MODULAR DAMAGE DETECTION FOR EXPANDABLE AND INFLATABLE STRUCTURES

Mark E. Lewis[(1)], Tracy L. Gibson[(2)], Pedro J. Medelius[(3)]

[(1)]NASA John F. Kennedy Space Center, NE-L6-B, Kennedy Space Center, FL 32899 USA,
Email: Mark.E.Lewis@nasa.gov
[(2)]Southeastern Universities Research Association, LASSO-008, Kennedy Space Center, FL 32899 USA,
Email: Tracy.L.Gibson@nasa.gov
[(3)]ASRC Federal Space and Defense, 150 Cocoa Isles Blvd, Suite 401, Cocoa Beach, FL 32931,
Email: Pedro.Medelius@asrcfederal.com

ABSTRACT

NASA has identified potential damage from micrometeoroid and orbital debris (MMOD) impacts as a primary threat to Commercial Crew Program vehicles. The International Space Station (ISS) and extraterrestrial habitats also exhibit the risk of damage caused by MMODs. Currently no integrated in-situ or real-time health monitoring damage detection system is being used for expandable and inflatable structures. A novel, modular damage detection system design that incorporates interchangeable and replaceable sensory panels in a foldable architecture is described. The design implements technologies that provide for situational awareness, self-configuration, and damage detection and localization. The system is applicable for the new Gateway and surface and ground support infrastructure.

1. INTRODUCTION

Millions of naturally occurring objects (micrometeoroids) and made-made debris can be encountered during a space mission. For decades, NASA and other agencies have tracked orbital debris but the tracking is limited to objects that are typically larger than 3 mm (in low-Earth orbit) [1]. During the Space Shuttle era, all the orbiters received damage from MMOD and MMOD was designated as the third greatest risk to losing an orbiter during a mission; launch and re-entry were the two highest risks [2]. Damage was observed to many orbiter windows during the first 63 Space Shuttle missions. A total of 177 debris impacts were identified on the Shuttleôs exterior windows with pitting significant enough to require replacing the windows [3]. Additionally, more than 70 Space Shuttle windows had to be replaced because of significant debris impacts during the first 88 missions [4]. In July 2014, MMOD impact caused damage to a panel on the Potential P4 Photovoltaic Radiator (PVR) on the International Space Station (ISS) [5]. During the Apollo program, debris damage was also observed due to landing operations. Two coupons from the Surveyor III spacecraft that were facing the Apollo 12 landing site were analyzed and an average of 103 pits/cm^2 was observed. The data indicated that the spacecraft was not exposed to the direct spray of the landing Lunar Module but was exposed to the fringes of the ejecta plume. It was speculated that if the spacecraft had been exposed to the direct spray, the damage would have been orders of magnitude higher [6].

NASA has identified a need for damage detection and health monitoring technologies in multiple NASA technology roadmaps. Technology gaps or needs for Space exploration include such areas as integrated health monitoring for space debris impacts [7], on government or commercial crew space vehicles [2], habitation structures, and expandable, inflatable or deployable structures.

In 2011, the Flagship Technology Demonstration program expressed an interest in the need to detect early damage to habitation structures from risks such as MMOD impacts for the planned Inflatable Hab (iHab) demonstration [8]. NASA Kennedy Space Center (KSC) initiated development of technologies for the early detection of MMOD damage to habitation structures. The primary technology investigated was a damage detection system based on sensing the electrical integrity of parallel conductive traces. When damage occurred, traces were broken and information was provided to the monitoring system. Multiple configurations were investigated, including those containing several sensing layers, where alternate layers were arranged orthogonally with respect to adjacent layers. The technology was demonstrated in the NASA Habitat Demonstration Unit (HDU) Deep Space Analog platforms and via a secure network for remote sensing using single and multi-panel approaches.

