for a small period.

sys Library
This library provides us access to some variables and functions that are managed by the interpreter itself. In the case of our script, we will be using this to terminate the script if we ever need to do so.

```
sense = SenseHat()
sense.clear()
```

The first line creates a link to the Sense Hat library and initializes itself so we can start making calls to it.

The second line tells the library to clear the LED Matrix, by default, this means switching off all the LEDs. It’s always a good idea to do this when dealing with the Sense HAT as it ensures nothing is being displayed already.

```
try:
      while True:
```

We set up our first **try statement**, and we need to do this so we can break out of our while loop by pressing **Ctrl + C** make sure you keep the indentation for the while True. This required spacing is because Python is sensitive to indentation. The next few lines of code will need a three-tab (12 spaces) indentation.

```
            temp = sense.get_temperature()
```

Getting the temperature is extremely simple, thanks to the Sense HAT Library. All we have to do is make a call to the library for it to retrieve the temperature from the sensor. The output that this will give us will be in Celsius, it also provides a larger decimal number, but we will deal with that on our next line of code.

We will explain how to convert the temperature to Fahrenheit if you would prefer to deal with that instead of Celsius.

Celsius

```
            temp = round(temp, 1)
```

Fahrenheit

```
            temp = 1.8 * round(temp, 1)  + 32
```

Here we give you two choices for this line of code, and the Celsius code utilizes the value we got from the sensor and rounds it to the nearest decimal place.

The Fahrenheit code is basically the same, the only difference being that we convert the value from Celsius to Fahrenheit.

Celsius

```
            print("Temperature C",temp)
```

Fahrenheit

```
            print("Temperature F",temp)
```

This bit of code is extremely basic and just prints the temperature to the terminal.

```
            humidity = sense.get_humidity()
            humidity = round(humidity, 1)
            print("Humidity :",humidity)

            pressure = sense.get_pressure()
            pressure = round(pressure, 1)
            print("Pressure:",pressure)
```

Both the humidity and pressure sensors can be read just like the temperature sensor. Luckily for us, the Sense HAT library makes this incredibly simple. Their values also come back with as a large decimal number, so we will again round them then display the values to the terminal.

There isn't much extra to say about these two code blocks as they operate just like the temperature code.

```
            time.sleep(1)
```

This line is a simple call to the time library that puts the script to sleep for approximately one second. It is mainly to reduce the rate at which the data is read and outputted.

You can speed up the read rate by decreasing this number or deleting the line. You can also slow it down further by increasing the number. The number should approximately be the number of seconds you want it to wait between reads.

```
except KeyboardInterrupt:
      pass
```

This code makes the try look for a **KeyboardInterrupt** exception. When it is triggered, we ignore the exception so we can have the script leave the while loop cleanly. We do this by calling **pass**.

3. With all the code entered into our Python file, you should end up with something that looks like below, of course, this will differ if you have used the Fahrenheit conversion code and not just straight Celsius.

```
#!/usr/bin/python
from sense_hat import SenseHat
import time
import sys

sense = SenseHat()
sense.clear()

try:
      while True:
            temp = sense.get_temperature()
            temp = round(temp, 1)
            print("Temperature C",temp)

            humidity = sense.get_humidity()
            humidity = round(humidity, 1)
            print("Humidity :",humidity)

            pressure = sense.get_pressure()
            pressure = round(pressure, 1)
            print("Pressure:",pressure)

            time.sleep(1)
except KeyboardInterrupt:
      pass
```

Once your code looks something like the one displayed above, and you are certain you have correctly indented your code you can quit and save, press **Ctrl+X**, then **Y** and then press **enter**.

4. We can now run our new Python script by running the following command in terminal.

```
sudo python ~/weather_script.py
```

5. You should now start to see text similar to the following appear in your terminal if everything is working as it should be.

```
('Temperature C', 30.0)
('Humidity :', 39.8)
('Pressure:', 1025.7)
```

Once you're happy with the data that is being displayed, you can stop the script by pressing **Ctrl + C**. This will terminate the script from running. Of course, you probably don't want to have to be looking at your Raspberry Pi's terminal to be able to get the current data from it.

Instead, we are going to show you two other methods for displaying data, the first of these is to display the data to the LED matrix.

The other method is to utilize a piece of software called Initial State that will allow us to graph the data, and if you decide to pay for the software, you can also store the data over a period of time.

Improving your weather station – Utilizing the LED Matrix

1. Changing our weather Python script to start displaying its data to the LED matrix is relatively easy. It will involve us concatenating all our prints together into a single string, then issuing the **show_message** command to the Sense Hat library.

Before we get started let's begin editing our weather script by running the following command.

```
sudo nano ~/weather_script.py
```

2. Now that we are in our weather script we can begin making some changes, above the **time.sleep** line we need to add an extra line of code. It basically handles everything in one simple line.

Above

```
            time.sleep(1)
```

Add

```
            sense.show_message("Temperature C" + str(temp) + "Humidity:" + str(humidity) +
"Pressure:" + str(pressure), scroll_speed=(0.08), back_colour= [0,0,200])
```

Make sure you have typed out this line of code all onto a single line. This change will make the temperature, humidity, and pressure scroll across the Sense HAT LED matrix.

We will explain a bit of what we are doing here. We firstly concatenate the temperature onto the end of "Temperature C", however since the temp variable is a number we need to wrap it in str() to convert it to a string.

Not converting the string will cause issues due to the way Python interprets adding a number to a string. We do this same process for the humidity and pressure.

We also decrease the scroll speed to 0.08, and we do that with the following part of the `code scroll_speed=(0.08)` you can increase or decrease this number to either speed up or decrease the speed the text scrolls.

The last bit of code we have on this line is `back_colour= [0,0,200]` , this part of the code sets the background color of the text to blue.

You can also change the color of the text itself. We can do this with our example of setting the colour of the text to a pinky color by adding the following: `, text_color=[200,0,200]` after `back_color[0,0,200]` .

3. With that line added there is one last thing we will want to add, that being a call to `sense.clear()` .

At the bottom of your script after the exception handling and pass, add the following line of code.

```
sense.clear()
```

This code will ensure that the LED matrix is completely cleared if we kill the script for any reason. It also stops the annoyance of having to deal with a partially turned on LED matrix and has the added benefit of saving a tiny amount of power, but there is no point on leaving something switched on when you are not utilizing it.

It is also good coding practice to make sure you clean-up everything when you stop a script from running. It helps prevent things like memory leaks from occurring.

