Analytic Velocity FFT Release

Test Case Document

V1.0

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Tester** | **Document**  **version** | **Reviewers** |
|  |  |  |  |
| 16-08-2023 | Nidhi Kulkarni | 1.0 | Chethan |
| 16-08-2023 | Purbasha Mahakal | 1.0 | Chethan |

|  |  |
| --- | --- |
| **Abbreviations** | **Description** |
| APP | Application |
| VRMS | Velocity Root-mean square |
| FFT | Fast Fourier transform |
| ACCFFT | Accelerometer FFT |
| VFFT | Velocity FFT |
| MQTT | Message queuing Telemetry Transport |

Contents

1. Introduction
2. Scope of Testing
3. Supporting Tools

3.1 SCISTATCALC-FFT CALCULATOR

3.2 Simple Harmonic Motion Calculator

3.3 MQTT.FX

4. Steps to run the app as executable in service

5. Test cases

5.2 Corner case testing

5.3 High Load Testing

5.4 Long-Term Testing

5.5 End-to-end testing

6. GIT Description

# 1. Introduction

This document outlines the test cases that have been executed and successfully passed during the testing phase of the analytic\_vel\_fft release. The purpose of this document is to provide a comprehensive overview of the tested scenarios, their expected outcomes, and the actual outcomes observed during testing.

2. Scope of Testing

The scope of testing aimed to ensure that the analytic\_vel\_fft software meets the specified requirements, functions as intended, performance, and provides a seamless user experience.

# 3. Supporting Tools

During the testing phase, a variety of supporting tools were employed to enhance the testing process, improve efficiency, and ensure accurate results. These tools contributed significantly to the successful execution of test cases and the overall quality of the software.

3.1 Scistatcalc FFT Calculator

**Link:** [Scistatcalc](https://scistatcalc.blogspot.com/) FFT Calculator

**Description**: This blog post implements a Fast Fourier Transform (FFT) or an Inverse Fast Fourier Transform (IFFT) on a complex input.

**Steps:**

1. Please enter the accelerometer data in the text areas below - one number per line, for each of the real and imaginary input text areas. Alternatively, you can choose to load a CSV file, which must be either a single column of numbers (for a real only input) or two comma-separated columns of numbers.
2. Specify the sampling frequency in arbitrary units (e.G. Hz) in the appropriately labeled text area.
3. To perform the FFT/IFFT, please press the button labeled "perform FFT/IFFT" below the results will populate the text areas below labeled "real output" and "imaginary output", as well as a text area at the bottom that will contain the real and imaginary output joined using a comma.
4. Copy the results to the CSV file.

3.2. Simple Harmonic Motion Calculator

Link: [simple-harmonic-motion-calculator](https://www.mide.com/simple-harmonic-motion-calculator)

DESCRIPTION:

* This tool calculates the variables of simple harmonic motion (displacement amplitude, velocity amplitude, acceleration amplitude, and frequency) given any two of the four variables
* When changing values for displacement, velocity, or acceleration the calculator assumes the frequency stays constant to calculate the other two unknowns.
* When changing the frequency value, the calculator assumes acceleration to be constant and calculates velocity and displacement using this new value for frequency.

Steps:

1. Please enter the frequency value (Hz) and amplitude value (g) in the text areas given.

2. Note the velocity value generated in the text area.

3. Multiply velocity with 0.707 to get VRMS.

**3.3 MQTT.FX**

**Version**:1.0.0

**Description:** MQTT.fx is a versatile and user-friendly MQTT client tool that facilitates communication and interaction with MQTT brokers. Designed to simplify MQTT-based communication, MQTT.fx provides a graphical interface that enables users to publish and subscribe to MQTT topics, monitor messages, and test MQTT functionalities.

**Setup MQTT.fx:**

1. Launch MQTT.fx on the testing machine.

* Configure the MQTT.fx connection settings to connect to the target MQTT broker
  + Broker Address: [Broker Address]
  + Port: [Port Number]
  + Security (if applicable): [TLS/SSL Configuration]

1. **Publish Data**

Using MQTT.fx, publish a test message to the [target topic] with a specific payload.

1. **Verify Message Received**

* Subscribe to the [target topic] using MQTT.fx.
* Verify that the published test message is received on the subscribed topic.
* Compare the received payload with the expected payload.

