**System Test Plan**

**For**

**EGR101Sim**

Emily Connearney

Luke Crump

Vivian Dang

Daniel Khalil

Keely Mashburn

|  |  |  |
| --- | --- | --- |
| Version | Author | Date |
| 1.0 | Daniel Khalil, Vivian Dang | 10/19/2021 |
| 2.0 | Luke Crump | 12/4/2021 |
|  |  |  |
|  |  |  |

Table of Contents

[1. Introduction 2](#_Toc89546446)

[1.1 Purpose 2](#_Toc89546447)

[1.2 Objectives 2](#_Toc89546448)

[2. Functional Scope 2](#_Toc89546449)

[3. Overall Strategy and Approach 2](#_Toc89546450)

[3.1 Testing Strategy 2](#_Toc89546451)

[3.2 System Testing Entrance Criteria 4](#_Toc89546452)

[3.3 Testing Types 4](#_Toc89546453)

[3.4 Suspension Criteria and Resumption Requirements 6](#_Toc89546454)

[4. Execution Plan 6](#_Toc89546455)

[4.1 Execution Plan 6](#_Toc89546456)

[5. Traceability Matrix & Defect Tracking 10](#_Toc89546457)

[5.1 Traceability Matrix 10](#_Toc89546458)

[5.2 Defect Severity Definitions 10](#_Toc89546459)

[6. Environment 11](#_Toc89546460)

[6.1 Environment 11](#_Toc89546461)

[7. Assumptions 11](#_Toc89546462)

[8. Risks and Contingencies 11](#_Toc89546463)

# Introduction

## Purpose

This document is a test plan for EGR101Sim System Testing, produced by the System Testing team. It describes the testing strategy and approach to testing the team will use to verify that the application meets the established requirements of the product owner prior to release.

## Objectives

* Meets the requirements, specifications, and rules of the product.
* Supports the intended product functions and achieves the required standards.
* Satisfies the Entrance Criteria for User Acceptance Testing.

# Functional Scope

The Modules in the scope of testing for the EGR101Sim System Testing are mentioned in the document attached in the following path :

1. The System Requirements Specification document:
2. Section 3.1 of this document.

# Overall Strategy and Approach

## Testing Strategy

EGR101Sim System Testing will include testing of all functionalities that the scope (Section 2) identified. System testing activities will include the testing of new functionalities, modified functionalities, screen level validations, workflows, functionality access, testing of internal & external interfaces.

The” Testing Types” section following this one will be discussing what needs to be tested. However, this section will describe how the types will be tested.

### Emulation Testing

* **Testing Objective**: The Arduino emulation functionality, its simulated pin connections, voltage at per component, and current of the system according to the specific requirements in the SRS.
* **Technique**: Java application that provides an instance of the emulation, and provides simulated voltages, connections, etc. warnings should be output if invalid connections and input voltage pairs are present during execution, otherwise if valid connections and input voltage pairs are present during execution the application should validate specific test cases.
* **Completion Criteria**: When all available test cases for emulation have been completed successfully

### Virtual Programmable Board Testing

* **Testing Objective**: The simulated Arduino hardware should perform actions based specifically on instructions compiled from the Arduino language according to the specific requirements in the SRS.
* **Technique**: Java application that provides a group of test cases with Arduino code and component configurations which will execute and validate expected behavior.
* **Completion Criteria**: When all available test cases for Virtual Programmable board testing have been completed successfully

### Communication Testing

* **Testing Objective**: Valid communication between Unity and the emulated Arduino according to the specific requirements in the SRS.
* **Technique**: Java application that executes a group of test cases that begin an instance of the application and capture TCP logging information from the communication between Unity and Java.
* **Completion Criteria**: When all available test cases for Communication testing have been completed successfully

### Simulation Testing

* **Testing Objective**: correct sensor readings and actions performed based on information captured from emulated Arduino connection in the 3-dimensional simulated environment.
* **Technique**: Python application that executes Unity executable simulation on a test-oriented course and logs information regarding sensor information and performed actions.
* **Completion Criteria**: When all available test cases for simulation testing have been completed successfully

### Bot Design Testing

* **Testing** **Objective**: design changes utilizing the bot design interface should change Arduino emulator component list.
* **Technique**: Unity based test that sends information to the Java Server and logs whether components, or connections were applied successfully to the emulator.
* **Completion** **Criteria**: When all available test cases for Bot Design testing have been completed successfully

## System Testing Entrance Criteria

In order to start system testing, certain requirements must be met for testing readiness. The readiness can be classified into usability testing, functional testing, and data and documentation testing.

