SmartCart

Pagnapech Chamroeun
Donald Bren School of Information & Computer Science
Henry Samueli School of Engineering
Irvine, CA, USA
pjchamro@uci.edu

Randy Chhan
Henry Samueli School of Engineering
Irvine, CA, USA
rchhan@uci.edu

Christian Rodriguez
Henry Samueli School of Engineering
Irvine, CA, USA
chriser2@uci.edu

Gary Villar
Donald Bren School of Information & Computer Science
Samueli School of Engineering
Irvine, CA, USA
garyv1@uci.edu

Kaavian Murugadass
Donald Bren School of Information & Computer Science
Samueli School of Engineering
Irvine, CA, USA
kmurugad@uci.edu

***Abstract* —** *The prevailing model for distributing products to the general public heavily relies on carts prioritizing minimal production costs over user satisfaction. Our innovative solution, SmartCart, introduces autonomous functionality to transform this traditional utility into a luxury experience. By enhancing customer interaction with the cart, SmartCart seeks to improve the overall shopping experience significantly.*

1. Introduction

The shopping cart can help the customer to shop in-store quickly and conveniently. It will save the customers time when doing groceries, running errands, or buying many items in less time. The cart is an intelligent cart that returns after the customer finishes their groceries. It also displays the items in the cart using a built-in screen with their price and quantity. During the checkout, the customer can directly check out via the screen on the cart. When the customer empties the cart in the parking lot, it automatically returns to the stall cart. Possible avoidance system and cart suggestion AI. Obstacle avoidance uses an AI to have ultrasonic sensors to detect distances. In addition to these features, we will explore the issues and deficiencies of current smart cart systems and include fixes. Due to the overall cost of the entire system, we will also implement security

features to deter any would-be thieves.

By the end of the fall quarter, we identified the objects/items we planned to support, including serializing all pertinent information of the items our computer vision model detected. Our software could draw bounding boxes around the items, update a text file, and play sounds depending on the actions of placing or removing items from the cart. Battery design schematics were drawn up on the hardware side, and powertrain/chassis design and setup began.

Most of our accomplishments were done during the winter quarter, realizing our most important software and hardware goals. The software we completed displayed a graphical user interface connected to our backend that was capable of registering and logging in users. In addition, our GUI supported real-time updates to the items in our physical cart that were detected by computer vision mode stored in the database. For the cart system, the backend reliably updated the database with the items written in the text file. The backend primarily served up API endpoints to the front end for actions such as signing up/in and checkout functionality. The enclosure was 3D-printed and mounted to the motor chassis, and hardware was implemented seamlessly inside. In addition to the user-facing systems, the cart system is capable of mobile autonomous movement, tracking, and following a customer around an environment.

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Object Identification**					
1.1	Detect the first 5 grocery items	P, G	10/9/23	10/16/23	7	100%
1.2	Send Data to Server	P, K, G	10/16/23	11/23/23	37	100%
1.3	Efficiency Detection (70% to 80%)	P	10/16/23	11/23/23	37	100%
1.4	Detect Put in/ Take Out	P	10/23/23	11/30/23	37	100%
1.5	Detect Another 10 grocery items	C, R, P, K, G	11/6/23	11/23/23	17	100%
2	**Database**					
2.1	Create FrontEnd	K	10/16/23	11/23/23	37	100%
2.2	Create DataBase Server	G	10/23/23	11/30/23	37	100%
2.3	Create Cloud Computing/ Base Server	C, R, P, G	10/30/23	11/30/23	30	100%
3	**Controller/Base Station Communication**					
3.1	Setup RaspberryPi and Object Ident.	P	10/30/23	11/30/23	30	100%
3.2	Setup Tablet and FrontEnd	C, P, K, G	10/30/23	11/30/23	30	100%
3.3	Setup Power Supply for RPi and Tablet	C, R, P, K, G	10/30/23	11/30/23	30	100%
4	**Powertrain and Hardware**					
4.1	Model Systems and Develop Code Structure	C, R	11/13/23	11/27/23	14	100%
4.2	Test Motor and Controller Code Functionality	C, R	11/23/23	12/1/23	8	100%
4.3	Develop and Test Dynamic Infrared Sensor System	C, R	11/30/23	12/7/23	7	CANCELLED
4.4	Mount Drivetrain System	C, R	11/30/23	12/7/23	7	100%
4.5	Develop Battery Management System	C, R	11/30/23	12/7/23	7	100%
5	**Obstacle Avoidance**					
5.1	Test Simulation	TBD	01/10/24	01/20/24	10	50%
5.2	Setup Front and Back Camera	TBD	01/10/24	01/15/24	5	50%
5.3	Physical Cart Testing	TBD	01/20/24	01/31/24	11	100%
5.3.1	Phase 1: Physical Testing (Momentum)	TBD	02/01/24	02/07/24	6	100%
5.3.2	Phase 2: Put Barriers	TBD	02/07/24	02/10/24	3	CANCELLED
5.3.3	Phase 3: Build up Barriers	TBD	02/10/24	02/20/24	10	CANCELLED
5.4	Navigation and Control	TBD	02/20/24	03/01/24	11	75%
5.4.1	Phase 1: Initial Test Navigation	TBD	03/01/24	03/10/24	9	100%
5.4.2	Phase 2: Initial Test Control	TBD	03/10/24	03/15/24	5	100%
5.4.3	Phase 3: Step up Navigation	TBD	03/15/24	03/20/24	5	100%
5.4.4	Phase 4: Step up Control	TBD	03/21/24	03/25/24	4	100%
5.4.5	Phase 5: Press Button for Cart Return	TBD	03/25/24	03/30/24	5	CANCELLED

