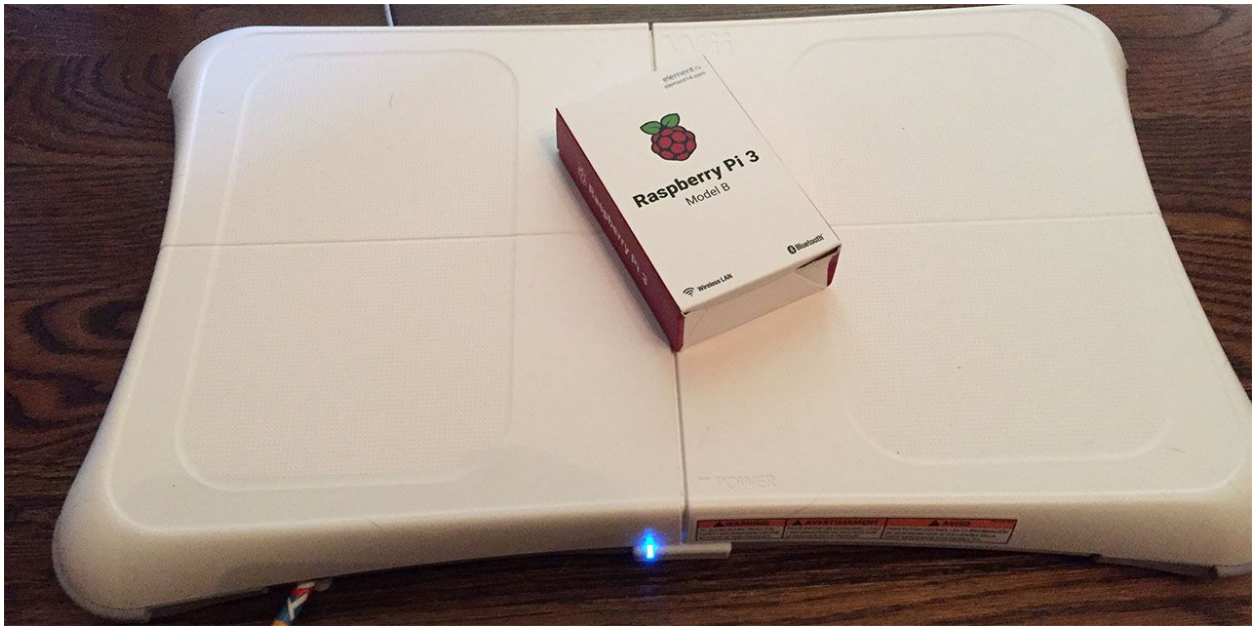
Documentation/ Procedures
Smart Scale 1st PHASE:

We are going to build a hackable, weight tracking, text messaging bluetooth scale.

- connect a Wii balance board to a Raspberry Pi through bluetooth
- run a Python script that measures your weight when you step on the balance board
- use a Raspberry Pi to stream your weight to a cloud service (Initial State)
- setup a SMS notification every time you weigh yourself
- build a weight-tracking dashboard you can access in your web browser

