Verification and Validation Report: Attitude Check

Adrian Sochaniwsky

April 6, 2024

1 Revision History

Date	Version	Notes
April 5, 2024	1.0	Initial version

2 Symbols, Abbreviations and Acronyms

symbol	**description**
T	Test
UT	Unit Test

Contents

List of Tables

This document reports the results of executing the VnV Plan.

3 Functional Requirements Evaluation

This section covers the evaluation of the functional requirements.

3.1 T1

This test evaluates the input and output requirements of Attitude Check. This test is implemented by the first 3 unit tests in https://github.com/adrian-soch/attitude_check/blob/main/test/attitude_check_test.cpp. This requirement is **satisfied** by Attitude Check.

3.2 T2

This test evaluates the calculations when accel, gyro, and mag data is provided. To conduct this test the following steps are required:

1. Create a csv file with sensor data and the ground truth with create_sensor_csv.py
2. Use basic_orientation_calculation.cpp to estimate quaternions.
3. Calculate RMSE for the inclination component of the quaternion using compare_results.py

Table 1: Accel, Gyro, Mag Test Results

Program	Inclination RMSE (deg)
Attitude Check	7.027 deg
AHRS	7.218 deg
% Difference	2.65 %

Since $2.65 < 3$ this test passes. Furthermore, Attitude Check is more accurate than the AHRS implementation for this test data.

3.3 T3

This test evaluates the calculations when accel and gyro data is provided. It is conducted using the procedure from T2.

Table 2: Accel and Gyro Test Results

Program	Inclination RMSE (deg)
Attitude Check	5.9709
AHRS	5.9708
% Difference	0.002 %

Since $0.002 < 3$ this test passes.

4 Nonfunctional Requirements Evaluation

This section covers the evaluation of the nonfunctional requirements.

4.1 Accuracy

See T2 and T3 for accuracy results.

4.2 Understandability

This NFR was outside the scope of the VnV Plan.

4.3 Performance

The runtime for the SparkFun Razor 9DoF IMU are reported:

- (Acc, Gyr, Mag) estimation: 1.5 ms
- (Acc, Gyr) estimation: 0.8 ms

Both are fast enough to keep up with average sensor update rates (20 - 600 Hz).

4.4 Maintainability

A likely change was not implemented, thus this test was not executed.

4.5 Portability

Attitude Check was successfully executed on x86 and ARM hardware, it compiles for Linux and Arduino. Thus, it passes the portability test.

5 Unit Testing

Each file with source code has its own unit test file (https://github.com/adrian-soch/attitude_check/tree/main/test). Each commit tot the main branch must pass all unit tests. There are 36 unit tests in total, covering 100% of the code.