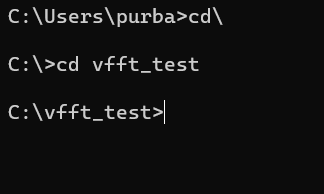
1. **Steps to run the exe as windows service:**
2. Check NSSM tool is present or not. If not download NSSM and move it in the C drive.
3. Copy the executable in desired folder.
4. Make service for the same by NSSM, name it as exe name -> “ class\_structure\_velo\_fft”
5. Start the service named “class\_structure\_velo\_fft”.
6. Change the properties as required and apply
7. Start MQTT server.
8. Database should be present.
9. Log folder should be present in the C drive.

**Path for Log Folder: 'C:\****logs\_Vfft'**

*Once service is started, test cases can be performed. No need to do these steps every time we check any test case. If there are any code changes then new executable needs to be created and add on service again after removing the previous one.*

* **Corner tests**
* ***Common procedure for test no 1-7*:**

1. Start the vSens Velocity service in machine 192.168.x.x and configure it for broker x.x.x.x which is mentioned on page on 4.
2. Type "CMD" in window search bar, and open command prompt.
3. Type: *cd/*
4. Type: *cd vfft\_test*



1. Run the data\_feeding.py script by executing the command:   
   *python data\_feeding.py* or   
   *py data\_feeding.py*
2. In the CMD an input window will appear:  
   " Please choose and enter any testing options from bellow list:
3. Type "S" for Sampling data missing in payload test
4. Type "W" for window size missing in payload test
5. Type "T" for timestamp missing in payload test
6. Type "D" for ACC data missing in payload test
7. Type "DW1" for ACC Data length is lesser than the window size in payload test
8. Type "DW2" for ACC Data length is higher than the window size in payload test
9. Type "DW3" for ACC Data length is higher than the window size, but not doubled in payload test "

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – Sampling data** |
| **Test Object** | Data in queue is incomplete - sampling rate missing in payload. |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "S" and enter. |
| **Expected Output** | 1. Drop the payload and log the following message "Sampling rate missing, packet is dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – window size** |
| **Test Object** | Data in queue is incomplete - window size missing in payload. |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "W" and enter. |
| **Expected Output** | 1. Drop the payload and log the following message "window size is missing, packet is dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – timestamp** |
| **Test Object** | Data in queue is incomplete - timestamp missing in payload |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "T" and enter. |
| **Expected Output** | 1. Drop the payload and log the following message " timestamp missing missing in payload, packet is dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – ACC data** |
| **Test Object** | Data in queue is incomplete - ACC Data missing in the payload |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "D" and enter. |
| **Expected Output** | 1. Drop the payload and log the following message "ACC Data missing in payload, packet is dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – comparing ACC data with window size (1)** |
| **Test Object** | Data in queue is incomplete - ACC Data length is lesser than the window size |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "DW1" and enter. |
| **Expected Output** | 1. Drop the payload and log the following message "ACC Data length is not matching with the window size in payload, packet is dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – comparing ACC data with window size (2)** |
| **Test Object** | Data in queue is incomplete - ACC Data length is higher than the window size ACC data length =8192  Window size =4096 |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "DW2" and enter. |
| **Expected Output** | 1. VFFT, VRMS, ACC FTT calculations and publishing will happen twice for each payload. 2. Timestamp will be calculated and published for each FFT window |

|  |  |
| --- | --- |
| **Test No and Name** | 1. **Payload format check to publish – comparing ACC data with window size (3)** |
| **Test Object** | Data in queue is incomplete - ACC Data length is higher than the window size ACC data length =8000  Window size =4096 |
| **Procedure** | 1. Please refer " Common procedure for test no 1-7" in page no 6 2. Please type "DW3" and enter. |
| **Expected Output** | 1. 1 FFT calculations and publishing will happen. 2. Other window calculations will not happen because of insufficient data, following log message "ACC Data length is not matching with the window size in payload, packet dropped" |

|  |  |
| --- | --- |
| **Test No and Name** | **8- MQTT Connect Test** |
| **Test Object** | Check Whether the Connection is Successful or Not |
| **Procedure** | 1. Provide accurate MQTT configurations and credentials by filling in the necessary details within the Vegam MQTT JSON file, including broker IP, broker port, username, and password. 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The log message “MQTT connection successful: **Broker IP, BROKER PORT”** will be savedin *mqtt\_subsribe\_publish\_logger.log.* 2. It indicates that the MQTT connection has been established successfully on the specified broker port and broker IP. 3. This message confirms that the communication between the client and the MQTT broker has been established without errors |