Table 1: This is our objective plan for Fall and Winter Quarter

2. Initial objectives

Our objective for the fall quarter was to have the product detection system operational and integrated with our project's other necessary software components. Our goal was to have an object detection machine learning model capable of detecting various products included in our database and have the application update a graphical user interface to display to the user the total items in the cart and the total price of these items.

Regarding the hardware, our objective was to have preliminary prototypes for the battery management systems and the powertrain.

Objectives for Fall Quarter

Product detection system

Object Identification

- Detect 5 grocery items
- Send data to the server
- Efficiency Detection (70% to 80%)
- Detect items put in/ take out

Controller/Base Station Communication

- Setup Raspberry Pi and Object Identification

Database

- Create FrontEnd (Graphic User Interface)
- Create Database server

Objectives for Winter Quarter

Based on our current progress from the fall quarter our revised goals.

Database

- Create Base Server (locally hosted website on Raspberry Pi)

Controller/Base Station Communication

- Setup Tablet and FrontEnd
- Setup Power Supply for Raspberry Pi and Tablet

Obstacle Avoidance

- Test Simulation
- Setup Front and Back Camera
- Physical Cart Testing
- Phase 1: Physical Testing (Momentum)
- Phase 2: Put Barriers
- Phase 3: Build up Barriers
- Navigation and Control
- Phase 1: Initial test Navigation
- Phase 2: Initial Test control
- Phase 3: Set up Navigation
- Phase 4: Set up Control
- Phase 5: Press Button for Cart Return

Powertrain

- Setup the Servos
- Setup the battery power
- Setup brake

Hardware System

- Drivetrain
- Battery and Management/Charging System
- Navigation and SLAM for Autonomous Movement

