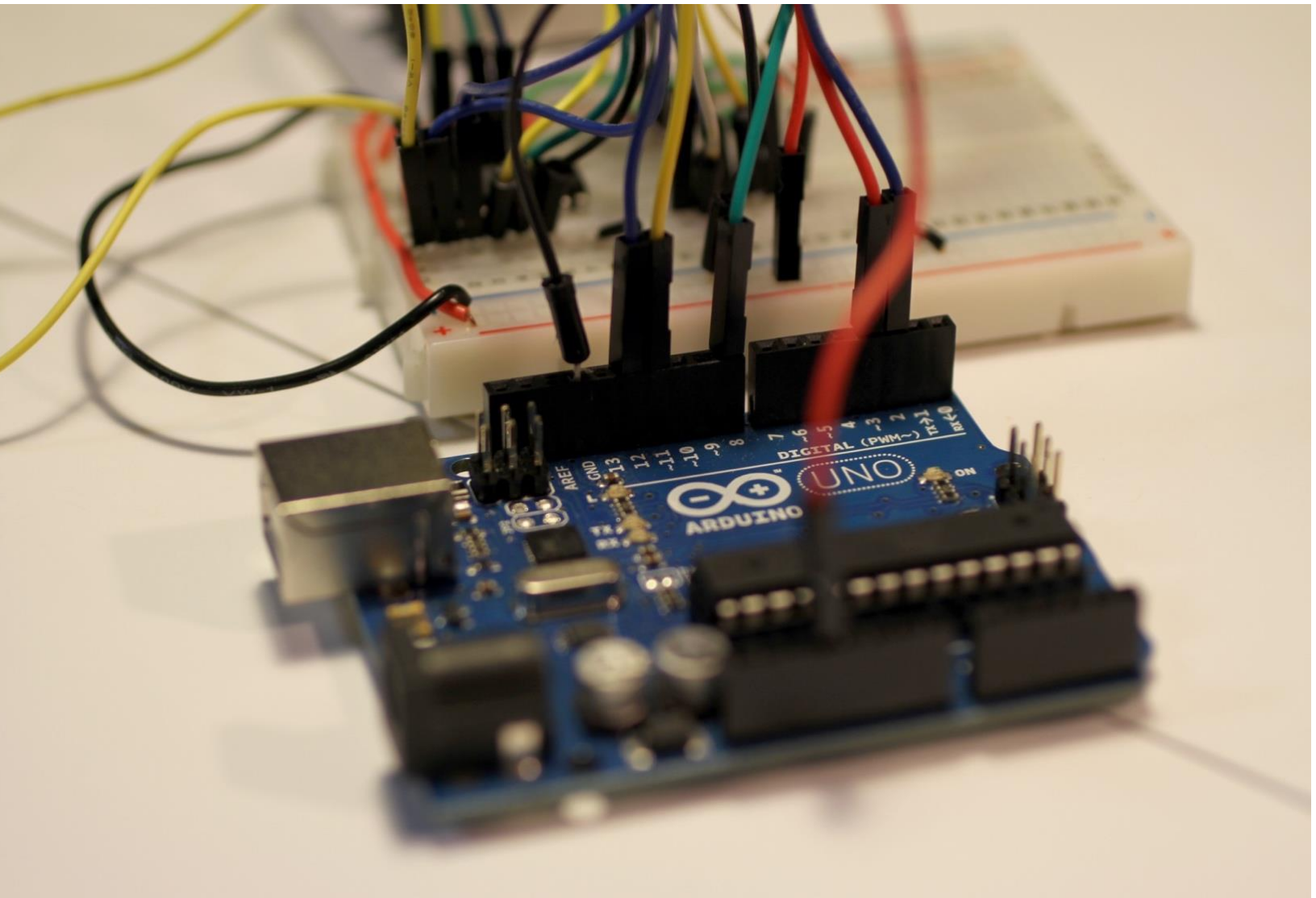
Energy Management in Low Power Wireless Sensor Networks

Testing Document

Cloud Seven Consulting

THE UNIVERSITY OF WESTERN AUSTRALIA



Energy Management in Low Power Wireless Sensor Networks
Team 14

Authors
Peter Bouvy (21299044)
Yung Ren Chin (21247413)
Aaron Hurst (21325887)
Khanh Tan (Jamie) Phan (21326604)
Matthew Ramanah (21317297)
Jake Sacino (21132001)
Andy Ta (21317377)

