ENT SENIOR DESIGN PROJECT REPORT

Sustainable Waste Sorter

Submitted to

Dr. Weissbach
Dr. Lin
Computer Engineering Technology Program
Engineering Technology Department

by

Elisabeth Garza
Aaron Smerdel
Jordan Staton

April 30, 2019

Sustainable Waste Sorter

ABSTRACT

The purpose of this project is to help people eliminate the confusion on whether they should throw their trash away or dispose of it in a recycling bin. The sustainable waste sorter is an informative device that tells the user where to place their trash. Our customer and the origin of the idea came from an organization called Roche Diagnostics Operations. Roche Diagnostics Operations is a multinational healthcare organization, the Indianapolis location focuses more on creating and developing their diabetic test strips. The device is created of four main components which include a Raspberry Pi 2 Model B, a camera module, an LCD screen, and a casing/mount that holds all of these components together. All of these components are compatible with the Raspberry Pi 2 Model B. The software was programmed in Python and the database in MySQL. During the development of the device, the most challenging task was learning how to develop in the new language, Python. Once the device reached a stable state it was piloted at Roche Diagnostics Operations. The purpose of the first of three pilot sessions was to verify that the device worked in the environment and that the items entered in the database were recognized; as a result, the device passed that test. The second pilot session had the same purpose as the first pilot session but with more items in the database. The device received more interaction during the second pilot session, though the team decided to schedule a third pilot session once all the items were entered into the database and a revamped user interface was completed. The team entered about 800 entries into the database and created a new and interactive user interface for the device. The third pilot session was a success; the items that were scanned by testers were recognized and the new user interface was a success as well. Overall, the sustainable waste sorter project was successful and educational. We, as students, took all of our fundamental learnings from our previous courses and applied them to this project. This allowed us to enhance our problem solving and project management skills. As people use the device, we hope that it educates them on how to properly recycle therefore improving the environmental state of our planet.

Sustainable Waste Sorter

TABLE OF CONTENTS

Sustainable Waste Sorter

REVISION HISTORY

Version	Date	Revised by	Description
1.0	10 Dec 2018	Aaron Smerdel, Jordan Staton, and Elisabeth Garza	Initial version

Sustainable Waste Sorter

1. INTRODUCTION

We are a group of seniors in the Computer Engineering Technology department at IUPUI. In our final year of school, we have to complete a Capstone Project that showcases the material and skill sets that we have learned. The project that was assigned to us as a group was to design and create a device for Roche Diagnostics. Roche is a multinational Pharmaceutical and Diagnostic company. The project is targeted toward recycling waste and creating a mechanism that informs individuals on how to sort waste into its respective bin. It is a large template that can be expanded upon for future instances and is more designed to teach the public about recycling. The scope of this report is to introduce readers into the topic, provide details for the research and testing that went into the project, system-wide designs decisions, system architectural design, and to provide recommendations moving forward.

1.1 Problem Statement

Recycling plays a large role in our environment. There are many waste items out there that can be converted into a reusable material. Though, some are not aware of how to recycle properly, which can cause waste companies to spend more time and money to sort out the recyclable items accordingly. Our customer, Roche Diagnostics, wanted us to create an educational sorting system that helps users make a smart decision when disposing of their waste. To fulfill our customer’s needs we used Raspberry Pi components including a camera, LCD touchscreen, and the Raspberry Pi board to create the sorting system. This system has a camera module that acts as a barcode scanner and an image processor. The barcode scanner and image processing features provide the user feedback on how to sort their waste.

1.2 System overview

The project began with our company sponsor Roche, pitching a sustainability initiative that they wanted to implement on their campus. The initiative was to create a device that can help people identify which items can be recycled and which ones cannot. The way the device operates is that an individual walks up to the waste station and scans their disposables. The device scans either the recycling triangle on the item or the barcode. The device then informs the user where their waste should be placed via on-screen instructions through an LCD display. The history of system development has been very stable. After a couple of meetings with our company sponsor, we had an idea of what we wanted to create for them. After sending over a proposal and discussing it, the sponsor was satisfied with what we had designed with as it had met their needs. Once the device was finalized and delivered to Roche, they piloted the device in a specified dining area. From there, they will monitor the effectiveness of the device and eventually implement it in more areas of their campus if it succeeds.

Sustainable Waste Sorter

2. REFERENCED DOCUMENTS

This section shall list the number, title, revision, and date of all documents referenced in this document. This section shall also identify the source for all documents not available through normal Government stocking activities.

Table 1: Reference Documents

Title	Document Reference Number	Comment
Project Proposal	v1.3.2	Submitted 10/05/2019
Requirement Specifications	v1.3.2	Submitted 01/25/2019
Process Flow Diagram	v1.3.2	Submitted 10/05/2019
Gantt Chart	v1.3.2	Submitted 01/25/2019
Price sheet	v1.3.2	Submitted 10/05/2019
Project Code	v5.2.0	Completed 04/23/2019
Test Specifications	v1.0.0	Completed 04/24/2019
Poster	v1.0.0	Submitted 4/15/2019
Weekly Reports	Weekly Reports 2 - 14	Weekly reports by each team member

3. SYSTEM-WIDE DESIGN DECISIONS

The design of the system regarding the inputs and the outputs are solely based on user interaction. The user makes the initial decision which then dictates how the system operates, after that any decision the user makes then results as an output or the system will prompt the user for another input.

The design decisions on the system's behavior are based on user inputs. The first input starts at the opening window of the program. The user presses a button if they have an item with a barcode or the second button without a barcode. The output from the home screen directs the user to the barcode reader window or the window with 3 buttons containing pictures of common items found in the Roche Diagnostics Operations cafeterias. Each of the informing messages that are outputted to the user is set to only display for 5-10 seconds depending on how long the message is. If the input from the user is the barcode option, then the output is a window that scans the barcode from an item. The next input is the detected barcode from the item, as a result, the output is a window that informs the user on which bin to place the item. If the input from the user is the common items option, then the output is a window that displays 3 different buttons. The next input requires the user to decide which button they need to press and as a result, the output informs the user on which bin to place the item. These decisions allow the system to decide between the different types of materials which are plastic, metal, paper, glass or trash. A reference to Appendix G shows the software flow of the system. This reference shows the detail of each possible decision that is made within the system.

The design decision of the system's database was designed to be simple and easy to maintain. A reference to Appendix H. This reference describes the name of the database, the table, and the attributes of the table.

The system itself did not have many safety protocols to it. Though, the device did have to meet the safety protocols of the environment that is lives in. Since the device requires an extension cord, the extension cord must be placed in a safe position where people are not susceptible to trip over. The device has low security and privacy requirements, the system only requires a user password in order to execute the program and the database only requires a user password for access.

The design and construction of the hardware and hardware-systems decision were made by decision matrices. A reference can be made to Appendix I which displays the four decision matrices that were used when picking hardware. The hardware decisions were made upon the microcontroller, touchscreen, image processing/barcode, and the casing. The project proposal containing the customer requirements and engineering requirements state that the device must weigh less than 15 pounds and should cover less than or equal to an area of one square foot. The hardware systems, the programming language, and the database system were made designed with knowledge from past experiences of courses. We, students of IUPUI, have taken database management courses which is why we chose MySQL as the database software system. The programming language, Python, was chosen as the foundation language for this project because it is a well-known language to develop with on a Raspberry Pi. As Python is known as a programming language that is currently taught as IUPUI, it was also an opportunity to showcase the skills that we have learned throughout our four years at IUPUI.

