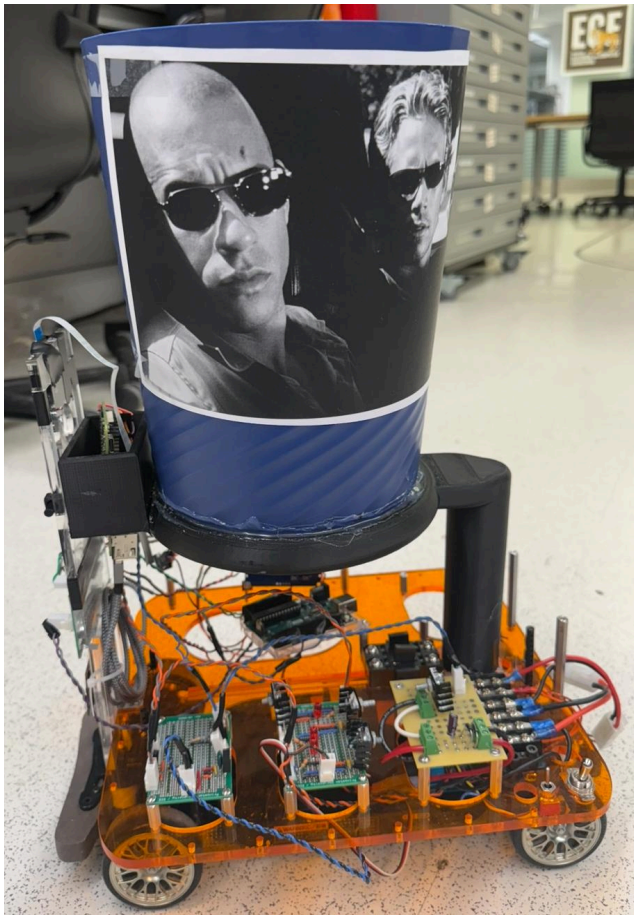
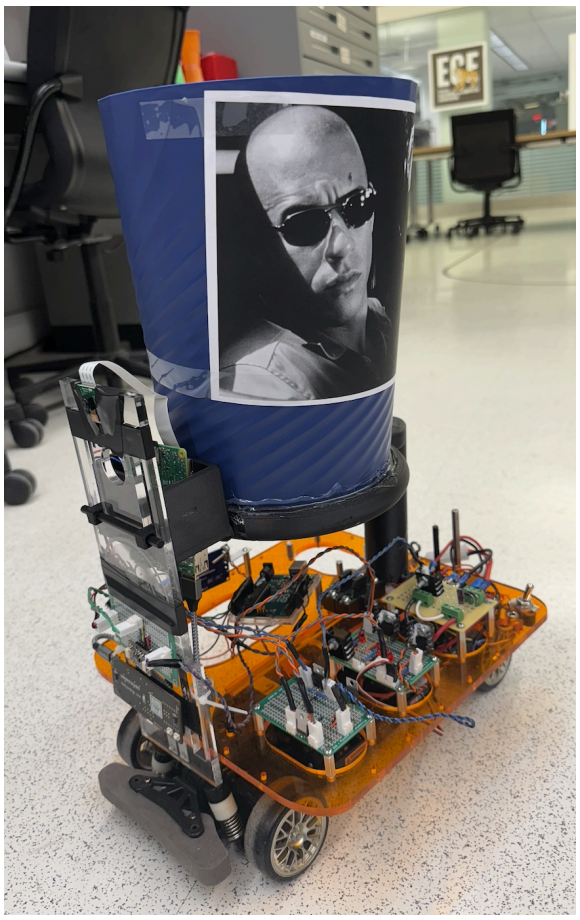
All circuits need schematic diagram. Need .zip with submission.