4. With those changes made your code should look similar to what is displayed below. Remember that the **sense.show_message** code should be contained within a single line.

```
#!/usr/bin/python
from sense_hat import SenseHat
import time
import sys

sense = SenseHat()
sense.clear()

try:
      while True:
           temp = sense.get_temperature()
           temp = round(temp, 1)
           print("Temperature C",temp)

            humidity = sense.get_humidity()
            humidity = round(humidity, 1)
            print("Humidity :",humidity)

            pressure = sense.get_pressure()
            pressure = round(pressure, 1)
            print("Pressure:",pressure)

            sense.show_message("Temperature C" + str(temp) + "Humidity:" + str(humidity) +
"Pressure:" + str(pressure), scroll_speed=(0.08), back_colour= [0,0,200])

          time.sleep(1)
except KeyboardInterrupt:
      pass

sense.clear()
```

Once your code looks something like the one displayed above and you are certain you have correctly indented your code you can quit and save, press **Ctrl+X** and then press **Y** and then press **enter**.

5. Now we can test our modified script to make sure that it functions correctly. Run the following command in terminal to run the script.

```
sudo python ~/weather_script.py
```

The text should now begin to scroll across the Sense Hat's LED matrix. Of course, as you will quickly notice, this isn't the easiest way of viewing all the sensors data.

The viewability of the display is mainly due to the small nature of the LED matrix making the text more difficult to read and only being able to display a single letter at one time. Of course, if you have an actual LED screen and not a light matrix, it will be a much better option for displaying your data on your Pi.

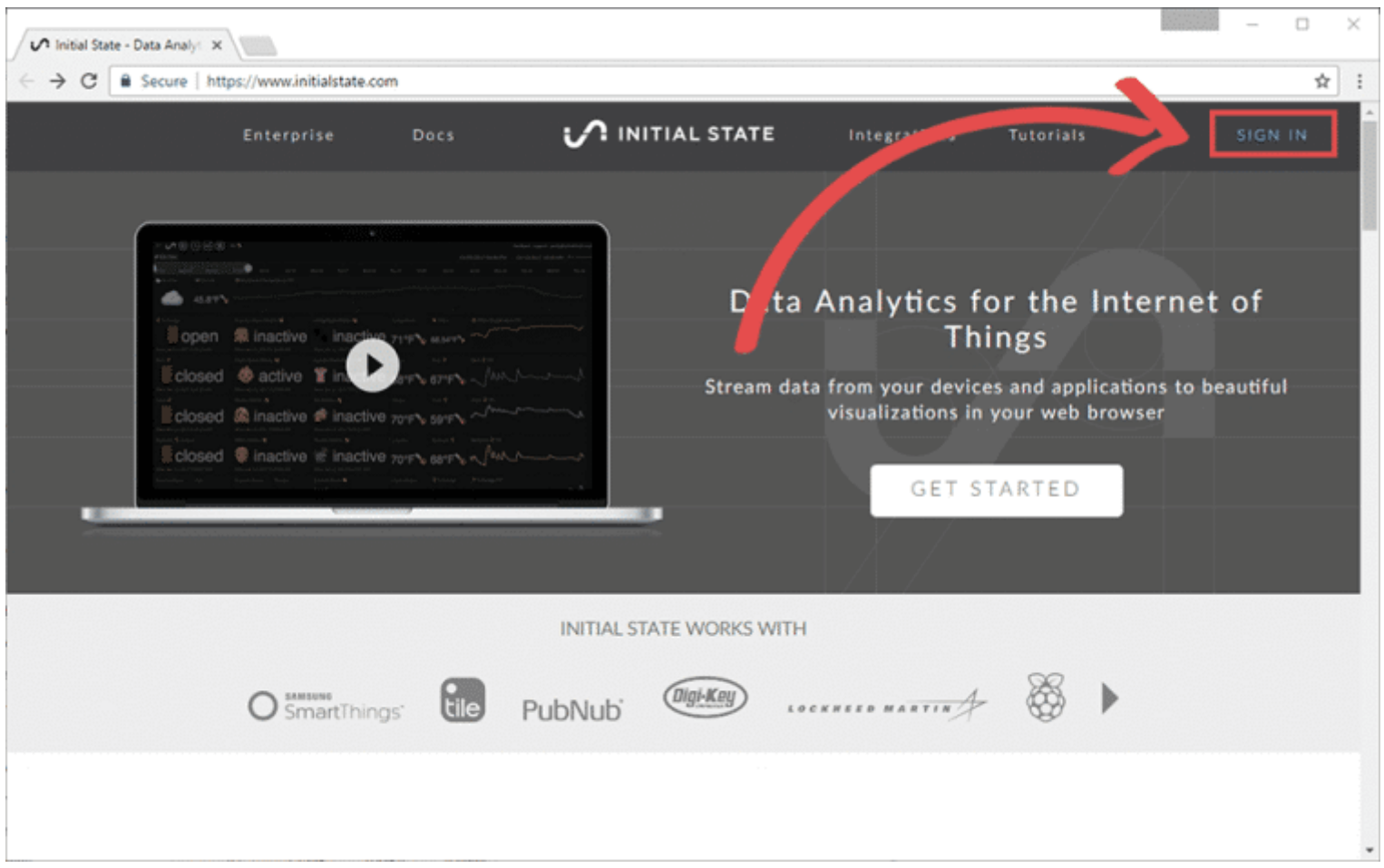
A nicer way of displaying your data is by utilizing a piece of software such as Initial State. We will go into setting up Initial state and getting our script to send data to it on the next page.

