Animal Detector

Thinh Truong, CompE, Arjun Rajapur, EE, Syed Muhammad, CompE, and Quan Minh Do, CompE, Julia Le CompE

***Abstract*—In this project, we want to design and prototype as a proof of concept, an automotive system that aids drivers in the prevention of collisions with large animals to prevent vehicle damage, injury, or death of the driver, as well as protection of wildlife.**

INTRODUCTION

Can we design a prototype of a system that can help a driver avoid collisions with large animals at speeds up to 60 mph?

Significance

"...about 1-2 million collisions occur yearly between cars and large wild animals in the U.S., causing approximately 200 human deaths, 26,000 injuries, and at least $8 billion in property damage [1]."

Context and Survey of Similar Solutions

There are many pre-existing systems out there that address this problem, with most having some limitations or the other. For example, Volvo cars have a collision avoidance system, with a particular feature of this being the ability to detect large animals, hence the name "Large Animal Detection System". The system uses radar and cameras in tandem to detect "large animals" such as moose or bear. If necessary, the system applies brakes or slows down the car. The limitations of this system are that only large moose or bear-sized animals are detected. Additionally, detection is only limited to when the side view of the animal is visible, and detection of animals from the front/rear is far less reliable. Partially obscure animals are unable to be detected as well.

Another approach comes from a paper that proposes a model using convolutional neural networks to detect animals using a camera. According to the paper, a 91% detection accuracy was achieved when mounting the camera on a dashboard and driving in various lighting conditions. What this system does not do is any sort of collision prevention mechanism such as braking/deceleration. Additionally, the paper doesn't specify at what speeds testing was done in. Its range of the size of animals detected was larger than that of Volvo's system.

Our system aims to bridge the gap between these two. Our system should be able to detect a wider range of animals than Volvo's system, as there are many animals smaller than moose that can make for a dangerous collision. It should also be able to briefly control the braking/speed of the car in a dangerous situation, and it should also function at high speeds, up to 60 mph.

Societal Impacts

Drivers will be positively impacted by our prototype, largely positively. They will have a system in their car that alerts them of potential animal threats on the road, automatically prevents imminent collisions with animals, and, in the worst-case scenario, minimizes collision damage as much as possible.

There are, however, a few potential negative impacts. Many people are not keen on autonomous control of their vehicles, as they lose control of their vehicle briefly.

Goals, Specifications and Testing Plan

For ease of testability, we are using a model car that is a fraction of the size of a real car. In other words the detection distance, and size specifications have been scaled down and made relative to the dimensions of the model car such that they don't depend on the model car we eventually use. The dimensions of the car will be specified as ***l***, ***w***, and ***h***. Our specifications and testing plan are in **Table 1** below.

Specification	Corresponding Testing Plan
Detect animals at a minimum size: <u>From the side:</u> 0.3*h tall and 0.5*w wide <u>From the front/rear:</u> 0.3*h tall and 0.1*w wise From a max distance of: (25 - 29)*l The car should be able to do this when going at a max speed of "60 mph" **relative to its size.**	Run recorded and real-time detection tests at various speeds on a straight track, with a small replica of an animal (deer for example) that is at the minimum specified size. The track will have flags/poles marking a certain distance increment. Test under following configurations: i)Animal parallel to line of sight, in front of car ii)Animal perpendicular to line of sight, in front of car iii)Moving animal, across car's path iv)Animal in positions described in i and ii but positioned on side of track When an animal has been detected, indicate in some form (LED, buzzer, etc.) Run back test footage and determine detection distance manually by measurement and using the flags/poles along the track as reference.
Car should slow down in events where an animal **could** cause a collision (examples include: an animal is on the side of the road about to jump onto the road, or an animal is	Two separate testing scenarios will be performed at various speeds and distances on a straight track: i)Replica animal further than 2m away from

already on the road), and come to a complete stop if the animal is at a distance less than 2m. If an animal is less than 2m away from the car, the car should hit the brakes and come to a full stop. This detection comes from the front sensor.	the car appears on track ii)Replica animal closer than 2m away from car appears on track A successful test is considered if for scenario i the car brakes before hitting the animal and in scenario ii the car is at least in the process of braking when it hits the animal.
Side sensors must be on at all times and , and should alert the driver via turning on an LED and buzzer if an object is closer than 0.3 times that of the car's width (0.3 *W)	Testing will be performed at various speeds on a track with barricades on the side that are within 0.3 * W from the car. The alert system should activate for at least 95% of tests.
Maintain at least 10 FPS (frames per second) on Raspberry Pi 5	Use Python timer to measure frame processing rate
Alert driver within 100 ms after detection	Measure delay in buzzer/LED activation
Maintain at least 75% accuracy under glare, shadow, or light rain	Test under 3 lighting/weather conditions (sunny, dusk, drizzle)