Part 1. Equipment

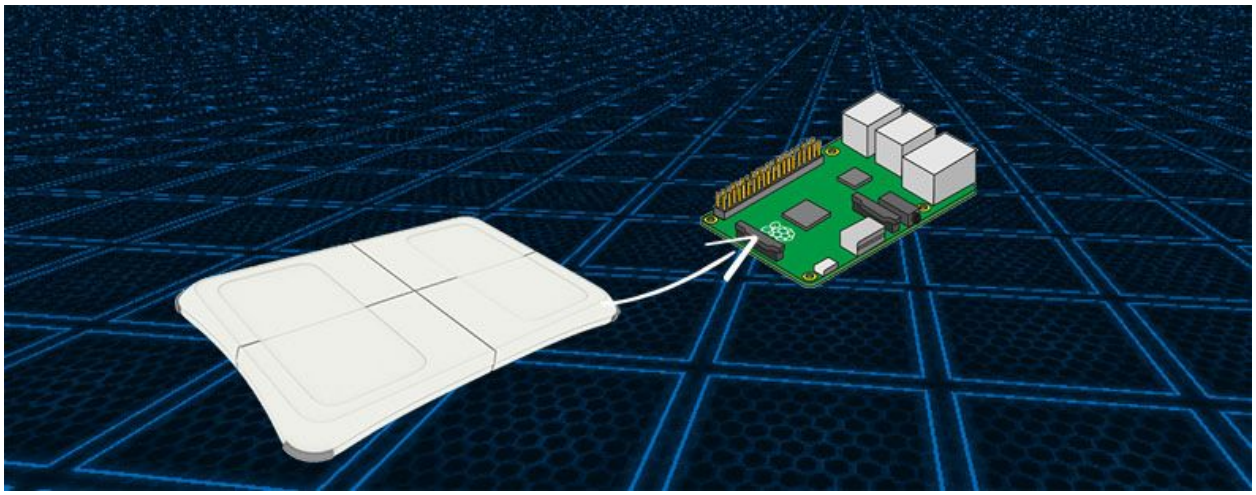


Here is a list of all the equipment that we will be using in this tutorial with links to where you can purchase each item.

- Raspberry Pi 3 with a SD card and Power Supply (https://init.st/psuufmj).
- Wii Balance Board (https://init.st/qg4ynjl)
- Wii Fit Rechargeable Battery Pack (https://init.st/iyypz2i)
- 3/8" Felt Pads (https://init.st/8gywmjj)

- Pencil (I'm not giving you a link to where to buy a pencil ... you should own one of these)

Part 2. Wii Balance Board Scale



Why a Wii Balance Board? It turns out that it is a really nice, durable scale that has bluetooth connectivity. This will allow us to connect it to a single-board computer (Raspberry Pi) to read your weight in a Python script and send those measurements to an online data service

Part 2. Bluetooth Setup

The Raspberry Pi 3 comes with bluetooth built in, all we need to communicate with the Wii Balance Board.
Power on your Pi (I am assuming you have already installed Raspbian and it boots up) and go to your Raspberry Pi terminal window. You can see the address of your bluetooth dongle with the "hcitool dev" command:

```
$ hcitool dev
Devices:
	hci0	00:1A:7D:DA:71:13
```

Install the bluetooth modules that we will be using in our Python scripts:

```
$ sudo apt-get install python-bluetooth
```

After installation completes, we are ready to connect and communicate with the Wii Balance Board. We will not be permanently pairing our Board with our Pi like we do with most of our bluetooth devices. The Wii Balance Board was never intended to be paired with anything other than a Wii, and permanent pairing proved to be quite the confusing challenge. Pairing will happen every time we run our Python script.

Part 2. Reading the Scale

It is time to connect our Wii Balance Board to our Raspberry Pi. We will do this by modifying a version of Stavros Korokithakis' Gr8W8Upd8M8.py script. The python script we will be using for this step is located here. You can copy the contents of this file to a file you create on your Raspberry Pi or you can clone all of the python files we will be using for this entire project. Let's do the latter. Type the following commands into a terminal on your Raspberry Pi:

```
$ cd ~
$ git clone https://github.com/InitialState/smart-scale.git
cloning into 'smart-scale'...
remote: Counting objects: 14, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 14 (delta 1), reused 8 (delta 0), pack-reused 0
Unpacking objects: 100% (14/14), done.
Checking connectivity... done.
```

You should see two python files in the new smart-scale directory - smartscale.py and wiiboard_test.py.

```
$ cd smart-scale
$ ls
README.md          smartscale.py           wiiboard_test.py
```

Run the wiiboard_test.py script to test communication and take weight readings from the Wii Balance Board:

```
$ sudo python wiiboard_test.py
```

You will see the following response:

```
Discovering board...
Press the red sync button on the board now
```

Remove the battery cover underneath the Board to locate the red sync button. Make sure you press the button within a few seconds of running the script or a timeout will occur. Once successful, you will see something similar to the following:

```
Found Wiiboard at address 00:23:CC:2E:E1:44
Trying to connect...
Connected to Wiiboard at address 00:23:CC:2E:E1:44
Wiiboard connected
ACK to data write received
84.9185297 lbs
84.8826412 lbs
84.9275927 lbs
```