Improving your Raspberry Pi weather station with Initial State

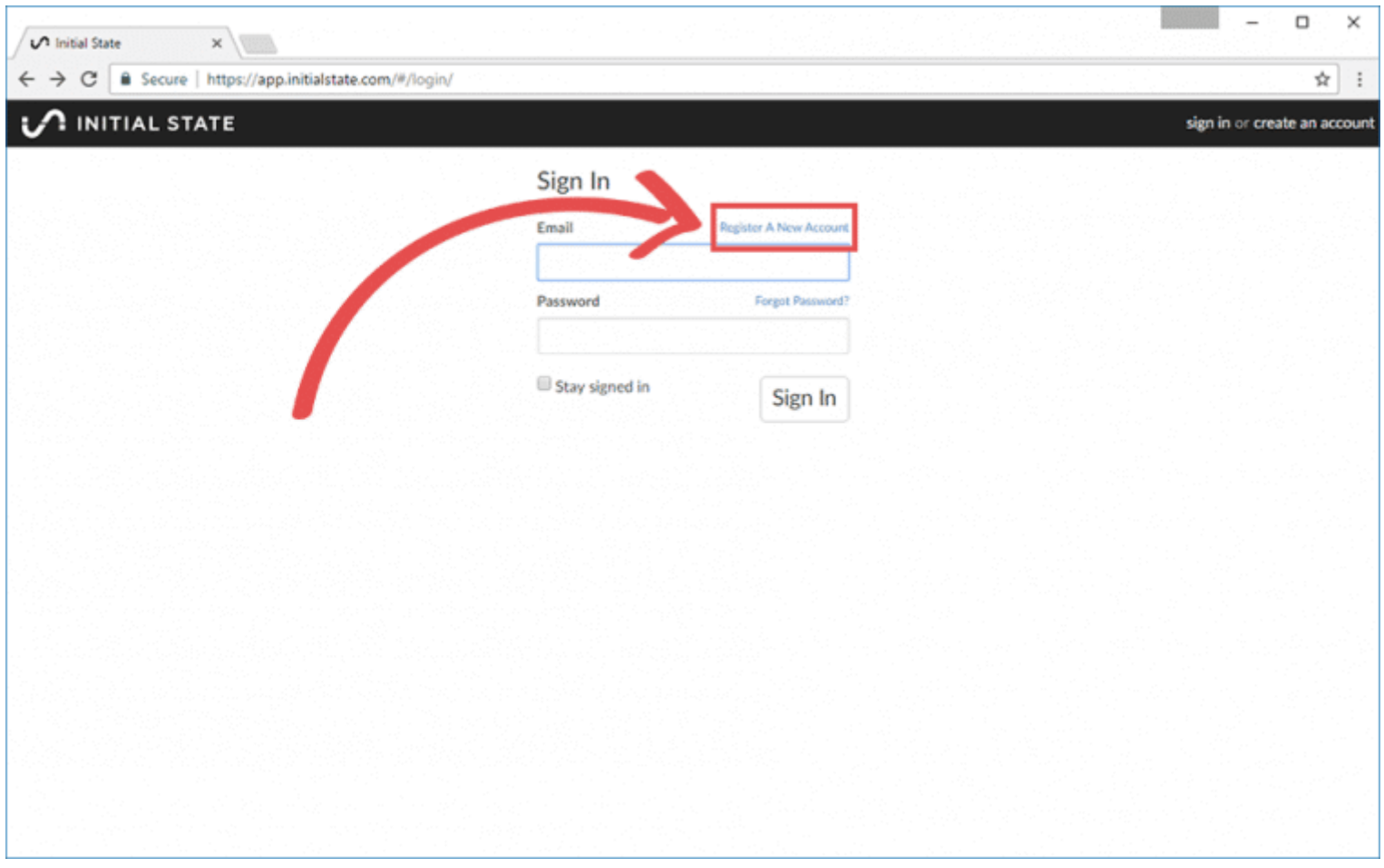
1. For those who don't know, Initial State is a website designed to act as a sort of data storage and data analytics site for the Internet of Things devices like the Raspberry Pi. We're using Initial state in this tutorial as the Cayenne IOT builder for the Raspberry Pi doesn't currently support the sense HAT, if they eventually add it, I will update this tutorial to reflect that.

Before we get started with implementing everything for the Raspberry Pi weather station, we will first have to sign up for a free account over at their website.

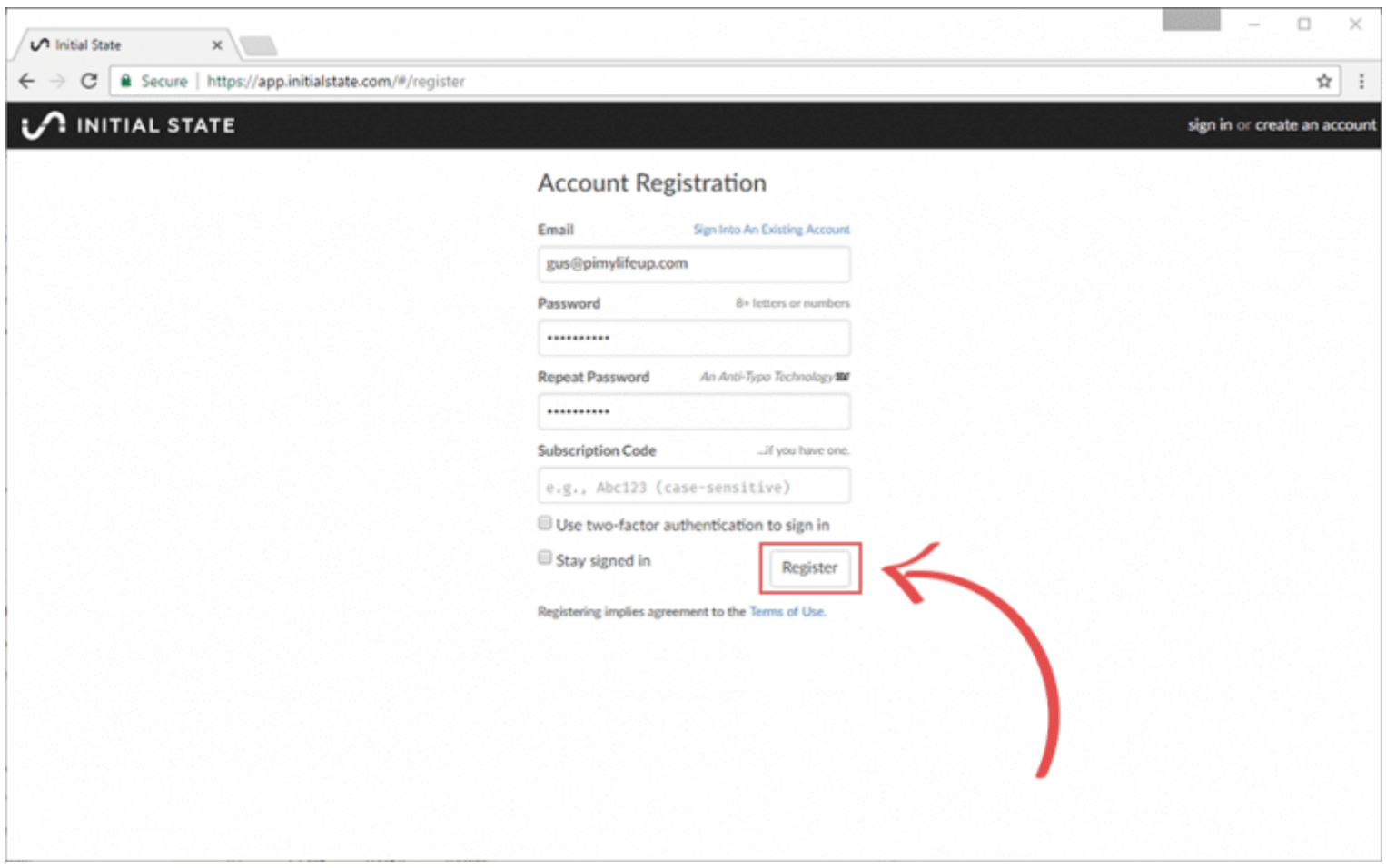
Once on the site, go to **SIGN IN**, in the top right-hand corner of the screen.