3.1 Hardware

Due to the device being software based, design calculations were not needed to complete the design. However, the components needed to be compatible with one another (connection based). The components we chose worked well together because they were designed for the Raspberry Pi and came from the same manufacturer, Adafruit. In lieu of design calculations, a majority of our decision was made through the aid of a decision matrix. The hardware chosen for the project as a result of the decision matrices was the Raspberry Pi 2 Model B (Appendix A), the camera module (Appendix B), 7" LCD screen (Appendix C), and a 2.5-amp power supply (Appendix D). All of these hardware components fit well into the casing mount.

3.2 Software

The software for this system runs off the operating system Ubuntu Mate. The program was developed in the programming language, Python. Our decision matrices (Appendix I) had concluded that Python and Ubuntu Mate would be the best-suited programming language and operating system for the device. All of the libraries used on the Raspberry Pi 2 Model B were downloaded from open-source libraries. The software architecture can be viewed in Appendix G.

3.3 Interface

The user interface was built with the Python library named Tkinter. Tkinter viewed as Python's standard GUI. The user interface was designed in a way that invites the user to interact with the device. The main screen consists of two buttons, one button is used to scan an item with a barcode and the other button is a list of standard items located throughout the sponsor's facility. If the user selected the button with the barcode, then a new window opens that prompts the user to scan their item. When the barcode is scanned the program checks the database for a match and then returns the result to the user and informs where to place the item. If the user selects the button of common items or an item without a barcode, then a new window opens. This new window contains three different buttons with pictures prompting the user to make a decision. When the user makes a decision then the device informs the user with appropriate disposal bin.

4. SYSTEM ARCHITECTURAL DESIGN

4.1 System components

The components that make up the entire system architecture include:

- Raspberry PI 2 microcontroller - MO1
  - This component is the processing power of the device. It holds all of the memory of the database and OS on a flash chip. It resides on the backside of the plastic case behind the plating. It is planned that this microcontroller can be replicated for mass use. It has USB, Ethernet, AV HDMI, micro SD, and micro USB connectors and it requires 5V and 700mA to operate. It has a 0.9 GHz processor, 1 GB SDRAM, and the OS is Linux. The hardware is upgradable due to the code being the most dynamic aspect of the device.
- Raspberry PI camera module v2 8 Megapixels - MO2
  - The camera module captures the image of barcodes to relay the information to the microcontroller. It is connected to the microcontroller by an FFC (flexible flat cable). It is also connected to the plastic case by Lego block connection, so it is easily detachable. The still resolution is 8 Megapixels which can be optimized. The hardware is upgradable due to the code being the most dynamic aspect of the device.
- Pi Touchscreen Display 7” - MO3
  - This LCD display is what the users interact with. It is connected to the microcontroller, to receive information from the software modules, and the plastic case. It is a touch display, so it is interactive. A power connection is created through the GPIO pins of the microcontroller as well as the conversion of data from the microcontroller. The hardware is upgradable due to the code being the most dynamic aspect of the device.
- 5V Power Adapter - MO4
  - The power adapter is what powers the entire device. It is 5V and 2.5A and its sole purpose is to power the microcontroller so it can power the other hardware components. It has a micro USB port, so it is interchangeable with similar power adapters. The hardware is upgradable due to the code being the most dynamic aspect of the device.
- Plastic Case and Legos - MO5
  - The plastic case is what holds all of the hardware components together. It consists of a front panel to fit the touchscreen in, space for the microcontroller to rest securely in. In the top portion of the case it has a LEGO docking area to allow for the connection of other components, such as the camera module. This case is used to hold Raspberry Pi parts together for one project. The hardware is upgradable due to the code being the most dynamic aspect of the device.

- Main Module - CSCI1
  - The main module contains all of the startup commands and initializations for each component attached to the microcontroller. It communicates with the barcode module whenever a user initiates the barcode option to scan a barcode. (Appendix J)
- Barcode Module - CSCI2
  - The barcode module contains the database search commands as well as image detection. These commands run on a loop until something is scanned by barcode. This module communicates with the main module to know when to start scanning and it communicates with the common items' module and the feedback module when a conclusion has been reached. (Appendix K)
- Common Items Module - CSCI3
  - The common items module is the code logic that detects and displays what the user selects on the touchscreen to provide an answer for the user. This module communicates with the barcode module when a result is found. (Appendix L)
- Feedback Module - CSCI4
  - The feedback module is the chunk of code that disputes and displays what the user scans to provide an answer for the user. This module communicates with the barcode module when a result is found. (Appendix M)
- User Interface - CSCI5
  - The user interface is the visual representation of the whole project. It is what users communicate with directly. It also communicates with all modules at some point for various displays.
- Database - CSCI6
  - The database is what houses all of the cataloged barcode information. The database gets called upon by the barcode module when a query is made to find a specific barcode. The design of the database allows for expansion so that more and more items can be added and so that improvements can be made to the whole system.

4.2 Concept of execution

The concept of executing starts with the relationship between all components of the device. The Raspberry Pi is the heart of the whole system. The Raspberry Pi is placed in a slot the casing. Connected to the Raspberry Pi 2 Model B are the FFC (flexible flat cable) from the camera module, the FFC (flexible flat cable) from the LCD screen, and the micro USB cable which is used to power the device. A reference to Appendix E showcases the relationship between all four of the components. During the execution of the program, once the user scans a barcode the program then makes a connection to the database source on the Raspberry Pi. If and/or when a match is found, then the database returns this information and the code manipulates the information returned. It then makes the decision on where the item should be placed. All windows that open during the program execution, except the main screen, have a timeout. When the user presses either of the two buttons on the main screen the corresponding window opens and then times out after 20 seconds. The concept of this design is intended for consistent usage of the device. For the situation in which a user walks up to the device, presses a button, then walks away from the device, the window will time out. This ensures the next user will see the home screen which will allow for easy usage and no confusion. All other message windows that open once the user has either scanned a barcode or picked one of the three buttons, times out after 5-10 seconds. If the barcode item that is scanned by the user is not found, then the program informs the user that the item was not found. The program then saves the barcode value scanned to a CSV file in the background for an entry in the future.

4.3 Interface design

4.3.1 Interface identification and diagrams

The interface diagram for the project can be referenced to Appendix E. There are three main interface connections between the Raspberry Pi, the camera module, and the LCD screen display. The Raspberry Pi is the linchpin for these connections. Data is passed through the Raspberry Pi from the camera module to be displayed on LCD.

4.3.2 (Raspberry PI 2 Microcontroller - MO1)

- a. Priority: High
- b. Type of interface: Processing unit of the device, interfaces with LCD and camera module
- c. Characteristics of Data Elements
    - 1) Raspberry Pi Camera Module
        - a) Project-unique identifier: MO2
        - b) Non-technical name: Camera
        - c) Technical name: Sony IMX219 Image Sensor
    - 2) Data type: image file
    - 3) Size and format: 16mB micro SD card
    - 4) Priority/Timing Restraints: Small Processing Power
    - 5) Security and Privacy: Local device
- d. Characteristics of Communication
    - 1) communicates with MO2 and M03
    - 2) connects with FFC cable and through GPIO
- e. Characteristics of Protocols
    - 2) Layer: Physical and Network
    - 3) Packet Routing: sends it through the GPIO pins
    - 4) Error Control: Everything is backed up on the micro SD card
    - 5) Synchronization: creates a connection to the database requiring a username and password.
    - 6) Other features: The microcontroller has enough processing power to not only power the LCD and camera, but it can also support 20 GPIO pins.