Bin Diesel: An Autonomous Trash Collection System

Callista Chong: cc0225@princeton.edu
Mayan Wasu: mw8270@princeton.edu
Group 308





Statement of Objective (5 pts)

Innovation takes time and experimentation; iterating through hardware test implementations produces broken crimps, wire and solder trimmings, pieces of electrical tape, burned componants, and more annoying clutter. Due to the nature of the space, the EQuad F112 lab benches are tight and cannot accommodate a dedicated waste bin, making the task of continuously collecting trash and keeping the bench clean a bit arduous. The Bin Diesel Autonomous Trash Collection System™ is the most natural solution: a small bin fitted atop an autonomous, key-word activated car that drives to the user via gesture detection, waits for trash deposits, then faithfully returns back to its starting position. Functionality of the Bin Diesel Autonomous Trash Collection System™ is contingent upon it's ability to 1) respond to the wake-work, "Bin Diesel!" (trained on Mayan Wasu's voice), 2) detect the human in it's line of sight with either arm raised to the side at 90° and lock on by assigning a tracking ID to that user, 3) use a standard control system to track the target using continuous corrective angle information, 4) stop when the first obstacle (assumed to be the user) is detected, and 5) detect and return to it's home positions, marked by a **WHAT"S THE QR CODE THING CALLED**. These goals were achieved through careful integration of new hardware and software, and iterative whole system tuning.

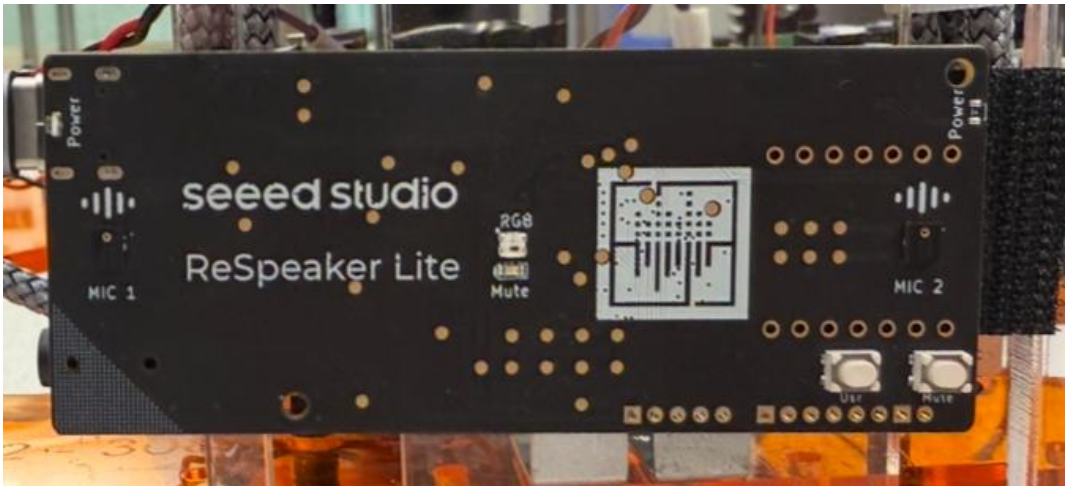
Overview of Key Sub-Systems and Components (30 pts)

Sensors

1. **Seeed Studio reSpeaker Lite USB 2-Mic Array with Picovoice Porcupine**

i) The device

The system uses a seeed studio reSpeaker Lite as it's primary audio sensing interface to implement a voice-based activation system. The reSpeaker provides a dual microphone array with onboard analog to digital conversion for seamless integration, and USB audio streaming to enable robust speech capturing with moderate environmental noise. The wake word enables hands-free, distant activation, so the vehicle can remain out of the way, stationary, and idle while awaiting the command. The module was mounted on the front mast of the vehicle using Velcro strips.



ii) Associated software

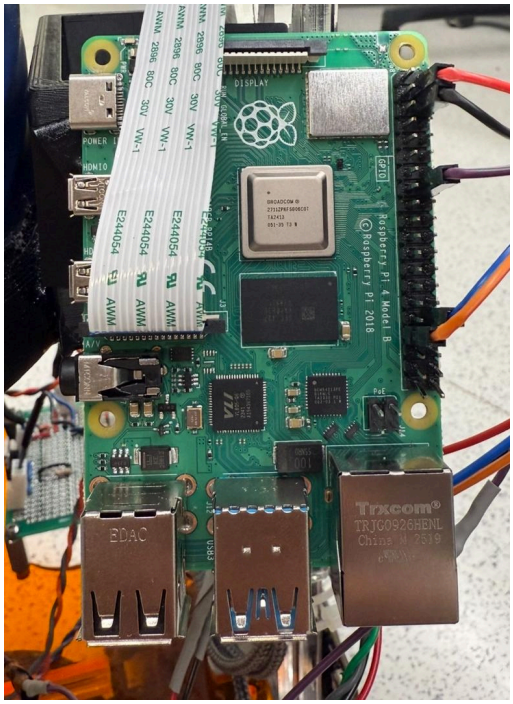
Wake word detection logic is contained in the *wake_word_detector.py* class, which manages audio streaming and inference. Audio is read frame by frame from the mic array and is converted from raw byte data into integer samples. Picovoice Porcupine's *process( )* function performs keyword inference using a custom wake word model file, returning a non-negative index when detection is flagged. For this project, the detection flag triggers a transition from the IDLE state into the TRACKING_USER state.