Testing Document

Project Partner:
Mr Mark Callaghan, ATAMO

Supervisor:
Mr Marcus Pham

Unit Coordinator:
Dr Sally Male

Group Meeting Day and Time:
Thursday 4pm

Version 1.6

[1]*Cover page image kindly provided by Visual Hunt [1]*

Revision History

Date	Version[2]	Description	Author
28/09/2017	1.0	Creation of initial template.	Jake Sacino
30/09/2017	1.1	Structure of tests	Jamie Phan
2017/10/05	1.2	Added Unit Tests: • TU_SH/DnSys_FlashBlink_Op • TU_SH/DnUart_Cli_Op • TU_SH/DpUart_External_Op • TU_SH/DnJoin_Cli_Op • TI_SH-NM/DnJoin_Mode4_Op	Jamie Phan
21/10/2017	1.3	Added Unit Tests: • TU_DB/Put_Exclusive_Op • TU_DB/Get_Exclusive_Op • TU_CI/DataTransfer_S3_Op	Andy Ta
23/10/2017	1.4	Added Unit Tests: • TU_SH/DpFrameHdr_PayloadHdr_Op • TU_SH/DpReserveField_Payload_Op • TU_SH/DpSample_SampleTime_Op • TU_SH/DpSample_SampleDiag_Op • TU_SH/DpSample_SampleSens_Op Added Integration Test: • TI_SH/DpSample_SampleMain_Op	Aaron Hurst
26/10/2017	1.5	Added Unit Tests: • TU_SH/ Load_Webpage_Op • TU_SH/User_Authenticate_Op	Jake Sacino
26/10/2017	1.6	Added Integration Test: • TI_DB-WA/Connection_Mock_Op	Andy Ta

[2]*Incrementing the version by 0.1 denotes a minor change; incrementing by 1.0 denotes a significant change*

Contents

1 Introduction

The proliferation of data monitoring technologies has enabled multifarious insights into environmental conditions, system reliability, consumer behaviour, and a myriad of other fields and industries [2, 3, 4]. Sensors are one such technology, able to detect events or changes in their environment and send this information to other electronics. Energy storage poses a constraint on sensor operation in rural environments; a lack of grid connectivity exposes the importance of sagacious energy management.

1.1 Overview

Cloud Seven Consultants (CSC) has been contracted by ATAMO (the client) to investigate energy management in low-power wireless sensor networks (WSN). The project requires integration of a third party WSN with an ATAMO Arduino based development platform. Sensor data (e.g. temperature, vibration) must be measured and periodically reported over the wireless network to a cloud-based database. The firmware in the sensor must provide for reliable communications while keeping tight constraints on battery energy usage. This information must be accessible on the web via a graphical user interface (GUI).

1.2 Purpose

The Test Plan document documents and tracks the necessary information required to effectively define the approach to be used in the testing of the project's product. The Test Plan document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user and other stakeholder whose input/approval into the testing process is needed.

2 Testing Plan

There are three suites of tests that are to be performed during the duration of the development in design, and during deployment of the system.

2.1 Risk Management

CSC employs the University of Western Australia's risk management approaches.

2.1.1 Likelihood Qualification

Table identifies 5 possible likelihood assessments and the qualification for each.

Likelihood	Qualification	Score
Rare	The event may only occur in exceptional circumstances. Has not occurred in the education Sector in Australia or elsewhere.	1
Unlikely	Unlikely to occur but could happen. Has happened in Australia in the education sector or elsewhere in the last 10 years	2
Possible	Could occur sometime in the next 3-5 years. Has happened in the education sector in the recent past.	3
Likely	Could occur sometime in the next 3-5 years. Has happened in the education sector in the recent past.	4
Probable	Expected to occur on a regular basis and has repeatedly occurred in UWA	5

2.2 Crucial Equipment

Crucial equipment refers to pieces of equipment that may be used in test however no readily available[1] source exists for the replacement of the equipment if it is damaged. Because these pieces of equipment are difficult to replace, extra care must be taken when they are required during testing.

Some items may be classified as 'crucial' in some circumstances only, for example the SmartMesh-IP Starter Kit may only be 'crucial' during development, as C7C does not have sufficient time to replace the item, and will cause project delivery delays.

Item	Reason	Note
SmartMesh-IP Starter Kit	Was provided by the client, and will be difficult to replace	This is only relevant during development

2.3 Test Descriptions

Standard test key-words must be used in descriptions to ensure communication of the test behaviour and outcomes is easily understood.

2.3.1 Expected Outcomes

The following keywords should be used when describing the expected outcome of the test:

1. "op" (Operational): To indicate the unit of work is expected to remain operational after the test
2. "nop" (Non-Operational): To indicate the unit of work is expected to be un-operational after the test
3. "ex" (Exception): To indicate the unit of work is expected to raises an exception to prevent system error

[1] Readily available refers not only to the ability to acquire but the lead time associate in procurement.

