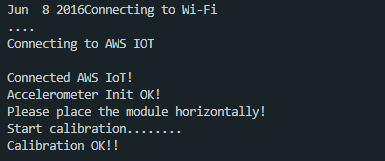
31/01/2020

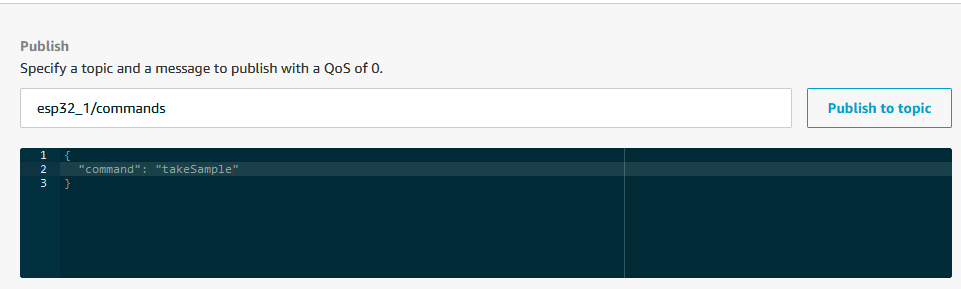
**Task:**

* Bring Project to AWS – IoT Core/ Serverless app
* Create AWS Educate account
* Create topics to Accelerometer Thing
* Use endpoint from AWS
* Run example code from AWS- Building an AWS IoT Core device using AWS Serverless
* Integrate ADXL357\_app lib in the Serverless example.
* Test integration with Lambda function (Published/Subscribed) and DynamoDB
* Forward JSON doc serialization/parsing to the topics.

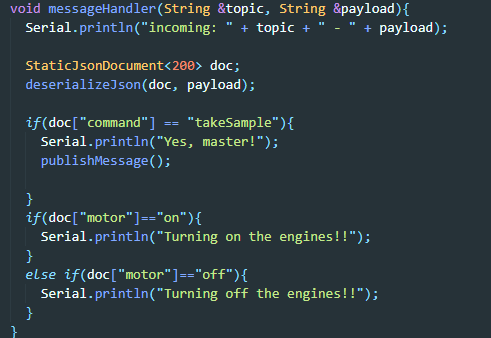
**Results:**

On the edge of the device, ESP32 connects to the Wi-Fi network, establish a connection to the AWS IoT Endpoint using MQTT protocol in the port 8883 and wait an event be triggered. Meanwhile accelerometer is initialized and calibrated.



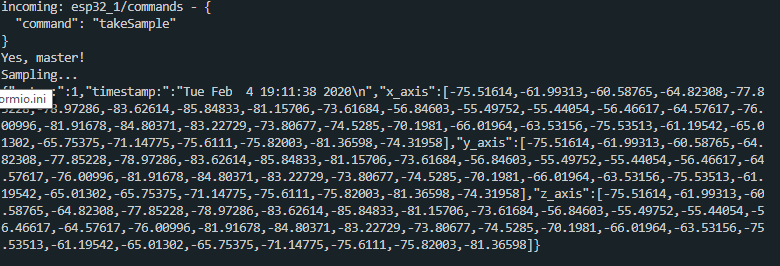
To test the application, I manually trigger the device using the AWS MQTT Client dashboard. Specifying the topic esp32\_1/commands and send the message “takeSample”. 

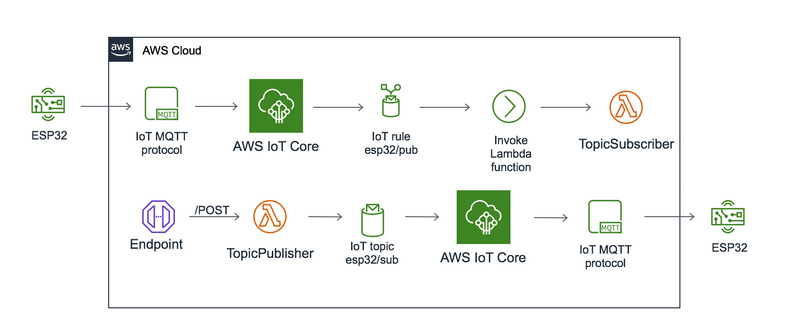
Back in the device the function messageHandler receives the payload/ topic and deserialize the JSON file. The message is then checked and the function publishMessage is called to send back the data in JSON format.





So far, the implementation of the publish method is incomplete, not sending back the payload to topic esp32\_1/samples. But is possible to test the implementation of the JSON doc with the results to be published.





**References:**

<https://aws.amazon.com/iot-core/getting-started/#kits>

<https://aws.amazon.com/blogs/mobile/iot-with-aws-appsync/>

<https://aws.amazon.com/blogs/compute/building-an-aws-iot-core-device-using-aws-serverless-and-an-esp32/>

<https://aws.amazon.com/blogs/iot/collecting-organizing-monitoring-and-analyzing-industrial-data-at-scale-using-aws-iot-sitewise-part-3/>

<https://aws.amazon.com/blogs/iot/asset-maintenance-with-aws-iot-services-predict-and-respond-to-potential-failures-before-they-impact-your-business/>

<https://aws.amazon.com/blogs/iot/using-aws-iot-services-for-asset-condition-monitoring/>

<https://aws.amazon.com/solutions/predictive-maintenance-using-machine-learning/>