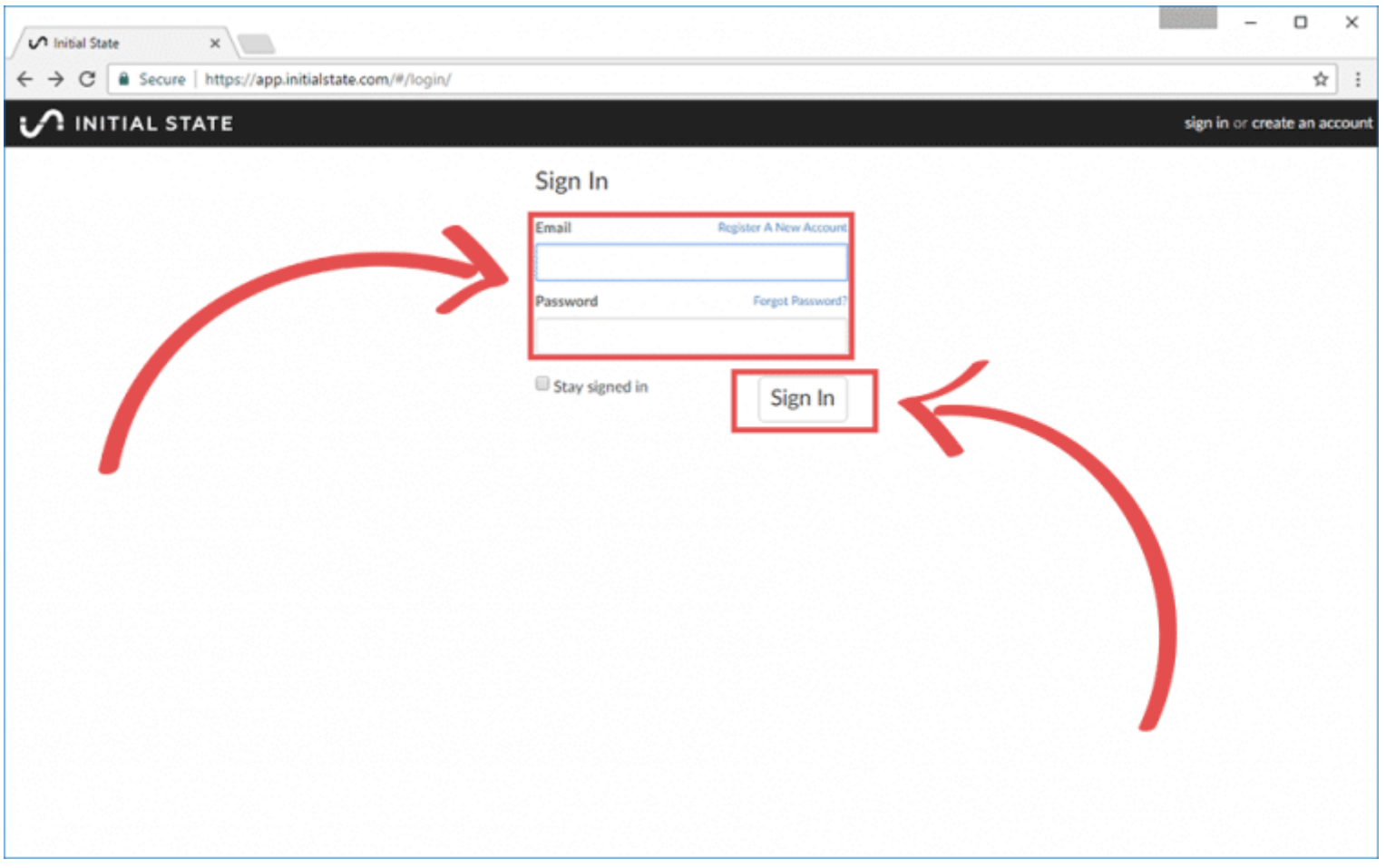
2. Now on the sign in screen, we need click on **Register A New Account**.