Successful objectives

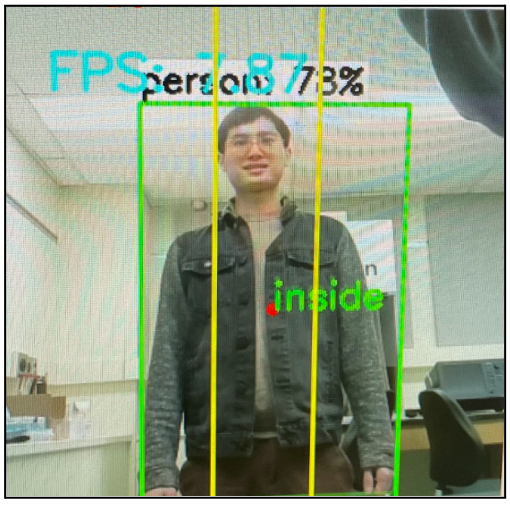

Figure 1: Navigation to follow a person

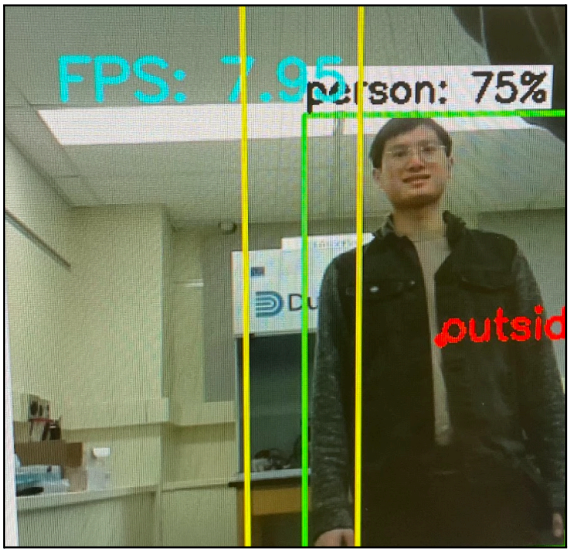

Figure 2: Navigate turn right

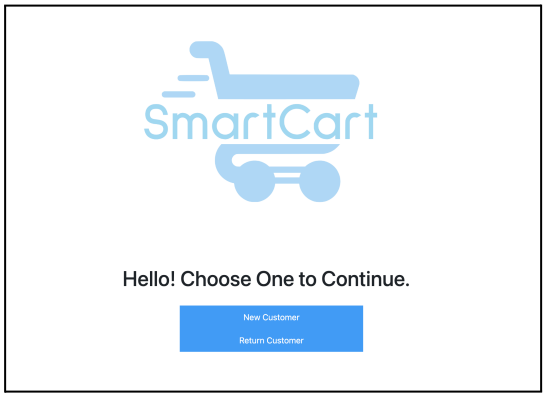

Figure 3: Landing page

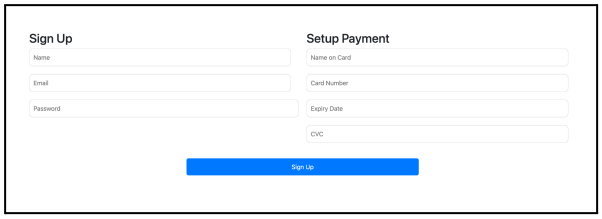

Figure 4: Sign-up

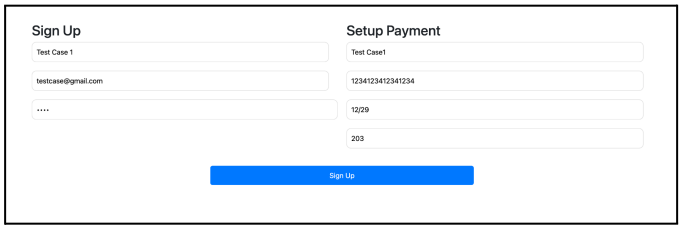

Figure 5: Sign up success

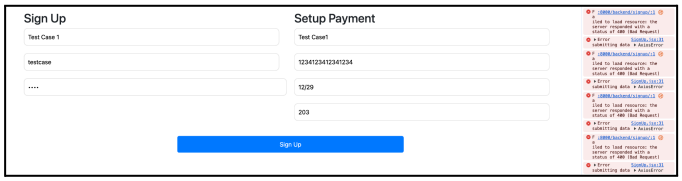

Figure 8: Sign up fail

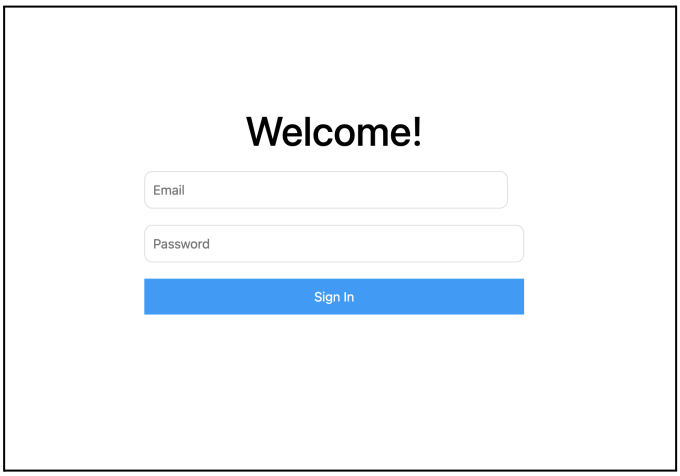

Figure 9: Login

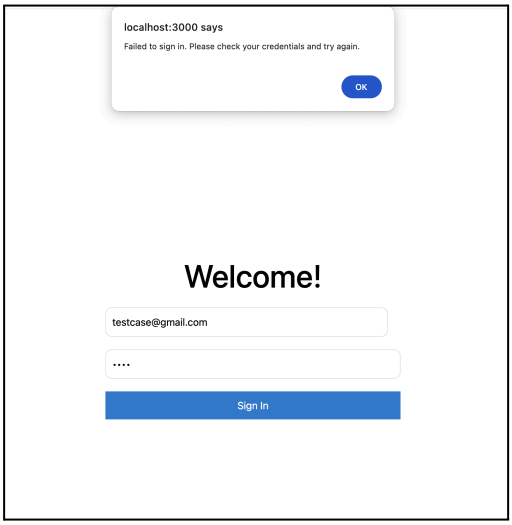

Figure 10: Login fail

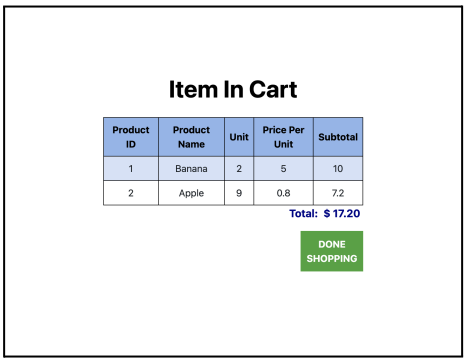

Figure 11: Shopping cart

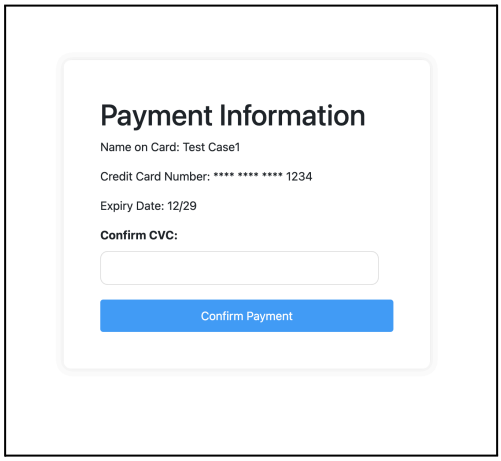

Figure 12: Confirm CVC

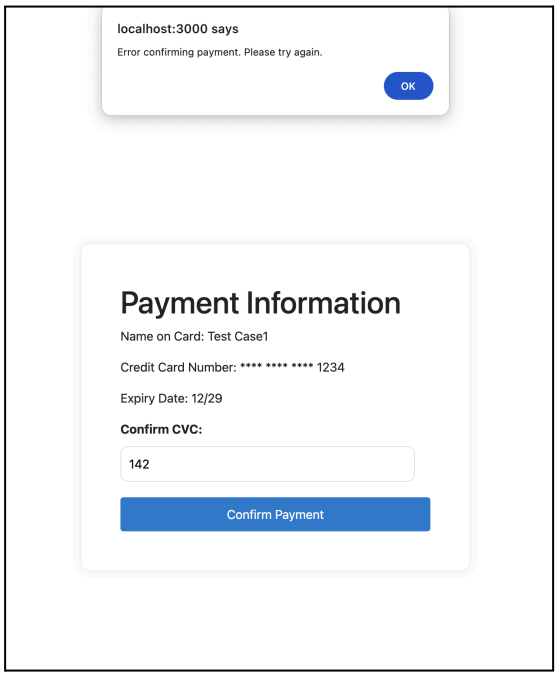

Figure 13: Confirm CVC fail

Unsuccessful objectives

Obstacle Avoidance

Due to simplifying our objectives and reaching a presentable goal, we decided to remove the obstacle avoidance system for our robot system and opted for the feasible objective of tracking and following a person

Automatic Cart Return

Due to time constraints, we could not implement the return cart system into our robot system as it would require a history of the distance traversed and memory to replay those directions to navigate back to a return station.

3. Setup Details for Project

Software

matplotlib 3.8.1

Python plotting package for creating static, interactive, and animated visualizations in Python.