2.4 Test Identification

Test will carry a unique identification as follows:

T[*suite*]_[*SubSystem*]/[*Unit*]_[*Circumstance*]_[*Outcome*]

Where;

- **Suite** is one of
  a. 'u' (unit)
  b. 'i' (integration)
  c. 's' (system)
- **Sub-system** (full' if all systems). This should be one of the following (if multiple are used, then they should be dash-delimited (e.g. 'SH-NM'):
  a. 'SH' for any sub-system related to the Sensor-Host
  b. 'NM' for any sub-system related to the Network Manager
  c. 'DB' for any sub-system related to the cloud database
  d. 'WA' for any sub-system related to the cloud web-app
  e. 'CI' for any sub-system relating the cloud infrastructure (that is not part of DB or WA)
- **Unit** of work being tested
- **Circumstances** of Test
  a. This should describe the unique circumstances or state of the test. For example, descriptions of 'how the device under test is connected', 'what was the last state it was under', 'what inputs are being used'. A short set of keywords should be used.
- Expected **outcome** of Test. This should be one of the following:
  a. 'op' to indicate an Operational outcome
  b. 'nop' to indicate a Non-Operational outcome
  c. 'ex' to indicate an Exception outcome

Note for the standardisation of names and useability within software, all non-alpha characters are to be removed and spaces and punctuation characters should be substituted with Camel-Casing. Underscore characters that would normally appear in a description field should also be substituted with Camel-Casing such that only 3 underscore characters are used (as seen above).

2.4.1 Revision History

Test revisions are to be tracked, however there is no need to include the revision of the test in the identifier, as only the most recent revision should be shipped to production. A revision history table must be provided with all tests.

3 Unit Testing

A unit, in the context of this testing plan, refers to a discrete testable item which is unable to be divided any further without resulting in loss of any testable functionality. Note that items provided to C7C which C7C considers 'black-boxed' are also considered units.

Unit testing is crucial in the development and deployment of the system. During development, it provides confidence in the unit, and significantly reduces effort in identification of errors in future development processes and tests. In deployment, unit tests assist in the identification of any manufacturing or assembly errors and mitigates the likelihood of defects in the final system.

3.1 Test Entry & Exit Criteria

The unit-testing phase begins with development and continues throughout as new features are designed. As the design of the system is test-driven, unit testing will be ubiquitous throughout development.

3.1.1 Dependencies

Unit tests have no dependencies; if a test requires more than one unit, it is promoted to integration testing in section 4.

3.1.2 Suspension Criteria

Unit tests will be suspended under the following conditions:

1. The test in question has a risk level of 'Moderate' or higher
2. The test in question poses a hazard to any persons operating the tests, or are nearby the testing site.
3. The test in question poses a risk of damage to any crucial pieces of equipment as described in section 2.2

3.2 Test Approach

Unit tests will be assessed with a two-level scale – pass or failure. Each test will test one single unit of functionality, and failure to pass the test will then require:

- Re-development efforts to ensure the unit will pass the test.
- Identification of failure and removal of defective item during deployment.

3.3 Test Success & Failure Criteria

A clearly defined criteria is required, with only two possible outcomes (usually in the form of 'expected' and 'everything-else/otherwise').

3.4 Test Deliverables

Unit tests do not mandate any deliverables.

3.5 Test Items

3.5.1 TU_SH/DpSys_FlashBlink_Op

Test Identifier	TU_SH/DpSys_FlashBlink_Op (rev. A)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	DuinoPro System
Details	
Purpose	Confirm system can be flashed with developer's machine
Test Entry Criteria	Before development begins on DuinoPro
Test Exit Criteria	Upon success or re-allocation of resources to more important issues
Required	• DuinoPro built with 96630eb962be26b19339f49d5dc076570b97c6a8 • No connections to DuinoPro • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Flash with simple Blink Test
Procedure	1. Modify the DuinoPro build such that; 2. Build the DuinoPro
Risks	
Outputs	
Test Success	The on-board LED should periodically blink
Test Failure	No blinking is observed
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/02	Jamie Phan	- DP built with 96630eb962be26b19339f49d5dc076570b97c6a8 -	Success

3.5.2 TU_SH/DnSys_HwareCheck_Op

Test Identifier	TU_SH/DnSys_HwareCheck_Op (Rev. A)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	Dusty Module
Details	
Purpose	Check for presence of Board Support Parameter (BSP) that configures external RAM size, Data bus verification of external RAM, Address bus verification of external RAM
Test Entry Criteria	Before development on the Dusty Module
Test Exit Criteria	Upon success or re-allocation of resources to more important issues
Required	• Dusty Module • Hwcheck utility (rev. 1.0.0.3) • ESP utility (rev. 1.1.1.6) • DC9004A and DC9006A • Personal computer • Serial Terminal Program (PuTTy)
Methodology	
Test Method	Manual
Circumstances / Case of Test	Using HwareCheck Flash image
Procedure	1. Connect the Dusty module to a computer using the DC9004A and DC9006A 2. Access the Dusty module via CLI with PuTTy 3. Extract the existing flash image on the Dusty as backup using the ESP utility 4. Erase the existing flash image from the Dusty using the ESP utility 5. Program the hwcheck utility onto the Dusty module
Risks	Loss of Dusty flash image
Outputs	
Test Success	Periodic output on CLI should show tests running and passing
Test Failure	
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A				