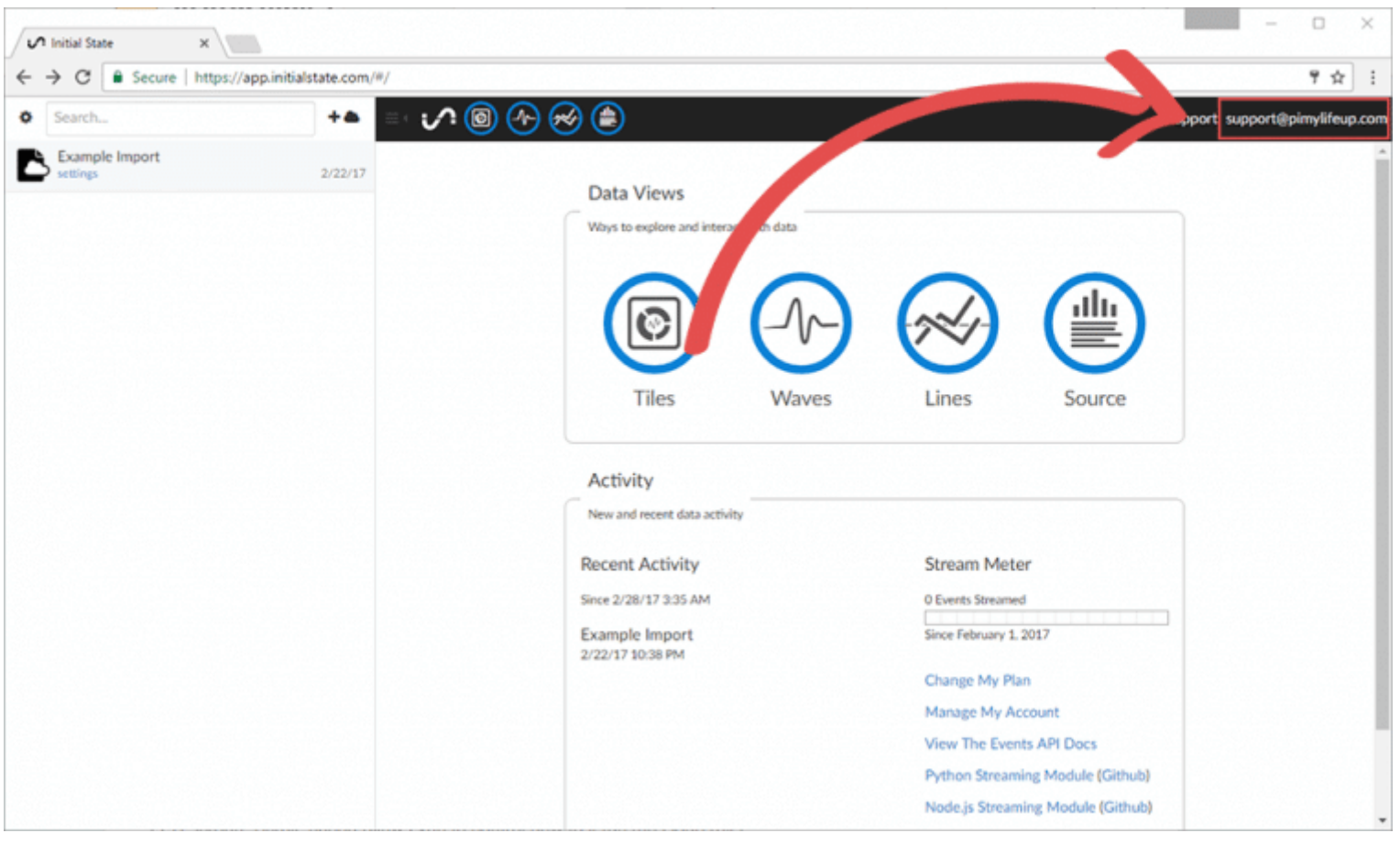
3. Finally, sign up your new user, make sure you set a strong password that you can remember as you will need this to get your API key and view your data.



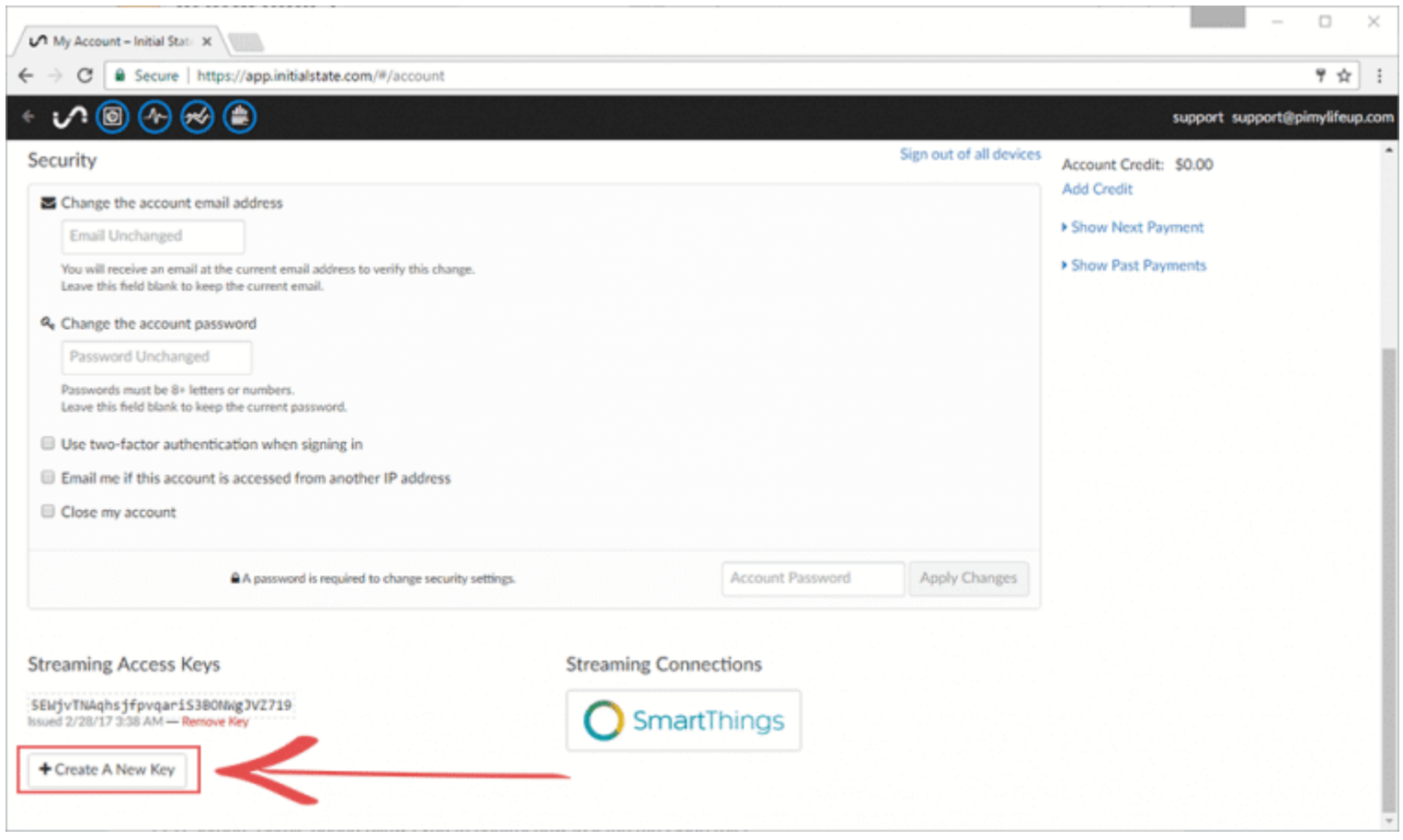
4. Now with your new user, go back to the sign in page and sign into your new Initial State user, we will need to grab the API key from here before we get started with our script.



