Bartini Drink Dispensing System

by

Lightweight Enterprises

Project Team: Noel Barron
Luke Mulder
Ben Hieltjes

Contact Person: Noel Barron
nbarron@sfu.ca

Submitted to: Dr. Andrew Rawicz - ENSC 440W
Mr. Steve Whitmore - ENSC 305W
School of Engineering Science
Simon Fraser University

Issue Date: March 28, 2016
Revision 1.0

Testing Overview

Lightweight Enterprises' testing has been divided into tests performed on the various subsystem and tests performed on the *Bartini* unit as a whole. The subsystems tested include software systems, mechanical systems, and electronic systems. This test plan has been created as a way for the engineering team to evaluate and fix problems associated with the project at a high level. A proper test plan will ensure a product quality. The proposed test plan is formulated using input/output combinations. Any output that does not match the expected one can be considered a failed test.

Software Testing

The *Bartini*'s software components can be natural split into two concurrent systems: the graphical user interface (GUI) and the software driving the control signals. Software testing is outlined in the following tables outlining expected input/output combinations. Table 1 outlines GUI tests while Table 2 outlines tests for the control software.

GUI Testing

Table 1: GUI Tests

Test No.	Input / Test Procedure	Expected Output
1.1.1	For every predefined drink option, click on the image.	A new window opens on top displaying the correct ingredients and quantities required to make the drink. Options to make the drink and close the window are present.
1..1.2	For every predefined drink option, choose option to invoke drink dispensing process.	Correct drink ID passed to control software. Correct quantities and liquids are used. Prevents user from selecting a drink which the machine is not able to produce. Quantities.txt is correctly updated.
1.1.3	For the custom drink option, click on the image.	A new window opens on top displaying 5 editable ingredients. Each ingredient can be chosen along with a respective quantity. Options to continue or to close the window are present.
1.1.4	In the custom drink window, edit the liquid and quantity fields.	No invalid liquids are permitted. No invalid quantities are permitted (Natural numbers no greater than the defined constant present in the code.)
1.1.4	In the custom drink window, edit a drink, choose the option to save, and close and re-open the custom drink window.	Valid entries in the saved drink persist in the new window to show success.
1.1.5	In any toplevel window including the drink customization, recipe view, and error messages, press the close button.	The intended window successfully closes leaving any other existing windows fully functional.
1.1.6	Press the exit button in the main window.	GUI should exit successfully printing an appropriate message in the terminal. Underlying platform should return to normal functioning state (CLI, Pi GUI, etc.)
1.1.7	Run the software on monitors of various sizes and type. Test using both mouse-based interface and touch-based interface.	GUI should not distort when running on different peripherals of the same size. Touch functionality should be analogous to mouse-based functionality.

Control Software Testing

Table 2: Control Software Test

Test No.	Input / Test Procedure	Expected Output
1.2.1	Invoke subroutine to rotate carousel.	Carousel rotates to desired position.
1.2.2	Invoke subroutine to dispense alcoholic beverage	Arm connected to the front servo articulates properly to hit the water-tap mechanism in the alcohol bottle
1.2.3	Invoke subroutine to mix beverage	Motor in the mixing container properly mixes the liquids contained within
1.2.4	Invoke base liquid dispensing mechanism	Only one of the base liquids are dispensed at a time
1.2.4	Invoke self-clean command	Water dispenses from the base liquids container onto the mixing chamber, which rinses the containers
1.2.5	Invoke subroutine to actuate solenoid valves, dc motors, servos, and any other switch-based peripherals.	Commanded peripherals actuate for the intended duration.

Mechanical Testing

Mechanical testing will focus on the moving parts of the project and any structural/enclosure concerns. Mechanical aspects are often the most fragile and as such, require frequent testing to ensure maximum performance. Mechanical tests are shown in Table 3.

Table 3: Mechanical Tests

Test No.	Input / Test Procedure	Expected Output
2.1.1	Apply a safe amount of force to the carousel.	Carousel rotates with minor friction resistance.
2.1.2	Place a reasonable load onto enclosure.	Enclosure does not collapse and maintains structural integrity.
2.1.3	Pick up the *Bartini* Unit	Enclosure stays intact, enough to move the unit around

Electronic Testing

Electronic testing is centred around the circuitry present in the *Bartini*. Most tests involve electronic measurements. Electronic tests are shown in Table 4.

Table 4: Electronic Tests

Test No.	Input / Test Procedure	Expected Output
3.1.1	Set Raspberry Pi GPIO pin to active.	Voltage difference measured at GPIO with respect to ground. Voltage is different than the GPIO pin when not active.
3.1.2	Measure voltage/current flowing through switch circuit with GPIO input both high and low.	Switch electronics “turn on” only when desired. “Off” state is sufficiently low such that the peripheral does no activate.

Unit Testing

Various unit tests are included to ensure the *Bartini* system is functional end-to-end. Tests reflect behavior and expectations of an end user. Unit tests are shown in Table 5.

Table 5: Unit Tests

Test No.	Input / Test Procedure	Expected Output
4.1.1	Order a valid drink using the GUI.	Mixed drink dispensed.
4.1.2	Order an invalid drink using the GUI.	Message is displayed communicating any errors or information pertinent to the situation.