|  |  |
| --- | --- |
| **Test No and Name** | **9- MQTT Improper Configuration - Invalid Broker IP** |
| **Test Object** | Verify that the service handles an **invalid broker IP** configuration properly and raises an appropriate error or exception |
| **Procedure** | 1. Modify the Vegam MQTT JSON file to include an invalid broker IP address (e.g., an IP that doesn't exist or is unreachable). 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The log message “MQTT connection unsuccessful: **Broker IP, BROKER PORT”** will be saved in *mqtt\_subsribe\_publish\_logger.log.* 2. The log message indicates that the MQTT connection is unsuccessful because of an invalid broker IP. |

|  |  |
| --- | --- |
| **Test No and Name** | **10- MQTT Improper Configuration - Invalid Broker Port** |
| **Test Object** | Verify that the service handles an **invalid broker port** configuration properly and raises an appropriate error or exception |
| **Procedure** | 1. Modify the data type of the broker port to a string in the Vegam MQTT configuration file. 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The log message “MQTT connection unsuccessful: **Broker IP, BROKER PORT”** will be saved in *mqtt\_subsribe\_publish\_logger.log.* 2. Log message indicates that the MQTT connection is unsuccessful because of an invalid broker port. |

**\**

|  |  |
| --- | --- |
| **Test No and Name** | **11- Empty MQTT Configuration File** |
| **Test Object** | Verify that the service handles an empty MQTT configuration file properly and raises an appropriate error or exception |
| **Procedure** | 1. Create an empty configuration file 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The log message “Vegam MQTT configuration file is empty”will be saved in *mqt\_subscribe\_publish\_logger.log.* 2. It indicates that the MQTT connection is unsuccessful because the configuration file is empty. |

|  |  |
| --- | --- |
| **Test No and Name** | **12- MQTT Configuration File is missing** |
| **Test Object** | Verify that the service handles a missing MQTT configuration file properly and raises an appropriate error or exception |
| **Procedure** | 1. Remove or temporarily move the Vegam MQTT JSON file to a different location to simulate a missing configuration file. 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The log message “Vegam MQTT configuration file is empty”will be saved in *mqt\_subscribe\_publish\_logger.log.* 2. The log message indicates that the MQTT connection is unsuccessful because the configuration file is missing. |

|  |  |
| --- | --- |
| **Test No and Name** | **13- MQTT DISCONNECT** |
| **Test Object** | Verify that the system correctly handles MQTT disconnection and raises appropriate events or exceptions when the connection is lost |
| **Procedure** | 1. Provide proper MQTT configurations and credentials by filling in the necessary details within the Vegam MQTT JSON file, including broker IP, broker port, username, and password. 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. 3. Switch off the MQTT broker 4. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected Output** | 1. The following log message will be savedin *mqtt\_subsribe\_publish\_logger.log* 2. “MQTT connection successful: **Broker IP, BROKER PORT”** 3. “MQTT connection dropped: **Broker IP, BROKER PORT”** |

|  |  |
| --- | --- |
| **Test No and Name** | **14- MQTT Auto-Reconnect** |
| **Test Object** | Verify that the system successfully handles MQTT auto-reconnect after a temporary network disruption and restores communication with the broker |
| **Procedure** | 1. Provide proper MQTT configurations and credentials by filling in the necessary details within the Vegam MQTT JSON file, including broker IP, broker port, username, and password. 2. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. 3. Switch off the MQTT broker 4. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics 5. Wait for 2 minutes 6. Switch on the MQTT broker 7. Monitor the logs and subscribe to the specified topics in MQTT.fx to observe if any messages or data are being received for the subscribed topics. |
| **Expected output** | 1. The following log message will be savedin *mqtt\_subsribe\_publish\_logger.log* 2. “MQTT connection successful: **Broker IP, BROKER PORT”** 3. “MQTT connection dropped: **Broker IP, BROKER PORT”** 4. “MQTT connection reestablished”: **Broker IP, BROKER PORT”** |