5. You should now be greeted by an empty dashboard screen like below, to get to the screen where you can generate an API key you will need to click on your **email address** in the top right-hand corner.



6. Now that we are on our account screen we need to scroll all the way to the bottom and look for **Streaming Access Keys**, underneath here you should see a **Create a New Key** button, press it. Once the key is generated write it down somewhere safe, as you will need this for our script.



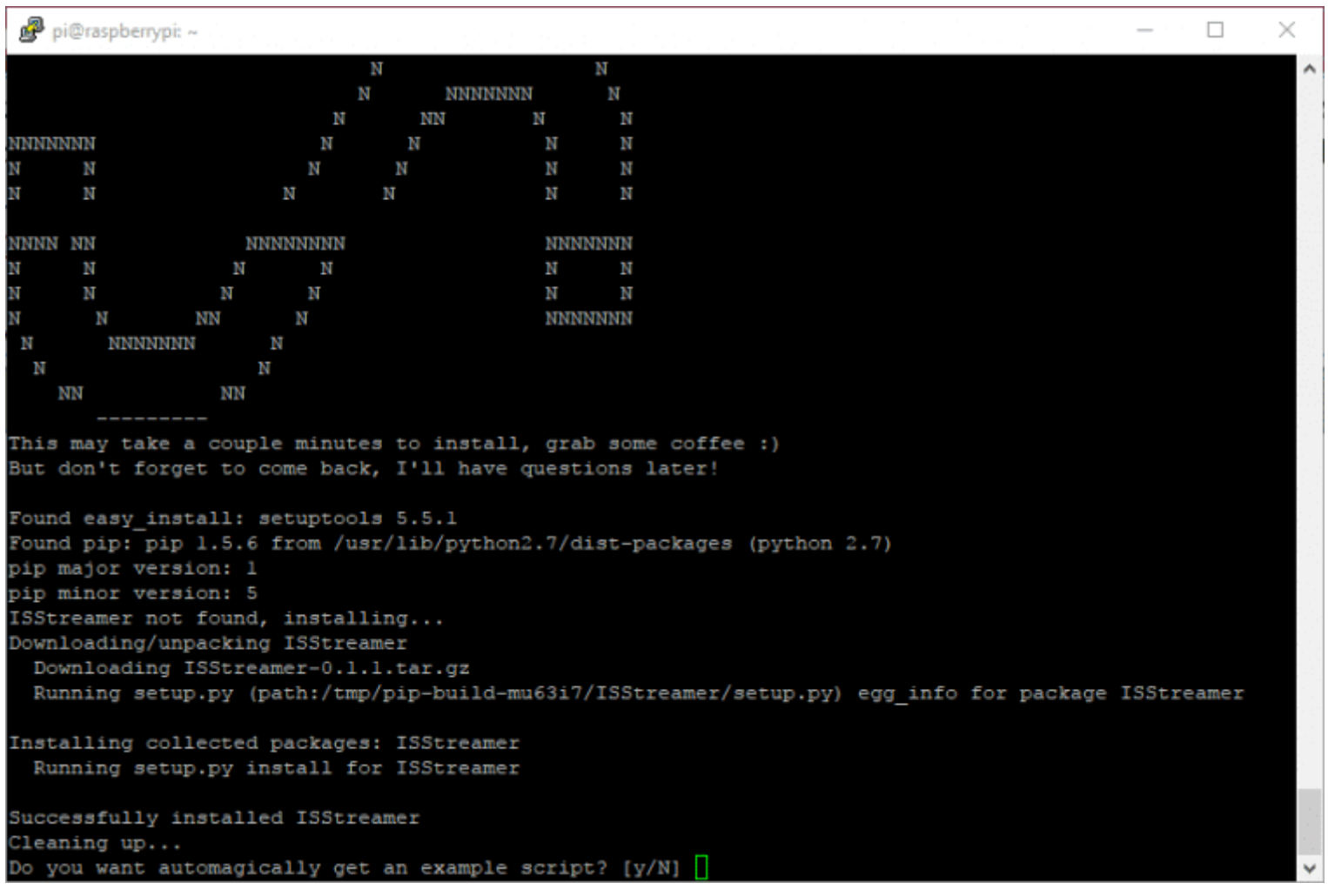
7. We can now get on with installing the Initial State python streamer, to do this we are going to grab their install script directly from their website. While it is not normally safe to run a script through CURL, Initial State is a trusted site. However, if you want to check the script yourself, you can view it by going to the following URL:

`https://get.initialstate.com/python`

If you are okay with running the script, then enter this command into the terminal.

```
curl -ssl https://get.initialstate.com/python -o - | sudo bash
```

8. Eventually, you will be greeted with the following prompt, press **N** to skip having to download the example code, and we will not need it for our tutorial.



9. Now we have the streamer installed, and we can begin editing our previous weather script. Run the following command to start editing the file.

```
sudo nano ~/weather_script.py
```

10. Now we will have first to make a couple of code additions to this file. Add the following lines to the file.

Below

```
import sys
```

Add

```
from ISStreamer.Streamer import Streamer
```

This line imports the Initial State streamer package, and this will allow us to make a connection to their website and stream our data to it.

Below

```
sense = SenseHat()
```

Add

```
logger = Streamer(bucket_name="Sense Hat Sensor Data", access_key="YOUR_KEY_HERE")
```

This line creates the Streamer and initializes a connection with Initial Senses servers, make sure you replace **YOUR_KEY_HERE** with the key you got in step 6.

Replace

```
print(
```

With

```
logger.log(
```

On this step you need to replace all occurrences of **print(with logger.log(** this will make the script log all your sensor data to Initial State's website. Luckily for us, the log method of the Streamer object follows the same parameters as print. So, we can just easily swap out the functions.