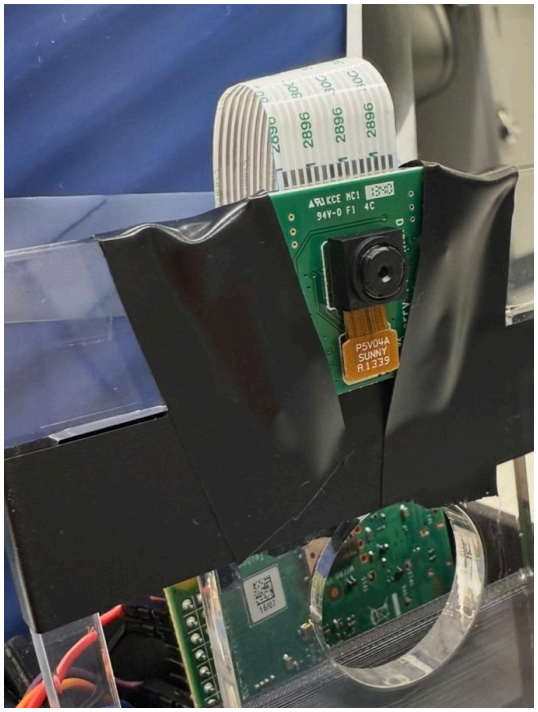
short audioframs to detect our custom wake word: "Bin Diese

2. PiCam with Raspberry Pi

i) The device

The Raspberry Pi 4 was our core processing unit. Inputs included PiCam, reSpeaker Lite, and the ToF trigger logic signal from Arduino.



