McMaster University

Software & Mechatronics Capstone

eBin

Verification and Validation

Anna Wei 400031943
Alan Yin 400007807
Ditong Liu 400009984
Huajie Zhu 001403438
Junni Pan 400024523
Zhihao Yang 400013899

April 11, 2020

Contents

List of Tables

List of Figures

1 Revisions

Date	Revision#	Authors	Comments
Feb.14 2020	Revision 0	Anna Wei Alan Yin Ditong Liu Huajie Zhu Junni Pan Zhihao Yang	Initial draft of Verification and Validation
Apr.10 2020	Revision 1	Anna Wei Alan Yin Ditong Liu Huajie Zhu Junni Pan Zhihao Yang	Add hardware part of interface, add software code walk through, add more concepts of Verification and Validation

Table 1: eBin Table of Revision

2 Purpose

The purpose of this document is to support the design and development process of eBin, the auto-sorting garbage bin. In particular, this document will show in detail the tests have been performed on the functionality of eBin, which ensures that the eBin will correctly recognize and dispose the garbage in to the corresponding sub-bin.

3 Scope

This document will focus on individual function modules in the system that have been discussed previously within the system design document. This document will focus on the test cases of the performance of each module, its purpose it to ensure that each feature within the system is working as expected. For each module, testing will be done individually by giving the system both normal and edge inputs and record the outputs, it is an indication of the completeness and robustness of the current system.

4 Background

Sorting or arranging how people throw away their garbage is an integral part of the whole recycling process because waste management companies need to sort everything out first before loading it into recycling machinery.So effective segregation of wastes is really important, it can reduce the cost of the whole waste management process and helps the environment to achieve sustainable development. In campuses and offices, students and stuffs throw away tons of garbage everyday, but the problem is whether the garbage is disposed in the correct

bin.To provide a more convenient way for users to sort and dispose their garbage, we introduce our eBin. When user throw garbage into eBin, it will determine the type of garbage using image recognition and ideally, displace it on a screen so that the user will know which type the garbage belongs to. Once the type is recognized, it will dispose it into the corresponding sub-bins. The purpose of our project is to provide a more convenient and effective way for users to sort and dispose their garbage.

5 Test Cases

5.1 Testing Plan and Testing Factors

Testing and validation for eBin is planned using bottom up method, where unit testing for every module and function is done first, followed by integration testing on controllers and systems. During integration testing for different systems, we applied black box approach to ensure that functionalities matches the requirements, without digging deeply into the implementation.
Test cases for validation purpose includes user input tests, interface tests, and software-hardware integration tests. We worked with target users(university students) to simulate the full eBin workflow in designated environment to assure the product meets the needs of customers.
Most software tests are done internally for verification purpose, to ensure the performance in various scenarios align with the regulations and requirements, because the core design and algorithms are not visible to clients and stakeholders.
Please see below for the detailed test plan for all the components and systems.

Component	Test Plan Test Factors
Detection system	The main goal for testing the Detection system is to make sure that the webcam can capture clear images, and the images must contain the object in the center. Also, the images taken by the webcam must be sent to the Core App system within a short period of time. Tests for camera module API need to be carried out as well, which means, for the camera module library functions that are used, we need to test them out one by one.
Core App System	Core App system is the system that arrange the work flow of the other systems. The test plan for Core App system is simulating the input output signal. By simulating all the possible input signal, we can easily tell by the output signal that the Core App system is working or not. The order of the input and output signal also mattes, which means, we need to feed the input signal to the Core App system and observe if the output signal is right sequence.

Garbage disposal system	Garbage disposal system consists of a motor, python application and a plate, each of these components were tested out. Similar to the Detection system, all the library functions were tested out for the motor's python application. What is more, we approved that the motor and the flipping plate combination we chose is qualified for all the operations as well. For example, we tried putting the heaviest garbage at the edge of the flipping plate(worse case scenario), the flipping plate should not be cracked and the motor should be able to rotate.
Interface System	Interface system involves an LCD with its library, a button and related programs. To test the LCD and its function, all APIs that were used in the programs need to be tested out individually. For the button, testings for its functions as well as reliability (for example, debouncing issue) need to be carried out. Lastly, a white box test should be conducted in the main to make sure that the interface system can work as one.
Analytic System	As the most important system for our project, Analytic System acts as the backbone. Thus the test cases for Analytic System is much more complex and they need to be carried out very carefully. Over 100 types of garbage are tried out by the machine learning model under analytic system, all of the garbage are found either from libraries or lecture halls, if 80 out of 100 garbage are recognized by the analytic system, we call it a pass.

5.2 Detection System

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Testing webcam position and angle	DEFR02 DEFR03	Put an object at different position on the plate	Image of the object	Image of the object	Pass
02	Testing camera resolution	DEFR02 DEFR03	Launch Application	Clear picture of the object	Clear picture of the object	Pass

03	Testing webcam robustness	DEFR01 DEFR02 DEFR03	Hit the webcam by a force that is less than 10 N from different angles	Webcam flexibility less than 0.5 mm	Webcam displacement 0.3 mm	Pass
04	Continuity testing between the webcam and the microcomputer	DEFR02 DEFR03	signal inputs in each pin	signal received in each pin	signal received in each pin	Pass
05	Testing camera initialization	DEFR03	Launch Application	Monitor Window	Monitor Window	Pass
06	Testing the function to capture the reference frame	DEFR02 DEFR03	Launch Application	Set the first frame	Set the first frame	Pass
07	Testing the function to monitor the plate	DEFR02 DEFR03	Launch Application	Continuous monitoring the current frame	Continuous monitoring the current frame	Pass
08	Testing the function to capture the frame once an object has been detected	DEFR02 DEFR03	Put an object on the plate	Image of the object	Image of the object	Pass
09	Testing the function to delete the captured image once the garbage has been disposed	DEFR02 DEFR03	Dispose an garbage	Previous image has been deleted	Previous image has been deleted	Pass

5.3 Core App System

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Testing Tensorflow modal initialization once (when the program is running)	CFR01	Run main	return the modal warning information and start string	return the modal warning information and start string	Pass
02	Testing Detection system initialization several times	CFR01	Run main several times	The camera is on and the footage is displayed	The camera is on and the footage is displayed	Pass
03	Testing accept images from Detection system	No reference	image.png	camera off and Get the photo and store it in picture folder	camera off and Get the photo and store it in picture folder	Pass
04	Testing analytic system initialization several times (when the program is running)	CFR01	run analytic program several times	Nothing (If something goes wrong, the program will report an error)	Nothing (If something goes wrong, the program will report an error)	Pass
05	Testing send image to analytic system	No reference	image.png	Nothing (If something goes wrong, the program will report an error)	Nothing (If something goes wrong, the program will report an error)	Pass
06	Testing get analyzed information from analytic system	No reference	image.png	image analysis results	image analysis results	Pass

07	Testing motor system initialization several times (when the program is running)	CFR01	Run Motor.py several times	Nothing (If something goes wrong, the program will report an error)	Nothing (If something goes wrong, the program will report an error)	Pass
08	Testing send analyzed information to motor system	No reference	image analysis results	Motor turns the garbage into proper classification	Motor turns the garbage into proper classification	Pass
09	Testing motor system return reset state	CFR02	Reset signal	Detection system, Analytic system and Motor system initialize	Detection system, Analytic system and Motor system initialize	Pass

5.4 Garbage disposal system

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Testing if the motor is fixed properly	DIFR01	drive the motor several times with different power input	Motor flexibility less than 0.6 mm, hinge backlash less than 3 degree	Motor flexibility less than 0.6 mm, hinge backlash less than 3 degree	Pass

02	Testing if the motor is able to handle worst case scenario	DIFR01 DIFR02	drive the motor several times with heaviest garbage that can be found	motor is still able to rotate the flipping plate fast and smooth without permanent damage to itself	motor is still able to rotate the flipping plate fast and smooth without permanent damage to itself	Pass
03	Motor robustness testing	DIFR01 DIFR02	Continuously drive the motor with high load for more than 72 hours	Motor is still working(no overheating, no performance degradation)	Motor is still working(no overheating, no performance degradation)	Pass
04	Motor's accuracy	DIFR01	Different angle	angles with an error less than 1.0 degree	angles with an error less than 1.0 degree	Pass
05	Flipping plate and hinge robustness testing	DIFR01 DIFR02	Place the heaviest possible garbage (2L Milk)	the flipping plate does not crack and the hinge is not bent, deformation less than 1.0mm	the flipping plate does not crack and the hinge is bent, deformation more than 1.0mm	Fail
06	Testing for clockwise rotation command	DIFR01	Rotate 90 degrees clockwise	Motor rotate 90 degrees clockwise	Motor rotate 90 degrees clockwise	Pass

07	Testing for counter-clockwise rotation command	DIFR01	Rotate 90 degrees counter-clockwise	Motor rotate 90 degrees counter-clockwise	Motor rotate 90 degrees counter-clockwise	Pass
08	Testing for reset after garbage disposed	DIFR01	Garbage disposed	Motor reset to initial state	Motor reset to initial state	Pass

5.5 Interface System

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Testing LCD screen brightness	IFR01	Power up LCD module	LCD screen displays a visible welcome message	'Welcome' message seen on LCD	Pass
02	Testing LCD screen content	IFR01	launch LCD API and try different display content	LCD displays different content according to the input	LCD displays different content according to the input	Pass
03	Testing Button functionality	IFR02	press the button while running main	System message: Button pressed	System message: Button pressed	Pass

5.6 Analytic System

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Testing model loading	AFR02	Launch Application	System message: Model loaded	System message: Model loaded	Pass
02	Testing image loading	AFR02	Capture the image	Image loaded	Image loaded	Pass
03	Testing image resizing	AFR02	Image loaded	Modified image with size 299*299	Modified image with size 299*299	Pass
04	Testing image flipping	AFR02	Image loaded	Modified image with horizontal flipping	Modified image with horizontal flipping	Pass
05	Garbage classifying for cardboard	AFR01	Cardboard image loaded	Prediction result cardboard	Prediction result cardboard	Pass
06	Garbage classifying for glass	AFR01	Beer bottle image loaded	Prediction result glass	Prediction result glass	Pass
07	Garbage classifying for metal	AFR01	Drink can image loaded	Prediction result metal	Prediction result metal	Pass
08	Garbage classifying for paper	AFR01	Newspaper image loaded	Prediction result paper	Prediction result paper	Pass
09	Garbage classifying for plastic	AFR01	Plastic bottle image loaded	Prediction result plastic	Prediction result plastic	Pass
10	Garbage classifying for trash	AFR01	Crumpled paper image loaded	Prediction result trash	Prediction result trash	Fail - Possible result: paper
11	Testing prediction result output	AFR01 AFR02	Analyzing finished	Material type	Material type	Pass

5.7 Integration Testing

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Outputs	Results
01	Motor system and Coreapp system integration testing	PR01	Coreapp system receives analytic system's garbage type signal	Garbage is disposed into the corresponding bin	Garbage is disposed into the corresponding bin	Pass
02	Ebin process classification under 6 seconds	PR02	Any garbage	Garbage is disposed into the corresponding bin under 6 seconds	Garbage is disposed into the corresponding bin under 6 seconds	Pass

6 Beyond Testing

6.1 Code Walk-through

File	Sample Code	Concern	Potential Solutions
camera.py	Line 40 - 48	*crop_capt = capt[150:-350,280:-120]* *gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY* The function captures the image and crops it to emphasize the object. However, the parameters used here are absolute pixel positions, which may not be optimized for other camera units.	Change to relative pixel positions so that the function can work for different camera models.

camera.py	Line 142 - 164	*cv2.imwrite('./images/'...* *os.system('py ./analy...)* Program is capturing every frame of object movement until the movement stops. There is a potential issue when the captured data becomes too large for the built in storage.	Automatically delete old frames, and allow the program to keep the 24 most recent frames in storage for referencing purpose.
analytic.py	Line 40 - 47	*top_k=predictions[0].so rt()[-len(predictions[0]):]* When finding the result by analysing multiple images, the algorithm sorts the entire array every time to find the top 3 elements, which can degrade the performance when array gets large.	Improve the algorithm so that it stops after the top 3 elements are found.
interface.py	Line 20, 43, & 66	*lcd = CharLCD(cols=16, rows=2,pin_rs=37, pin_e=35,pins_data=[33...* The initial setup was included in multiple methods, therefore the hardware specification parameters were entered multiple times across the file. It will be hard to maintain when the program scales up.	Declare reusable constants at the beginning of program to enhance maintainability.

motor.py	Line 15 - 18 & Line 47 - 50	*def clockwiseByDeg():... GPIO.setmode (GPIO. BOARD)...* Motor class and interface class shares the same GPIO with different modes. Setting IO mode inside of a functional component may not be a good idea because there is a chance of deadlock when LCD and motor work at the same time.	Define a separate initialization method for motor class to avoid deadlock in setting GPIO mode.

6.2 Code Reviews

Code reviews were performed by both the software team and the hardware team. As the code of each module was constructed by different team members while the combination of software and hardware requires the understanding of both parts. The code integration and optimization was done mostly by the software team.

7 Supporting Material

Please refer to previous design documents.

8 Traceability Matrices

The following is the Traceability Matrix for the functional and non-functional requirements according to their test case classification:

Test Cases:	**Functional and Non-Functional Requirement:**
Detection System	Detection Functional Requirement 1
	Detection Functional Requirement 2
	Detection Functional Requirement 3

Table 10: Detection System Traceability

Test Cases:	**Functional and Non-Functional Requirement:**
Core App System	Core Functional Requirement 1
	Core Functional Requirement 2

Table 11: Core App System Traceability

Test Cases:	Functional and Non-Functional Requirement:
Garbage Disposal System	Disposal Functional Requirement 1
	Disposal Functional Requirement 2

Table 12: Garbage disposal Traceability

Test Cases:	Functional and Non-Functional Requirement:
Interface System	Interaction Functional Requirement 1
	Interaction Functional Requirement 2

Table 13: Interface Traceability

Test Cases:	Functional and Non-Functional Requirement:
Analytic System	Analytics Functional Requirement 1
	Analytics Functional Requirement 2

Table 14: Analytic Traceability

Test Cases:	Functional and Non-Functional Requirement:
Integration System	Performance Requirement 1
	Performance Requirement 2

Table 15: Integration Traceability