Numpy 1.25.2

Fundamental package for array computing with Python, providing support for large, multi-dimensional arrays and matrices.

opencv-python 4.8.1.78

Wrapper package for OpenCV python bindings, used for computer vision and image processing tasks.

pandas 2.1.3

Powerful data structures for data analysis, time series, and statistics in Python.

pyyaml 6.0.1

YAML parser and emitter for Python, allowing YAML file reading and writing.

requests 2.31.0

Python HTTP for Humans, simplifying the process of making HTTP requests.

torch 2.1.1

Tensors and Dynamic neural network library with strong GPU acceleration, primarily developed for deep learning.

torchvision 0.16.1
image and video datasets and models for torchvision, including image transformations.

ultralytics 8.0.210
Ultralytics YOLOv8 for SOTA object detection, providing state-of-the-art algorithms for object detection tasks.

Django 3.2.5
A high-level Python web framework that encourages rapid development and clean, pragmatic design.

rest_framework 3.12.4
A powerful and flexible toolkit for building Web APIs in Django.

pygame 2.0.1
A set of Python modules designed for writing video games, providing functionality for creating games and multimedia programs.

opencv-python 4.8.1.78
Wrapper package for OpenCV python bindings, used for computer vision and image processing tasks.

TFLite 2.5.0
A set of tools that enables on-device machine learning by optimizing TensorFlow models for inference on mobile, embedded, and IoT devices.

tensorflow 2.6.0
An end-to-end open source platform for machine learning, providing a comprehensive ecosystem of tools, libraries, and community resources.

pycoral 2.0
Used in conjunction with TensorFlow Lite to deploy machine learning models on Edge TPU devices. Installation requires an extra step due to lack of support on PyPi:
python3 -m pip install --extra-index-url https://google-coral.github.io/py-repo/ pycoral~=2.0

NPM DEPENDENCIES

axios 0.21.1
Promise based HTTP client for the browser and node.js, simplifying the process of sending asynchronous HTTP requests.

react 17.0.2
A declarative, efficient, and flexible JavaScript library for building user interfaces. It enables developers to create large web applications that can change data, without reloading the page.

Ros Humble Hawksbill
Robot Operating System (ROS) installed on Raspberry Pi that contains a set of software libraries and tools for building robot applications, originally developed to run on Debian Packages

Ubuntu 22.04
Ubuntu is an open-source software operating system that allows ROS2 to run on Raspberry Pi

Hardware

The software is meant to be run on a Raspberry Pi 4 model B in conjunction with a PiCam, Coral Edge TPU, and a speaker. We will be using a 6V DC motor with a TB6612FNG motor controller connected to another Raspberry Pi 4 model B. The battery will be a 12V 5 Ah (8 cells, 4s2p) Samsung 18650-25R battery pack managed by the Shenzen BMS-40A-4S-E BMS with multiple P-channel power mos's. Voltage regulators for powering the Pi, as well as motors, are XLSEMI XL6019 5A switching regulators with passive components to step down the voltage from 12 to 5.1 and 5.75 V. There are multiple fuses scattered throughout the individual supplies in case of overcurrent resulting from component failure. The black 3D-printed box contains most of the electronic components and is attached to the camera mounted on the front of the shopping cart.

Hardware Design Process

For the design process we knew we wanted a mobile cart, meaning a battery would be necessary to supply power without wires connected to a power supply. Firstly, we needed to decide on the necessary battery capacity and discharge rate, so we benchmarked the devices that would be drawing power off a lab bench power supply. Then we would convert that to the power drawn by the input buck converter to get the maximum current that would be running through the battery at any one time. There is a worst-case efficiency of the buck converter of 70% with an average case around 80%+, but to safely design around the maximum current draw, we chose to calculate based on the worst case.

Our original design was to use two 12 V motors to control cart movement and coordination, but later in my analysis and space constraints, it proved to be unnecessary for the load that we would be moving. We later went with 4 1.5A maximum stall motors on a small metal chassis. With the maximum current of our switching IC being around 5A, we have to limit the current supplied to the motors. This is done with the TB6612FNG h-bridge motor driver and PWM control. Because of this factor, we decided to use 2 separate switching converters to power the Raspberry Pi and the motors. Because the Pi has the possibility to pass our recorded maximum current, this system would make the Pi supply consistent and more reliable.