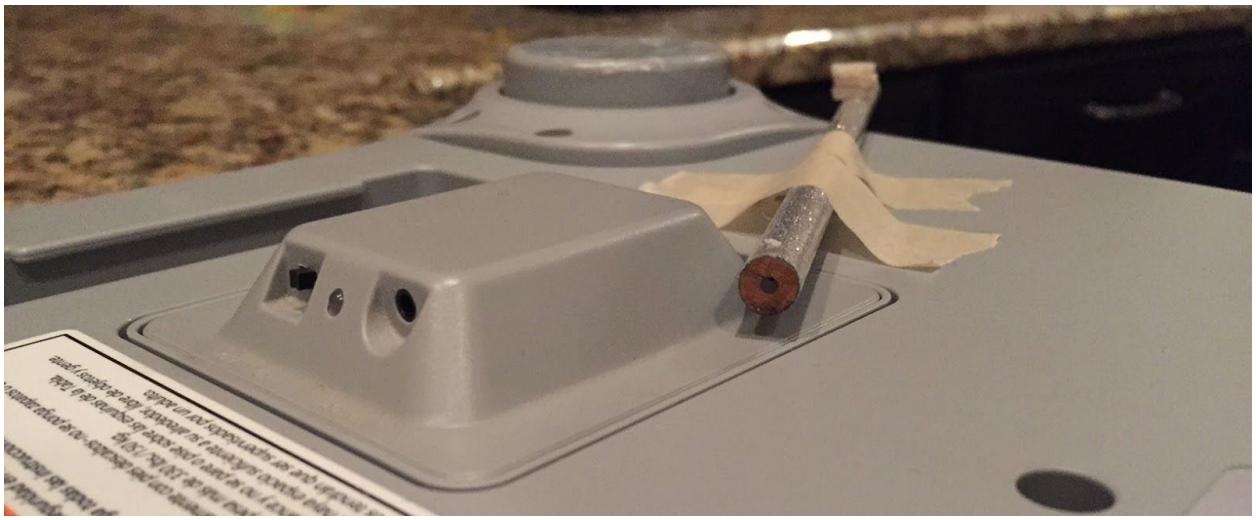
The wiiboard_test.py script is taking the number of weight measurements specified on line 10 and outputting the average:

```
# --------- User Settings ---------
WEIGHT_SAMPLES = 500
# ---------------------------------
```

You can play with this number by changing the value and re-running the script to see the variation in weight measured and time required for each measurement. Weigh yourself, weigh your dog, weigh whatever and see if the measurements make sense. To stop the script, press CTRL+C.

You have now successfully converted your Wii Balance Board into a Raspberry Pi connected scale.

Part 2. Hardware Tweaks



Nintendo assumed you would always power your Wii Balance Board with four AA batteries and included no AC power adapter. Having only battery power would be inconvenient because we cannot permanently pair our Wii Board to our Pi through bluetooth. We need to sync it, then allow it to remain synced without draining the batteries so we can simply step on the scale and weigh. Luckily, there are several third-party adapters made for the Wii Balance Board that we can use to provide constant power from a wall outlet. Replace the batteries with a battery pack and plug the ac adapter into a wall outlet.

Having to pair the Wii Balance Board and Raspberry Pi every time we run our Python script presents another inconvenience due to the location of the sync button. The sync

button is at the bottom of the Wii Board, which means we would have to flip it over every time we need to sync. We can fix this by making a little lever using a pencil and three 3/8" felt pads as shown above. The rechargeable battery pack exposes the sync button to the underneath surface of the Board. Tape a pencil (or something similar) that spans from the sync button to the outside front of the Board. Stack three 3/8" felt pads (or something similar) on the center of the pencil to create a stationary pivot. Be careful to not expose too much of the pencil out from the Board as you don't want someone to accidentally kick it out. Flip the Board over and you can press the sync button by simply pressing down on the lever. A bit of a hack but effective.

Depending on how you store your Wii Board, you may want to remove the rubber grip pads from the feet of the Board (the pads are simply stickers you can pry off). 3/8" felt pads can be placed on the Board's feet for easy sliding.

Part 3. Initial State

We want to stream our weight/data to a cloud service and have that service turn our data into a nice dashboard that we can access from our laptop or mobile device. Our data needs a destination. We will use Initial State as that destination.

Step 1: Register for Initial State Account

Go to https://iot.app.initialstate.com and create a new account.