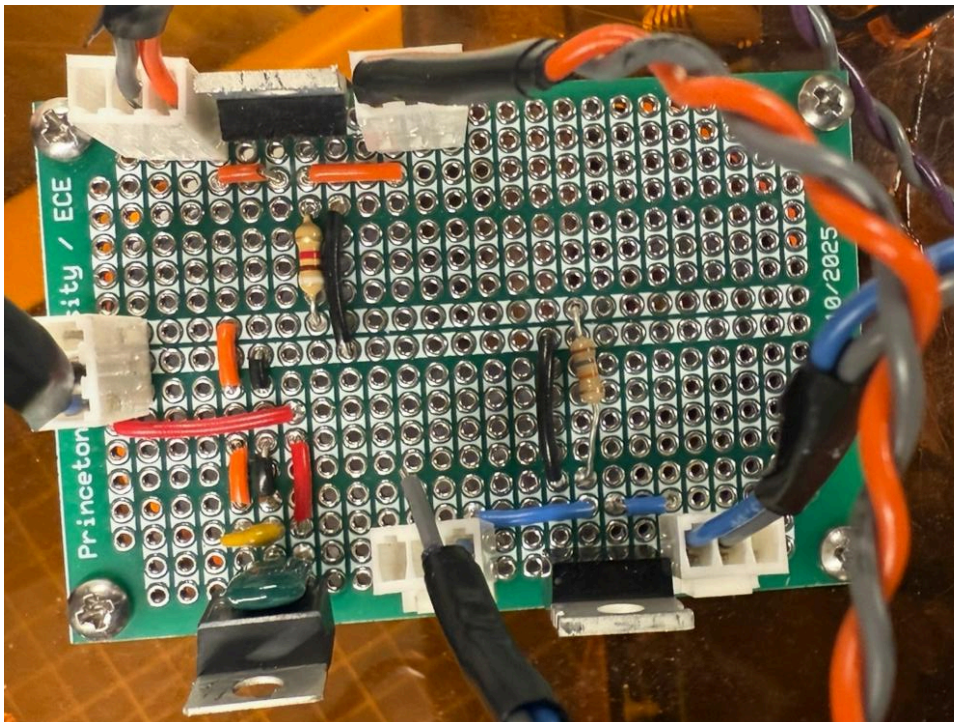


The Raspberry Pi requires a stable 5V supply and operates exclusively with 3.3V logic on its GPIO pins. To meet these requirements, several power regulation and logic-level conversion stages were necessary implementations.

A buck converter was used to safely step down the 9.6V battery to a regulated 5V suitable for powering the Raspberry Pi. This ensured a stable supply voltage despite battery discharge and load variations, which is critical to protect the Pi and ensure stable operation.

The 5V digital proximity flag derived from the ToF sensor (discussed more in the following section) is sent from the Arduino to the Raspberry Pi, requiring logic-level 5V to be safely translated to 3.3 V before being read by the Raspberry Pi GPIO. This was implemented using a dedicated bidirectional level-converting IC, which needed 5 V and 3.3 V reference voltages.

Conversely, control PWM signals from the Raspberry Pi to the motor and steering servo required translation from 3.3 V to 5 V. A pull-up inverter circuit was used to accomplish this: the 3.3 V PWM outputs drive the gates of low-threshold NMOS transistors, with pull-up resistors connected to 5 V, provided from the 9.6V battery using onboard leveling via an LM7805. This configuration inverts the PWM signal, but creates a 5V PWM suitable for the motor and servo control inputs.



ii) Associated software

test_yolo_pose_tracking.py
test_apriltag_detection.py

3. Adafruit VL53L0X ToF Distance Sensor with Arduino Uno

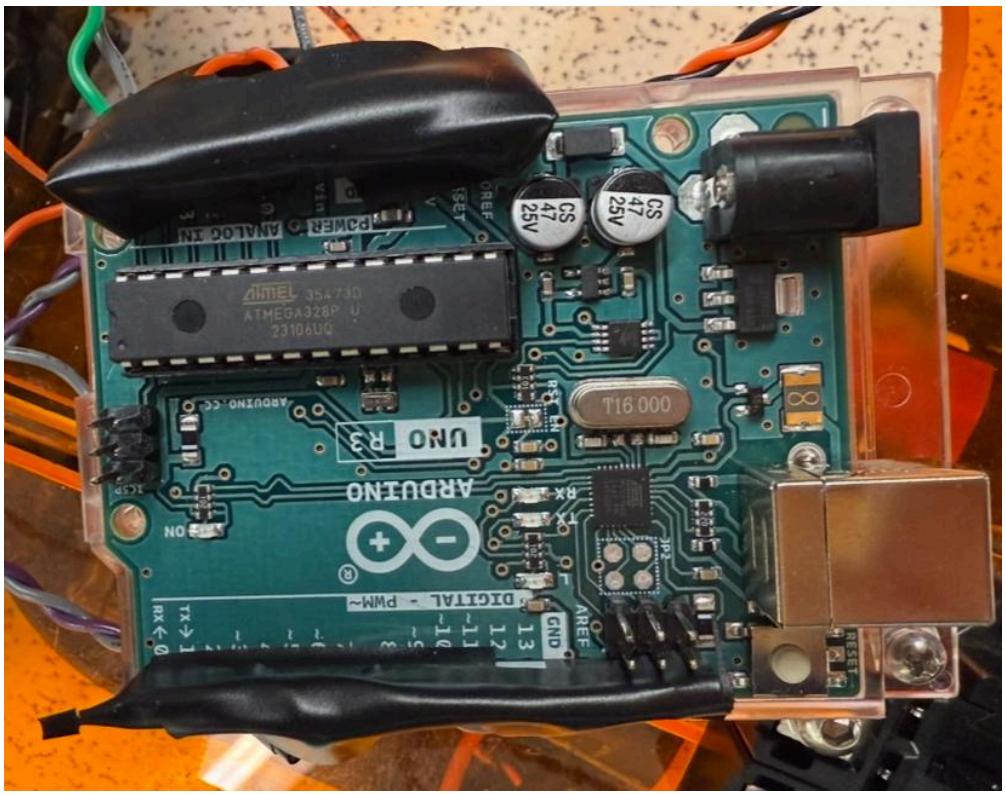
The system uses the VL53L0X time-of-flight (ToF) sensor to provide proximity-based stopping instructions. The ToF is first read on an Arduino Uno over I^2C, which produces a distance measurement in millimeters. This distance is thresholded to produce a digital output high/low signal used to flag stopping. The Arduino outputs 5V logic, so it is level shifted to 3.3V to make it compatible to Raspberry Pi GPIO inputs.

