BinBot

Software Development Plan

REVISION HISTORY

Revision #	Author	Revision Date	Comments
1.0	Michael Savitski	September 28, 2019	Outline
1.1	Sean DiGirolamo	September 29, 2019	Development & VC
1.2	Michael Savitski	September 29, 2019	Activities & Schedule
1.3	Sean Reddington	September 29, 2019	Tasks, updated Gantt Chart, and various revisions
2.0	Sean Reddington	October 17, 2019	Added System overview section, logo, updated gantt chart, milestone demo 1
2.1	Sean Reddington	October 19, 2019	Remaining milestone demos, formatting and clean up before resubmission

Table of Contents

System Overview

BinBot is a waste-collection robot intended to patrol specific areas such as university grounds, stadiums, boardwalks, or schools. BinBot will identify waste that is laying on the ground and collect it to be disposed of properly. BinBot will have a camera module and on-board microcontroller unit that communicates with a server via wi-fi. To outsource the heavy data processing from the on-site robot; a Linux server will process the images sent by BinBot using data representations created with a deep learning algorithm to identify pieces of waste. The server will then inform BinBot of information regarding the photos, such as if any waste has been identified, and if so, how far it is and how BinBot should navigate to the waste. The waste will then be collected using a mechanical arm and place it in a waste bin attached to the robot. The image feed from BinBot, with additional visual indicators of identified waste, can be transmitted to a mobile application for observing BinBot's progress and success.

One of the key components for BinBot to function as needed will be deep learning software for training a neural network model, so that BinBot's other software will be able to identify waste objects in the images from its camera module. This will entail the use of the OpenCV library for processing images, and the use of the open-source deep learning library TensorFlow. Ideally, this separate software component will be run on a GPU supported computer system. The neural network data model will be trained using hundreds to thousands of images of waste objects, captured in the expected resolution of BinBot's camera. Also included in the training data with these images will be information such as camera height, object size, and object distance, so that BinBot will be able to make estimations to allow it to traverse to the waste objects and successfully collect them. After feeding images to the neural network, the neural network will be tasked with returning whether or not waste is located in the image, and if so, how far the waste is and at what angle it is from the robot.

The completed data model will be applied to software which will run on a server along with a communication socket service, which BinBot will continuously connect to via wi-fi. The images continuously collected by BinBot's camera module will be sent to the server for processing via the OpenCV library, which will use the trained neural network data model to identify waste objects. Once the waste has been identified, as well as the distance and angle, movement instructions will then be calculated and sent back to BinBot's on-board computer board so that it may travel to the nearby waste objects and pick them up. For the purposes of this simple project, BinBot will simply be instructed to turn towards the waste, and travel to it in a straight line.

BinBot's on-board computer will be an Arduino Mega 2560 board with limited functionality. It will be to collect images from the camera module and then transmit them to the data processing API. Receiving data back from the data processing API about waste objects in BinBot's camera's view, the primary purpose of the board will be to operate BinBot's robotic components. BinBot will have robotic treads for traversing across the floor to approach waste objects, as well as a robotic arm for collecting the objects. The software running on the board will need to use the constantly updated information about distance and size of identified objects to operate these components efficiently and successfully collect waste.

Additionally, a companion mobile application will be developed for users of BinBot to install on Android devices. This companion application will be able to receive images from the data processing server which will allow the user to watch a live feed of BinBot's camera view. The server will modify these images visually to have boxes around identified waste objects, and text about BinBot's current progress in its process of collecting the objects such as the estimated distance and status messages like "traversing" or "collecting".