## Testing Types

### Usability Testing

User interface attributes, cosmetic presentation and content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is comfortable to use and provides the user with consistent and appropriate access and navigation through the functions of the application.

|  |  |
| --- | --- |
| **SRS Requirement #** | **Description** |
| 4.3.2.8 | The system shall restart the simulation if the restart button is selected. |
| 4.3.2.9 | The system shall end the simulation if the end button is selected. |
| 4.3.2.10 | The system shall pause the simulation if the pause button is selected. |
| 4.3.2.11 | The system shall play the simulation if the play button is selected. |
| 4.1.2.3 | The system shall record the pin locations that are connected by a wire in a file. |
| 4.1.2.4 | The system shall have a button to save the current wire configuration. |
| 4.1.2.5 | The system shall save the current configuration of wires when the “save” button is pressed. |
| 4.1.2.6 | The system shall have a button to exit the Wiring and Design Interface. |
| 4.1.2.7 | The system shall exit to the main view screen when the exit button is pressed. |
| 4.2.3.1 | The system shall have a File button that shows file operations when clicked on. |
| 4.2.3.13 | The system shall have a save button appear when the File button is clicked. |
| 4.2.3.3 | The system shall save the Arduino script when the save button is clicked. |
| 4.2.3.4 | The system shall have an open button appear when the File Button is clicked. |
| 4.2.3.5 | The system shall open a new Arduino script when selected. |
| 4.2.3.6 | The system shall have a “save configuration” button appear when the File button is clicked. |
| 4.2.3.7 | The system shall save a configuration file when the “save configuration” button is clicked. |
| 4.2.3.8 | The system shall produce a configuration file that contains the wiring setup and the Arduino code when the “save configuration” button is pressed. |

### Functional Testing

The objective of this test is to ensure that each element of the component meets the functional requirements of the business as outlined in the:

* Software Requirements Specifications Document
* EGR101 Instructors rules or conditions

|  |  |
| --- | --- |
| SRS Requirement # | Description |
| 4.4.3.1 | The system shall allow for pin connections between the Arduino pins and components. |
| 4.4.3.2 | The system shall simulate the Arduino clock by counting in microseconds after executing. |
| 4.4.3.3 | The system shall allow for calls from Arduino code to delay programmable interaction with the components. |
| 4.4.3.4 | The system shall match the behavior defined from compiled Arduino code. |
| 4.4.3.5 | The system shall allow for digital writing to pins by providing voltage via simulated Pulse Width Modulation (PWM). |
| 4.4.3.6 | The system shall allow for analog writing to pins by providing a voltage. |
| 4.4.3.7 | The system shall provide traditionally used libraries like Servo.h and Serial.h to interface with. |
| 4.4.3.8 | On execution the system shall push voltage to each pin sequentially beginning at the IO and power ports of the Arduino. |
| 4.4.3.9 | On execution the components shall work if and only if they have adequate ground connection, and enough voltage to satisfy the potential of the component |
| 4.1.2.1 | The system shall allow the user to select a wire. |
| 4.1.2.2 | The system shall allow the user to move the wire to connect two pin locations. |
| 4.3.2.1 | The system shall prompt the user to select a course, the course selected will be displayed and executed on. |
| 4.3.2.2 | The emulation thread shall send each components behavior to the simulation UI via TCP connection per cycle. |
| 4.3.2.3 | The simulation UI shall send each components sensor data to the Arduino Emulation via TCP connection per update iteration. |