3.5.3 TU_SH/DpUart_external_op

Test Identifier	TU_SH/DpUart_External_Op (rev. A)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	DuinoPro API UART (DpUart)
Details	
Purpose	To confirm UART communication on DP is functional
Test Entry Criteria	During development of the DuinoPro UART C-Library
Test Exit Criteria	Upon success or re-allocation of resources to more important issues

Required	• DuinoPro built with 5e52ff2c4de8661a5b1c87fdea9bd76edf5f616e
	  ○ All T-Ctrl flags set to zero except DPUART control.
	• DP-External connection (GND, TX and RX)
	  ○ Use Oscilloscope probe to TX (with GND tie)

Methodology	
Test Method	Manual
Circumstances / Case of Test	External MCU connected via UART

Procedure	1. Modify the test control macros in the 'tester' module such that only DPUart is set.
	2. Build the DuinoPro
	3. Prepare the Oscilloscope for triggering
	4. Connect the DuinoPro (DP) to the External device (Ext) as follows;
	  a. DP/M7.Gnd@2 → Ext/GND
	  b. DP/M7.GPIO5.USART_TX → Ext/Probe
	  c. DP/M7.GPIO4.USART_RX → Ext/Nonte
	5. Observe the output on the Oscilloscope

Risks	
Outputs	
Test Success	The oscilloscope correctly shows 9-bits of data received to be as sent from the DP (character '1', including start bit)
Test Failure	The oscilloscope does not detect/cannot read the data
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/02	Jamie Phan	- DP built with 3045482f237726a0706b3a5400001c2a9db3475a - Used external Arduino UNO (see resources)	Failure
B	2017/10/03	Jamie Phan	- DP built with 5e52ff2c4de8661a5b1c87fdea9bd76edf5f616e	Success

3.5.4 TU_SH/DnJoin_Cli_Op

Test Identifier	TU_SH/DnJoin_Cli_Op (rev. B)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	Dusty-to-DuinoPro API UART (DnUart) interface
Details	
Purpose	To confirm UART communication between the DuinoPro and Dusty module is operational
Test Entry Criteria	During development of the DuinoPro UART C-Library
Test Exit Criteria	Upon success or re-allocation of resources to more important issues
Required	• DuinoPro built with 43fb1b2d9ad0fa1aa02d99f59d0894a8ad75610a • Dusty Module flashed with IPM_A01 • Dp-Dn Rev. 0.0.1 • DC9004A and DC9006A devices • Serial Terminal Program (PuTTY) • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Dusty module under CLI debugging mode
Procedure	1. Modify the DuinoPro build such that the join in the DN Join Test Case has a unique Network Id (e.g. ‘1300’) 2. Build the DuinoPro 3. Assert the Dusty Flash (IPM_A01) loads a default Network Id (1299) (Use CLI command ‘info’) 4. Connect the DuinoPro to the Dusty and run the test build 5. Record via CLI (use CLI command ‘info’) the Dusty module’s Network Id after communication with the DuinoPro
Risks	
Outputs	
Test Success	The CLI reports that the Network Id changed
Test Failure	The CLI reports the default Network Id (1299)
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/09/30	Jamie Phan	- DP built with 43fb1b2d9ad0fa1aa02d99f59d0894a8ad75610a - Dp-Dn Rev. 0.0.1 - Dusty flashed with IPM_A01	Failure
B	2017/10/05	Jamie Phan	- DP built with f00317305189b1868c581c0565f45b23eea80d6c - Dp-Dn Connection Rev. 0.0.1 - Dusty flashed with IPM_A02_Fuse	Failure

3.5.5 TU_SH/DpFrameHdr_PayloadHdr_Op

Test Identifier	TU_SH/DpFrameHdr_PayloadHdr_Op (Rev. A)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO pack_field_header function
Details	
Purpose	To confirm that the pack_field_header function successfully constructs the correct field header for a given field type and stores this at the correct location in the payload. Input validation is also tested.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit bd0e0cb43e40550c64e8275143bc97f5ce102e9d • Personal Computer or duinoPRO
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer (requires modification of pass/fail notification functions in TestCase class) or push built application to duinoPRO.
Procedure	Using personal computer: 1. Comment out dp_payload_flush and #include "../globals.h" from frame.cpp 2. Modify TestDpFraming.cpp: remove all t_pass() and t_fail() calls and replace these will corresponding printf statements. 3. Write brief C++ main file (with main function) that calls test cases in TestDpFraming.cpp. 4. Compile the C++ main file, linking the appropriate source files (i.e. frame.cpp) 5. Run the executable and check console output for success/failure. Using duinoPRO: 1. Build code in Atmel Studio 7 2. Port code to duinoPRO 3. Examine LED(s) for success/failure
Risks	
Outputs	
Test Success	pack_payload_header returns FAILURE for invalid inputs and generates expected outputs for valid inputs.
Test Failure	Does not return FAILURE with invalid inputs or generates unexpected outputs for valid inputs.
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/16	Aaron Hurst	Test completed with personal computer using above methodology	Success