Table 1: Specifications and Testing Plan

DESIGN

A. Overview

The primary goal of our system is to detect animals approaching the roadway and assess the potential risk they pose to an oncoming vehicle. Our initial machine-learning approach relied on a simple image-classification model that identified whether an object in a frame was an animal. After classification, we attempted to estimate distance using bounding-box size and object position within the frame. While this approach allowed us to detect animals and generate a basic proximity estimate, it processed each frame independently and lacked any understanding of movement or behavioral trends. For our MDR prototype, we transitioned away from standalone classification and implemented a more practical, real-time detection pipeline. Using an AI camera module connected to the Raspberry Pi, the system streams continuous video and processes each frame with a YOLOv8n detection model. The updated design outputs bounding boxes, detection confidence, and an estimated distance to the detected animal, providing a significantly more robust foundation than the initial prototype. While the current implementation runs at approximately 10–15 FPS, we plan to improve latency and achieve stable real-time performance (20–30 FPS) through model optimization, hardware-accelerated inference, and resolution tuning. To further enhance accuracy and reduce overfitting, we will expand the dataset with additional online images, apply stronger data augmentation, and fine-tune the YOLOv8n model specifically for roadside detection scenarios. These improvements will strengthen the system's ability to compute a reliable hazard score in later development stages.

In addition to the camera-based detection system, we are implementing proximity sensors around the vehicle, to enhance driver awareness and decision-making. This feature will use side-mounted IR Time of Flight (VL53L0X) sensors to detect nearby vehicles in adjacent lanes and monitor blind-spot zones. A sensor on the front, also VL53L0X will monitor the distance between the car and the animal. Combined with the front camera, it will allow us to more accurately assess the risk level of an animal's behavior, to determine the appropriate response of our alert/braking system. The side sensors, interfaced to our Raspberry Pi 5 via I2C, will alert the driver when an object comes too close to the side of the vehicle. The main purpose of this is to provide a means of alerting the driver, if they plan on making any sudden lange changes as a panic response to the animal/ or the car autonomously slowing down/braking. During MDR, we

prototyped the sensor-based alert system with simple HC-SR04 Ultrasonic sensors, simply as a prototype, however our intention is to use the IR sensors because they are much more accurate, and have a much larger detection range.

The alert system, to alert the driver, is simple. Two pairs of Red and Orange LEDs, positioned on the right and left of the car, provide visual alert to the driver. A buzzer will provide an auditory alert. The alert system's state will not only be determined by the sensors and what they detect, but also the camera and what our model detects. Our alert system interfaced with the sensors was demonstrated in MDR.

Our PCB, which hasn't been printed yet but has been designed and routed, will interface all the sensors to the Raspberry Pi, providing I2C data connection and power. It also supports the alert system and interfaces it with the Raspberry Pi GPIO pins. All power supplied to the PCB will come from the +3V3 and +5V rails of the Pi.

Our car's braking system, controlled by our processing unit, the Raspberry Pi 5, was implemented and controlled using a 50 Hz RC Control PWM signal from the Pi's Hardware PWM Pin , whose duty cycle is varied for a slowing down, and hard brake of the car. Both deceleration and hard braking were implemented via Python functions, and shown during MDR.

B. Hardware Design Overview

The hardware subsystem in our prototype integrates sensing, processing, and control into a compact embedded system. The Raspberry Pi 5 serves as the central processing unit, running real-time object detection using the Pi Camera Module 3 and communicating with 5 laser time-of-flight sensors (VL53L0X) over the I²C bus.

A custom PCB handles power supply and distribution to the sensors. The PCB will also interface the I2C bus to sensors as well as other GPIO to the alert system. Power to the components of the PCB will be supplied by the Raspberry Pi's voltage rails.

The Pi sends PWM signals to the car's electronic speed controller to brake or slow the vehicle when a hazard is detected, while the alert system uses LEDs and buzzers to issue caution and urgent warnings.