On the Raspberry Pi, the ToF is treated as a high-priority safety interlock, being polled at the beginning of every min control-loop interaction before any other processing is allowed to happen. If the ToF flag is raised while the car is in a state with active motion, the

controller immediately sends a *motor.stop()* command, assumes the user has been reached, and executes the rest of the turning and return-to-home procedures. To reduce false triggers, the ToF receiving function requires multiple consecutive logic high readings before triggering a stop flag.

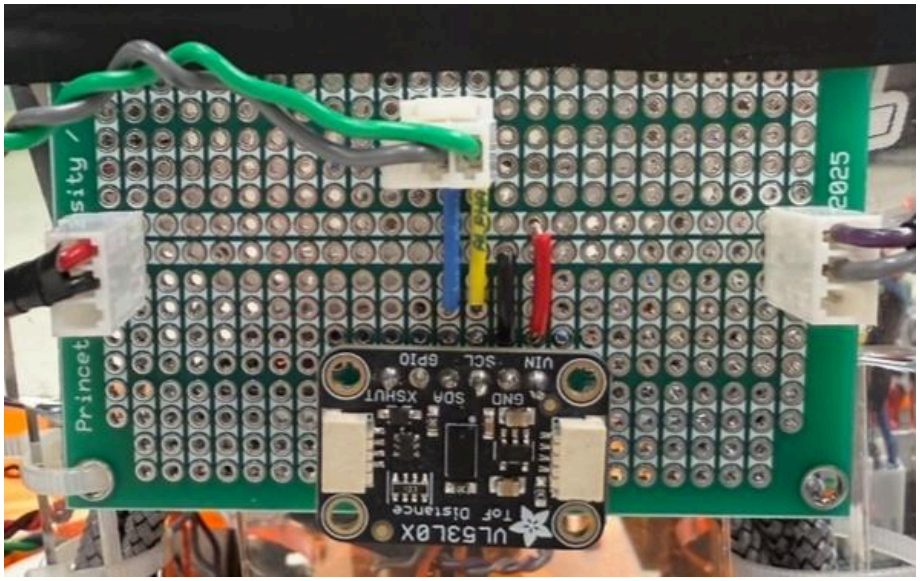
a. Arduino Uno

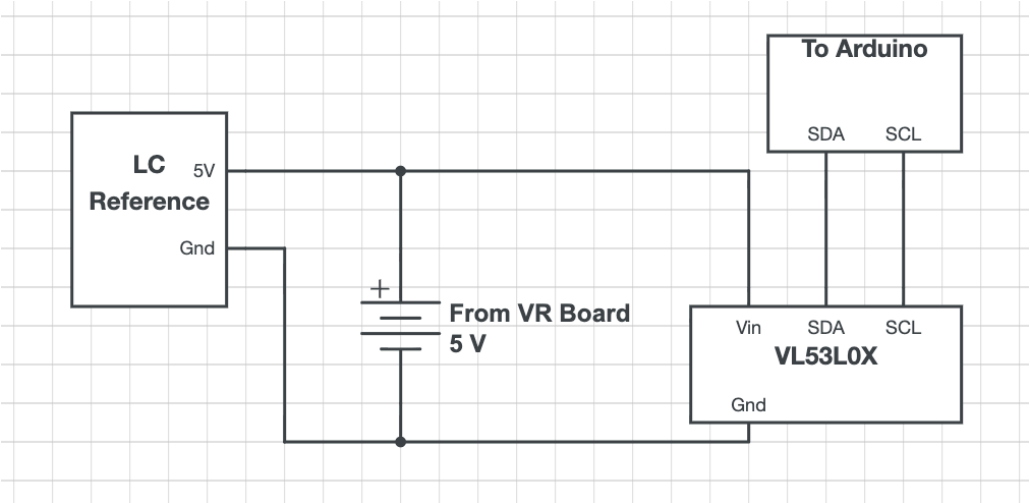
The Arduino operates with 5 V logic, which is compatible with the VL53L0X I^2C interface. However, its 5 V digital output is incompatible with the 3.3 V Raspberry Pi GPIO inputs, necessitating a level converter to safely translate the proximity flag. The Arduino is powered from the system's voltage regulator (5 V and GND), provides 3.3 V and GND references to the level converter, communicates with the ToF sensor via SDA and SCL, and outputs the thresholded proximity signal on digital pin 7. Offloading I^2C communication and thresholding to the Arduino simplifies the Raspberry Pi interface.