3.5.6 TU_SH/DpReserveField_Payload_Op

Test Identifier	TU_SH/DpReserveField_Payload_Op (Rev. A)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO reserve_field function
Details	
Purpose	To confirm that the reserve_field function successfully reserves the correct amount of space in the payload by incrementing the payload pointer and successfully calls pack_field_header to store the correct payload header.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit bd0e0cb43e40550c64e8275143bc97f5ce102e9d • Personal Computer or duinoPRO
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer (requires modification of pass/fail notification functions in TestCase class) or push built application to duinoPRO.
Procedure	Using personal computer: 1. Comment out dp_payload_flush and #include "../globals.h" from frame.cpp 2. Modify TestDpFraming.cpp: remove all t_pass() and t_fail() calls and replace these will corresponding printf statements. 3. Write brief C++ main file (with main function) that calls test cases in TestDpFraming.cpp. 4. Compile the C++ main file, linking the appropriate source files (i.e. frame.cpp) 5. Run the executable and check console output for success/failure. Using duinoPRO: 1. Build code in Atmel Studio 7 2. Port code to duinoPRO 3. Examine LED(s) for success/failure
Risks	
Outputs	
Test Success	reserve_field returns FAILURE for invalid inputs and generates expected outputs for valid inputs.
Test Failure	Does not return FAILURE with invalid inputs or generates unexpected outputs for valid inputs.
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/16	Aaron Hurst	Test completed with personal computer using above methodology	Success

3.5.7 TU_SH/DpSample_SampleTime_Op

Test Identifier	TU_SH/DpSample_SampleTime_Op (Rev. C)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO sample_time function
Details	
Purpose	To confirm that the sample_time function successfully calls reserve_field and stores a correct (mock) timestamp in the payload.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer
Procedure	1. Checkout commit DustyDuinoPro commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 on personal computer. 2. Compile sample_offline_test.cpp using the command specified on line five of that file. 3. Run the executable test.exe. 4. View command line output to verify success/failure.
Risks	
Outputs	
Test Success	Expected payload entries are observed
Test Failure	Expected payload entries are not observed
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Timestamp values stored at incorrect positions in payload.	Failure
B	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Timestamp entry stored in reverse order (LSB first).	Failure
C	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology	Success

3.5.8 TU_SH/DpSample_SampleDiag_Op

Test Identifier	TU_SH/DpSample_SampleDiag_Op (Rev. B)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO sample_diagnostic function
Details	
Purpose	To confirm that the sample_diagnostic function successfully calls reserve_field and stores correct (mock) diagnostic data (battery voltage) in the payload.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer
Procedure	1. Checkout commit DustyDuinoPro commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 on personal computer. 2. Compile sample_offline_test.cpp using the command specified on line five of that file. 3. Run the executable test.exe. 4. View command line output to verify success/failure.
Risks	
Outputs	
Test Success	Expected payload entries are observed
Test Failure	Expected payload entries are not observed
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Battery voltage entry stored in reverse order (LSB first).	Failure
B	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology	Success

3.5.9 TU_SH/DpSample_SampleSens_Op

Test Identifier	TU_SH/DpSample_SampleSens_Op (Rev. D)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO sample_sensor function
Details	
Purpose	To confirm that the sample_sensor function successfully calls reserve_field and stores correct (mock) sensor data in the payload.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer
Procedure	1. Checkout commit DustyDuinoPro commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 on personal computer. 2. Compile sample_offline_test.cpp using the command specified on line five of that file. 3. Run the executable test.exe. 4. View command line output to verify success/failure.
Risks	
Outputs	
Test Success	Expected payload entries are observed
Test Failure	Expected payload entries are not observed
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Error: sizeof does not return correct size of a buffer within a function to which the buffer has been passed (i.e. within sensor_read).	Failure
B	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Sensor data stored in wrong location within payload.	Failure
C	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. 16-bit sensor measurements stored in correct order, but the two bytes for each measurement are stored in the wrong order (LSB first).	Failure
D	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology	Success

3.5.10 TU_NM/WSNHandshake_APIConnNoSerialMux_Op

Test Identifier	TU_NM/WSNHandshake_APIConnNoSerialMux_Op (Rev. A)
Test Type	Unit
Sub-System(s) Involved	NM
Type of Works	Unit Functionality
Works Under Test	Python Serial connection object
Details	
Purpose	Confirm the Python application can connect to the Network Manager
Test Entry Criteria	During development/post install
Test Exit Criteria	Upon success or re-allocation of resources to more important issues
Required	• DustyPyManager with commit 47b40e37bbff42f88f1b76ab276c6dd6a6b89834 • DC2274A-A • Personal Computer • Python 2.7 Environment
Methodology	
Test Method	Software
Circumstances / Case of Test	API connection to Network Manager with no Serial Mux in place
Procedure	1. Run the Python unit-test with the DC2274A-A connected
Risks	
Outputs	
Test Success	Retrieval of Network Manager status information and no exceptions raised
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/10	Jamie Phan	- DustyPyManager built with 47b40e37bbff42f88f1b76ab276c6dd6a6b89834	Success