With the maximum current running through the battery at any one time, we could work on designing the actual battery. We decided between NiMH, Sodium Ion, 18650 LiFePO4, and lithium polymer. Balancing between safety and space/energy density, we decided to go with 8 Samsung 25R INR18650 LiFePO4 cells arranged in a 4s2p topology.

Once received, we tested each cell by weighing them and confirming they were all within the rated weight provided by the supplier. We then tested each voltage and found a maximum of 1mV difference, meaning these cells were genuine. We connected battery spacers to each side of each cell and attached them together in a 4x2 fashion. We then spot-welded the parallel cells with 0.2x10mm nickel strips to ensure each strip was adequately connected. Once the parallel cells were verified by the multimeter, we then worked on welding the series strips starting from the ground. Once this was done, we checked the voltage at each stage and verified the work was done correctly. We then soldered wires to a 40A Passive BMS to each battery stage, passively balancing/protecting the cells. After analyzing the battery, we determined the

nickel strip to be the constraint when it comes to the current from the battery. With a 0.2 mm thickness nickel strip at 7 mm width, the maximum safe current drawn is around 20A (10A through each singular 0.2 series connection). We knew we would have to limit current and fuse the power supply input so as not to overload the switching converters.

Peak system current draw from battery:
(1A projected 100% motor utilization + 0.83 max Pi current) 2 A

This is well under the required current draw limited by the nickel strip thickness of the battery (20A), so we will definitely need to fuse the current input to the buck converter, in case of short failure.

Figure 13: 4s2p LiFePO4 battery with passive BMS

4. STANDARDS USED/ CONSIDERED:

IEEE Std 802.11

It is used to send data wirelessly via WiFi.
Send data from the microcontroller to the tablet.
High data rate transmission of the video streaming.
Secure protocol network transmission.

SAE J2402_202311

It is used to indicate when the shopping cart is moving or turning.
It is used to indicate possible malfunction of devices on the shopping cart.
Prevent danger to oncoming traffic and users with indicators and symbols

IEEE Std 2030.2.1-2019

Battery management system disconnects power at undervoltage, overvoltage, and is fused for overcurrent
Each cells state of charge (SOC) has been cycled, and cells have been balanced
Charge and discharge rates are being followed or safely under the maximum provided in the cell datasheet
PCS regulates voltage, and connected IC's regulate current

IEEE Std 1872-2015

Methodology for knowledge representation and reasoning in robotics and automation
Fully autonomous robot: A role for a robot performing a given task in which the robot solves the task without human intervention while adapting to operational and environmental conditions
RobotMotion: is any process of movement where the agent is a robot and the patient is one of its (robot) parts. Given that, it is any process in which the robot moves one of its parts.

TCP:

used for constant encoded video communication with the base station

created using the low-level, wifi communication socket and socket-server libraries
better for constant video stream than UDP because of out-of-order packets

I2C:

It is a 2-wire digital communications link for the serial data line (SDA) and serial clock line (SCL), which allows for relatively fast and reliable data transfer.
Protocol allows for multiple communication devices to be connected with the same wire by having different addresses for the individual communication device
It is acknowledged after receiving the data, and verifying that it is sent and received. This helps the server and user to get the acknowledgment received.
These standard protocols allow us to keep a clean and efficient way to design our SmartCart.