b. ToF Board

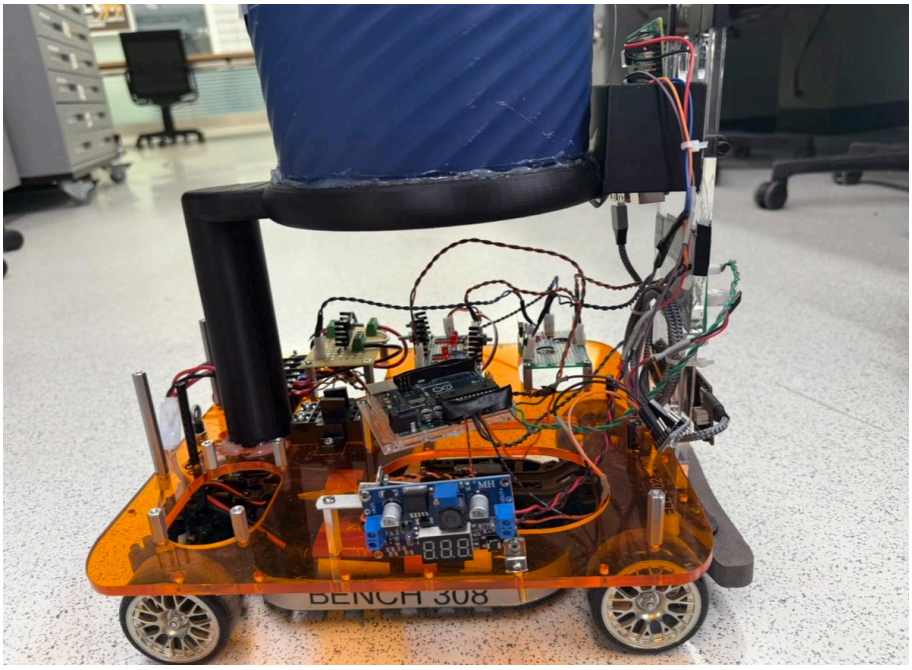
The ToF board is powered from the system's voltage regulator (5 V and GND), which is shared as a reference voltage with the level converter, and transmits SDA and SCL to the Arduino.





Hardware

3D Printed Bin & Raspberry Pi Holder



Code Architecture Overview

The Raspberry Pi code is organized around the main control script, *maincallista.py*, which executes the car's high-level behavior and procedures. *maincallista.py* runs the main loop, polls sensor inputs, and commands motor/servo PWMs. The main module is directly dependent on *statemachine.py*, which defines the system's operating states and monitors transitions between them, and utilizes the following specialized modules to support functionality:

Perception and input modules

- *wake_word_detector.py:* detects wake word, "Bin Diesel", and triggers activation from IDLE state to TRACKING_USER state
- *test_yolo_pose_tracking.py:* executes gesture detection using YOLO-based pose tracking to lock onto the user and triggers transition from TRACKING_USER to FOLLOWING_USER
- *tof_sensor.py:* reads the ToF binary proximity flag from the Arduino through a GPIO input to detect when the user has been reached
- *test_apriltag_detection.py:* detects an AprilTag used to mark the car's home position and facilitates navigation back home after trash has been deposited.