The car used for this project is the Traxxas Slash 2-Wheel Drive 1/10 RC Car capable of speeds up to 30 mph. The car is about 2 feet long and 1 foot wide. The ESC used for the car is the XL-5, which is connected to the Raspberry Pi, instead of the remote controller that comes with it. The ESC contains only three wires, Power, GND, and its PWM wire. We simply cut the PWM wire from the car's receiver and attach it to the Pi's PWM pin, and connect the GND wire to the Pi's GND.

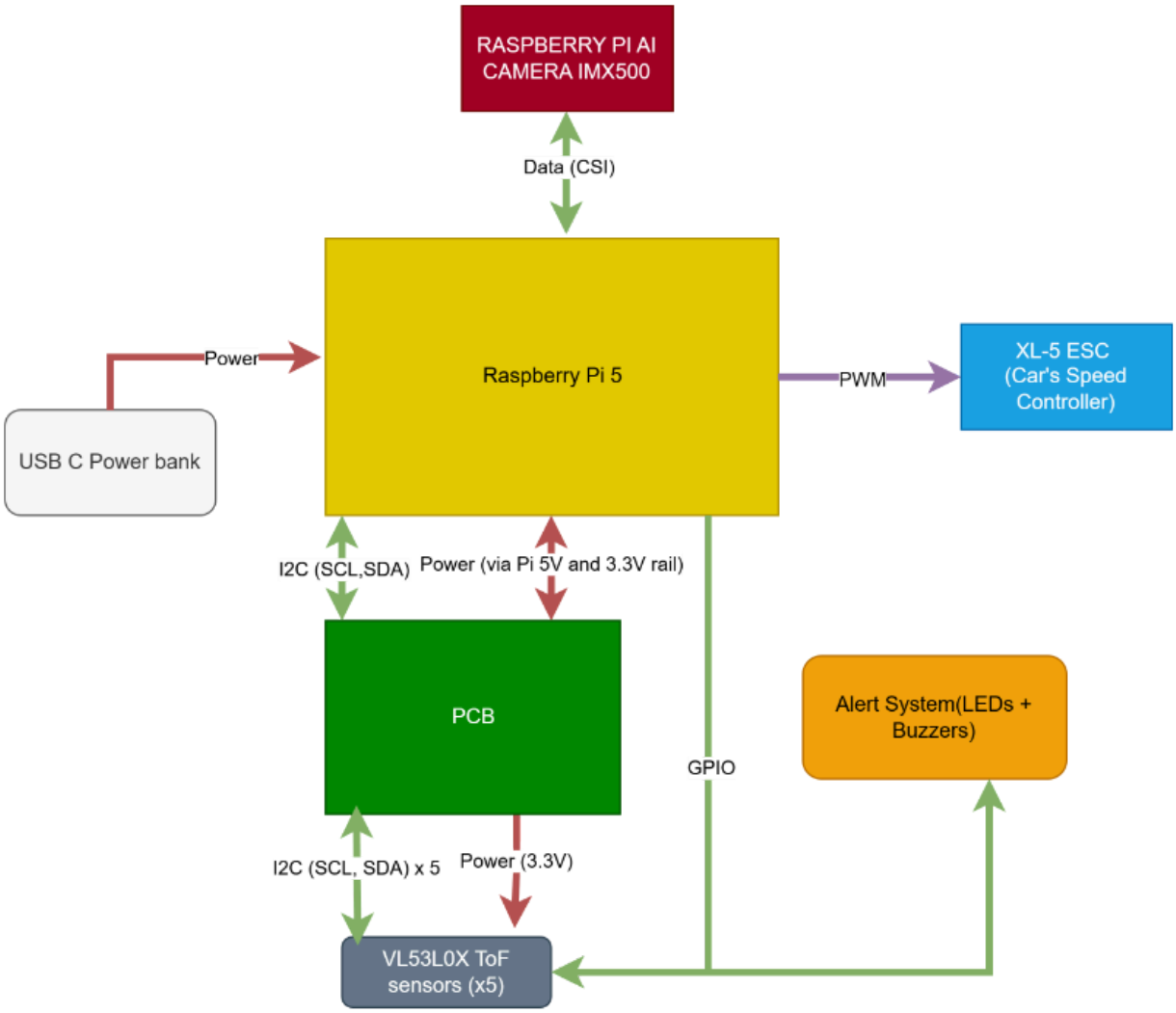

Figure 1: Hardware Block Diagram

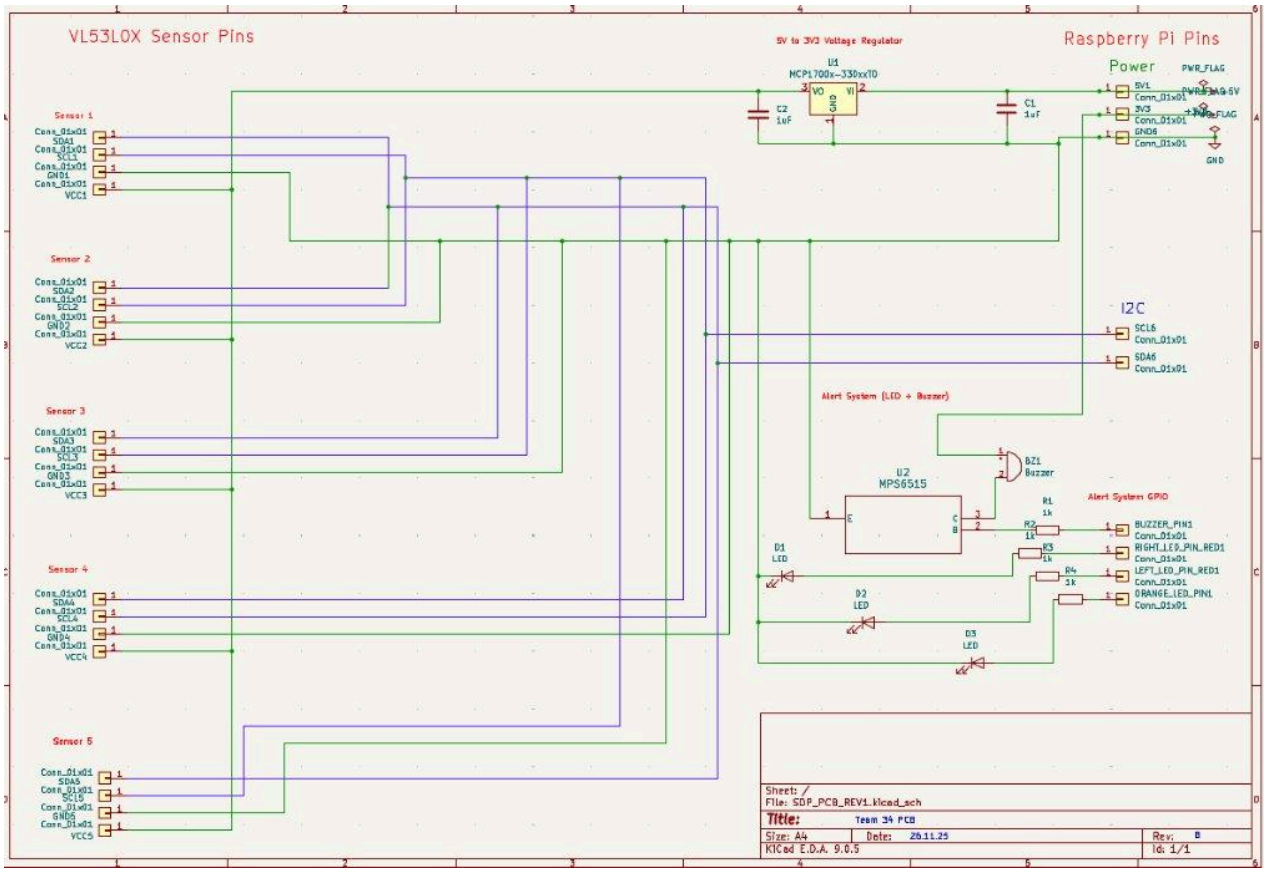