Currently, KSC is investigating the use of flexible materials and modular designs for use in spacecraft, smart, wearable fabrics, solar arrays, military shelters,

and ground infrastructure. The modular designs use embedded algorithms for situational awareness, self-configuration, and damage detection and localization. Each panel is aware of its spatial relationship to the other panels. Additionally, each panel is identical in hardware and software, which greatly enhances the modularity and tailorability of the system. The current version of the Modular Damage Detection System (MDDS) is focused on detecting damage to infrastructure associated with ground systems that support launch operations such as composite cryogenic storage tanks.

2. SENSOR SYSTEM DESIGN

The MDDS expands on the previously demonstrated and NASA-patented Flat Surface Damage Detection System (FSDDS) and the Flexible Damage Detection System (FLEX-DDS) capabilities and technologies [9]. The sensory system is an intelligent damage detection system that could be embedded into or added to structures, providing a lightweight in-situ health monitoring capability for space or aircraft vehicles, and for expandable, inflatable, and deployable structures and other applications.

The MDDS is composed of three main systems: 1) the Sensory Panel; 2) the embedded software for situational awareness and damage detection; and 3) the mobile device with a graphical user interface (GUI) that is used to operate and monitor the Sensory Panels wirelessly. The architecture is flexible and expandable, supporting one Sensory Panel or many panels organized two-dimensionally in a grid pattern. A notional block diagram with four Sensory Panels is shown in *Figure 1*.

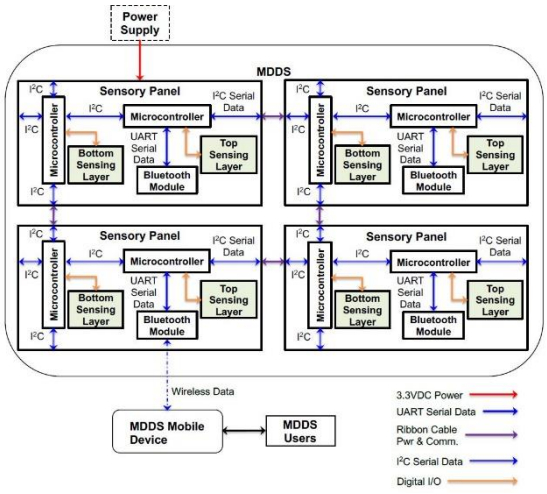

Figure 1. Notional MDDS Architecture Block Diagram with Four Sensory Panels

The Sensory Panels consist of two microcontrollers with embedded software, two Sensing Layers, and a Bluetooth low-energy (BLE) module for wireless communication. The Sensory Panels are interchangeable and operate independently, scanning for damage periodically and waiting to be connected to a Bluetooth-enabled device. The Sensory Panelôs embedded software, running on the microcontrollers, executes algorithms for situational awareness and damage detection, and reports the results upon request.

The MDDS GUI runs on a mobile device and provides users with the ability to configure, command, and monitor the Sensory Panels in the system. The GUI allows users to set Sensory Panel attributes and the panel scanning rate of the application. It also provides the ability to scan, discover, and connect to the Sensory Panels wirelessly via Bluetooth. The active Sensory Panels are displayed on the main view and detailed damage information is provided in graphical and tabular format.

The intent for this prototype system is to be operated as a technology demonstration. A variety of configurations will be tested with at least four intelligent, interchangeable, and configurable Sensory Panels connected at any one time. It will demonstrate the Sensory Panelôs modularity and situational awareness capabilities.

3. SENSORY PANEL

The Sensory Panels are designed to be identical and interchangeable. Each Sensory Panel consists of a custom printed circuit board (PCB) with two 32-bit microcontrollers with embedded software, two Sensing Layers, and a BLE module for wireless communication (*Figure 2*). Each Sensory Panel also includes four Molex connectors that can be used for power distribution and panel-to-panel communication if multiple Sensory Panels are utilized. The Molex connectors are oriented to allow panels to be connected to adjacent panels from any of the four sides of the PCB.