3.5.11 TU_DB/Put_Exclusive_Op

Test Identifier	TU_DB/Put_Exclusive_Op
Test Type	Unit
Sub-System(s) Involved	DB
Type of Works	Unit Functionality
Works Under Test	Successful Python Data Puts
Details	
Purpose	Confirm the Python application can successfully upload data into DynamoDB
Test Entry Criteria	During development
Test Exit Criteria	Upon multiple successful uploads
Required	• Personal Computer • Python 3 Environment • Internet Access • Access to AWS Identity & Access Management • AWS's Python API (Boto3)
Methodology	
Test Method	Software
Circumstances / Case of Test	Configure the AWS Command Line Interface to access the IAM credentials and ensure that the personal computer has internet access.
Procedure	1. Run the tailored Python DataInsertion.py script in conjunction with the Python Put.py script
Risks	
Outputs	
Test Success	Successful acknowledgement of Data Puts
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/10	Andy Ta	- Executed 10 Data Put Tests using mock data	Success

3.5.12 TU_DB/Get_Exclusive_Op

Test Identifier	TU_DB/Put_Exclusive_Op
Test Type	Unit
Sub-System(s) Involved	DB
Type of Works	Unit Functionality
Works Under Test	Successful Python Data Gets
Details	
Purpose	Confirm the Python application can successfully query data into DynamoDB
Test Entry Criteria	During development
Test Exit Criteria	Upon multiple successful queries
Required	• Personal Computer • Python 3 Environment • Internet Access • Access to AWS Identity & Access Management
Methodology	
Test Method	Software
Circumstances / Case of Test	Configure the AWS Command Line Interface to access the IAM credentials and ensure that the personal computer has internet access.
Procedure	1. Run the tailored Python Query.py script and cross-reference it with AWS DynamoDB Console
Risks	
Outputs	
Test Success	Successful acknowledgement of Data Gets
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/10	Andy Ta	- Executed 10 Data GetTests using mock data	Success

3.5.13 TU_CI/DataTransfer_S3_Op

Test Identifier	TU_DB/Put_Exclusive_Op
Test Type	Unit
Sub-System(s) Involved	DB
Type of Works	Unit Functionality
Works Under Test	Successful Data Transfer from DynamoDB to S3
Details	
Purpose	Confirm the Python application can successfully transfer data from DynamoDB to S3 in a comma separated variable format
Test Entry Criteria	During development
Test Exit Criteria	Upon multiple successful csv transfers
Required	• Personal Computer • Python 3 Environment • Internet Access • Access to AWS Identity & Access Management
Methodology	
Test Method	Software
Circumstances / Case of Test	Configure the AWS Command Line Interface to access the IAM credentials and ensure that the personal computer has internet access.
Procedure	1. Run the tailored Python NetworkQuery.py script
Risks	
Outputs	
Test Success	Successful acknowledgement of Data Transfers
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/10	Andy Ta	- Executed 10 Data Transfers using mock data	Success

3.6.14 TU_SH/DnDp_DUT_Nop

Test Identifier	TU_SH/DnDp_DUT_Nop
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	Successful bare-board testing of Dusty Interposer PCB
Details	
Purpose	Ensure the PCB of Dusty module is functioning properly.
Test Entry Criteria	During development
Test Exit Criteria	Upon successful of PCB fabrication
Required	• Personal Computer • KICAD Environment • Voltmeter • Test Jig (Dummy Board with LED)
Methodology	
Test Method	Software and Hardware
Circumstances / Case of Test	Ensure the connection of the breakout is properly connect to power source.
Procedure	1. Check to make sure that no current flow between separate nets by measuring the amount resistance between them. 2. Check to make sure there is current flow from one 'node' to the next for every net on the board.
Risks	PCB may be damaged if not carefully handle
Outputs	
Test Success	Successful PCB fabrication and 100% Net List Test
Test Failure	Otherwise
Deliverables	None

Rev.	**Date**	**Tester**	**Description**	**Outcome**
A				

3.6.15 TU_SH/DnDp_FT _Nop

Test Identifier	TU_SH/DnDp_FT_Nop
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	Successful functional test of Dusty Interposer PCB when electrical component is placed.
Details	
Purpose	Ensure the PCB of Dusty module to duinoPRO is functioning properly when interface with duinoPRO
Test Entry Criteria	During development
Test Exit Criteria	Upon successful of Dusty Interposer PCB module to duinoPRO
Required	• Personal Computer • Arduino IDE or Atmel Studio • Jumper wire • PuTTy • Rev0.0.6
Methodology	
Test Method	Software and Hardware
Circumstances / Case of Test	Ensure the connection of the breakout is properly connect to duinoPRO via jumper wire
Procedure	1. Connect the Dusty to duinoPRO 2. Determine the Network ID via CLI after Dusty connects to duinoPRO
Risks	
Outputs	
Test Success	Successful Integration between duinoPRO and Dusty where Network ID changed
Test Failure	Otherwise
Deliverables	None