4.3.3 <u>(Raspberry PI camera module v2 8 Megapixels - MO2)</u>

- a. Priority: High
- b. Type of interface: Real-time video feed
- c. Characteristics of Data elements
  - 1) Raspberry PI 2 B Microcontroller
    - a) Project-unique identifier: M01
    - b) Raspberry Pi
    - d) Technical name: N/A
    - e) Abbreviation or synonymous names: RasPi
  - 2) Data type (alphanumeric, integer, etc.)
  - 3) Size and format:
  - 4) Units of measurement: Sends frame captures of live feed
  - 5) Range: N/A
  - 6) Accuracy: >95% accuracy reading barcodes
  - 7) Constrained to updating until the barcode is fully in the frame and held still
  - 8) Security and privacy constraints: N/A
  - 9) Sources: Setting/sending - Camera Module, Recipients using/receiving entities - Raspberry Pi 2 B Microcontroller
- 2) Frames are captured and immediately replaced if a barcode is not found in the frame
- 3) Medium: The frames come directly from the video feed as an object in the code
- 4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
- 5) Relationships among assemblies, such as sorting/access characteristics
  - d. Characteristics of Communication
  - 1) communicates with MO1
  - 2) connects with FFC (flexible flat cable) and through GPIO ports

- e. Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
  - 1) Communication between MO1 and MO2
  - 2) Physical communication through hardwired connection between Raspberry Pi and camera via an FFC (flexible flat cable)

4.3.4 <u>(Pi Foundation Display - 7" Touchscreen Display for Raspberry Pi MO3)</u>

- a. Priority: High
- b. Type of interface: Real-time data transfer and communication with the user
- c. Characteristics of individual data elements that the interfacing entity(ies):
  - 1) Raspberry PI 2 B Microcontroller
    - a) Project-unique identifier: M01
    - b) Raspberry Pi
    - d) Technical name: Raspberry Pi 2 Model B
    - e) Abbreviation or synonymous names: RasPi
  - 3) Size and format: 800 x 400 pixels
  - 4) Units of measurement: Sends frame captures of live feed
  - 6) Accuracy: >95% accuracy reading barcodes
  - 7) Constrained to updating until the barcode is fully in the frame and held still
  - 9) Sources: Setting/sending - Camera Module, Recipients using/receiving entities - Raspberry Pi 2 B Microcontroller
- d. Characteristics of data element assemblies:
  - Communicates with M01and M02
  - Connect to Raspberry Pi 2 Model B through FFC cable
- 2) Data elements in the assembly and their structure:
  - Reference Appendix E
- 4) Visual and auditory characteristics of displays and other outputs:
  - The visual characteristic of the LCD includes the graphic user interface created to interact with its user.

4.3.5 User Setup and Operation

Setup for this device includes gathering all of the necessary parts and putting them together. The waste sorter is entirely made up of Raspberry Pi branded items: Raspberry Pi 2 microcontroller board, Raspberry PI camera module v2 8 Megapixels, 5V Power adapter, Pi Touchscreen Display 7", plastic case for a 7" display and camera, and 2 10x2 legos. There is a ram chip that contains all of the OS data and applications required for operations. This can be flash-copied, and mass produced for any number of devices. Initially, remove the plastic casing on the back side of the LCD mount. Then, place the Raspberry Pi inside of the enclosure. The camera module has a special FFC (flexible flat cable) that must be securely fastened into the PI board. Thread the cable from the camera module through the plastic cover into the Raspberry Pi. The camera module can then be attached to the top of the case using the legos. The power adapter is fairly short, so an extension cable is recommended.

The program that is needed should already be on the flash chip. All that is required of the operator is to plug in the device and set it in the desired location next to some waste bins, turn on the Raspberry Pi, and start the application on the desktop. The camera may have to be tuned by rotating the lens. The main screen gives the user two options of identifying an item: a barcode and common cafeteria items. Select the option desired or scan a barcode. The result is displayed on the screen and then the program returns to the main screens.

Sustainable Waste Sorter

5. CONCLUSIONS AND RECOMMENDATIONS

This device was successfully created as it met all engineering requirements that were drafted when the scope of the device was being defined. The system correctly identifies different types of waste located throughout Roche Diagnostics' campus. It can correctly discern between the items through two distinct methods. The system can detect a barcode and search through a database to retrieve the disposal method of the item. As well as display a visual representation of commonly found items on the campus, which are sorted into different categories (different recycling bins or waste bin).

The most valuable experience from this project is the experience of how engineers may function between one another in an actual business. The software aspects were divided into categories and each team member had a main focus for the software design. These individual modules then had to be combined together into one working system. Other engineer experience included reaching project milestones by set deadlines so that progress can be accurately monitored.

Lastly, in retrospect, this project may have been better suited to being built in C# as the coding language and Windows IoT for the operating system. This device becomes very centralized on the graphical user interface. Python lacks in comparison to C#; software limitations were seen throughout the design process. These limitations may not have appeared in C# as it is more robust in the user interface field. However, it is a great learning experience to display the skill of being able to learn a new language. As well as apply previous skills to create a comprehensive design in an unfamiliar environment.

Sustainable Waste Sorter

NOTES

Technical References

Python - High Level Programming Language

C# - High Level OOP Language

FFC - Flexible Flat Cable

Linux - Family of Open Source Operating Systems

MySQL - Open Source Relational Database Management

IoT - Internet of Things

Raspberry Pi - Small, credit card sized, portable computer

Sustainable Waste Sorter

REFERENCES

A. R. (2018, May 21). An OpenCV barcode and QR code scanner with ZBar. Retrieved from https://www.pyimagesearch.com/2018/05/21/an-opencv-barcode-and-qr-code-scanner-with-zbar/

Chapter 7 Export Excel Data into MySQL. (n.d.). Retrieved from https://dev.mysql.com/doc/mysql-for-excel/en/mysql-for-excel-export.html

Python GUI Programming. (n.d.). Retrieved from https://www.w3schools.in/python-tutorial/gui-programming/

Python Tutorial. (n.d.). Retrieved from https://www.w3schools.com/python/

Unittest.mock - getting started. (n.d.). Retrieved from https://docs.python.org/3/library/unittest.mock-examples.html

Sustainable Waste Sorter

APPENDIXES

Appendix A

Raspberry Pi

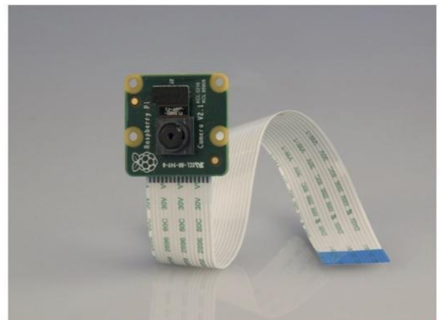
Raspberry Pi 2, Model B

Product Name	**Raspberry Pi 2, Model B**
Product Description	The Raspberry Pi 2 delivers 6 times the processing capacity of previous models. This second generation Raspberry Pi has an upgraded Broadcom BCM2836 processor, which is a powerful ARM Cortex-A7 based quad-core processor that runs at 900MHz. The board also features an increase in memory capacity to 1Gbyte.
Specifications	
Chip	Broadcom BCM2836 SoC
Core architecture	Quad-core ARM Cortex-A7
CPU	900 MHz
GPU	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V, 2A
Connectors:	
Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4)
Audio Output	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
JTAG	Not populated
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Micro SDIO

Sustainable Waste Sorter

Appendix B

Raspberry Pi



Camera Module

Product Name	**Raspberry Pi Camera Module**
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Numer	**913-2664**

Specifications

Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464
Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4″
Dimensions	23.86 x 25 x 9mm
Weight	3g

Sustainable Waste Sorter

Appendix C

(images/104110009 01_01.jpg)

Raspberry Pi 7" Touchscreen Display

SKU: 104110009

Description

The 7" Touchscreen Monitor for Raspberry Pi gives users the ability to create all-in-one, integrated projects such as tablets, infotainment systems and embedded projects. The 800 x 480 display connects via an adapter board which handles power and signal conversion. Only two connections to the Pi are required; power from the Pi's GPIO port and a ribbon cable that connects to the DSI port present on all Raspberry Pi's. Touchscreen drivers with support for 10-finger touch and an on-screen keyboard will be integrated into the latest Raspbian OS for full functionality without a physical keyboard or mouse.

Features:

- 7" Touchscreen Display.
- Screen Dimensions: 194mm x 110mm x 20mm (including standoffs)
- Viewable screen size: 155mm x 86mm
- Screen Resolution 800 x 480 pixels
- 10 finger capacitive touch.
- Connects to the Raspberry Pi board using a ribbon cable connected to the DSI port.
- Adapter board is used to power the display and convert the parallel signals from the display to the serial (DSI) port on the Raspberry Pi.

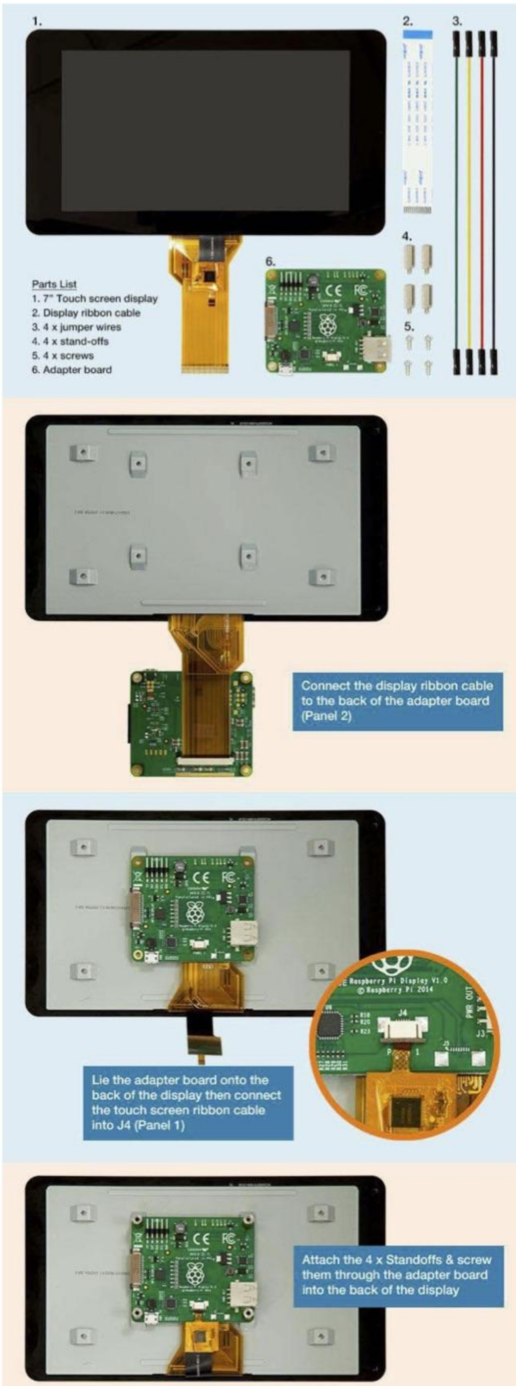
Part List:

- 1 x 7" Touchscreen Display
- 1 x Adapter Board
- 1 x DSI Ribbon cable
- 4 x stand-offs and screws (used to mount the adapter board and Raspberry Pi board to the back of the display
- 4 x jumper wires (used to connect the power from the Adapter Board and the GPIO pins on the Pi so the 2Amp power is shared across both units)

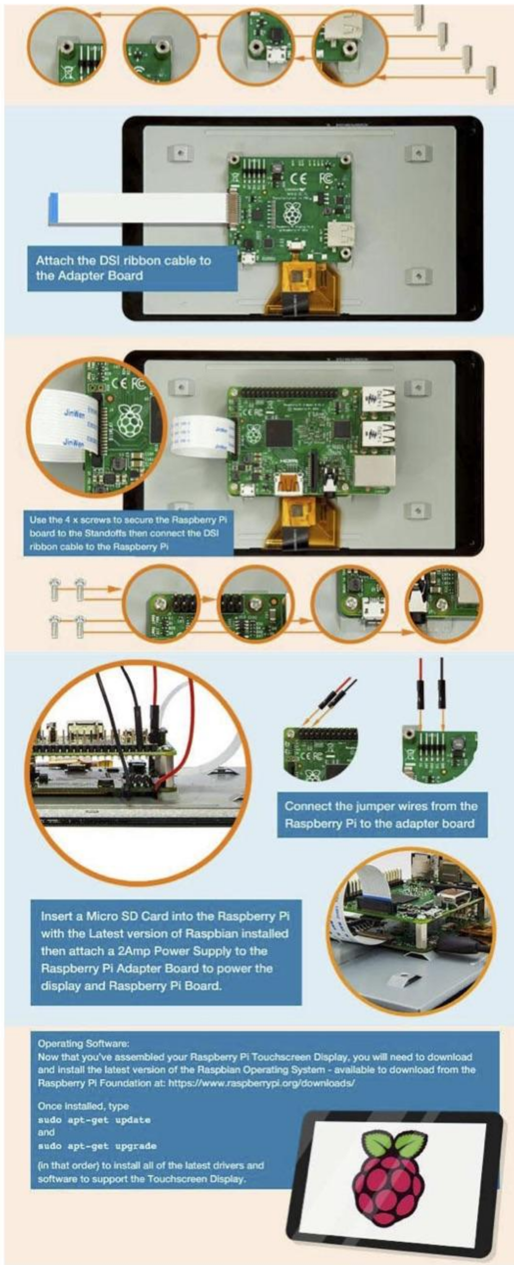
(http://www.seeedstudio.com/depot/Maker-Pro-t-1672.html?ref=pinfo) Designer: Raspberry Pi Foundation (http://www.raspberrypi.org/)
Other Products From This Designer (http://www.seeedstudio.com/depot/Raspberry-Pi-Foundation-m-93.html?ref=pinfo)
Weight: 514 g

Sustainable Waste Sorter

Page 2 of 4



Sustainable Waste Sorter



Overview

Sustainable Waste Sorter

Appendix D

Raspberry Pi Power Supply

Features:

Built specifically for use with Raspberry Pi

Class II design 5vdc 1A output via Micro USB

Energy efficient to ErP stage 2

Description:

A 5vdc 1A Euro Micro USB power supply is manufactured specifically for use with the Raspberry Pi device. It offers a highly efficient output meeting latest CEC stage 2 (V) requirements and is safety approved. This unit has fixed European pins and features short circuit and over current protection as standard. This Raspberry Pi power supply has M.T.B.F of 50K hours at 25 degrees C.