Sensory Panel communication is accomplished using two primary serial peripheral interfaces. The Universal Asynchronous Receiver/Transmitter (UART) of the microcontroller is used to configure the BLE module (RN4020). Once the module is configured, a Bluetooth-enabled device is paired to it, and a Microchip Low-energy Data Profile (MLDP) service is open, the UART is used to transmit serial data wirelessly to a mobile device. The MLDP is a private BLE service that allows

serial data (up to 50 kbps) to be transported over Bluetooth. It basically turns the interface into a simple UART, bridging the UART serial data from the microcontroller. The I^2C serial peripheral interfaces are used for panel-to-panel communication. It is a serial protocol for low-speed, two-wire communication.

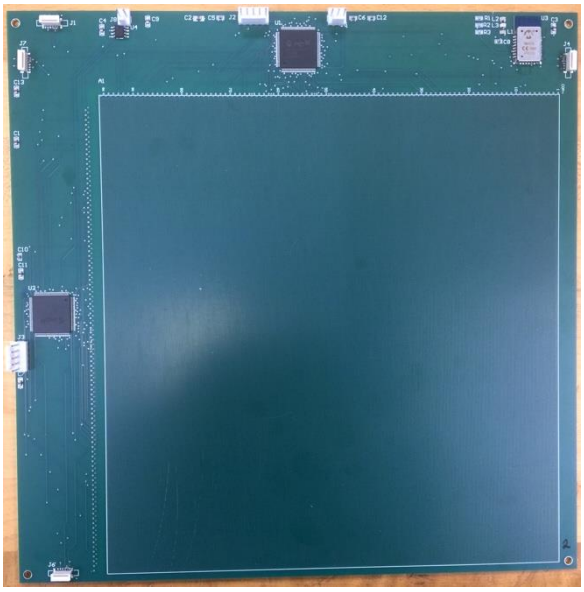

Figure 2. MDDS PCB

The Sensory Panels have two Sensing Layers, each consisting of 96 conductive traces that run parallel to each other. The top Sensing Layer has traces that span vertically 7.67 inches with a trace-to-trace spacing of approximately 50 mils. The bottom Sensing Layer is identical except it is oriented orthogonal to the top layer, thereby creating a two-dimensional grid pattern. Damage is detected when one or more conductive traces are broken.

The Sensory Panels require 3.3VDC input voltage and consume approximately 450mW of power. Additional Sensory Panel attributes include PCB dimensions (9.5 x 9.5 x 0.062 in. (24.13 x 24.13 x 0.16 cm) (W x L x D)) and Sensing Layer dimensions (7.67 x 7.67 in. (19.48 x 19.48 cm) (W x L)).

4. EMBEDDED SOFTWARE

Stored in the non-volatile memory of the Sensory Panels microcontrollers is an embedded program that is responsible for the configuration and operation of each Sensory Panel. The embedded software initializes and configures the serial peripheral interfaces and the BLE module. It also processes and responds to commands sent from the mobile device and reports Sensory Panel health information upon request. In addition, the embedded program executes algorithms that provide three main MDDS functions: 1) generate an active panel map; 2) determine potential Master Panels (MPs) and set the MP once a connection has been made with the GUI; and 3) monitor the health status of all active panels. *Figure 3* provides an overview of the embedded software.

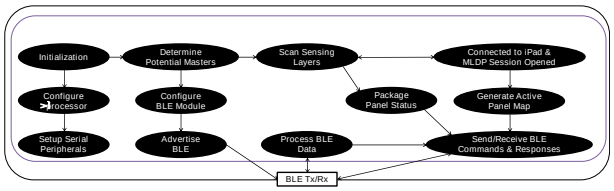

Figure 3. Sensory Panel Embedded Software Logical and Structural Overview

The MDDS active panel mapping algorithm scans potentially adjacent panels to determine which panels are active. The algorithm within each individual active panel then performs a scan to determine which panels could be the MP. This is accomplished by each panel checking if an active panel is present to its left and/or below it using the I^2C serial interfaces. If there are no active panels to the left or below the panel, then it is considered a potential MP and the BLE module is configured as such. Since the configuration of the system is arbitrary, the MDDS can have one or more potential MPs. Using the GUI, a user will arbitrarily connect to one of them and assign it the role of MP. All the other potential MPs become normal, active panels. The newly-assigned MP starts a progressive scan by communicating with adjacent panels to first determine its closest neighboring panels. Next, the MP requests each of its adjacent neighbors to report the status of their respective neighbors. This operation continues until no new panels are found. The MP keeps a record of the configuration of the MDDS and the path to follow to access any active panel within the system. Upon request from the GUI, the MP transmits back the coordinates of the active panels in the MDDS.