Rev.	**Date**	**Tester**	**Description**	**Outcome**
A				

3.5.14 TU_WA/Load_Webpage_Op

Test Identifier	TU_WA/Load_Webpage_Op
Test Type	Unit
Sub-System(s) Involved	WA
Type of Works	Unit Functionality
Works Under Test	Successfully Load Webpage
Details	
Purpose	Confirm Webpage can be viewed post-deployment
Test Entry Criteria	During development
Test Exit Criteria	Upon multiple successful page renderings with various internet browsers
Required	• Personal Computer • Python 2 Environment • Internet Access • Google Chrome/Safari/Firefox
Methodology	
Test Method	Software
Circumstances / Case of Test	Access hosted domain *atamo.ap-southeast-1.elasticbeanstalk.com*; ensure that the personal computer has internet access.
Procedure	Access *atamo.ap-southeast-1.elasticbeanstalk.com* using Google Chrome, Safari & Firefox. Ensure page renders successfully
Risks	
Outputs	
Test Success	Successfully loaded web application post-deployment
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/24	Jake Sacino	- Successfully loaded web application post-deployment with above methodology	Success

3.5.15 TU_WA/User_Authenticate_Op

Test Identifier	TU_WA/User_Authenticate_Op
Test Type	Unit
Sub-System(s) Involved	WA
Type of Works	Unit Functionality
Works Under Test	Successfully Authenticate User
Details	
Purpose	Confirm User is able to sign in
Test Entry Criteria	During development
Test Exit Criteria	Upon successful sign in and redirect after providing valid user credentials
Required	• Personal Computer • Python 2 Environment • Internet Access
Methodology	
Test Method	Software
Circumstances / Case of Test	Access hosted domain *atamo.ap-southeast-1.elasticbeanstalk.com*; ensure that the personal computer has internet access; set valid login credentials
Procedure	Access *atamo.ap-southeast-1.elasticbeanstalk.com;* enter valid login credentials; ensure page renders successfully
Risks	
Outputs	
Test Success	Successfully authenticated user and redirected to web application
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/24	Jake Sacino	- Successfully authenticated user and redirected to web application with above methodology	Success

3.6 Test Risk & Issues

As units are the smallest divisible unit of functionality, risk impact is generally low – however as it is the first suite of tests to be performed, the likelihood of test error is also significantly high.

Risk Description	Likelihood	Impact	Risk Level	Mitigation Strategy
Damage to				

4 Integration Testing

Integration testing is an essential step in the development and deployment of a system. During development, ensures that individual modules or unit in a complex system are compatible with each other and that the interface between these two modules functions as required.

4.1 Test Entry & Exit Criteria

The integration-testing phase begins with the development a test program to establish communication between two modules. After various iterations of unit testing integration testing may be completed to ensure that the updates in the design of various units remains compatible with the modules it connects to.

4.1.1 Dependencies

Integration testing by its very nature tests the compatibility of various units within a larger system. The dependencies of the integration testy come from the interface between the unit and the devices it must connect to. These devices may provide inputs used by the unit or may receive the data produced by the unit.

4.1.2 Suspension Criteria

Unit tests will be suspended under the following conditions:

1. The test in question has a risk level of 'Moderate' or higher
2. The test in question poses a hazard to any persons operating the tests, or are nearby the testing site.
3. The test in question poses a risk of damage to any crucial pieces of equipment as described in section 2.2

4.2 Test Approach

Integration tests will be assessed with a two-level scale – pass or failure. Each test will test the interface between a minimum number of two units and will ensure that unit continue to function correctly when sending and/or receiving across this interface. Failure to pass the test will then require:

- Re-development efforts in the design of one or more of the units to ensure the interface will pass the test.
- Identification of failure and removal of defective item during deployment.

4.3 Test Success & Failure Criteria

A clearly defined criteria is required, with only two possible outcomes (usually in the form of 'expected' and 'everything-else/otherwise').

4.4 Test Deliverables

Integration tests do not mandate any deliverables.