System Block Diagram

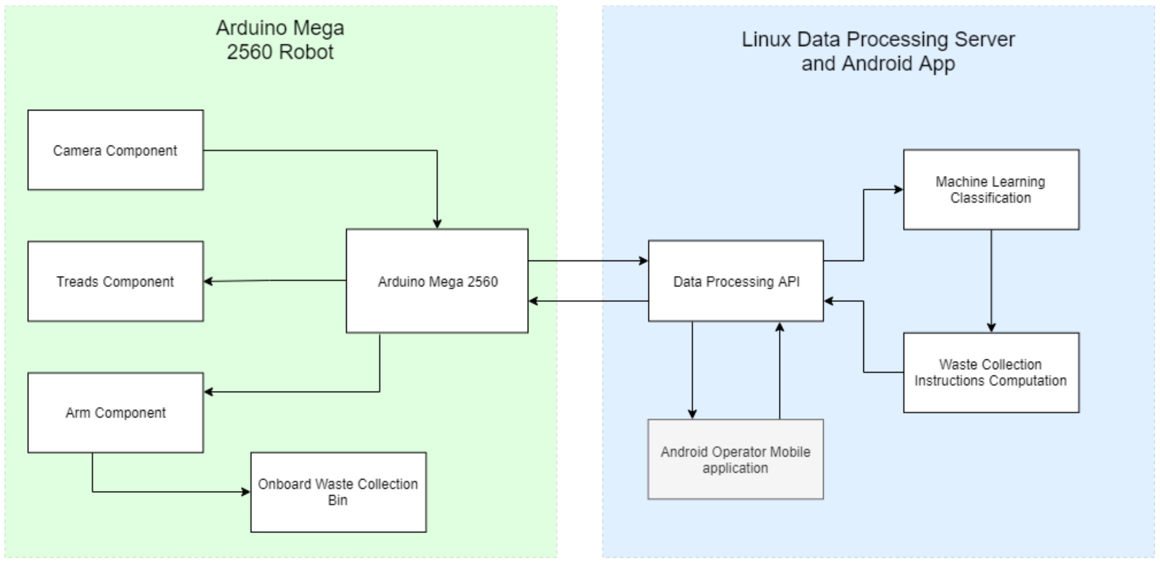

Arduino Mega 2560 Robot
Camera Component
Treads Component
Arduino Mega 2560
Arm Component
Onboard Waste Collection Bin
Linux Data Processing Server and Android App
Machine Learning Classification
Data Processing API
Waste Collection Instructions Computation
Android Operator Mobile application

Fig 1. Block diagram illustrates the flow of data from the Camera feed to BinBot disposing of the waste.

The block diagram shows a high level overview of BinBot project's system architecture. Each of the hardware components on the robot kit will have a corresponding controller interface in the Sketch program loaded onto the Arduino Mega 2560. The Arduino Mega 2560 will exchange data with the Linux server over Wi-Fi via a data transmission interface. The Linux server will be hosting a data processing API that will pass BinBot's camera feed to the machine learning classification processing; passing the results to the Android operator mobile application. If there is collectable waste, the server will then computer instructions for the robot to collect the target waste object and send it back to the Arduino board. The Arduino will then pass the instructions to the tread and arm interfaces to collect the object, placing it into an onboard waste collection bin.

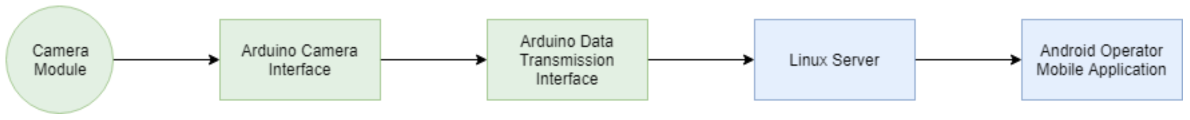

Camera Module
Arduino Camera Interface
Arduino Data Transmission Interface
Linux Server
Android Operator Mobile Application

Fig 2. Diagram displaying interaction between BinBot's Camera Module, Linux Server, and Android Application.

The camera module will capture images at 1 fps via the Camera Interface. This image feed will be passed to the Arduino's Data Transmission interface which will send the images to the Linux server to be processed. Along with the data processing, the image feed will be passed to the Android operator mobile application for monitoring and manual intervention of BinBot during testing.