Part Number	RPI-PSU-EU-MK1
Output	5vdc 1A maximum
Current Min.	0.01A
Power (watts)	5W
Line Reg	+/-5% at rated load
Total Output Regulation	+/-5 % at 0—100% load
Ripple & Noise (mV p-p)	200mV P-P
Protections	Over Current and Short Circuit
Case Size	54 x 24 x 38mm
Weight (approx.)	70g
DC Cord	1.8 Metres
DC Plug	Micro USB
Rated Input Voltage	100-240Vac
Full Input Voltage Range	90-264Vac
Rated Frequency	50-60Hz
Full Frequency Range	47-63Hz
Efficiency	68.7%

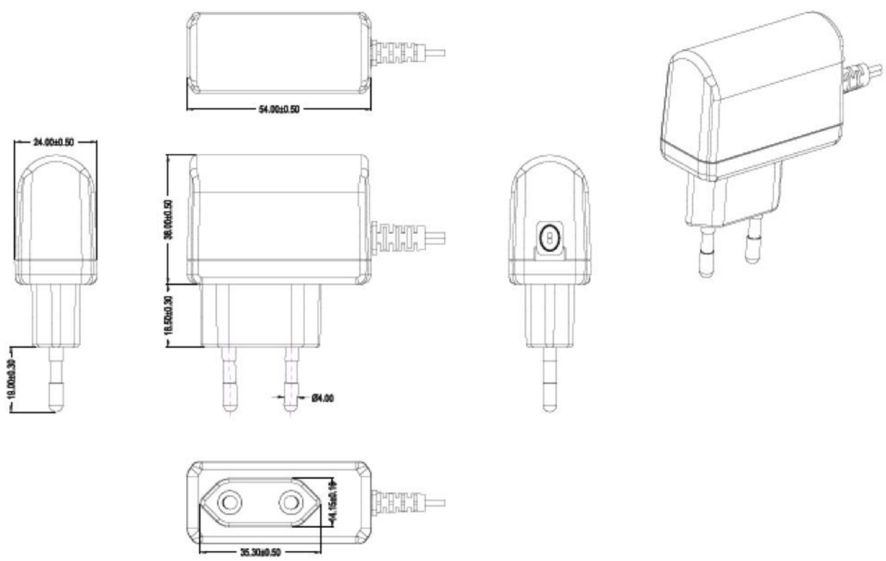
Leakage Current	shall not exceed 0.25mA
Input Power	7.72W max
Input Current (RMS Max.)	0.18A max
Hi-Pot Spec	3000Vac 10mA 1 min. (I.P. to O.P.)
No load power consumption	0.3W max
Operating Temperature	0 to 40 degrees C
Storage Temperature	-20 to 80 degrees C
Operating Humidity	10% to 90%
Safety Approvals	TUV/VDE 60950-1 (UL)
EMC Standards	EN55022:2006+A1:2007 / EN6100-3-2 / EN6100-3-3
Pb-free	Yes RoHS Compliant
MTBF	50K Hours at 25 degrees C

See mechanical drawing and DC cable drawing on page 2.

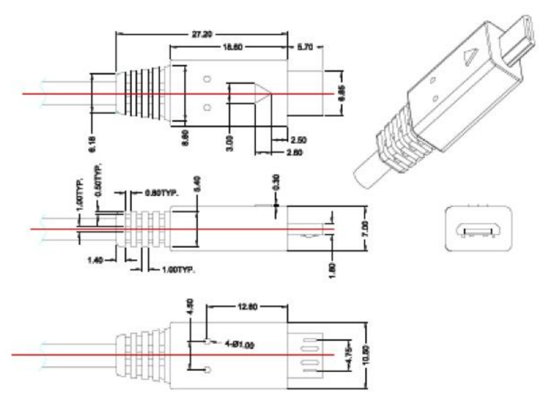
Sustainable Waste Sorter

Raspberry Pi Power Supply

Mechanical drawing:

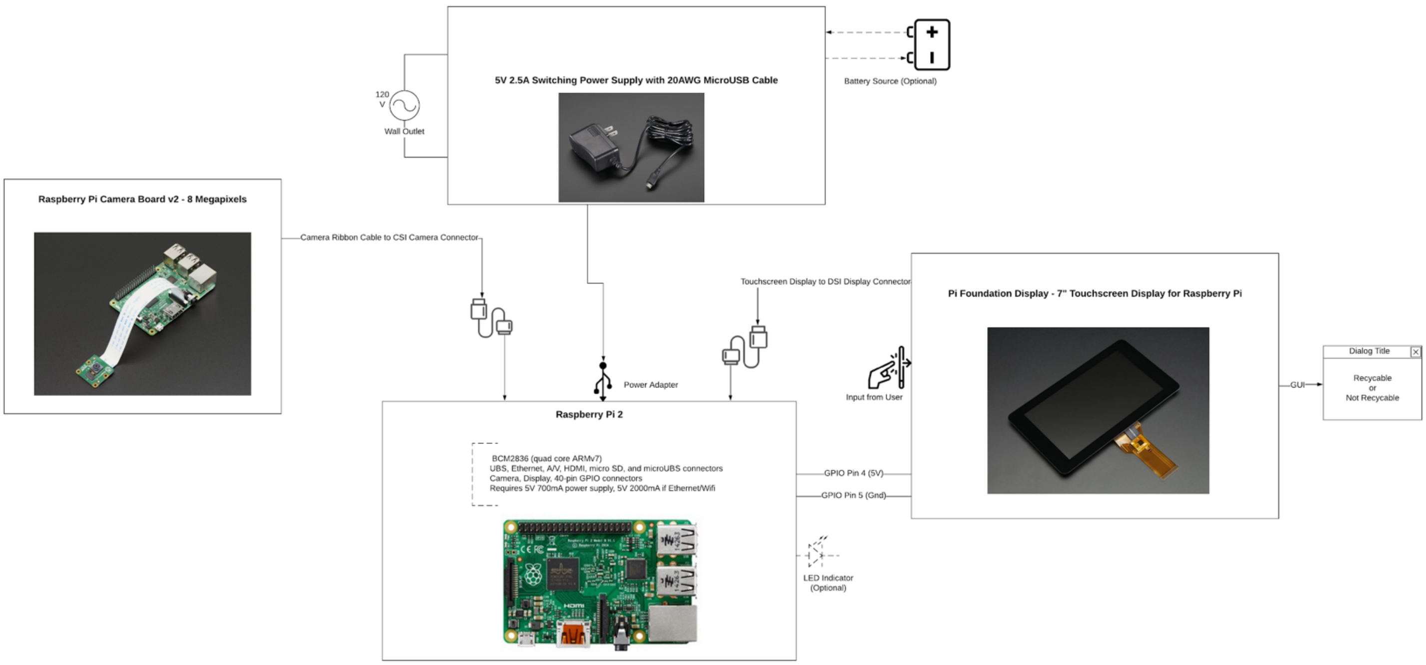


Output connector

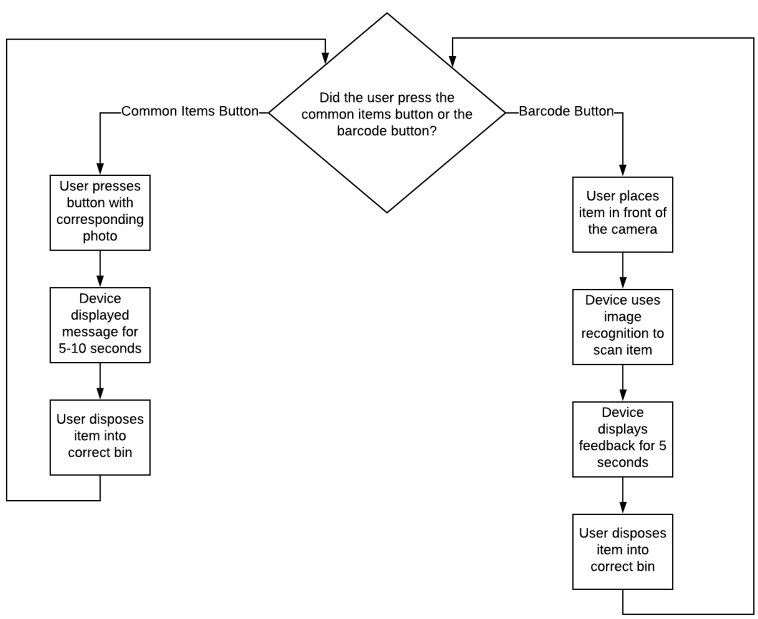


Sustainable Waste Sorter

Appendix E

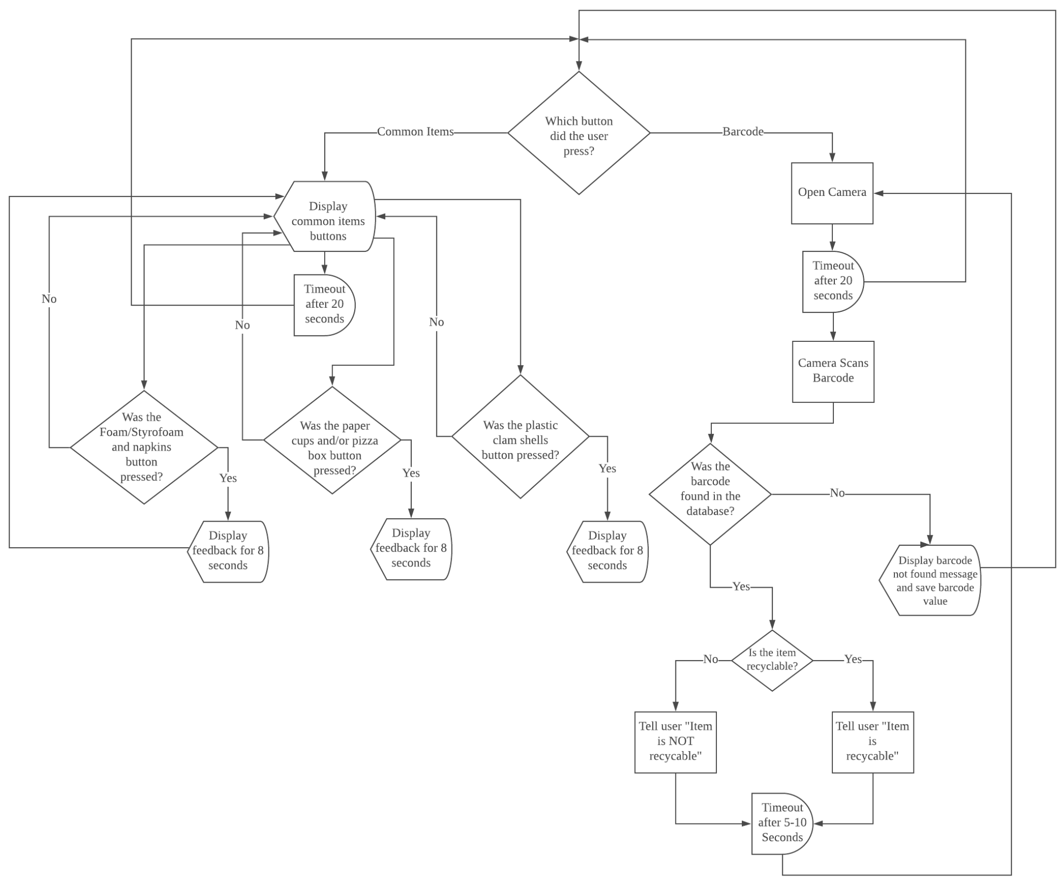


Appendix F



Sustainable Waste Sorter

Appendix G



Appendix H

Database: product_barcodes Table: barcode_information	
Attributes:	Data Type:
name	varchar(300)
barcode	varchar(300)
description	varchar(250)
isRecycable	tinyint(1)

Sustainable Waste Sorter

Appendix I

High-Level Decision Matrix					
Microcontroller					
Criteria	Weighting	Raspberry Pi 2 B		Raspberry Pi 3 B+	
		Score	Rating	Score	Rating
Cost	0.3	3	0.9	4	1.2
Compatibility	0.4	5	2	2	0.8
Processing Speed	0.3	3	0.9	5	1.5
Total			3.8		3.5
Touchscreen					
Criteria	Weighting	Raspberry Pi Touchscreen 7"		Raspberry Pi Touchscreen 5"	
		Score	Rating	Score	Rating
Cost	0.3	5	1.5	2	0.6
Size	0.2	4.5	0.9	4	0.8
Resistivity	0.5	3.5	1.75	5	2.5
Total			4.15		3.9
Image Processor/Barcode					
Criteria	Weighting	Adafruit Barcode Reader		Camera	
		Score	Rating	Score	Rating
Cost	0.35	3	1.05	5	1.75
Compatiblity	0.5	2	1	4	2
Size	0.15	4	0.6	4.5	0.675
Total			2.65		4.425
Casing					
Criteria	Weighting	Plastic		Metal	
		Score	Rating	Score	Rating
Weight	0.25	5	1.25	3.5	0.875
Cost	0.2	4	0.8	3	0.6
Conductivity	0.1	4	0.4	4	0.4
Durability	0.35	3.5	1.225	5	1.75
Total			3.675		3.625
Total = weighting * score					

Sustainable Waste Sorter

Appendix J

```python
from Tkinter import *

import SelectBarcode
import SelectNoBarcode

# Constants
WHITE = "#ffffff"
GRAY = '#797D7F'
FONT = "Constantia"
FONT_BUTTONS = "Constantia 14 bold"
FONT_HEADER = (FONT, 36)


def noBarcode():
    SelectNoBarcode.NoBarcode.createWindow()


def barcode():
    SelectBarcode.Barcode.createWindow()


def main():
    window = Tk()
    window.title("Main Window")
    window.configure(bg=GRAY)
    window.bind("<q>", lambda e: window.destroy())
    window.overrideredirect(True)
    window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(), window.winfo_screenheight()))

    label = Label(window, text="Not sure if it can be recycled? \n Find out here!", font=FONT_HEADER, bg=GRAY,
                  fg=WHITE)
    label.place(x=45, y=50)

    barcode_button = Button(window, text="Press Here to scan a barcode!", font=FONT_BUTTONS, command=barcode)
    barcode_button.place(x=120, y=205, width=190, height=215)

    no_barcode_button = Button(window, text="Common items\nfound in this\n cafeteria.\n\nPress here!",
                               font=FONT_BUTTONS, command=noBarcode)
    no_barcode_button.place(x=500, y=205, width=190, height=215)

    close_button = Button(window, text="X", font="Constantia 14", width=1, height=1,
                          command=lambda: window.destroy())
    close_button.place(x=758, y=5)

    window.mainloop()


main()
```

Sustainable Waste Sorter

Appendix K

```
import cv2
import Tkinter as tk
from Tkinter import *
from ttk import Frame
from PIL import Image, ImageTk
from pyzbar.pyzbar import decode
import argparse
import datetime
import mysql.connector
import ResultFunctions

# Constants
WHITE = "#ffffff"
GRAY = '#797D7F'
FONT = "Constantia"
FONT_BUTTONS = (FONT, 12)
FONT_HEADER = (FONT, 28)
CATEGORY1 = 0
CATEGORY2 = 0
CATEGORY3 = 0
CATEGORY4 = 0


class Barcode:

    def __init__(self):
        pass

    @staticmethod
    def createWindow():
        # initialize communication with database
        mydb = mysql.connector.connect(
            host="127.0.0.1",
            user="root",
            password="password"
        )
        mycursor = mydb.cursor()
        mycursor.execute("use product_barcodes")

        # construct the argument parser and parse the arguments
        ap = argparse.ArgumentParser()
        ap.add_argument("-o", "--output", type=str, default="new-barcodes.csv",
                        help="path to output CSV file containing barcodes")
        args = vars(ap.parse_args())

        csv = open(args["output"], "w")
        found = set()

        # Graphics window
        window = tk.Toplevel()
        window.title("Sustainability")
        window.configure(bg=GRAY)
```