Step 2: Install the ISStreamer

Install the Initial State Python module onto your Pi:

At a command prompt (don’t forget to SSH into your Pi first), run the following command:

```
$ cd /home/pi/
$ \curl -sSL https://get.initialstate.com/python -o - | sudo bash
```

Step 3: Make some Automagic

After Step 2 you will see something similar to the following output to the screen:

```
pi@raspberrypi ~ $ \curl -sSL https://get.initialstate.com/python -o - | sudo
bash
Password:
Beginning ISStreamer Python Easy Installation!
This may take a couple minutes to install, grab some coffee :)
But don't forget to come back, I'll have questions later!

Found easy_install: setuptools 1.1.6
Found pip: pip 1.5.6 from /Library/Python/2.7/site-packages/pip-1.5.6-
py2.7.egg (python 2.7)
pip major version: 1
pip minor version: 5
ISStreamer found, updating...
Requirement already up-to-date: ISStreamer in /Library/Python/2.7/site-packages
Cleaning up...
Do you want automagically get an example script? [y/N]
```

(the output may be different and take longer if you have never installed the Initial State Python streaming module before)

When prompted to automatically get an example script, type y. This will create a test script that we can run to ensure that we can stream data to Initial State from our Pi. You will be prompted:

```
Where do you want to save the example? [default: ./is_example.py]:
```

You can either type a custom local path or hit enter to accept the default.

You will be prompted for your username and password that you just created when you registered your Initial State account. Enter both and the installation will complete.

Step 4: Access Keys

Let's take a look at the example script that was created.

```
$ nano is_example.py
```

On line 15, you will see a line that starts with `streamer = Streamer(bucket_ ....` This lines creates a new data bucket named "Python Stream Example" and is associated with your account. This association happens because of the `access_key="..."` parameter on that same line. That long series of letters and numbers is your Initial State account access key. If you go to your Initial State account in your web browser, click on your username in the top right, then go to "My Settings", you will find that same access key at the bottom of the page under "Streaming Access Keys".

Every time you create a data stream, that access key will direct that data stream to your account (so don't share your key with anyone).

Step 5: Run the Example

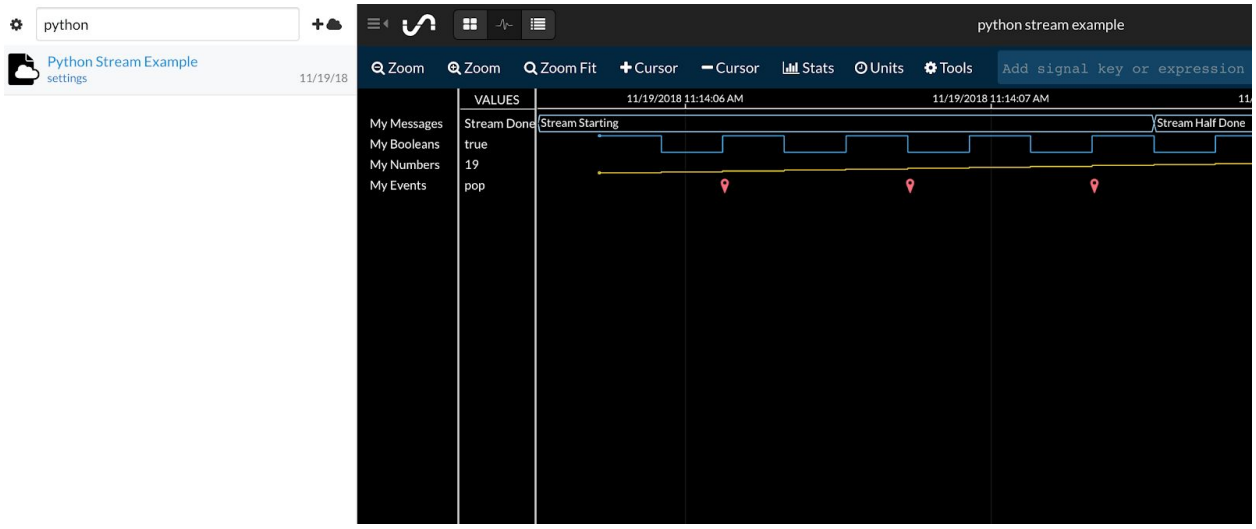
Run the test script to make sure we can create a data stream to your Initial State account. Run the following:

```
$ python is_example.py
```