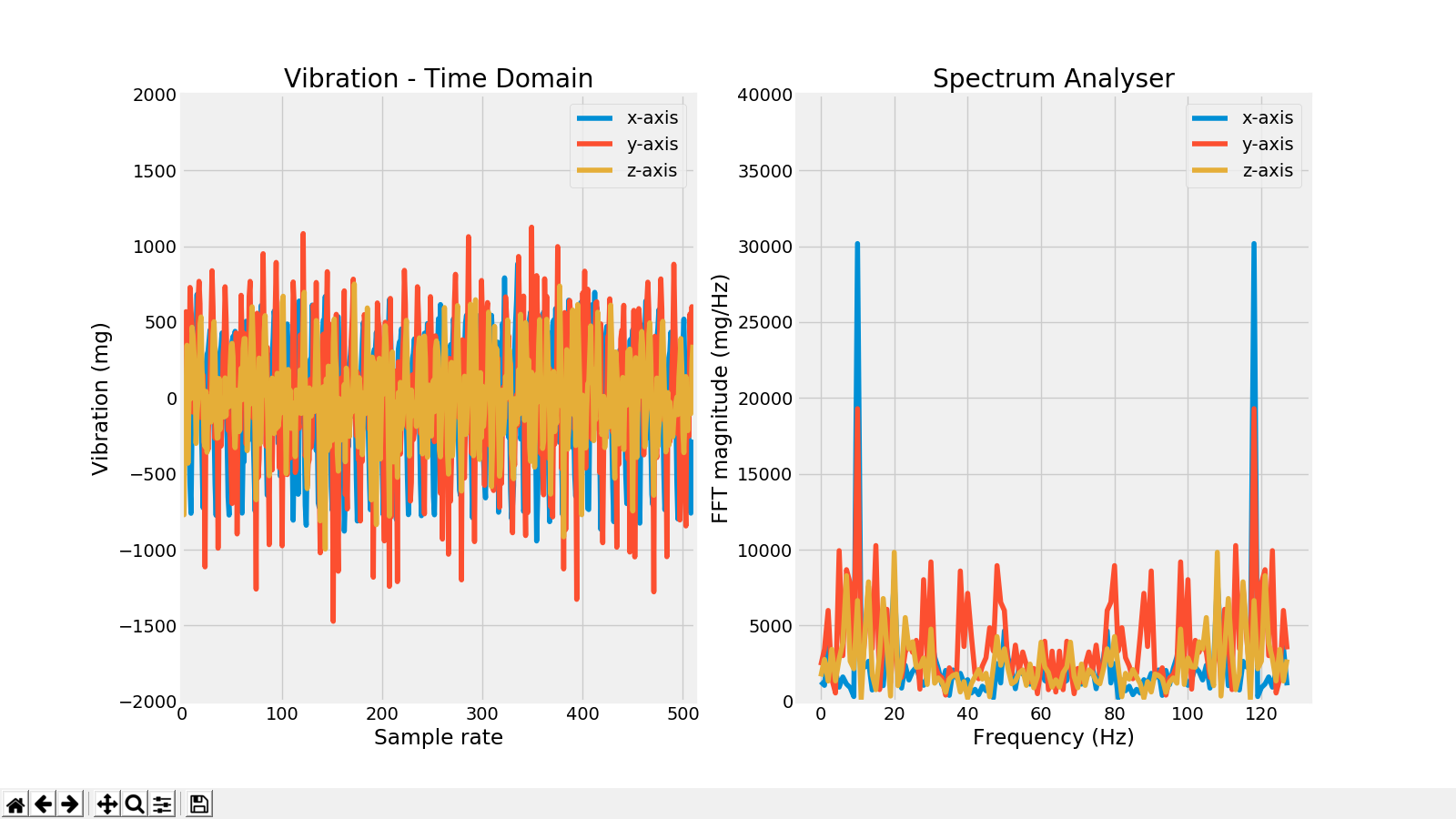
<https://github.com/aws-samples/aws-iot-events-accelerators/tree/master/integratingmachinelearning>

<https://arduinojson.org/v6/doc/>

**Solution:** After investigation two methods can be applied to remove DC offset from the original signal before performing FFT:

* Applying High-Pass Filter
* Subtractiong the Mean of the Original Signal

I opted to subtract the mean of the signal and made use of the detrend() method available in Scipy library to eliminate this:



As shown above I could remove the DC offset.

**References**

<http://blog.originlab.com/data-handling/how-to-remove-dc-offset-before-performing-fft>

<https://www.mathworks.com/help/dsp/ref/dsp.spectrumanalyzer-system-object.html>

<https://makersportal.com/blog/2018/8/14/real-time-graphing-in-python>

<https://www.analog.com/en/analog-dialogue/articles/intro-to-mems-vibration-monitoring.html>

<https://www.researchgate.net/post/What_is_the_effect_of_DC_on_fft_in_MATLAB>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.fft.html>

<https://stackoverflow.com/questions/25735153/plotting-a-fast-fourier-transform-in-python>

<https://stackoverflow.com/questions/51429792/unexpected-fft-results-with-python>

<https://matplotlib.org/3.1.0/gallery/style_sheets/style_sheets_reference.html>

<http://greenteapress.com/thinkdsp/thinkdsp.pdf>

<https://open.umn.edu/opentextbooks/textbooks/think-dsp-digital-signal-processing-in-python>

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.subplots.html>

<https://www.python-course.eu/matplotlib_subplots.php>

<https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py>

<https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem>