Test Plan for Proof Productions CCB

Prepared By: Barret Vogtman

Version: 1.0

Date: April 22, 2019

Project Team:

Scrum Master: Madeline Presnell

Project Owner: Shafin Siraj

Dev Team: Connor Astemborski

Emily Barnes

Jaxon Terrell

Barret Vogtman

Table of Contents

[1. Introduction 1](#_Toc6930413)

[1.1 Document Purpose 1](#_Toc6930414)

[1.2 Document Organization 1](#_Toc6930415)

[1.3 Project Background 1](#_Toc6930416)

[1.4 Related Documents 1](#_Toc6930417)

[2. Validation Strategy 2](#_Toc6930418)

[2.1 Testing Levels and Objectives 2](#_Toc6930419)

[2.2 System Requirements 2](#_Toc6930420)

[3. System Testing 3](#_Toc6930421)

[3.1 Automated State Testing 3](#_Toc6930422)

[3.2 Automated Stress Testing 3](#_Toc6930423)

[4. Validation Methodology 3](#_Toc6930424)

[4.1 General Information 3](#_Toc6930425)

[4.2 Organization of Validation Procedures 3](#_Toc6930426)

[4.3 Validation Data Files 3](#_Toc6930427)

[4.4 Statistical Methods for Evaluating Results 3](#_Toc6930428)

[4.5 Test Case/Requirements Traceability 3](#_Toc6930429)

[5. Validation Procedures 4](#_Toc6930430)

[5.1 Validation Cases 4](#_Toc6930431)

[5.2 Validation Scripts 4](#_Toc6930432)

[5.2.1 Protocol 1 4](#_Toc6930433)

[5.2.2 Protocol 2 4](#_Toc6930434)

[5.2.3 End User Testing 5](#_Toc6930435)

[5.3 Protocol 1 Test Script 7](#_Toc6930436)

[5.4 Protocol 2 Test Script 11](#_Toc6930437)

[5.5 End User Test Script 15](#_Toc6930438)

# Introduction

## Document Purpose

This Test Plan Document sets out testing procedures to be conducted according to the corresponding Validation Plan.

## Document Organization

The remainder of this document is organized as follows:

* Section 2, Validation Strategy, contains the test levels and objectives, and the hardware required for testing
* Section 3 contains a summary of completed System Testing
* Section 4, Validation Methodology, explains the scope of the testing
* Section 5, Validation Procedures, contains the actual testing procedures.

## Project Background

This document contains the testing procedures for the latest additions to Proof Production’s CCB. This round of development included the addition of the following elements:

* Jog button and joystick
* Deadman switch
* Slider value (Integrated from another dev team)
* CRC-32 Protocol Protection

## Related Documents

|  |  |
| --- | --- |
| **Document** | **Author** |
| Project Requirements Document for Stage and Theatre Automation Systems Phase 4 | Shafin Siraj |
| Validation Plan | Connor Astemborski |

# Validation Strategy

## Testing Levels and Objectives

* + 1. Testing Objectives

This protocol is written to accomplish the following testing objectives:

* CRC-32 protocol works as intended
* Additional buttons function as intended
* Integration has been completed as intended
  + 1. Testing Levels

As there is only one type of end user intended for this system, there is only one level of testing required.

## System Requirements

* + 1. Hardware Configuration

Since access to the Theater Automation System, or TAS, is not available for testing, the testing will occur utilizing the two (2) 1 GB Raspberry Pi, buttons, and laptops supplied by Proof Productions.

* + 1. Software Configuration

The testing will occur on each of the laptops utilizing Microsoft Visual Studio with the Client Side CCB code and Raspberry Pi with the Server Side CCB code installed. The Raspberry Pi will also require Raspberry.System3 and Raspberry.IO.GeneralPurpose3 nuget libraries installed.

* + 1. Software Developer

The software was developed by students of Rowan University over several semesters. Phase IV development was done by the dev team cited on the title page of this document.

* + 1. Source Code

Source Code is available for inspection by authorized employees of Proof Productions, Inc.

# System Testing

## Automated State Testing

Protocol 1 was developed and run to ensure all states would be entered according to a state diagram submitted to the development team by Proof Productions. Protocol 1 was completed and passed testing on April 9, 2019.

## Automated Stress Testing

Protocol 2 was developed and run to ensure all states would be entered according to a state diagram submitted to the development team by Proof Productions. Protocol 2 was completed on April 9, 2019. Protocol 2 was run starting on April 19. Testing was successfully completed on April 22, 2019.

# Validation Methodology

## General Information

As this is Phase IV of the software development, there will be no unit or integration testing performed. Additions are being made to already existing programs making these types of testing infeasible. System and Performance/Stress testing has been performed according to protocols in scripts provided in this document.

## Organization of Validation Procedures

|  |  |
| --- | --- |
| **Test Procedures** | |
| Test Procedure Number | Test Case/Script Included |
| 1 | Protocol 1 – Automated State Testing |
| 2 | Protocol 2 – Automated Stress Testing |
| 3 | End User Testing |

## Validation Data Files

There are no validation data files used in this testing.

## Statistical Methods for Evaluating Results

There are no statistical methods for evaluating results.

## Test Case/Requirements Traceability

See Traceability Matrix.

# Validation Procedures

This section contains the test cases and scripts needed to verify that the requirements are met. A test case describes what is being tested and a test script is the detailed procedures for executing a test case. See the forms that follow this page for the test cases/scripts.

## Validation Cases

## Validation Scripts

### Protocol 1

#### Startup Systems

1. Open terminal window Raspberry Pi
2. Change directory to Downloads/bin/Debug
3. On laptop open ClientSideApplication CCB.cs in Visual Studio
4. In terminal window of pi, run sudo mono ConsoleApp1.exe
5. In Visual Studio, start the CCB.cs code
6. Ensure CRC32 value matches on Raspberry Pi monitor

#### Run Protocol 1

### Protocol 2

#### Startup Systems

1. Open terminal window Raspberry Pi
2. Change directory to Downloads/bin/Debug
3. On laptop open ClientSideApplication CCB.cs in Visual Studio
4. In terminal window of pi, run sudo mono ConsoleApp1.exe
5. In Visual Studio, start the CCB.cs code
6. Ensure CRC32 value matches on Raspberry Pi monitor

#### Run Protocol 2

### 5.2.3 End User Testing

#### Startup Systems

1. Open terminal window Raspberry Pi
2. Change directory to Downloads/bin/Debug
3. On laptop open ClientSideApplication CCB.cs in Visual Studio
4. In terminal window of pi, run sudo mono ConsoleApp1.exe
5. In Visual Studio, start the CCB.cs code
6. Ensure CRC32 value matches on Raspberry Pi monitor

#### Next Button

#### Go Button

#### Deadman Button

#### Go Button

#### Stop Button

#### Previous Button

#### Joystick

#### Jog Button

#### Slider value

## Protocol 1 Test Script

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Step** | **Input or Prerequisites** | **Procedure** | **Expected Results** | **Actual Results** | **Initial & Date/Time** | **Attachment Ref.** | **Pass/Fail** |
| 1 | 1. Prepare Raspberry Pi |  | Open terminal window on Raspberry Pi | Terminal window opens |  |  |  |  |
|  | 2. |  | Change directory to Download/bin/Debug | Directory is changed |  |  |  |  |
| 2 | 1. Prepare laptop |  | Open ClientSideApplication CCB.cs in Visual Studio on Laptop | CCB.cs loads in Visual Studio |  |  |  |  |
| 3 | 1. Start Server side |  | Run sudo mono ConsoleApp1.exe on the Raspberry Pi | Program runs and shows “Waiting for connection…” on screen. |  |  |  |  |
| 4 | 1. Start Client Side |  | Start the CCB.cs code by pressing the play button in Visual Studio | The code starts and the GUI with buttons, arrows, and a slider show on screen |  |  |  |  |
| 5 | 1. Check CRC32 |  | Ensure CRC32 code on Raspberry Pi monitor matches | Both numbers should match |  |  |  |  |
| 6 | 1. Start Simulation |  | Press Protocol 1 Button | Automated test script will begin disabling GUI |  |  |  |  |
|  | 2. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1 and 4 are enabled |  |  |  |  |
|  | 2. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
|  |  |  | Cue timer finishes. | Only Button 4 is enabled. |  |  |  |  |
|  | 3. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
|  | 4. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
|  | 5. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
|  | 6. Stop Button |  | Output in Visual Studio Window says “Stop Button Pressed” | Button 2 shows depressed.  Physical Stop Button flashes.  Button 3and 4 are enabled |  |  |  |  |
|  | 7. Previous Button |  | Output in Visual Studio Window says “Previous Button Pressed” | Button 3 shows depressed.  Physical Previous Button flashes.  Button 1 and 4 are enabled |  |  |  |  |
|  | 8. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
|  | 9. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
|  | 10. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
|  | 11. Previous Button |  | Output in Visual Studio Window says “Previous Button Pressed” | Button 3 shows depressed.  Physical Previous Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
|  | 12. Stop Button |  | Output in Visual Studio Window says “Stop Button Pressed” | Button 2 shows depressed.  Physical Stop Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
|  | 13. |  | Output in Visual Studio Window says “End of Simulation” | GUI is reenabled |  |  |  |  |

## Protocol 2 Test Script

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Step** | **Input or Prerequisites** | **Procedure** | **Expected Results** | **Actual Results** | **Initial & Date/Time** | **Attachment Ref.** | **Pass/Fail** |
| 1 | 1. Prepare Raspberry Pi |  | Open terminal window on Raspberry Pi | Terminal window opens |  |  |  |  |
|  | 2. |  | Change directory to Download/bin/Debug | Directory is changed |  |  |  |  |
| 2 | 1. Prepare laptop |  | Open ClientSideApplication CCB.cs in Visual Studio on Laptop | CCB.cs loads in Visual Studio |  |  |  |  |
| 3 | 1. Start Server side |  | Run sudo mono ConsoleApp1.exe on the Raspberry Pi | Program runs and shows “Waiting for connection…” on screen. |  |  |  |  |
| 4 | 1. Start Client Side |  | Start the CCB.cs code by pressing the play Button in Visual Studio | The code starts and the GUI with Buttons, arrows, and a slider show on screen |  |  |  |  |
| 5 | 1. Check CRC32 |  | Ensure CRC32 code on Raspberry Pi monitor matches | Both numbers should match |  |  |  |  |
| 6 | 1. Start Simulation |  | Press Protocol 2 Button | Automated test script will begin disabling GUI |  |  |  |  |
|  | 2. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1 and 4 are enabled |  |  |  |  |
|  | 3. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
|  |  |  | Cue timer finishes. | Only Button 4 is enabled. |  |  |  |  |
|  | 4. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
|  | 5. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
|  | 6. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
|  | 7. Stop Button |  | Output in Visual Studio Window says “Stop Button Pressed” | Button 2 shows depressed.  Physical Stop Button flashes.  Button 3and 4 are enabled |  |  |  |  |
|  | 8. Previous Button |  | Output in Visual Studio Window says “Previous Button Pressed” | Button 3 shows depressed.  Physical Previous Button flashes.  Button 1 and 4 are enabled |  |  |  |  |
|  | 9. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
|  | 10. Go Button |  | Output in Visual Studio Window says “Go Button Pressed” | Button 1 shows depressed.  Physical Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
|  | 11. Next Button |  | Output in Visual Studio Window says “Next Button Pressed” | Button 4 shows depressed.  Physical Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
|  | 12. Previous Button |  | Output in Visual Studio Window says “Previous Button Pressed” | Button 3 shows depressed.  Physical Previous Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
|  | 13. Stop Button |  | Output in Visual Studio Window says “Stop Button Pressed” | Button 2 shows depressed.  Physical Stop Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
| 7 | 1. Loop |  | Simulation loops back to second Next Button press and continues | All outputs listed as above for the corresponding button |  |  |  |  |
| 8 | 1. | 72 hours later | Stop simulation in Visual Studio | Screen should show that the program was executing until stopped by tester |  |  |  |  |

## End User Test Script

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Step** | **Input or Prerequisites** | **Procedure** | **Expected Results** | **Actual Results** | **Initial & Date/Time** | **Attachment Ref.** | **Pass/Fail** |
| 1 | 1. Prepare Raspberry Pi |  | Open terminal window on Raspberry Pi | Terminal window opens |  |  |  |  |
|  | 2. |  | Change directory to Download/bin/Debug | Directory is changed |  |  |  |  |
| 2 | 1. Prepare laptop |  | Open ClientSideApplication CCB.cs in Visual Studio on Laptop | CCB.cs loads in Visual Studio |  |  |  |  |
| 3 | 1. Start Server side |  | Run sudo mono ConsoleApp1.exe on the Raspberry Pi | Program runs and shows “Waiting for connection…” on screen. |  |  |  |  |
| 4 | 1. Start Client Side |  | Start the CCB.cs code by pressing the play button in Visual Studio | The code starts and the GUI with buttons, arrows, and a slider show on screen |  |  |  |  |
| 5 | 1. Check CRC32 |  | Ensure CRC32 code on Raspberry Pi monitor matches | Both numbers should match |  |  |  |  |
| 6 | 1. Next Button | All button presses will be actual buttons, not GUI buttons | Press the Next Button | Button 4 shows depressed.  Next Button flashes.  Button 4 is enabled on pi monitor |  |  |  |  |
| 7 | 1. Go Button |  | Press Go Button | Should be no reaction.  Button should not flash.  Only Button 4 should be enabled |  |  |  |  |
| 8 | 1. Deadman Button |  | Press and hold Deadman Button | Deadman Button should light.  Button 1 should enable.  Button 4 should remain enabled. |  |  |  |  |
| 9 | 1. Go Button | Deadman Button still held | Press Go Button | Button 2 shows depressed.  Go Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
|  |  |  | Cue timer finishes. | Only Button 4 is enabled. |  |  |  |  |
| 10 | 1. Next Button |  | Press Next Button | Button 4 shows depressed.  Next Button flashes.  Button 1, 3, and 4 are enabled |  |  |  |  |
| 11 | 1. Go Button | Deadman button pressed and held | Press Go Button | Button 1 shows depressed.  Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
| 12 | 1. Next Button |  | Press Next Button | Button 4 shows depressed.  Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
| 13 | 1. Stop Button |  | Press Stop Button | Button 2 shows depressed.  Stop Button flashes.  Button 3 and 4 are enabled |  |  |  |  |
| 14 | 1. Previous Button |  | Press Previous Button | Button 3 shows depressed.  Previous Button flashes.  Button 1 and 4 are enabled |  |  |  |  |
| 15 | 1. Next Button |  | Press Next Button | Button 4 shows depressed.  Next Button flashes.  Button 3 and 4 are enabled |  |  |  |  |
| 16 | 1. Go Button | Deadman Button pressed and held | Press Go Button | Button 1 shows depressed.  Go Button flashes.  Button 2, 3 and 4 are enabled. |  |  |  |  |
| 17 | 1. Next Button |  | Press Next Button | Button 4 shows depressed.  Next Button flashes.  Button 2, 3, and 4 are enabled |  |  |  |  |
| 18 | 1. Previous Button |  | Press Previous Button | Button 3 shows depressed.  Previous Button flashes.  Button 2 and 4 are enabled. |  |  |  |  |
| 19 | 1. Stop Button |  | Press Stop Button | Button 2 shows depressed.  Stop Button flashes.  Button 3 and 4 are enabled |  |  |  |  |
| 20 | 1. Joystick |  | Move Joystick forward and backward | No output should be shown on screen |  |  |  |  |
| 21 | 1. Jog Button |  | Press and hold Jog Button | Jog Button should light |  |  |  |  |
| 22 | 1. Joystick | Jog Button held | Move Joystick backward | Value of 0 to 512 should be displayed on Channel 0 |  |  |  |  |
| 23 | 1. |  | Move Joystick forward | Value of 512 to 1000 should be displayed on Channel 1 |  |  |  |  |
| 24 | 1. Slider |  | Move slider | Slider value of 0 to 100 should be displayed |  |  |  |  |