Each unit test and its result are below:

```
  Test project /home/adrian/dev/attitude_check/build/test
      Start  1: ACheck_Test_Fixture.invalid_init
 1/36 Test  #1: ACheck_Test_Fixture.invalid_init ........................   Passed
      Start  2: ACheck_Test_Fixture.invalid_update
 2/36 Test  #2: ACheck_Test_Fixture.invalid_update ......................   Passed
      Start  3: ACheck_Test_Fixture.marg_zero_gyro
 3/36 Test  #3: ACheck_Test_Fixture.marg_zero_gyro ......................   Passed
      Start  4: ACheck_Test_Fixture.marg_zero_mag
 4/36 Test  #4: ACheck_Test_Fixture.marg_zero_mag .......................   Passed
      Start  5: ACheck_Test_Fixture.marg_zero_acc
 5/36 Test  #5: ACheck_Test_Fixture.marg_zero_acc .......................   Passed
      Start  6: ACheck_Test_Fixture.imu_zero_gyro
 6/36 Test  #6: ACheck_Test_Fixture.imu_zero_gyro .......................   Passed
      Start  7: ACheck_Test_Fixture.imu_zero_acc
 7/36 Test  #7: ACheck_Test_Fixture.imu_zero_acc ........................   Passed
      Start  8: ACheck_Test_Fixture.set_quaternion
 8/36 Test  #8: ACheck_Test_Fixture.set_quaternion ......................   Passed
      Start  9: ACheck_Test_Fixture.set_get_gain
 9/36 Test  #9: ACheck_Test_Fixture.set_get_gain ........................   Passed
      Start 10: ACheck_Test_Fixture.get_initial_orientation_imu
10/36 Test #10: ACheck_Test_Fixture.get_initial_orientation_imu .........   Passed
      Start 11: ACheck_Test_Fixture.get_initial_orientation_marg
```

```
11/36 Test #11: ACheck_Test_Fixture.get_initial_orientation_marg ........  Passed
      Start 12: ACheck_Estimator_Test_Fixture.update_marg_with_intitial
12/36 Test #12: ACheck_Estimator_Test_Fixture.update_marg_with_intitial .  Passed
      Start 13: ACheck_Estimator_Test_Fixture.update_imu_with_intitial
13/36 Test #13: ACheck_Estimator_Test_Fixture.update_imu_with_intitial ..  Passed
      Start 14: quat_test_suite.invalid_init
14/36 Test #14: quat_test_suite.invalid_init ............................  Passed
      Start 15: quat_test_suite.conjugate_f
15/36 Test #15: quat_test_suite.conjugate_f .............................  Passed
      Start 16: quat_test_suite.conjugate_d
16/36 Test #16: quat_test_suite.conjugate_d .............................  Passed
      Start 17: quat_test_suite.product
17/36 Test #17: quat_test_suite.product .................................  Passed
      Start 18: quat_test_suite.norm_d
18/36 Test #18: quat_test_suite.norm_d ..................................  Passed
      Start 19: quat_test_suite.norm_f
19/36 Test #19: quat_test_suite.norm_f ..................................  Passed
      Start 20: quat_test_suite.scalar_f
20/36 Test #20: quat_test_suite.scalar_f ................................  Passed
      Start 21: quat_test_suite.add_f
21/36 Test #21: quat_test_suite.add_f ...................................  Passed
      Start 22: quat_test_suite.subtract_f
22/36 Test #22: quat_test_suite.subtract_f ..............................  Passed
      Start 23: quat_test_suite.subtract_equals_f
23/36 Test #23: quat_test_suite.subtract_equals_f .......................  Passed
      Start 24: quat_test_suite.set_f
24/36 Test #24: quat_test_suite.set_f ...................................  Passed
      Start 25: quat_test_suite.to_array
25/36 Test #25: quat_test_suite.to_array ................................  Passed
      Start 26: utilities_test_suite.euler_d
26/36 Test #26: utilities_test_suite.euler_d ............................  Passed
      Start 27: utilities_test_suite.euler_f
27/36 Test #27: utilities_test_suite.euler_f ............................  Passed
      Start 28: utilities_test_suite.euler1_f
28/36 Test #28: utilities_test_suite.euler1_f ...........................  Passed
      Start 29: utilities_test_suite.euler2_f
29/36 Test #29: utilities_test_suite.euler2_f ...........................  Passed
      Start 30: utilities_test_suite.rotm_d
30/36 Test #30: utilities_test_suite.rotm_d .............................  Passed
      Start 31: utilities_test_suite.rotm_f
```

```
31/36 Test #31: utilities_test_suite.rotm_f .............................  Passed
      Start 32: initializers_test_suite.acc_d
32/36 Test #32: initializers_test_suite.acc_d ...........................  Passed
      Start 33: initializers_test_suite.acc1_f
33/36 Test #33: initializers_test_suite.acc1_f ..........................  Passed
      Start 34: initializers_test_suite.acc2_f
34/36 Test #34: initializers_test_suite.acc2_f ..........................  Passed
      Start 35: initializers_test_suite.mag_d
35/36 Test #35: initializers_test_suite.mag_d ...........................  Passed
      Start 36: initializers_test_suite.mag1_f
36/36 Test #36: initializers_test_suite.mag1_f ..........................  Passed

100% tests passed, 0 tests failed out of 36
```

6 Changes Due to Testing

Two bugs were caught when creating unit tests for the Attitude Check module, they were arithmatic errors that were corrected on the spot. Furthermore, during this process 2 bugs were found in a popular open source repository. See:

- https://github.com/Mayitzin/ahrs/issues/111
- https://github.com/Mayitzin/ahrs/issues/112

7 Automated Testing

The unit tests are setup to run automatically when a Pull Request is opened, and after any commit is made to the `main` branch. The GitHub workflow runs the same command as the `build.sh` script that is used locally to build and run tests.

8 Trace to Requirements

Table 3 shows the traceability between tests and requirements.

Table 3: Relation of Test Cases to Requirements.

	R1	R2	R3	R4	R5	NFR1	NFR2	NFR3	NFR4	NFR5
T1	X				X					
T2	X	X	X		X	X				
T3	X	X		X	X	X				
T4						X				
T5							X			
T6								X		
T7									X	
T8										X

9 Trace to Modules

Table 4 shows the traceability between tests and modules.

Table 4: Relation of Test Cases to Modules.

	M0	M1	M2	M3	M4	M5	M6
T1						X	
T2						X	
T3						X	
T4	X	X	X	X	X	X	
T5	X	X	X	X	X	X	
T6						X	X
T7							
T8							X

10 Code Coverage Metrics

The CI pipeline automatically uploads test coverage on the `main` branch here: https://app.codecov.io/gh/adrian-soch/attitude_check/tree/main/src (scroll to the bottom and click on individual files). Test coverage is 100% for all 5 files.