Sustainable Waste Sorter

```
    window.bind("<q>", lambda e: window.destroy())
    window.overrideredirect(True)
    window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(), window.winfo_screenheight()))

    # Create Labels and buttons
    label = Label(window, text="Position Barcode in Front of the Camera", font=FONT_HEADER, bg=GRAY, fg=WHITE)

    # Place widgets into the canvas
    label.place(x=30, y=50)

    closeButton = Button(window, text="X", font=FONT_BUTTONS, width=1, height=1, command=lambda: window.destroy())
    closeButton.place(x=758, y=5)

    mainFrame = Frame(window)
    mainFrame.place(x=150, y=150)

    # Capture video frames
    showBarcode = tk.Label(mainFrame, width=500, height=250)
    showBarcode.pack(pady=0, padx=0)

    def readBarcode():
        # read the current frame
        cap = cv2.VideoCapture()
        ok, frame = cv2.VideoCapture(0).read()

        if ok:
            # convert to color
            cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
            img = Image.fromarray(cv2image).resize((500, 250))
            barcodes = decode(img)

            # decode barcode
            for barcode in barcodes:
                barcodeData = barcode.data.decode('utf-8')

                # stop once a barcode is read
                if barcodeData == barcodeData:
                    cv2.VideoCapture(0).release()

                    # get data from database OR search database with barcode found
                    query = "SELECT * FROM product_barcodes.barcode_information WHERE barcode = %s"
                    barcodeData = (barcodeData,)

                    mycursor.execute(query, barcodeData)
                    search = mycursor.fetchall()
                    count = mycursor.rowcount

                    if count == 0:
                        barcode = barcode.data.decode('utf-8')
                        if barcode not in found:
                            csv.write("{},{}\n".format(datetime.datetime.now(), barcode))
                            csv.flush()
```

Sustainable Waste Sorter

```
                        found.add(barcodeData)
                    ResultFunctions.ResultFunctions.notFound()
                    global CATEGORY4
                    CATEGORY4 += 1
                    print "Not Found called {} time(s)".format(CATEGORY4)

                for result in search:
                    if (result[2] == "plastic" or result[2] == "metal" or result[2] == "glass") and result[3] == 1:
                        ResultFunctions.ResultFunctions.glassMetalsPlastics()
                        global CATEGORY1
                        CATEGORY1 += 1
                        print "Glass/Metal/Plastic called {} time(s)".format(CATEGORY1)
                        break
                    elif result[2] == "paper" and result[3] == 1:
                        ResultFunctions.ResultFunctions.foundPaper()
                        global CATEGORY2
                        CATEGORY2 += 1
                        print "Paper called {} time(s)".format(CATEGORY2)
                        break
                    elif result[3] == 0:
                        ResultFunctions.ResultFunctions.notRecycable()
                        global CATEGORY3
                        CATEGORY3 += 1
                        print "Trash called {} time(s)".format(CATEGORY3)
                        break

        # display frame
        imgTk = ImageTk.PhotoImage(image=img)
        showBarcode.imgTk = imgTk
        showBarcode.configure(image=imgTk)
        showBarcode.after(5, readBarcode)

    readBarcode()
    window.mainloop()
```

Sustainable Waste Sorter

Appendix L

```python
import Tkinter as tk
from Tkinter import *
from PIL import ImageTk
import ResultFunctions

# Constants
WHITE = "#ffffff"
GRAY = '#797D7F'
FONT = "Constantia"
FONT_BUTTONS = (FONT, 12)
FONT_SMALL = (FONT, 8)
FONT_HEADER = (FONT, 25)
FONT_RESULTS = (FONT, 24)
FONT_NOT_FOUND = (FONT, 18)


class NoBarcode:
    def __init__(self):
        pass

    @staticmethod
    def createWindow():
        # Graphics window
        window = tk.Toplevel()
        window.title("No Barcode")
        window.configure(bg=GRAY)
        window.bind("<q>", lambda e: window.destroy())
        window.overrideredirect(True)
        window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(), window.winfo_screenheight()))

        mainLabel = Label(window, text="Which item do you have?",
                          font=FONT_HEADER, bg=GRAY, fg=WHITE)
        mainLabel.place(x=200, y=30)

        backButton = Button(window, text="Back", font=FONT_BUTTONS, width=5, height=1,
                            command=lambda: window.destroy())
        backButton.place(x=700, y=420)

        plastic = Button(window, command=ResultFunctions.ResultFunctions.pressedPlastics)
        photo1 = ImageTk.PhotoImage(file='/Users/elisabethgarza/Desktop/roche-capstone-project/images/plastics.jpg')
        plastic.config(image=photo1, width=210, height=130)
        plastic.place(x=290, y=320)

        cups = Button(window, command=ResultFunctions.ResultFunctions.pressedCups)
        photo2 = ImageTk.PhotoImage(file='/Users/elisabethgarza/Desktop/roche-capstone-project/images/cups.jpg')
        cups.config(image=photo2, width=215, height=165)
        cups.place(x=440, y=120)

        plates = Button(window, command=ResultFunctions.ResultFunctions.pressedPlates)
        photo3 = ImageTk.PhotoImage(file='/Users/elisabethgarza/Desktop/roche-capstone-project/images/plates.jpg')
        plates.config(image=photo3, width=210, height=165)
        plates.place(x=140, y=120)

        window.after(20000, lambda: window.destroy())
        window.mainloop()
```

Appendix M

```
import Tkinter as tk
from Tkinter import *
from PIL import ImageTk

# Constants
WHITE = "#ffffff"
GRAY = '#797D7F'
FONT = "Constantia"
FONT_BUTTONS = (FONT, 12)
FONT_SMALL = (FONT, 8)
FONT_HEADER = (FONT, 25)
FONT_RESULTS = (FONT, 24)
FONT_NOT_FOUND = (FONT, 18)


class ResultFunctions:
    def __init__(self):
        pass

    @staticmethod
    def glassMetalsPlastics():
        window = tk.Toplevel()
        window.title("Result")
        window.configure(bg=GRAY, cursor='none')
        window.overrideredirect(True)
        window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(), window.winfo_screenheight()))

        label = tk.Label(window,
                         text="This item is recyclable!\nPlace in Glass/Metals/Plastics bin.\nPlease empty all "
                              "contents into the TRASH bin.",
                         font=FONT_RESULTS, fg=WHITE, bg=GRAY)
        label.place(x=30, y=40)

        canvas1 = tk.Canvas(window, width=310, height=230)
        canvas1.place(x=250, y=180)
        window.photo = photo1 = ImageTk.PhotoImage(
            file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/metals.jpg')
        canvas1.create_image(156, 116, image=photo1)

        window.after(5000, lambda: window.destroy())

    @staticmethod
    def foundPaper():
        window = tk.Toplevel()
        window.title("Result")
        window.configure(bg=GRAY, cursor='none')
        window.overrideredirect(True)
        window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(),
                                        window.winfo_screenheight()))
        label = tk.Label(window, text="This item is recyclable! \nPlace in Paper Products bin.",
                         font=FONT_RESULTS, fg=WHITE, bg=GRAY)
        label.place(x=170, y=40)

        canvas1 = tk.Canvas(window, width=312, height=259)
        canvas1.place(x=235, y=145)
        window.photo = photo1 = ImageTk.PhotoImage(
            file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/paper.jpg')
        canvas1.create_image(156, 130, image=photo1)

        window.after(5000, lambda: window.destroy())
```