5. Updated Prototypes

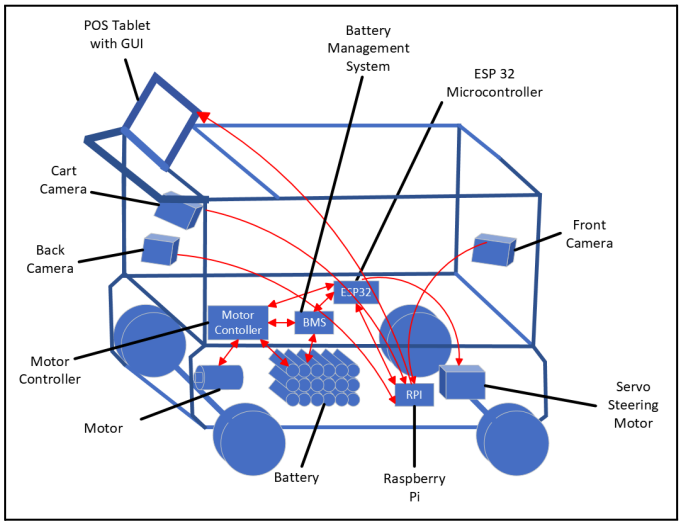

Figure 14: Initial Design

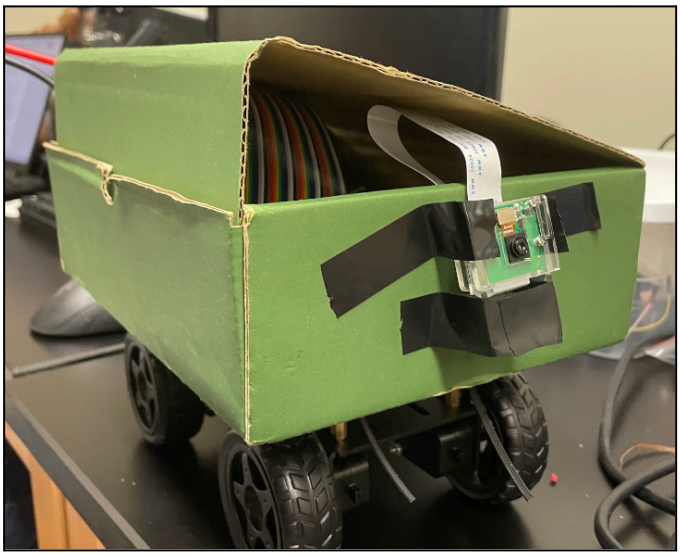

Figure 15: Prototype of components in show box attached to wheels

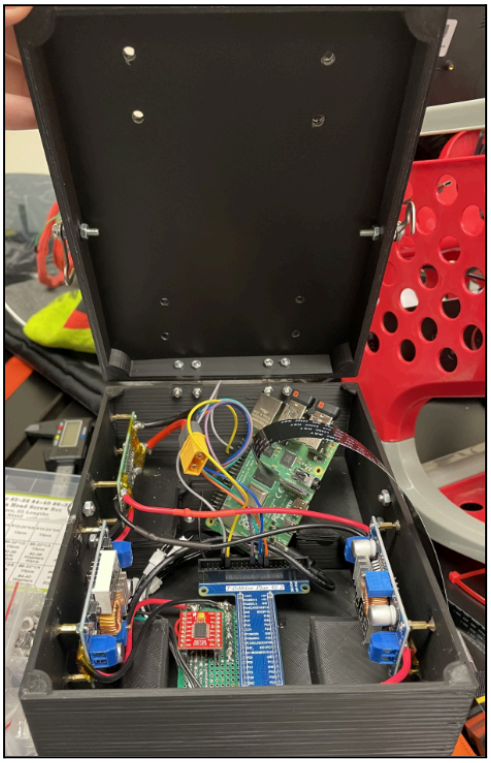

Figure 16: 3D-printed box to fit components and connected to wheels

Figure 17: Top of cart with touch screen connected

6. Conclusions/Results

Figure 19: Fully autonomous cart

Nominal Cases
Inputs: Apple, Orange, Banana, Bottle and Cup
Using Coral Edge TPU
Distance from Camera (10 inches +/- 0.5 inches)
Output: Accuracy: average 86.12%
Expected Output:
Accuracy: average 85%

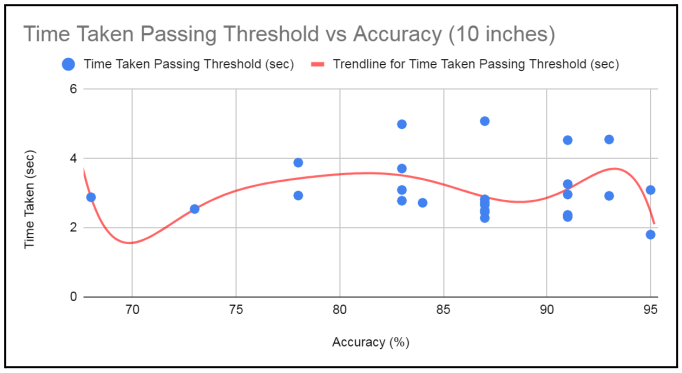

Figure 20: Time Taken vs Accuracy (10 inches)