### Data and Documentation Testing

Data and documentation cover all the user guides, installation guides, readme files, and set up a manual that is provided with the software to ensure that the user understands the EGR101 Simulation System. The objectives of this type of testing: check if what is stated in the documents is available in the software and check if the explanation of the system is correctly explained in the documentation.

|  |  |
| --- | --- |
| **SRS Requirement #** | **Description** |
| 4.1.2.3 | The system shall record the pin locations that are connected by a wire in a file. |
| 4.2.3.7 | The system shall save a configuration file when the “save configuration” button is clicked. |
| 4.2.3.8 | The system shall produce a configuration file that contains the wiring setup and the Arduino code when the “save configuration” button is pressed. |
| 4.3.2.2 | The emulation thread shall send each components behavior to the simulation UI via TCP connection per cycle. |
| 4.3.2.3 | The simulation UI shall send each components sensor data to the Arduino Emulation via TCP connection per update iteration. |
| 4.3.2.7 | The system should display console logs in the Arduino IDE during simulation execution if logging is present within Arduino code. |

## Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan.

### Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the hardware, software, or database, it is up to the Testing Manager to determine whether the test plan will be re-executed, or part of the plan will be re-executed.

### Resumption Requirements

Resumption of testing will be possible when the functionality that caused the suspension of testing has been retested successfully.

# Execution Plan

## Execution Plan

The execution plan will detail the test cases to be executed. The Execution plan will be put together to ensure that all the requirements are covered. The execution plan will be designed to accommodate some changes, if necessary, if testing is incomplete on any day. All the test cases of the projects under test in this release are arranged with the subsystem they belong to with the more important at the top.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SRS Requirement #** | **Test Case #** | **Input** | **Expected Behavior** | **Pass/Fail** |
| 4.4.3.1 | 1.0 | The user shall click on a pin on the Parallax board and a pin located on a component | The pins will be connected by a wire. | Pass |
| 4.4.3.2 | 1.1 | The tester will print out the time in milliseconds while the program is running | The system will print out the time variable in the console in milliseconds. | Pass |
| 4.4.3.3 | 1.2 | The tester will use the delay function to delay the operation of code for 1000 milliseconds. | The system will delay for the time specified. | Pass |
| 4.4.3.4 | 1.3 | The tester will program a script in Arduino | The simulation will perform the code in the Arduino script. | Pass |
| 4.4.3.5 | 1.4 | The tester will use the digitalWrite function with a pin number as the argument | The corresponding pin will receive a 5V voltage. | Pass |
| 4.4.3.6 | 1.5 | The tester will use the analogWrite function with a pin number as the argument. | The corresponding pin will receive a voltage specified in the function call. | Fail |
| 4.4.3.7 | 1.6 | The user shall call the Servo.h library in a script | The user can call a function from the Servo.h library | Pass |
| 4.4.3.7 | 1.6.1 | The user shall call the Serial.h library in a script | The user can call a function from the Serial.h library | Pass |
| 4.4.3.8 | 1.7 | The tester will print the voltage of the IO pins | The pins will sequentially get power | Fail |
| 4.4.3.9 | 1.8 | The tester will hook up a component with too much power. | The component will not work, and an error will be thrown | Fail |
| 4.4.3.9 | 1.8.1 | The tester will hook up the same component in test case 1.8 but will have too little power | The component will not work, and an error will be thrown | Fail |
| 4.4.3.9 | 1.8.2 | The tester will hook up the same component in test case 1.8 but will have the correct amount of power | The component will work as intended | Fail |
| \*The system is in the Arduino IDE Scene for the following test cases: | | | | |
| 4.2.3.1 | 2.0 | The user will click on the file tab in the Arduino IDE | The file tab will drop down to show file operations | Pass |
| 4.2.3.2 | 2.1 | The user will click on the file tab in the Arduino IDE | The file tab will drop down to show the save button | Pass |
| 4.2.3.3 | 2.1.1 | The user will click on the file tab in the Arduino IDE and then click on the save button | The script currently opened will be saved to the system. | Fail |