Actuation modules:

- *motor_controller.py*: sends PWM signals to the level converter board intended to control speed. PWM values are configured according to an inverted PWM, and clamps on PWM values are in place for safe functionality. This program takes in speed requests as percentages of maximum duty, defined in the config file.
- *servo_controller.py:* sends PWM signals to the level converter board intended to steer the car. PWM values are similarly configured according to the inverted PWM with clamps on maximum angles. This module also converts angles into PWM duty cycles for ease of use.

Data Showing Expected Functionality (25 pts)

1. ReSpeaker wake word detection: see how far it detects/how reliable?
2. PiCam gesture detection and tracking ID
3. ToF signal reliability: stopping distance from different objects?
    a. Run car at wall with only distance sensor stopping the car and measure the stopping distance to judge response time (900mm trigger)
4. Raspberry Pi: PWM traces
    a. Measure pre-inverter level correction and post, see that signal is flipped and pulled up to 5V from 3.3V
5. Time Limited Open-Loop Steering Correction: measure corrective path with a single target
    a. Collect each pwm update and graph to see how direct the path is.

Discussion of Challenges and Design Choices (10 pts)

Our original project plan aimed to use the reSpeaker to estimate the angle of the wake word in order to steer the car towards the user. However, the speaker did not have a reliable way to separate the audio received by either microphone to perform any kind of angle estimation. Instead, a wake-word-based activation scheme was chosen in conjunction with PiCam gesture tracking to provide a low-power interface. By decoupling the system

activation from the perception module, we avoid continuously running the computationally intensive vision algorithms. This prevents unnecessary CPU load on the Raspberry Pi and improves system responsiveness.

Although the PSoC was already configured for motor/servo control, the final design completely removed reliance on the PSoC. Since signals were time-critical (corrective steering, stop signals), direct control from Pi to actuators minimized communication overhead and simplified the control path. The result was a more predictable and faster system.

One of our objectives going into this project was to emphasize electrical engineering principles: to us, this meant physically integrating sensors and actuators before relying too heavily on software. However, this meant interfacing between multiple voltage domains. The Raspberry Pi had a 3.3V GPIO limitation that had to be coupled with the 5V logic required by the external actuators and ToF sensor routed through the Arduino. A simple level converter was sufficient for the 5V to 3.3V translation from the Arduino, but an NMOS pull-up inverter circuit was more suitable for the task of raising the 3.3V PWM signals being sent to the motor and servo.

Another deviation from the original project proposal was that, instead of doing one angle estimation and driving until a user was met, continuous corrective navigation was used throughout the entire path to the user. For this reason, simply tracking time traveled and executing a return path based on that was no longer a reliable way to return to home. AprilTags were selected as a robust and visually distinct marker to represent the home location, and were easily integrated into the already existing PiCam and visual processing foundation.

Possible Improvements (10 pts)

There are a number of natural extensions to this project that would enhance the reliability of the system, improve the user experience, and reduce latency:

1. Improved return to home procedure

The current HOME state procedure relies on the AprilTag being within the PiCam field of vision, or a slow search sweep is executed in an effort to find it. A very beneficial improvement combines AprilTag detection with dead-reckoning (using a gyroscope or wheel odometry) to assist in estimating how much to turn after a trash deposit to ensure the AprilTag is in view.

2. Natural language voice control

The audio processing capabilities of the system currently only serve wake word detection, but it could easily be extended to support basic spoken commands in a manual control mode, such as “forward”, “stop”, or “turn left”. This would allow manual override/course correction, so in environments where the camera is not reliable (low-light, high noise), the user can override dependence on it.

3. Adaptive speed and steering control

The vehicle’s navigation control system accounts for latency in the error-angle delivery by using fixed-duration steering and speed commands. Future iterations of the Bin Diesel Autonomous Trash Collection System™should include adaptive control based on target distance, and expand upon existing frameworks that adjust speed based on tracking confidence. We already increase speed when the angle error is low, but extensions can be made to slow speed when approaching a user, and adjust speed/angle adjustments based on user tracking confidence.

Appendix of Code, Adequate Comments (5 pts)

Key Code to add:

Modular:

→ Wake Word
→ Pose Detection
→ Tracking
→