4.5 Test Items

4.5.1 TI_SH-NM/DnJoin_Mode4_Op

Test Identifier	TI_SH-NM/DnJoin_Mode4_Op (Rev. A)
Test Type	Integration
Sub-System(s) Involved	SH, NM
Type of Works	Interface between sub-systems
Works Under Test	Dusty to Network Manager
Details	
Purpose	Confirm that the Dusty module can connect to a network
Test Entry Criteria	After development of the Dusty API UART
Test Exit Criteria	Upon success or re-allocation of resources to more important issues
Required	• DuinoPro built with 43fb1b2d9ad0fa1aa02d99f59d0894a8ad75610a • Dusty Module flashed with IPM_A01 • Dp-Dn Rev. 0.0.1 • DC9004A and DC9006A devices • Serial Terminal Program (PuTTY) • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Dusty module under CLI debugging mode
Procedure	1. Modify the DuinoPro build such that the join in the DN Join Test Case has a unique Network Id (e.g. ‘1300’) 2. Build the DuinoPro 3. Assert the Dusty Flash (IPM_A01) loads a default Network Id (1299) (Use CLI command ‘info’) 4. Connect the DuinoPro to the Dusty and run the test build 5. Record via CLI (use CLI command ‘info’) the Dusty module’s Network Id after communication with the DuinoPro
Risks	
Outputs	
Test Success	The sensor-host successfully joins the Network
Test Failure	The CLI reports the default Network Id (1299)
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/05	Jamie Phan	- DP built with f00317305189b1868c581c0565f45b23eea80d6c - Dp-Dn Connection Rev. 0.0.1 - Dusty flashed with IPM_A02_Fuse	Success

4.5.2 TI_SH/DpSample_SampleMain_Op

Test Identifier	TI_SH/DpSample_SampleMain_Op (Rev. D)
Test Type	Unit
Sub-System(s) Involved	SH
Type of Works	Unit Functionality
Works Under Test	duinoPRO sample_sensor function
Details	
Purpose	To confirm that the sample function calls the correct sample_* function(s) for a given input, generated the correct payload and returns failure if given invalid inputs.
Test Entry Criteria	During development of duinoPRO application layer
Test Exit Criteria	Upon successful construction of field headers of each possible type
Required	• duinoPRO application software from commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 • Personal Computer
Methodology	
Test Method	Manual
Circumstances / Case of Test	Build test code locally with personal computer
Procedure	1. Checkout commit DustyDuinoPro commit 2fa2fb22c5398eaa4193c7e483eb518f5783b451 on personal computer. 2. Compile sample_offline_test.cpp using the command specified on line five of that file. 3. Run the executable test.exe. 4. View command line output to verify success/failure.
Risks	
Outputs	
Test Success	Expected payload entries are observed
Test Failure	Expected payload entries are not observed
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology.	Failure
B	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Timestamp field header not in payload, timestamp value at wrong position.	Failure
C	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology. Diagnostic data output expected values incorrect.	Failure
D	2017/10/19	Aaron Hurst	Test completed with personal computer using above methodology	Success

4.5.3 TI_DB-WA/Connection_Mock_Op

Test Identifier	TI_DB-WA/Connection_Op
Test Type	Unit
Sub-System(s) Involved	DB
Type of Works	Integration Functionality
Works Under Test	Successful Web Application Data Gets
Details	
Purpose	Confirm the Web Application can successfully query data into DynamoDB
Test Entry Criteria	During development
Test Exit Criteria	Upon multiple successful queries
Required	• Personal Computer • Python 3 Environment • Internet Access • Access to AWS Identity & Access Management • ElasticBeanStalk • Plot.ly APIs
Methodology	
Test Method	Software
Circumstances / Case of Test	Configure the AWS Command Line Interface to access the IAM credentials and ensure that the personal computer has internet access.
Procedure	1. Run the dataAutomation.py script to automatically generate random data points every second 2. Authenticate the database and the web application 3. Compare the data writes against the reads from the web application
Risks	
Outputs	
Test Success	Successful acknowledgement of Data Gets
Test Failure	Otherwise
Deliverables	None

Test History				
Rev.	**Date**	**Tester**	**Description**	**Outcome**
A	2017/10/20	Andy Ta, Jake Sacino	Connection was successful and able to query a dynamic database	Success

4.6 Test Risk & Issues

Integration tests are at one level higher in terms of functionality and involve the interconnection between different modules. As a result, there is a higher risk of damage to equipment. Most of integration tests however have been software based with low likelihood for damage to equipment or injury to human operators.

Risk Description	Likelihood	Impact	Risk Level	Mitigation Strategy
Damage to				

5 References

[1] G. Levine. (2017). *Arduino* [Image]. Available: https://visualhunt.com/photo/126201/.].

[2] M. Yen, "Wireless sensor network monitors structural data and environmental contaminants," (in English), *Materials Performance*, Article vol. 47, no. 3, pp. 19-20, Mar 2008. Available: <Go to ISI>://WOS:000253665300005

[3] Y. Pang, G. Lodewijks, and Bindt, "Machinery maintenance prediction using pattern recognition of condition monitoring data," (in English), *8th International Conference on Condition Monitoring and Machinery Failure Prevention Technologies 2011, Vols 1 and 2*, Proceedings Paper pp. 772-781, 2011. Available: <Go to ISI>://WOS:000399622600078

[4] K. Ellis, S. R. Mounce, B. Ryan, M. R. Templeton, and C. A. Biggs, "Use of on-line water quality monitoring data to predict bacteriological failures," (in English), *12th International Conference on Computing and Control for the Water Industry, Ccwi2013*, Proceedings Paper vol. 70, pp. 612-621, 2014. doi:10.1016/j.proeng.2014.02.067. Available: <Go to ISI>://WOS:000341500600067