Your code should now look like something similar to what is displayed below.

```
#!/usr/bin/python
from sense_hat import SenseHat
import time
import sys
from ISStreamer.Streamer import Streamer

sense = SenseHat()
logger = Streamer(bucket_name="Sense Hat Sensor Data", access_key="YOUR_KEY_HERE")
sense.clear()

try:
      while True:
           temp = sense.get_temperature()
           temp = round(temp, 1)
           logger.log("Temperature C",temp)

            humidity = sense.get_humidity()
            humidity = round(humidity, 1)
            logger.log("Humidity :",humidity)

            pressure = sense.get_pressure()
            pressure = round(pressure, 1)
            logger.log("Pressure:",pressure)

           time.sleep(1)
except KeyboardInterrupt:
      pass
```

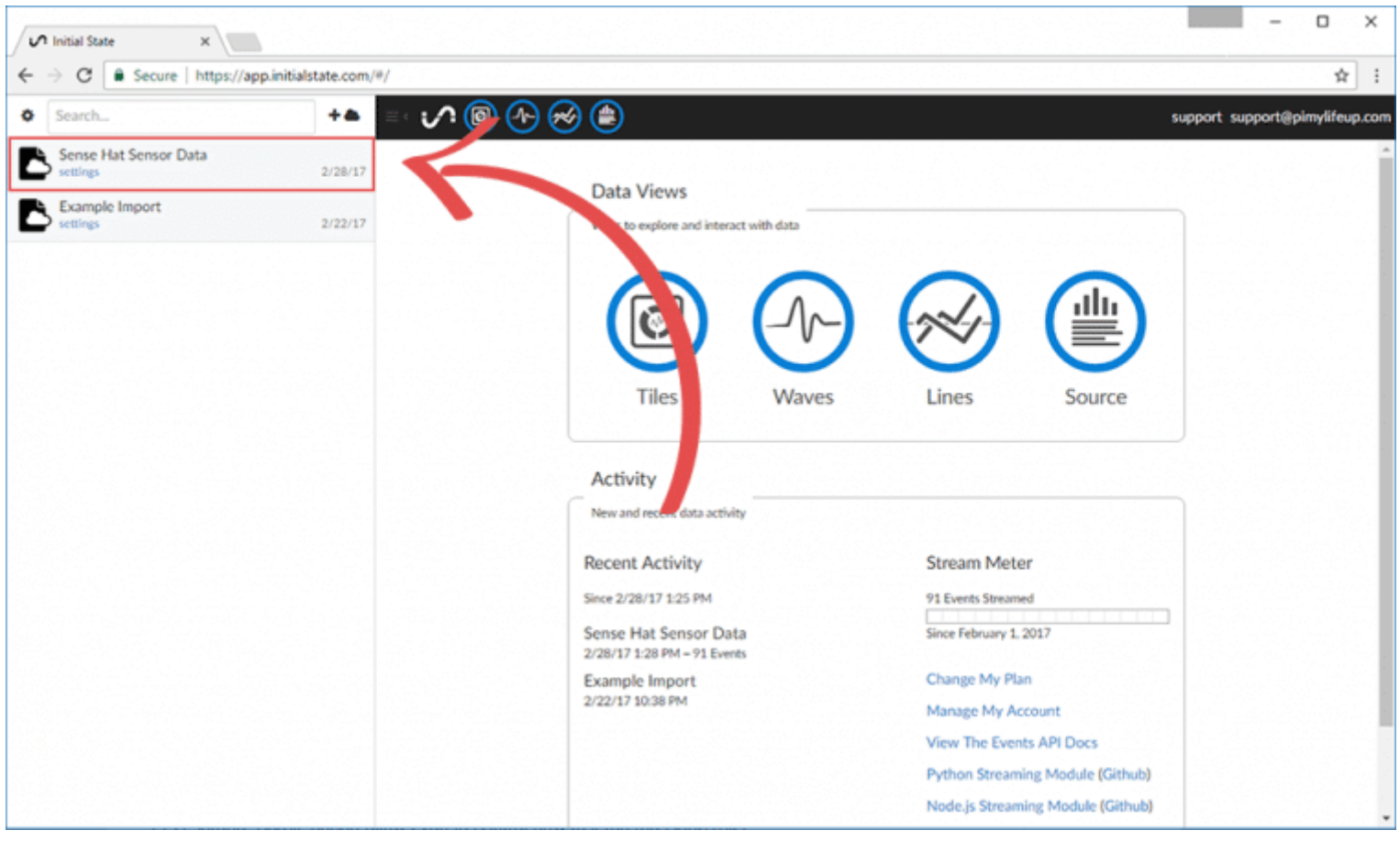
Once your code looks something like the one displayed above, and you are certain you have correctly indented your code you can quit and save, press **Ctrl+X** and then **Y** and then press **enter**.