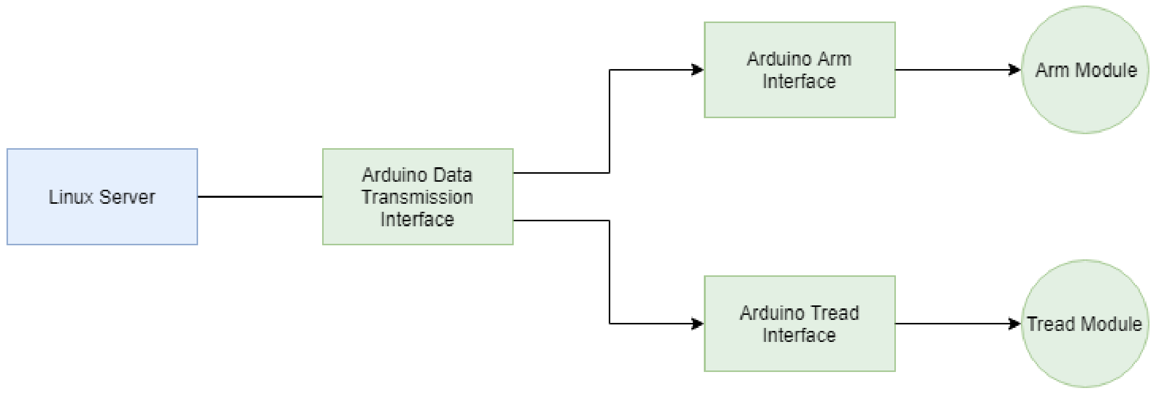


Fig 3. Diagram displaying interaction between the Linux Server and the Arduino's interfaces in order to operate the robot.

The Linux Server will exchange data with the Arduino over Wi-Fi through the Arduino's Data Transmission interface. This includes computed instructions for BinBot to travel to or collect the target object. The Arduino will then pass the instructions to the corresponding interface. These interfaces will provide the functions to move the arm or tread modules.

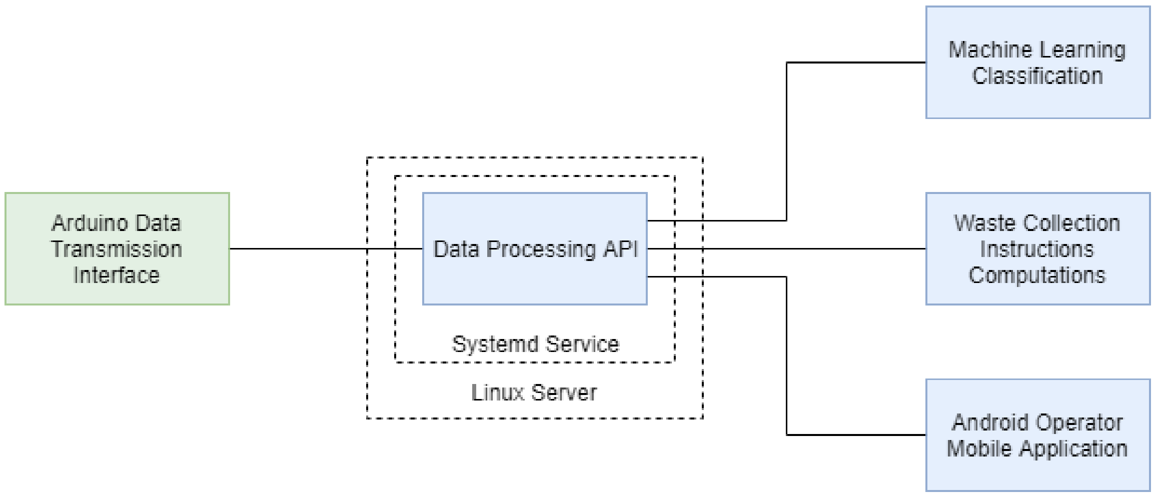

Arduino Data Transmission Interface
Data Processing API
Systemd Service
Linux Server
Machine Learning Classification
Waste Collection Instructions Computations
Android Operator Mobile Application

Fig 4. Diagram displaying the interaction between the Arduino's Data Transmission Interface, Linux data processing server, and Android Application.

The Arduino Data Transmission interface will send the image feed to the Linux processing server. The proposed method is to host a data processing API application that will execute via Systemd services. The data processing API will relay the image feed to the machine learning classification processing. If the machine learning model identifies collectable waste, instructions for the robot kit to collect the waste will be computed. These instructions include navigation instructions for the tread module and collection instructions for the arm module. The image including the results of the object classification will also be sent to the Android mobile application.