The health of each Sensory Panel is continuously monitored by the embedded code. When the GUI requests a health status of the MDDS, the MP starts by requesting the status from each of the active panels in the system, calculates the actual location of any faults, and reports the location of the broken conductive traces to the GUI. When damage occurs to the top Sensing Layer only, the determination of the exact location of the damage is not possible, thus the system reports the damage as having occurred along the corresponding trace(s). When the damage is detected on both Sensing Layers, the damage location is determined by: 1) a single point when only one trace on each layer is broken, 2) a rectangle,

when multiple traces are broken in at least one layer. The rectangle is bound by the number of traces affected in each of the corresponding Sensing Layers. A relative time stamp is associated with each damage event to establish the proper order of events. This helps organize and identify the location of damage if subsequent damages occur at a later time on the same panel.

If a Sensory Panel which was previously active were to become unresponsive, the embedded software autonomously starts a new re-configuration scan to establish a new set of active panels. Since multiple paths are possible to reach a panel, the failure of any one panel will not necessarily prevent panels further away from the MP from being accessed and monitored using an alternate communication/power path.

5. GUI

The MDDS application (app) is a custom-developed program that runs on a 3rd generation iPad Pro Wi-Fi model with iOS 12. The app was written in Swift 4.2, an open-source programming language developed by Apple Inc., using Xcode 10 integrated development environment (IDE) for macOS. The app provides a GUI for users to configure, command, control, monitor, and display the active Sensory Panels in the MDDS. It also provides the ability to scan for BLE advertisements and to connect to BLE peripherals (*Figure 4*).

Figure 4. MDDS Application Home Page

The MDDS app was designed to provide users the ability to interact with the system. Commands are issued by users and sent wirelessly using Bluetooth technology. Command responses and Sensory Panel health statuses are received from the MP and processed and displayed textually and/or graphically depending on the data type. User interaction with the GUI is accomplished using standard tap and touch gestures. Device rotation is also supported. *Figure 5* illustrates the high-level structure and functionality of the application relevant to MDDS operation. Specific details of iOS and Swift functions are not included in the overview.

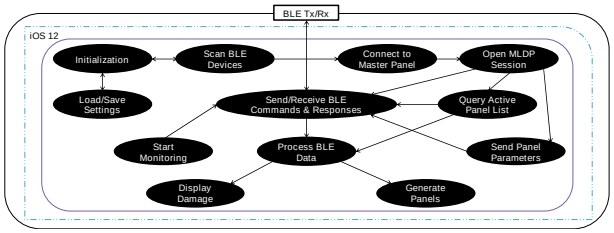

Figure 5. Mobile Device Application Logical and Structural Overview

Swift is a general-purpose programming language that can be used to target the Apple iOS platform. It provides access to libraries, frameworks, classes, methods, and etc. required for iOS app development. The MDDS app uses two such frameworks, *UIKit* and *CoreBluetooth*.

These frameworks are essential for the GUI development. *UIKit* provides the essential infrastructure for the GUI such as window, view, and event handling architectures. The *CoreBluetooth* framework provides the necessary classes to communicate with Bluetooth 4.0 low-energy devices. To accomplish the functionality illustrated in *Figure 5*, numerous classes, methods, and objects were needed. These classes were extended to expand their functionality beyond their default implementation for MDDS use.

The MDDS *MainViewController* class expands the functionality of specific *UIKit* framework classes. The *UIViewController* class is used to provide methods to manage and coordinate all views and controller objects. The *panelButton* class was written to expand the functionality of the *UIButton* class and to customize a Sensory Panel. This class enables tap gesture control for each active panel displayed to open a detailed view that provides additional panel-specific information. The layout of the panels is accomplished using the *UIStackView* class. This class organizes the panels in a grid pattern which represents the physical layout of the Sensory Panels.

The *MainViewController* class also implements methods of the Bluetooth communication protocol. This allows the delegate, the *MainViewController* class, to monitor the discovery, connectivity, and retrieval of BLE peripheral devices. When a Sensory Panel is connected

wirelessly to the iPad, the *MainViewController* begins monitoring the designated MP BLE services and characteristics. When the user opens a MLDP session with the MP, the MLDP private service bridges the panelôs primary microcontroller UART to the iPad. The iPad GUI uses this service to send commands and receive command responses and data from the microcontroller. Data is transported and packaged in a private data service characteristic of the MLDP service.

The *MainViewController* class also provides custom functions to perform other MDDS-related tasks. One of the major tasks is to process BLE data. Data received from the MP is in string format UTF-8 encoded. The data string consists of a header, comma delimited data, followed by a carriage return, and a line feed. The header typically begins with ó$ô followed by an abbreviated version of the echoed command. The data string is parsed into its syntactic components and the data is converted into the proper data type. The panelôs damage information is stored in a two-dimensional array. Other functions include generating the *panelButtons* and updating their graphics with health status indications. When the Active Panel List is received from the MP, a function updates the fill and border colors of the sensory panel graphics to represent the health status received. Panels with no damage are displayed with a green fill color and panels with damage are displayed red.

The *SettingsTableViewController* class implements methods to display, load, and save the MDDS appôs settings. The *SettingsTableViewController* class expands the functionality of the *UITableViewController* and the *UserDefaults* classes. The *UITableViewController* class is used to manage the *TableViewController* for displaying and setting the attributes of the app and the Sensory Panels. To enable the app to retain its settings, the app-ôs defaults database is utilized. The database is accessed using methods from the *UserDefaults* class. The app was designed to store specific Sensory Panel parameters in the defaults database for as long as the app is installed. When the app is initialized, these parameters are loaded and when the user closes the Settings view (*Figure 6*), the parameters are saved to the database.

Finally, displaying each Sensory Panelôs detailed health status is accomplished using three custom classes. These classes configure, manage, and coordinate the views and controller objects. The *PanelViewController* class is responsible for managing and coordinating the Sensory Panel and the Collection views and calculating the damage attributes to be displayed in tabular format. The *SensoryPanelView* class expands the functionality of the *UIView* class to display damage graphically in the *PanelViewController*. The *PanelViewLayout* class expands the role of the *UICollectionViewLayout* class configuring the layout and displaying the damage attributes in tabular format on the *PanelViewController*.

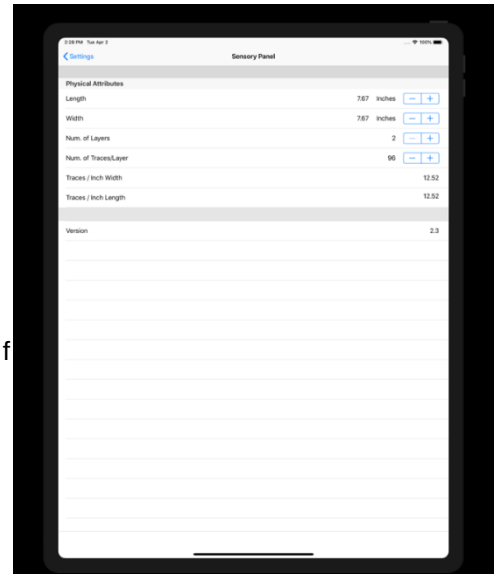

Figure 6. Sensory Panel Settings View

The *SensoryPanelView* class implements methods to display detailed damage information graphically to the user. The *setupPanelLayout* function calculates the viewôs size using the actual defined by the user in the *SettingsTableViewController* class and scales it appropriately to fit within the usable bounds of the *PanelViewController*. Mobile device orientation is also used as a parameter in determining the scaling factor. At this point, the axis range and view transform are calculated. The axis range is calculated by rounding the actual Sensory Panel physical dimensions to the nearest integer. This range is used to draw the x- and y-axes and the horizontal and vertical lines in the view. In order to scale the graphical objects in the viewôs graphics context accurately, an affine transformation matrix is used. This transform scales, rotates, and translates the graphical objects representing damage to the correct locations considering the position of the axis origin in relation to the view origin. The *plotDamage* and the *plotBlindSpots* functions are responsible for graphically drawing damage in the view. Both methods have identical plotting functions, except for the designed shape layer. Damage is drawn in red on the *Damage CAShapeLayer*, indicating impact damage that results in broken traces on both the top and bottom sensing layers of a Sensory Panel. In contrast, blind spots, or damage occurring on the top sensing layer only, is drawn in yellow on a different *CAShapeLayer*. Since a y-

coordinates assigned to this type of damage, it is represented differently to the user. These methods also use the same function to draw the objects in the view. The *rectsDots* function draws rectangular or circular graphical objects based on the points (x-, y-coordinates) passed to the function. If more than one trace on each sensing layer is broken, the function draws a rectangular-shaped object corresponding to the width and height of the reported damage. If only one trace on each sensing layer is broken, the function draws a circle to pinpoint the damage location.

6. TESTING AND DEMONSTRATION

A set of simulated test data was generated to evaluate the performance of the MDDS. The test data set utilized five Sensory Panels laid out in an inverted-T pattern (*Figure 7*) with damage on two of the Sensory Panels. The test data set followed the MDDS data protocol and was loaded onto the MP as part of the embedded software.

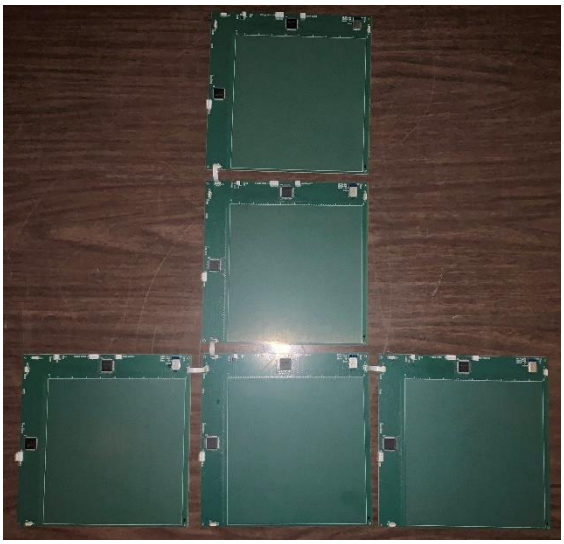

Figure 7. Sensory Panel Layer for Simulated Testing

Testing was initiated by applying power to one Sensory Panel. The Sensory Panel performed various initialization steps and configured itself as a potential MP, at which time the panelôs BLE module broadcasted its name as ñMDDS-MPò. The User then utilized the MP to scan for advertisements from Bluetooth-enabled devices. The User selected the potential MP and connected to it, making it the MP. The User then pressed the Open MLDP Session and Query Active Panel List buttons on the GUI (*Figure 4*). The GUI received information from the MP and displayed the page shown in *Figure 8*. The User then pressed the configuration gear image in the toolbar located in the bottom of the view to open the app settings and the Sensory Panel Settings View (*Figure 6*).

The User verifies the settings and makes any required changes. The User returns to the Home Page and presses the Send Parameters button, which sends the calculated traces/inch parameter. To initiate the GUI to monitor the Sensory Panels health status, the User then pressed the ñStart Monitoringò button. The MP responded with the simulated Sensory Panel health status information as shown in *Figure 9*. Sensory Panels with red backgrounds and borders indicate panels with damage while green-filled panels indicate Sensory Panels without damage.

Figure 8. GUI Showing APL

Figure 9. GUI Showing APL and Sensory Panel Health Status

To display detailed Sensory Panel health information, the User can press anyone of the buttons representing a Sensory Panel. When the User pressed the red