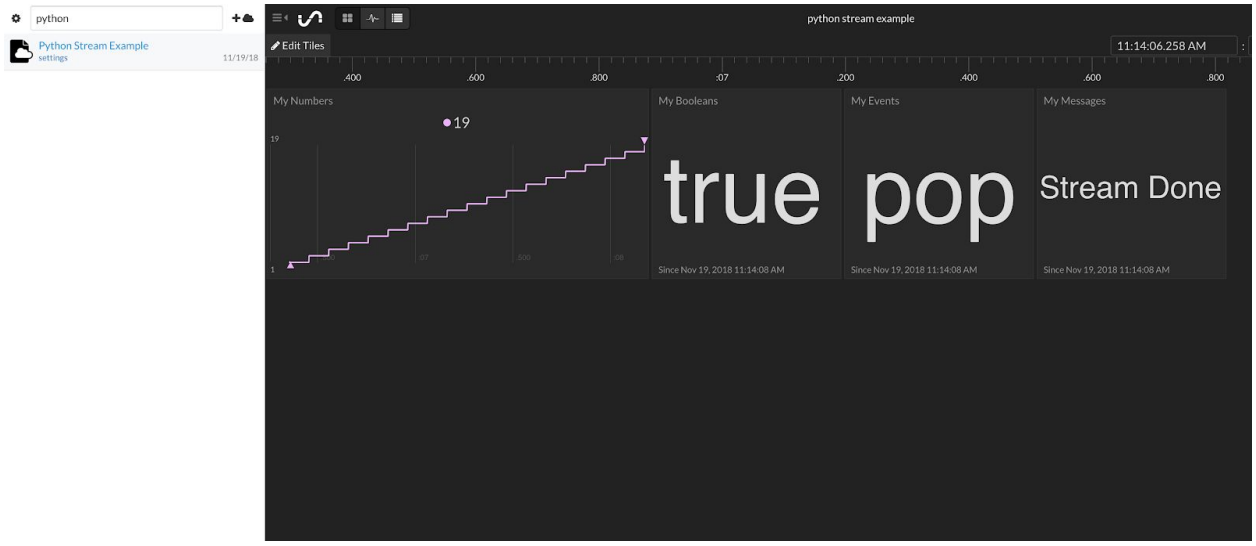
Step 6: Profit

Go back to your Initial State account in your web browser. A new data bucket called "Python Stream Example" should have shown up on the left in your log shelf (you may

have to refresh the page). Click on this bucket and then click on the Waves icon to view the test data.



You will want to step through the Waves tutorial to familiarize yourself with how to use this data visualization tool. Next, view the data in Tiles to see this same data in dashboard form.



You are now ready to start streaming real data from your scale.

Part 3. The Final Script

Assuming you ran the "git clone https://github.com/InitialState/smart-scale.git " command in Part 2, the final script that puts everything together is called smartscale.py

in your ~/smart-scale directory.
(*https://github.com/InitialState/smart-scale/blob/master/smartscale.py*)

A few settings need to be set in the script before you can run it. Open up smartscale.py in your favorite text editor such as nano.

```
$ cd ~
$ cd smart-scale
$ nano smartscale.py
```

Near the top of this file, there is a User Settings section.

```
# --------- User Settings ---------
BUCKET_NAME = ":apple: My Weight History"
BUCKET_KEY = "weight11"
ACCESS_KEY = "PLACE YOUR INITIAL STATE ACCESS KEY HERE"
METRIC_UNITS = False
WEIGHT_SAMPLES = 500
THROWAWAY_SAMPLES = 100
WEIGHT_HISTORY = 7
# ---------------------------------
```

- BUCKET_NAME sets the name of the Initial State data bucket that your weight/data will be streamed into. This can be set here and changed later in the UI.
- BUCKET_KEY is the unique bucket identifier that specifies where your data will stream into. If you want to create a different bucket/dashboard, use a different identifier here (*note, if you archive a bucket, you cannot reuse its key in a new bucket).
- ACCESS_KEY is your Initial State account key. If you do not put your ACCESS_KEY in this field, your data will not show up in your account.
- METRIC_UNITS allows you to specify your weight in kg if set to True or lb if set to False.
- WEIGHT_SAMPLES specifies how many measurements are taken and averaged together to get your actual weight. 500 measurements takes about 4-5 seconds and provides fairly accurate results.
- THROWAWAY_SAMPLES specifies the number of samples that are thrown away when you first step on the board. This prevents the initial steps and shifting from throwing off the final measurement. This should always be much less than WEIGHT_SAMPLES.