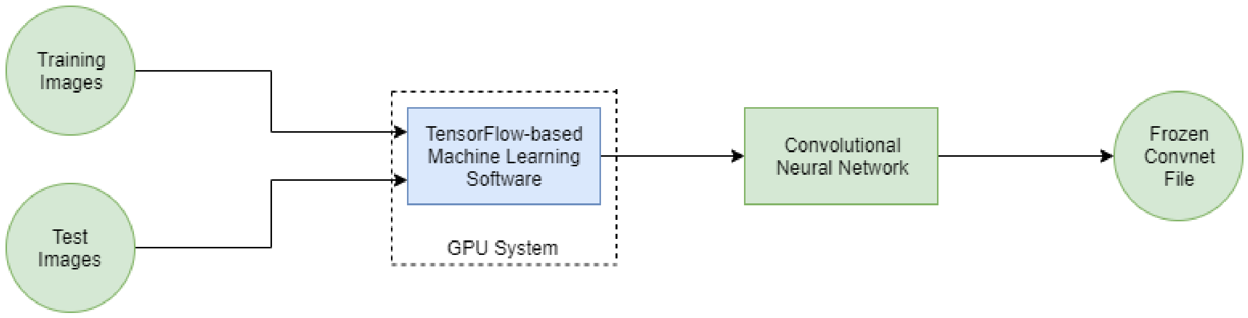

Training Images
Test Images
TensorFlow-based Machine Learning Software
GPU System
Convolutional Neural Network
Frozen Convnet File

Fig 5. Diagram illustrating the process of training the neural network data model that will enable BinBot to recognize objects.

The machine learning algorithm will take as input approximately one thousand images for each trash objects we want to make BinBot's image recognition able to identify. Running on a GPU system and implementing Google's TensorFlow API, the software will train a convolutional neural network data model. There will also be several hundred images reserved as a test data set, to verify the neural network's success on data it was not trained on. This model can then be exported, a process called "freezing", to be implemented in the server-side OpenCV implementation.

ACTIVITIES

Research

The first and most primary concern for beginning the project is research into several topics. A very important topic for the project's success is that of machine learning. To that end, we will be utilizing the TensorFlow tutorials (https://www.tensorflow.org/), as well as the book upon which much of the material in those tutorials is based, "Deep Learning with Python" by Francois Chollet. Through this research we will gather the necessary understanding of how a deep learning and networks work and can be implemented for our task of identifying waste objects.

The next most important research for our success will be that of using the Arduino board and development environment, specifically to operate a camera module, send and receive data via the wi-fi module, and operate the robotic components by controlling the voltage sent to them. It is especially important for us to gain an understanding of this before we can begin development, as we might not realistically have access to the robot as early as we might like to.

We must also research into how we will integrate the robot, machine learning processing, and mobile app using a Linux server. We must decide the best methods to transmit real-time data streams between each of these components. While we have some experience with webserver tools and frameworks such as RabbitMQ, Pika, Flask, etc.; our goal is to find a simpler and more "low-level" way without the need of a large web application server.

The final aspects of our research are development oriented. Though members of our team have varying levels of experience with them, some research will need to be done in the implementation of Android mobile applications, a Linux server, and streaming an image feed.

Requirements Gathering

There are two primary requirements that must be gathered for BinBot. The first of which is, of course, BinBot itself. We have researched several options for a simple robot kit that will meet our needs. As none of us has much experience in mechanical engineering, a self-contained kit with instructions was a crucial choice. Once acquired, we will need to build the robot and get it operational.

The second requirement will be the gathering of our images to train the neural network for identifying waste. It is our intention to capture hundreds to thousands of individual images of waste objects in a controlled space, making sure the size of the object, height of the camera, and distance from the camera of the object are recorded as well. This will be done using a DSLR camera we have access to.

The third requirement will be coordinating the use of professor Dr. Charlie Wang's Linux server to host our heavy data processing. While we have access to less powerful Linux machines, we hope we can incorporate the use of the Nvidia GPU into our machine learning model training.

Top-Level Design

The top-level design for BinBot consists of four separate elements. The first is BinBot itself, which will be a small-scale simple robot with treads for movement, a mechanical arm, and an on-board Arduino microcontroller. The Arduino will be programmed to perform the necessary tasks of operating BinBot's camera, its mechanical components, and communication between itself and the server.