Sustainable Waste Sorter

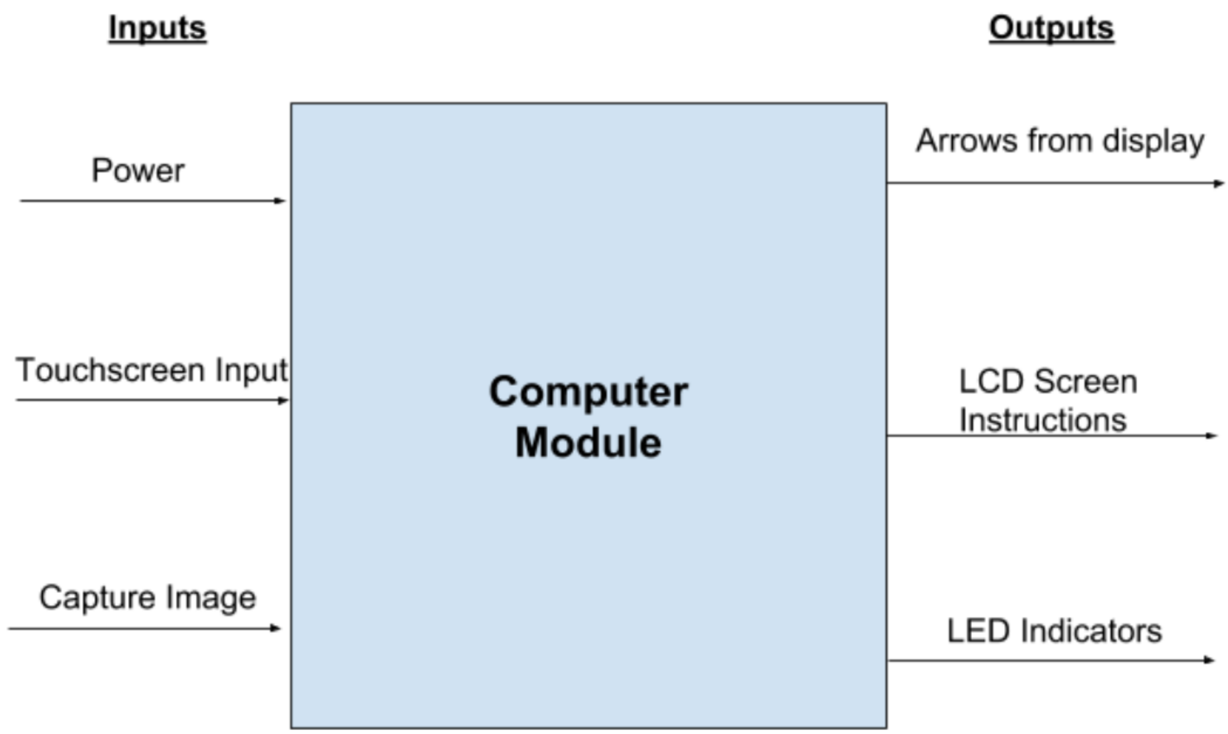
```
@staticmethod
def notRecycable():
    window = tk.Toplevel()
    window.title("Result")
    window.configure(bg=GRAY, cursor='none')
    window.overrideredirect(True)
    window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(),
                                   window.winfo_screenheight()))
    label = tk.Label(window, text="This item is NOT recycable! \nPlace in Food/Trash bin.",
                     font=FONT_RESULTS, fg=WHITE, bg=GRAY)
    label.place(x=180, y=30)

    canvas1 = tk.Canvas(window, width=313, height=266)
    canvas1.place(x=245, y=140)
    window.photo = photo1 = ImageTk.PhotoImage(
        file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/trash.jpg')
    canvas1.create_image(156, 135, image=photo1)

    window.after(5000, lambda: window.destroy())

@staticmethod
def notFound():
    window = tk.Toplevel()
    window.title("Result")
    window.configure(bg=GRAY, cursor='none')
    window.overrideredirect(True)
    window.geometry("%dx%d+0+0" % (window.winfo_screenwidth(),
                                   window.winfo_screenheight()))

    label = tk.Label(window, text="We did not find this item in our records."
                                  + "\nThis item's information has been saved and will be added."
                                  + "\n\n Not sure which recycle bin to place the item?" + "\n Place in the "
                                                                                           "Food/Trash bin.",
                     font=FONT_NOT_FOUND, fg=WHITE, bg=GRAY)
    label.place(x=50, y=170)
    window.after(8000, lambda: window.destroy())

@staticmethod
def pressedCups():
    message = tk.Toplevel()
    message.title("Cups")
    message.configure(bg=GRAY, cursor='none')
    message.bind("<q>", lambda e: message.destroy())
    message.overrideredirect(True)
    message.geometry("%dx%d+0+0" % (message.winfo_screenwidth(), message.winfo_screenheight()))

    label = Label(message,
                  text="Paper cups and pizza boxes may be placed in the Paper Products Bin. \n\nPlease empty "
                       "all contents into the trash bin BEFORE recycling.",
                  font="Constantia 16", bg=GRAY, fg=WHITE)
    label.place(x=30, y=60)

    canvas1 = tk.Canvas(message, width=313, height=266)
    canvas1.place(x=245, y=165)
    message.photo = photo1 = ImageTk.PhotoImage(
        file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/paper.jpg')
    canvas1.create_image(156, 135, image=photo1)

    message.after(8000, lambda: message.destroy())
```

Sustainable Waste Sorter

```
@staticmethod
def pressedPlates():
    message = tk.Toplevel()
    message.title("Plates")
    message.configure(bg=GRAY, cursor='none')
    message.bind("<q>", lambda e: message.destroy())
    message.overrideredirect(True)
    message.geometry("%dx%d+0+0" % (message.winfo_screenwidth(), message.winfo_screenheight()))

    label = Label(message,
                  text="Items that are styrofoam, plastic, and/or napkins are trash.\n\nPlease place in the "
                       "Trash bin.",
                  font="Constantia 18", bg=GRAY, fg=WHITE)
    label.place(x=40, y=60)

    canvas1 = tk.Canvas(message, width=313, height=266)
    canvas1.place(x=235, y=165)
    message.photo = photo1 = ImageTk.PhotoImage(
        file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/trash.jpg')
    canvas1.create_image(156, 135, image=photo1)

    message.after(8000, lambda: message.destroy())

@staticmethod
def pressedPlastics():
    message = tk.Toplevel()
    message.title("Plastics")
    message.configure(bg=GRAY, cursor='none')
    message.bind("<q>", lambda e: message.destroy())
    message.overrideredirect(True)
    message.geometry("%dx%d+0+0" % (message.winfo_screenwidth(), message.winfo_screenheight()))

    label = Label(message,
                  text="Items that are foam (also known as 'Styrofoam'), and napkins are trash.\n\nAdditionally, "
                       "some mixed plastic items, such as plastic cutlery and plastic lids, are trash.\n\nPlease "
                       "place in the Trash bin",
                  font="Constantia 14", bg=GRAY, fg=WHITE)
    label.place(x=5, y=60)

    canvas1 = tk.Canvas(message, width=313, height=266)
    canvas1.place(x=245, y=165)
    message.photo = photo1 = ImageTk.PhotoImage(
        file=r'/Users/elisabethgarza/Desktop/roche-capstone-project/images/metals.jpg')
    canvas1.create_image(156, 135, image=photo1)

    message.after(8000, lambda: message.destroy())
```

Sustainable Waste Sorter

Appendix N



Appendix O