| 4.2.3.4 | 2.3 | The user will click on the file tab in the Arduino IDE | The file tab will drop down to show the open button | Pass |
| 4.2.3.5 | 2.3.1 | The user will click on the file tab in the Arduino IDE and then click the open button | The application will open the file explorer for the respective OS | Pass |
| 4.2.3.6 & 4.2.3.7 & 4.2.3.8 | 2.4 | The user will click on the file tab in the Arduino IDE and then click the save configuration button | The application will save a copy of the code and a copy of the wiring connections to a txt file | Fail |
| \*The system is in the Simulation Scene for the following test cases: | | | | |
| 4.3.2.1 | 3.0 | The user will select a scene from the scene list | The scene will start running in the simulation | Fail |
| 4.3.2.2 | 3.1 | The tester will print out the messages being sent to the simulation via the emulator | The console will print out the instructions that the boe-bot is performing in the simulation | Fail |
| 4.3.2.3 | 3.2 | The tester will print out the data the sensors are sending to the emulator | The console will print out the values the data sensors are reading | Fail |
| 4.3.2.5 | 3.3 | The user will input a script that causes the boe-bot to drive in circles | The boe-bot will update its direction in the simulation to match the turn | Pass |
| 4.3.2.6 | 3.4 | The user will have a script that causes both wheels to spin in the same direction | The boe-bot will move in the direction of the wheels turning | Pass |
| 4.3.2.7 | 3.5 | The user will have a script that uses logging. | The simulation will print out data values to the Arduino IDE console | Fail |
| 4.3.2.8 | 3.6 | The user will click the restart button | The simulation will restart. | Fail |
| 4.3.2.9 | 3.7 | The user will click the end button | The simulation will end | Fail |
| 4.3.2.10 | 3.8 | The use will click the pause button | The simulation will pause | Fail |
| 4.3.2.11 | 3.9 | The user will click the play button | The simulation will play | Fail |
| 4.3.2.12 | 3.10 | The use will upload a script to cause the boe-bot to have a collision | The boe-bot will not be able to move through the object | Pass |
| \* The system is in the Wiring Scene for the following test cases: | | | | |
| 4.1.2.1 | 4.0 | The user will click on a pin | The system will save this pin to the file and will set a start point for a wire | Fail |
| 4.1.2.2 | 4.1 | The user will click on a pin and then another pin | The system will save the pin to the file and set the end point for a wire | Fail |
| 4.1.2.4 & 4.1.2.5 | 4.2 | The user will click on the save configuration button | The system will save the values of the pins to a file. | Fail |
| 4.1.2.6 & 4.1.2.7 | 4.3 | The user will click on the exit button | The system will exit to the main view screen | Fail |
| 4.1.2.8 | 4.4 | The user will click on the resistor button | The system will add a resistor to the board | Fail |
| 4.1.2.9 & 4.1.2.10 | 4.5 | The user will connect a resistor to a pin on one end and a wire on the other | The system will connect the resistor to the wire and the pin | Fail |

Table 1. This table hoes into detail about how each requirement in the SRS document should be tested and the desired output.

# Traceability Matrix & Defect Tracking

## Traceability Matrix

List of requirements, corresponding test cases

## Defect Severity Definitions

|  |  |
| --- | --- |
| **Critical** | The defect causes a catastrophic or severe error that results in major problems and the functionality rendered is unavailable to the user. A manual procedure cannot be either implemented or a high effort is required to remedy the defect. Examples of a critical defect are as follows:   * System abends * Data cannot flow through a business function/lifecycle * Data is corrupted or cannot post to the database |
| **Medium** | The defect does not seriously impair system function can be categorized as a medium Defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of a medium defect are as follows:   * Form navigation is incorrect * Field labels are not consistent with global terminology |
| **Low** | The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of a low defect are as follows:   * Repositioning of fields on screens * Text font on reports is incorrect |

# Environment

## Environment

The system testing environment will be used for System Testing. In order to conduct the testing, the tester needs to have the following installed onto their computer:

* Unity v2020.3.18f1(64-bit)
* Java 1.8.0.\*

# Assumptions

Contains assumptions made specific to this project.

* The user has a computer that is made after 2015
* The user has access to the internet to download the application

# Risks and Contingencies

Define risks and contingencies.

# 