- WEIGHT_HISTORY sets the number of measurements taken before an extra update is sent. Only measurements taken two hours or more apart count toward the history.

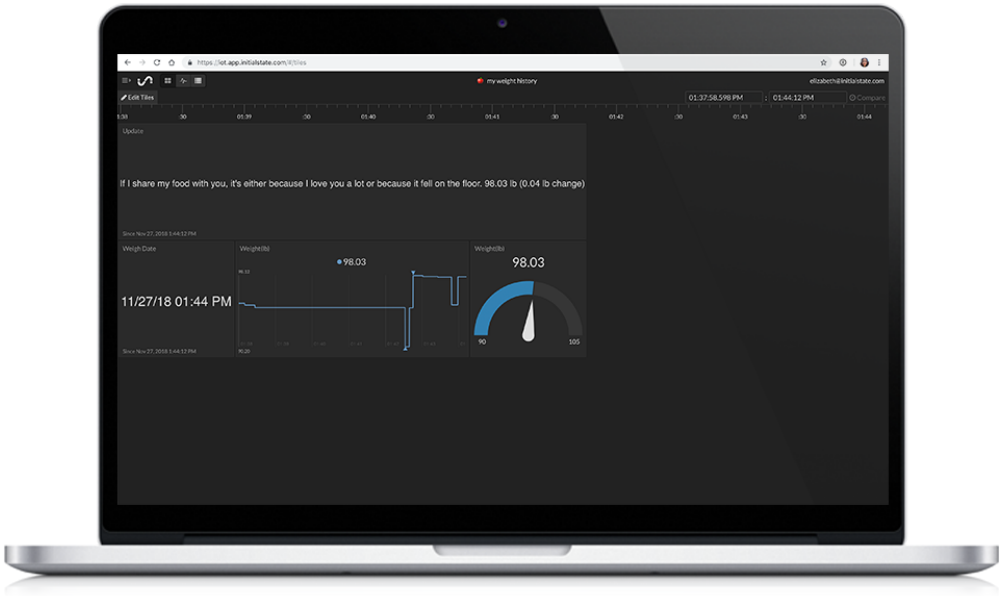
Once you have specified each parameter in this section and saved your changes, you are ready to run the final script. Before we run the script, let's go through what it is going to do.

- At the start of the script, you will be asked to pair your Wii Balance Board with your Raspberry Pi. Use the lever that you hacked together in section Part 2: Hardware Tweaks to press the sync button when prompted.
- Once the script is running, step on the Wii Board to begin measuring your weight. After a 4-5 seconds, your weight will be automatically sent to your Initial State account.
- After we setup SMS notifications (in a couple of steps), you will receive a text message soon after your measurement.

Run the script to start the magic.

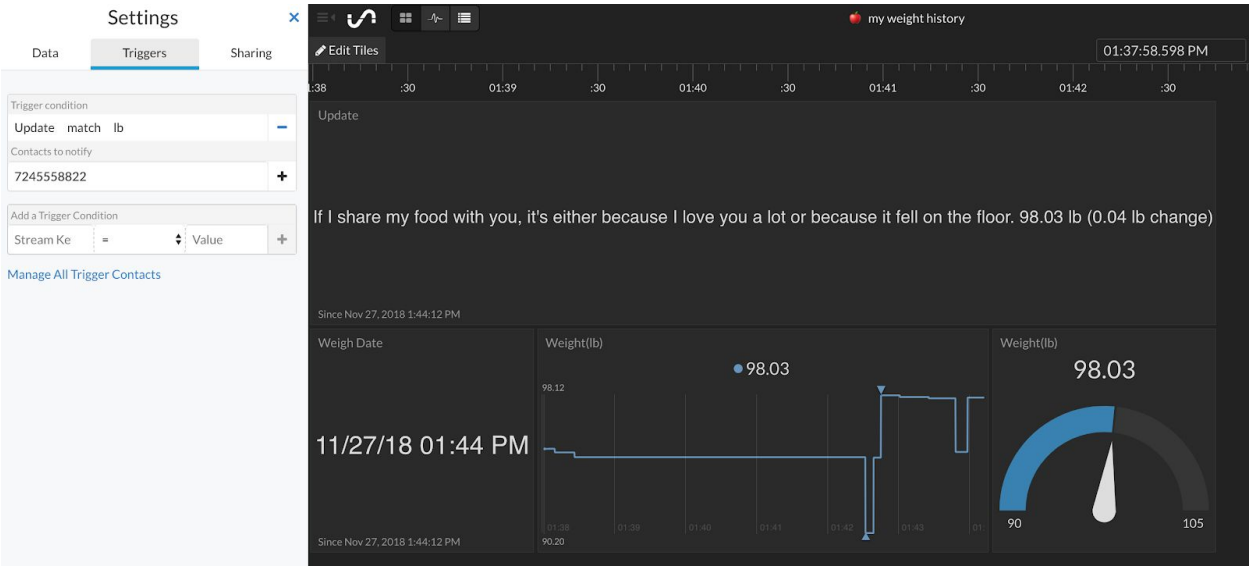
```
$ sudo python smartscale.py
```

Part 3. Dashboard



Go to your Initial State account and click on the new data bucket with the name corresponding to the BUCKET_NAME parameter (i.e. 🍎 My Weight History). Click on Tiles to view your weight history dashboard. You should see three tiles the first time you view your data in Tiles - Update, Weight Date, and Weight (lb). You can customize your dashboard by resizing and moving tiles as well as changing view types and even adding tiles. This dashboard gives you the ability to see your weight history at-a-glance. It is mobile friendly and you can even share it with other people.

Part 3. SMS

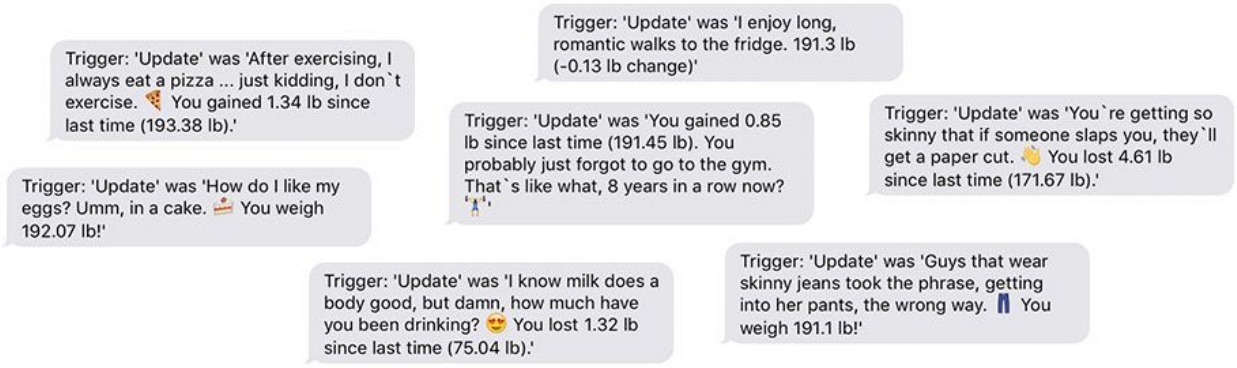


Let's create a SMS alert whenever the scale takes a weight measurement.
We are going to follow the Trigger notification setup process outlined here.

1. Make sure your weight history data bucket is loaded.
2. Click on the bucket's settings (under its name) in the data bucket window.
3. Click on the Triggers tab.
4. Select the data stream to trigger on. You can use the drop-down list to select from existing streams once a data bucket has loaded or you can type in the stream name/key manually. In the example screenshot above, "Update" is selected.
5. Select the conditional operator, in this case 'match'.
6. Select the Trigger value that will trigger an action (manually type in the desired value). Type in lb if you are not using metric units or type in kg if you are using metric units. Whenever the stream "Update" contains "lb" (or "kg"), you will get a text message notification.
7. Click the '+' button to add the Trigger condition.

8. Enter your email address or phone number in the "Contacts to notify" field.
9. Click the '+' button to add the contact information.
10. Input any verification code if adding a new phone number to complete setup.
11. Click the Done button at the bottom to return to the main screen.

Your trigger is now live and will fire when the condition is met.



Once setup is complete, you will get a SMS every time you weigh yourself that contains your weight, how much your weight changed since the last measurement.

Part 4. Conclusions

There is unlimited options for you to build on what you've now created.