Error/Edge Cases
Inputs: Apple, Orange, Banana, Bottle and Cup
Using Coral Edge TPU
Distance from Camera (8 inches +/- 0.5 inches)
Output: Accuracy: average 84.96%
Expected Output:
Accuracy: average 92%

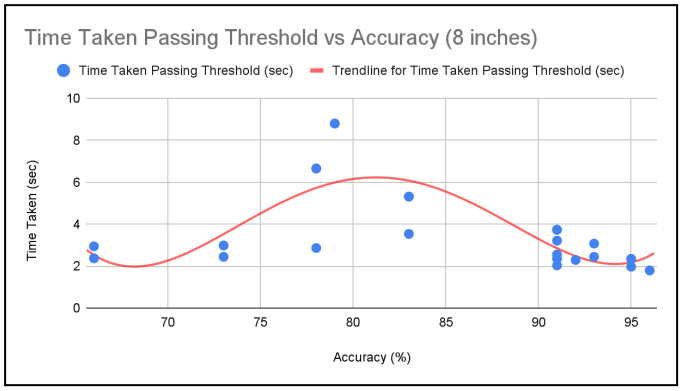

Figure 21: Time Taken vs Accuracy (8 inches)

5.1V input Raspberry Pi 4 Model B with PiCam (Idle)

Time of Recording (s)	Supply Voltage (V)	Supply Current(A)	Power Draw (W)
1	12.01	0.11	1.3211
2	12.01	0.29	3.4829
3	12.01	0.33	3.9633
4	12.01	0.32	3.8432
5	12.01	0.31	3.7231
6	12.01	0.27	3.2427
7	12.01	0.33	3.9633
8	12.01	0.28	3.3628
9	12.01	0.26	3.1226
10	12.01	0.26	3.1226

Figure 22: 5.1V input Raspberry Pi 4 Model B with PiCam (Idle)

5.1V input Raspberry Pi 4 Model B with PiCam (Under Progra Load)

Time of Recording (s)	Supply Voltage (V)	Supply Current(A)	Power Draw (W)
1	12.01	0.49	5.8849
2	12.01	0.56	6.7256
3	12.01	0.61	7.3261
4	12.01	0.59	7.0859
5	12.01	0.56	6.7256
6	12.01	0.51	6.1251
7	12.01	0.55	6.6055
8	12.01	0.59	7.0859
9	12.01	0.53	6.3653
10	12.01	0.59	7.0859

Figure 23: 5.1V input Raspberry Pi 4 Model B with PiCam (Under Program Load)

4 5.75V Motors at 50% Duty Cycle TB6612FNG

Time of Recording (s)	Supply Voltage (V)	Supply Current(A)	Power Draw (W)
1	12.01	0.42	5.0442
2	12.01	0.48	5.7648
3	12.01	0.51	6.1251
4	12.01	0.48	5.7648
5	12.01	0.49	5.8849
6	12.01	0.50	6.005
7	12.01	0.51	6.1251
8	12.01	0.50	6.005
9	12.01	0.49	5.8849
10	12.01	0.49	5.8849

Figure 24: 4 5.75V Motors at 25% Duty Cycle TB6612

References

[1] Evan, "TensorFlow Lite Object Detection on Android and Raspberry Pi," *GitHub*, Apr. 23, 2023. Available:https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/deploy_guides/Raspberry_Pi_Guide.md

[2] "Django," *Django Project*. Available: https://docs.djangoproject.com/en/5.0/topics/http/urls/ (accessed Feb. 26, 2024).

[3] "ROS 2 Documentation — ROS 2 Documentation: Humble documentation," *docs.ros.org*. https://docs.ros.org/en/humble/index.html

[4] "Model: BMS-40A-4S-E / B / S BMS-40A-4S-S Standard BMS 4 cell 16.8V 40A lithium battery protection board (with recovery function -AUTO Recovery)." Accessed: Feb. 26, 2024. [Online]. Available: https://www.mantech.co.za/Datasheets/Products/BMS-40A-4S_SGT.pdf

[5] J. Zhao, S. Liu, and J. Li, "Research and Implementation of Autonomous Navigation for Mobile Robots Based on SLAM Algorithm under ROS," *Sensors*, vol. 22, no. 11, p. 4172, May 2022, doi: https://doi.org/10.3390/s22114172