The second element is the mobile application. The primary purpose of this application will be to receive images from the server sent by BinBot, marked to show waste objects that have been successfully identified using the neural network data model. The application will also have a very simple interface for making BinBot perform basic movements. This application will be crucial to our ability to test and demo BinBot's functionality.

The third element will be our server, which will perform the object identification processing and transmit information and commands to BinBot to collect the waste objects. It will also act to convey the image feed from BinBot by storing the images sent by BinBot's camera, marking identified objects, and sending the modified images to the mobile application.

The last element of the design is the deep learning implementation. This is how we will produce a usable neural network data model that the server software will use to identify waste objects. It is our intent to utilize the TensorFlow open-source machine learning library and a GPU system for this aspect of the design. We will design software that can take the input of our waste object images, as well as the associated distance and size data, and train the neural network with these.

Detailed Design Planning

BinBot's various components, spread across different components and devices, will prove a challenge to successfully implement in tandem and ensure good communication between the parts that must communicate. To this end, detailed planning will be required. First, we will need to compose class diagrams of all our intended functionality for the individual components. A part of this will be designed methods by which the different components, all existing on different hardware and many written in differing programming languages, will be able to communicate with each other successfully. To ensure these communication interfaces are designed properly, we will also need to create a component diagram that properly maps all potential communications between the components of the BinBot system.

Testing

As is always the case in software development, testing will be crucial to our success. We will need to find ways to incrementally test and determine the stability of our features as we develop them, and ensure the individual components work together correctly. Much of our process will require exhaustive real-time testing, so most of our testing will not be automated. The process will generally be to develop on aspect of a feature, test it for stability, and then test for stability of all existing elements after the new element is implemented.

Tasks

BinBot Robot

The initial requirements for the robotics part of the project consist of getting approval from the depart for the Xiaorgeek robot kit to be provided to our group and receiving the robotics hardware. Since the robotics kit consists of individual pieces, the robot will have to be manually built by the group. After successfully building the robot kit, our first goal is to load a simple script onto its Arduino demonstrating basic functionality such as tread and arm movement to confirm the robot was built correctly. The next steps will be to build out interfaces for each of the robot's components. This includes interfaces for the treads, arm, camera, power supply management, and data transmission to the Linux server. Our first priority will be the camera and data transmission functionality so that these interfaces can be provided for the machine learning and mobile application parts of the project early on. This includes successfully setting up a network socket connection over Wi-Fi to send frames captured by the camera to the Linux server in real-time. Once the robot can exchange data from the Linux server, the final goals are to implement the functionality for the Arduino to receive instructions from the server on whether or not waste has been identified in an image and to navigate and retrieve that waste via the tread and arm interfaces.

BinBot Server

In order for the robot to communication with the machine learning software and mobile application, a processing server is required. Once we decide on the methods of exchanging data as mentioned in the research section above, we will need to configure a Linux server and implement several services to be ran on it. First, we will need to write software that will send and receive data in real-time from both the Arduino board and the mobile operator testing application. Next is to host the machine learning software onto the server to abstract the heavy data processing from the robot's onboard Arduino microcontroller. Finally, we will need to design an application that can take data from the machine learning model and convert it into simple instructional input for the robot. This includes computing the location of the target object from the robot and dynamically generating tread and arm instructions to collect it. We foresee this as one of the more difficult tasks for the BinBot project.

BinBot App

For testing and demo purposes, we must build out an Android application to relay instructions to the robot. Our first goal is having the app send and receive data to the server. This starts with receiving camera feed from the server. This data feed will include the machine learning model's classification results, in which we will include *accept/reject* buttons to provide human feedback to the model's live processing. We will finally implement simple results feeds from the server, slowly removing required human operation of BinBot. From there we will implement simple commands to the robot to demonstrate early functionality of the robot's components. This will include commands such as start, stop, directional tread and arm movements, etc. From there, we will shift focus to

Machine Learning Software

After identifying the tools and machine learning model to use for waste recognition, we will work on the image training software. If we are ahead of schedule with the machine learning processing, we will try to integrate the use of the Nvidia GPU for training. We will then focus on training the model with specific object types as followed:

Waste Object 1 - Train neural network to identify paper scraps and return their distance and angle by feeding the network images of waste and their distance and angle from the camera.

Waste Object 2 - Train neural network to identify aluminum cans and return their distance and angle by feeding the network images of waste and their distance and angle from the camera.

Waste Object 3 - Train neural network to identify chip bags and return their distance and angle by feeding the network images of waste and their distance and angle from the camera.

Waste Object 4 - Train neural network to identify plastic bags and return their distance and angle by feeding the network images of waste and their distance and angle from the camera.

Waste Object 5 - Train neural network to identify small cardboard boxes and return their distance and angle by feeding the network images of waste and their distance and angle from the camera

MILESTONE DEMONSTRATIONS

Demo 1

For the BinBot team's first demo, we will be focusing on the machine learning and waste identifying/classification features of BinBot. This will include TensorFlow implementation of the machine learning model training done in the *BinBot_MachineLearning* repository as well as the OpenCV object classification portion done in the *BinBot_Processing* repository. Both of these repositories can be found on the capstone course's GitHub page.

The machine learning model will have trained on images taken by the BinBot team and preexisting data provided by TensorFlow. The training should then allow the object classification to detect up to 5 types of waste. This includes paper scraps, crushed cans, chip bags, plastic bags, and small cardboard boxes. The object classification will be demonstrated by feeding the model images containing these waste types. The results will be a modified version of the image containing drawn bounding boxes surrounding which objects it detected, as well as a label describing the type of waste the object was classified as.

Demo 2

The second demo for BinBot will demonstrate mobility features of the robot kit. These features will include most of the Arduino Sketch implementations found in the *BinBot_ArduinoSketch* repository as well as the integration required for the data processing server and Android operator mobile application also found in the *BinBot_Processing* and *BinBot_OperatorMobileApp* repositories. All of these repositories can be found on the capstone course's GitHub page.

The main focus of this demo is to show that the robot kit was successfully built and functioning. This includes simple movements of the robot's treads and arm mechanisms via instructions sent from the processing server. The demo will also include the ability to control the robot via the Android application. The app will have start and stop buttons, which when pressed, will send a command to the processing server; instructing the Arduino to change the robot to a *collecting* or *standby* state.

Demo 3

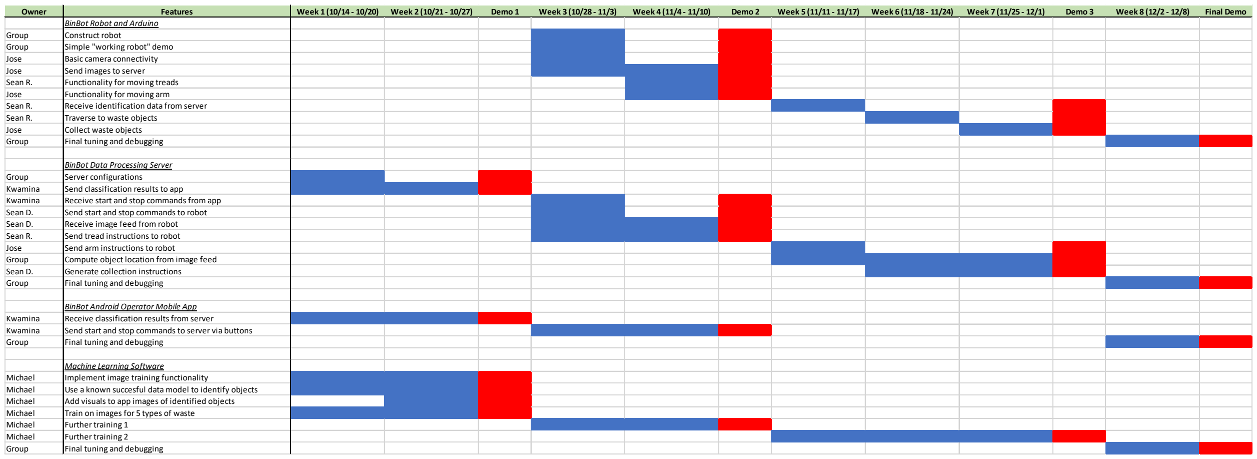
The third demo will feature the waste collecting abilities of BinBot. This will require unified integration between all of the BinBot components to realize its overall purpose. The main requirements for this demo revolve around computing the waste collection instructions for BinBot. BinBot will send a continuous image feed to the processing server to identify any waste in its surroundings. If waste has been found, the processing server will compute the location and size of the target waste based off the results found by object identification machine learning model. Then the server will translate this into instructions for BinBot to move to the target and collect it.

Demo Final

BinBot Team's final demo will showcase the final version of the BinBot capstone prototype. This will consist of any fine tuning, fixed bugs, or features that were not completed by their original deadline. This demo will be similar to the previous while featuring any additional updates or fixes that were missing in previous demos.

Schedule

For a better view of the development schedule, please see the accompanying Excel spreadsheet from which this image was captured. This Gantt chart represents our projected completion goals for each feature. Theses features will be implemented through weekly sprints with specific features being ready by each of the listed demo dates.



Owner	Features	Week 1 (10/14 - 10/20)	Week 2 (10/21 - 10/27)	Demo 1	Week 3 (10/28 - 11/3)	Week 4 (11/4 - 11/10)	Demo 2	Week 5 (11/11 - 11/17)	Week 6 (11/18 - 11/24)	Week 7 (11/25 - 12/1)	Demo 3	Week 8 (12/2 - 12/8)	Final Demo
	BinBot Robot and Arduino												
Group	Construct robot				🟦		🟥						
Group	Simple "working robot" demo				🟦		🟥						
Jose	Basic camera connectivity				🟦		🟥						
Jose	Send images to server				🟦	🟦	🟥						
Sean R.	Functionality for moving treads					🟦	🟥						
Jose	Functionality for moving arm					🟦	🟥						
Sean R.	Receive identification data from server							🟦			🟥		
Sean R.	Traverse to waste objects								🟦		🟥		
Jose	Collect waste objects									🟦	🟥		
Group	Final tuning and debugging											🟦	🟥
	BinBot Data Processing Server												
Group	Server configurations	🟦		🟥									
Kwamina	Send classification results to app	🟦	🟦	🟥									
Kwamina	Receive start and stop commands from app				🟦		🟥						
Sean D.	Send start and stop commands to robot				🟦		🟥						
Sean D.	Receive image feed from robot				🟦	🟦	🟥						
Sean R.	Send tread instructions to robot				🟦	🟦	🟥						
Jose	Send arm instructions to robot							🟦			🟥		
Group	Compute object location from image feed							🟦	🟦	🟦	🟥		
Sean D.	Generate collection instructions								🟦	🟦	🟥		
Group	Final tuning and debugging											🟦	🟥
	BinBot Android Operator Mobile App												
Kwamina	Receive classification results from server	🟦	🟦	🟥									
Kwamina	Send start and stop commands to server via buttons				🟦	🟦	🟥						
Group	Final tuning and debugging											🟦	🟥
	Machine Learning Software												
Michael	Implement image training functionality	🟦	🟦	🟥									
Michael	Use a known succesful data model to identify objects	🟦	🟦	🟥									
Michael	Add visuals to app images of identified objects		🟦	🟥									
Michael	Train on images for 5 types of waste	🟦	🟦	🟥									
Michael	Further training 1				🟦	🟦	🟥						
Michael	Further training 2							🟦	🟦	🟦	🟥		
Group	Final tuning and debugging											🟦	🟥

Fig 6: BinBot team's Gantt chart

Development Environments

Hardware for BinBot

- GFS Robot Tank Car
- Motor Drive: L298P two high-power motor driver chip
- MCU: Arduino Mega 2560
- Working Voltage: 12V Lithium battery
- Signal range: <= 80M
- Camera: Second-generation HD camera with manual adjustment of focus

Software Environment

- Arduino programming language and IDE
- Mobile application programmed in the Android Studio IDE using Java and XML
- Linux server
- Server processing programmed using Java or Python (JetBrains's IDE)
- Neural Network using OpenCV and Tensorflow implemented using Python

Version Control

Version control will be handled with several GitHub hosted Git repositories for the mobile application, server, and Arduino software. Each repository will have a main master branch. Changes will be committed to other branches and will be merged into the master branch once they receive group approval.