11. Now we have made those changes we can finally run our updated script by running the following command.

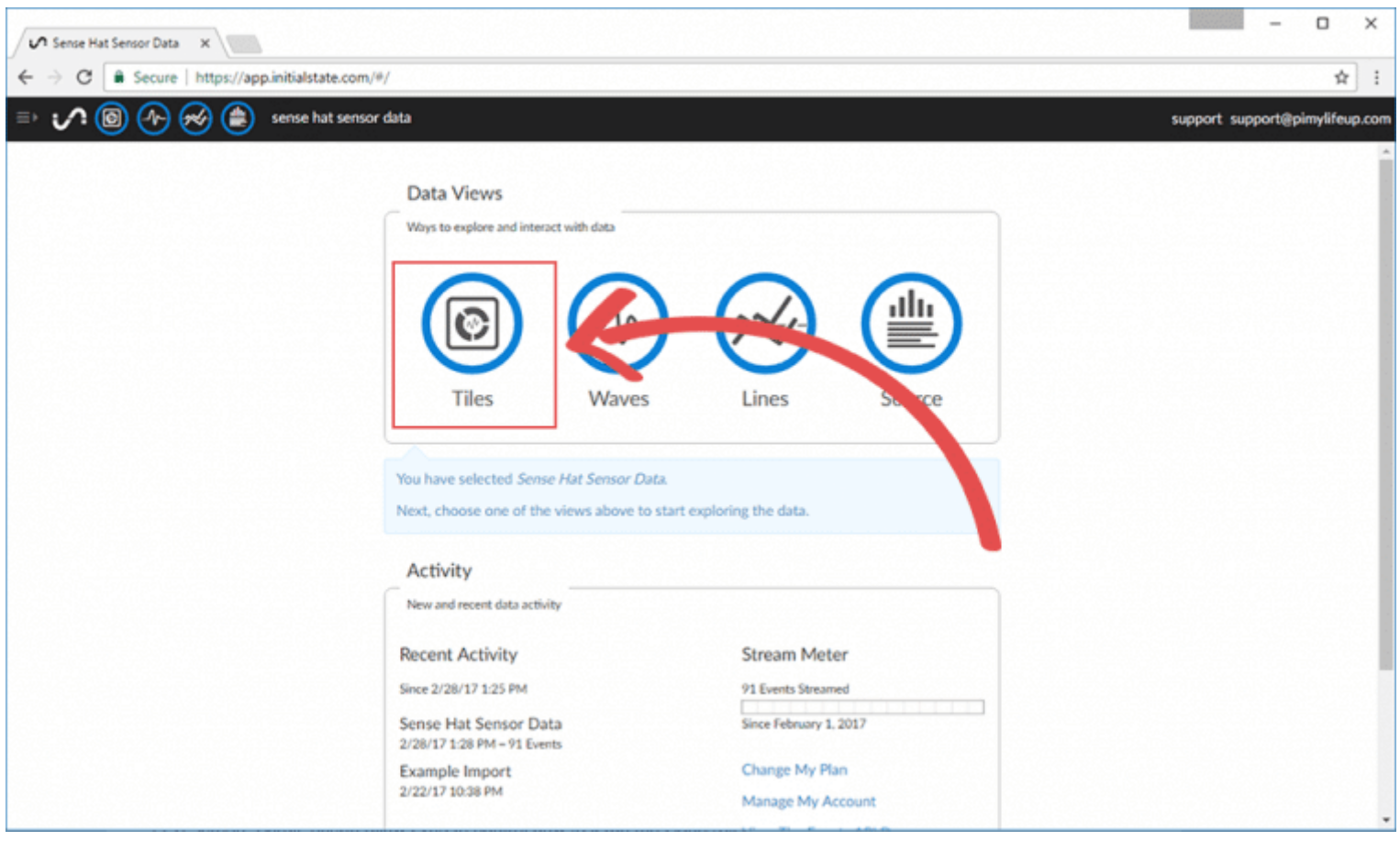
```
sudo python ~/weather_script.py
```

The script will immediately start to send your data to the website.

12. Now going back to your Initial State dashboard, you should now see your new data set on the left-hand side, click it.



13. With the data set selected, we can now choose how we want to view the data, for this tutorial we will select **Tiles** as this is the most versatile option.



14. You should now be greeted with a display like the one below, and this will continually update itself with the new values as they are sent from your Raspberry Pi.

If you want to change the graph that is being displayed for any of the values, you can right click on them and click on **Configure Tile**.

Improving your Weather Station – Start script at startup

1. Before we get started with setting up our script, we first need to install an additional package we may have to rely on. This package is **dos2unix**, and this converts DOS-style line endings into something that is Unix friendly.

Run the following Linux command in the terminal to install the package.

```
sudo apt-get install dos2unix
```

2. To set up our Python script as a service, so it starts on boot we will need to write up a short script. The advantage of this script is that we will be able to stop it and force a restart easily.

To begin let's start writing our new script by running the following command.

```
sudo nano /etc/init.d/weatherstation
```

Now write the following into the file, this is quite a bit of code. You can find all the code we use in this tutorial over at the GitHub page for this tutorial.

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          weatherstation
# Required-Start:
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start/stops the weatherstation
# Description:       Start/stops the weatherstation
### END INIT INFO

DIR=/home/pi
DAEMON=$DIR/weather_script.py
DAEMON_NAME=weatherstation

DAEMON_USER=root

PIDFILE=/var/run/$DAEMON_NAME.pid

. /lib/lsb/init-functions

do_start () {
    log_daemon_msg "Starting system $DAEMON_NAME daemon"
    start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --user
$DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON
    log_end_msg $?
}
do_stop () {
    log_daemon_msg "Stopping system $DAEMON_NAME daemon"
    start-stop-daemon --stop --pidfile $PIDFILE --retry 10
    log_end_msg $?
}

case "$1" in
    start|stop)
        do_${1}
        ;;

    restart|reload|force-reload)
        do_stop
        do_start
        ;;

    status)
        status_of_proc "$DAEMON_NAME" "$DAEMON" && exit 0 || exit $?
        ;;
    *)

        echo "Usage: /etc/init.d/$DAEMON_NAME {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```

3. Now we have written all that code into the file, and we can now save and exit it by pressing **Ctrl + X** then pressing **Y** and then **Enter**.

4. Now we have the file saved there is a few things we will need to do to make sure this will correctly work. First things first we will run **dos2unix** on our newly created file. This program will ensure the line endings are correct.

```
sudo dos2unix /etc/init.d/weatherstation
```

5. With that done, we need to now change the permissions for our python script. Otherwise, our **init.d** bash script will fail to work. Type the following into the terminal to change its permissions.

```
sudo chmod 755 /home/pi/weather_script.py
```

6. We also need to modify the permissions of our weatherstation bash script, and we need to give it execution rights. We do that by entering the following command.

```
sudo chmod +x /etc/init.d/weatherstation
```

7. Now finally, we need to create a symbolic link between our bash script and the **rc.d** folders. We can do that by running the following command in terminal.

```
sudo update-rc.d weatherstation defaults
```

8. Everything should now be set up, and we can now interact with our new bash file like any other service.

To start up our Python script, we can just run the following command.

```
sudo service weatherstation start
```

9. Now everything should be correctly set up, and the weatherstation service should now automatically start on boot. We can also stop the script, reload it, and check the status of the script like most services.

Below is a list of commands you can call to interact with the weatherstation service.

This command starts up the service which keeps track of your **weather_script.py** file.

```
sudo service weatherstation start
```

The following command stops the weatherstation service and kills the process that is running our **weather_script.py**.

```
sudo service weatherstation stop
```

This command reloads weatherstation service by killing the process and reloading it.

```
sudo service weatherstation reload
```

The last command shown below retrieves the status of the weatherstation service and our **weather_script.py** script.

```
sudo service weatherstation status
```

I hope that this tutorial has shown you how to set up a basic Raspberry Pi weather station that utilizes the Sense HAT and that you haven't run into any issues. It's certainly a great project for anyone who wishes to set up a cheap weather station. If you have some feedback, tips or have come across any issues that you would like to share, then please don't hesitate to leave a comment over at our forum.

Installing Ubuntu Mate on the Raspberry Pi

Using HDMI-CEC on a Raspberry Pi

Installing Java on the Raspberry Pi

How to Setup Raspberry Pi Remote Desktop

Basic Guide to Voltage Dividers

How to Setup Raspberry Pi NFS Server

14 Comments

Loïc on August 21, 2019 at 5:47 am

Hello,

Very good tutorial. Thank a lot. But how to display the temperature, humidity etc. on 2 digits ? please, because actually it's not easy to read.

Have a good day.

Loic

Reply

Gus on August 22, 2019 at 7:42 pm Editor

Just edit the Round function parameters.

For example, change the following line, so the found function no longer uses the second parameter.