* **High Load tests**

|  |  |
| --- | --- |
| **Test No and Name** | **1- High load test** |
| **Test Object** | **Stress testing the application with multiple subscribers and publishers** |
| **Procedure** | 1. Start *Monitor\_System\_Behavior* script to document the base level CPU and memory utilization before starting the vSens Velocity service and get the result in a text file with a time interval of 1 hour and continuous checking for the same, make sure no unnecessary services are running on this machine 2. Start the vSens Velocity service in machine 192.168.x.x and configure it for broker x.x.x.x which is mentioned on page on 4. 3. Configure the *sensor\_data\_generator* script with the following configurations  #with file name###    1. Number of sensors: 50    2. Burst Interval Time: 10 minutes    3. Data Type: Raw Data 8192 samples 3 Axis 4. Configure and start the *sensor\_data\_collector* script with the following configurations:    1. Number of sensors: 50    2. Burst Interval Time: 10 minutes    3. Data Type: Raw Data 8192 samples 3 Axis    4. Test time: 12 hours 5. Start the *sensor\_data\_generator* script 6. The *Monitor\_System\_Behavior script* will log the results in a text file after 12 hours *[service\_behaviour.txt*] |
| **Expected Output** | 1. In the result file, the following results are expected:  * Data Loss %: < 0.1% * Data Integrity: 0% error * Maximum CPU utilization: < X% * Maximum Memory Utilization: < X% * Average CPU utilization: < X% * Average Memory Utilization: < X% * Number of service breakdowns or restarts: 0 |

* **Long Term tests**

|  |  |
| --- | --- |
| **Test No and Name** | **1-Long Term test** |
| **Test Object** | **To evaluate the system's performance, stability, data integrity, and resource utilization under continuous high load conditions for 7 days.** |
| **Procedure** | 1. Start the vSens Velocity service in machine 192.168.x.x and configure it for broker x.x.x.x which is mentioned on page 4. 2. Start *Monitor\_System\_Behavior* script to document the base level CPU and memory utilization before starting the service and get the result in a text/excel file with time interval of 1 hour and make sure no unnecessary services are running on this machine. 3. Configure the *sensor\_data\_generator*  script with the following configurations:    1. Number of sensors: 5    2. Burst Interval Time: 10 minute    3. Data Type: Raw Data 8192 samples 3 Axis 4. Configure and start the *sensor\_data\_collector* script with the following configurations:    1. Number of sensors: 5    2. Burst Interval Time: 10 minute    3. Data Type: Raw Data 8192 samples 3 Axis    4. Test time: 7 days 5. Start the *sensor\_data\_generator* script 6. The *Monitor\_System\_Behavior script* will log the results in a text file after 12 hours *[service\_behaviour.txt*] |
| **Expected Output** | 1. In the results file, the following results are expected:  * Data Loss %: < 0.1% * Data Integrity: 0% error * Maximum CPU utilization: < X% * Maximum Memory Utilization: < X% * Average CPU utilization: < X% * Average Memory Utilization: < X% * Number of service breakdowns or restarts: 0 |

* **End to End testing with IoT Platform**

|  |  |
| --- | --- |
| **Test No and Name** | **1-End-to-End Testing** |
| **Test Object** | Verify the end-to-end behaviour of the IoT system involving simulated sensor data transmission, data processing, and data publishing. |
| **Procedure** | 1. Start the vSens Velocity service in machine 192.168.x.x and configure it for broker x.x.x.x which is mentioned on page 4. 2. Configure the *sensor\_data\_generator*  script with the following configurations:   Number of sensors: 5  Burst Interval Time: 10 minute  Data Type: Raw Data 8192 samples 3 Axis published at different frequencies.   1. Configure and start the *sensor\_data\_collector* script with the following configurations:   Number of sensors: 5  Burst Interval Time: 10 minute  Data Type: Raw Data 8192 samples 3 Axis published at different frequencies.  Test time: 2 hours  4. Start the *sensor\_data\_generator.* |
| **Expected output** | 1. Check if messages were published to the expected 9 topics with the correct payloads in MQTT fx.   For example, VRMSX output should be published on **sensor\_id/VelocityRMSX,** VRMSY output should be published on **sensor\_id/VelocityRMSY,** VRMSZ output should be published on **sensor\_id/VelocityRMSZ**   1. Verify whether peaks are observed at the same frequency as the published data frequency 2. If the charts on the dashboard are not being displayed, proceed to restart the test. |

**GIT DESCRIPTION:**

**GIT LINK:**  [GIT\_LINK](https://github.com/VegamSolutions/Vegam_Analytics/tree/updated_dir_structure)

**GIT COMMIT ID:** 0beb44b5227bb965abf62c6e5742fbd218ee32db