This document explains how to set up boto3 on the local system to access a table from DynamoDB and the flow of the program static\_graph.py.

* Follow this to install boto3 and set up the necessary config and credentials file:

<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html>

and refer here for location of config and credentials file :

<https://docs.aws.amazon.com/credref/latest/refdocs/file-location.html>

Caution: In the config file, the region must be the same as the table’s region in DynamoDB.

* To get the credentials to be placed, follow this link:

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SettingUp.DynamoWebService.html>

* For working with dynamodb, look here: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html>

Code:

* The resource that we want to work with is specified as a resource.
* The table name IMU\_Data is specified with the Table parameter to write data into it.
* The table.scan command scans the entire table and gets the data of all the attributes.

Documentation for scan : <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Table.scan>

Api Reference:

<https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Scan.html>

Stack Overflow:

<https://stackoverflow.com/questions/36780856/complete-scan-of-dynamodb-with-boto3>

* The scan command returns a dictionary with a lot of values(mentioned in docs above). The data from the table is under the key ‘Items’ inside the dictionary. We store that in data.
* Then we get the data and store them in individual variables using list comprehension.
* The timestamp is split and only the time part is taken using the split() which will return a list with the date at index 0 and time at index 1. So, we specify [1] to get the time. Then in the time, we don’t want to display the milliseconds as it will be too much clutter. So, [:-4], will make us get until the “end-4” index of the time, which ignores the milliseconds part.
* Each data list is sent to split\_axes function to split into x,y, and z axis data to make it easier to plot.
* All the coordinate data is in the form a string in this form [‘[0, 0, 0]’,’[1, 2, 3]’, …..]. So, we take each set from the list and then strip the brackets( ][ ). The, there is a space before the numbers, so we remove that with replace and then split them by the commas and store them in a list. So, [‘[0, 0, 0]’,’[1, 2, 3]’, …..]. will become [[‘0’,’0’,’0’], [‘1’,’2’,’3’],……]
* Now the string numbers should be converted to float in the next list comprehension.
* Now, all the numbers in the 0th position(i.e, the x-axis) are stored in x and so on.
* These are returned and stored in mag\_x, mag\_y etc, respective to the data.
* Now the plotgraph function plots the data using subplots( allows to have multiple plots in same screen).
* plt.subplots(3, sharex=True, figsize=(10, 15))

This line means that 3 graphs are plotted one below the other and that the x-axis is common.