Figure 2: PCB Schematic Diagram

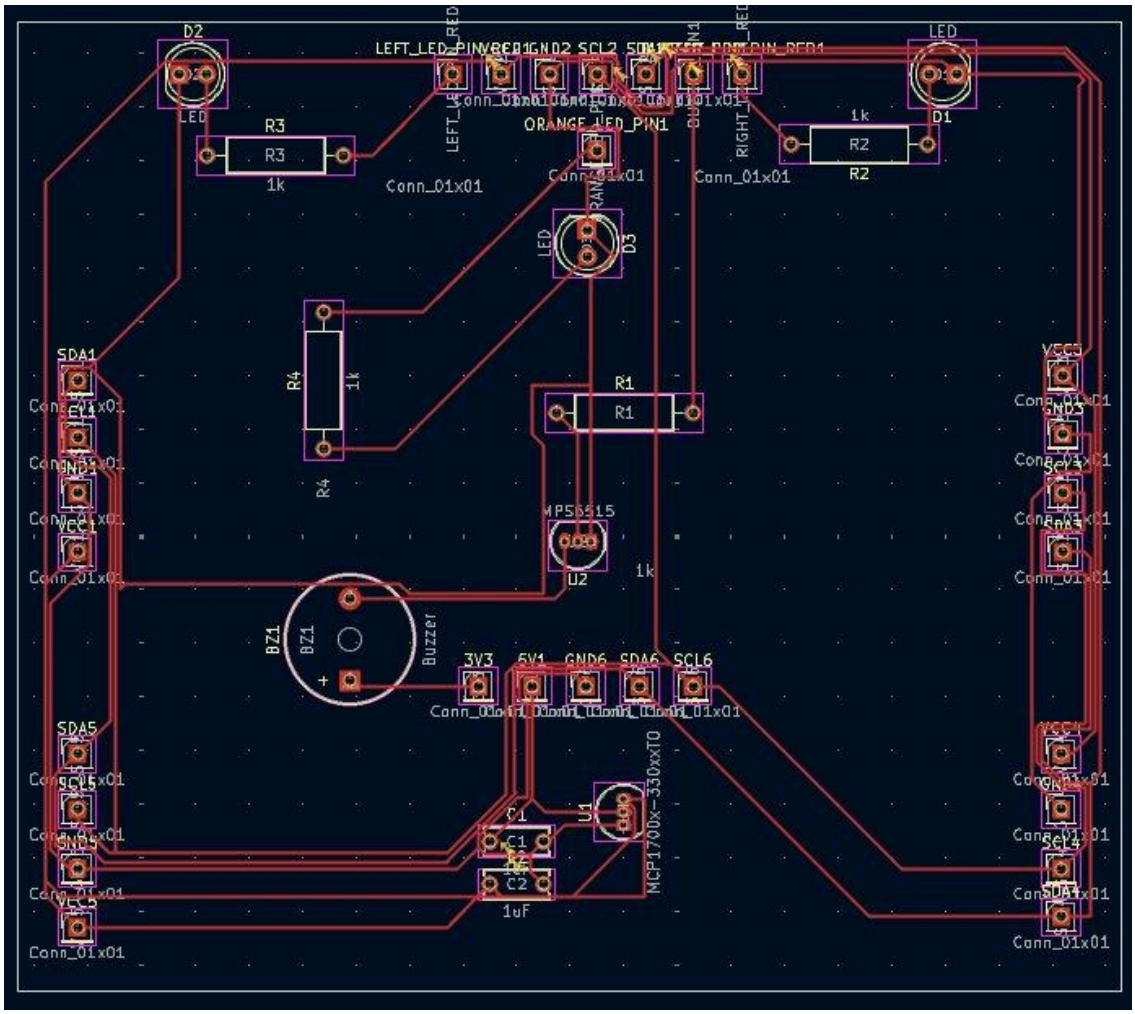

Figure 3. PCB Routing Diagram

C. Software Design Overview

Our software system processes live input from two primary sensing components: the front-facing AI camera module and multiple side-mounted proximity sensors. The camera provides a continuous video stream that is analyzed in real time using the YOLOv8n detection model. Each frame is processed on the Raspberry Pi using Python, OpenCV, and the Ultralytics YOLO framework, producing bounding boxes, detection confidence, and estimated animal distance.

This allows the software to identify potential hazards directly in front of the vehicle.

In parallel, the system reads distance measurements from the VL53L0X ToF sensors mounted on the sides and front of the car. The side sensors monitor adjacent lanes and blind-spot regions, while the front ToF sensor provides an additional distance check to validate or supplement the camera's detection results. For our MDR, in-place of the VL53L0X, we used HC-SR04 Ultrasound sensors.

All sensor inputs are connected in the Raspberry Pi to determine the appropriate alert state. When an animal is detected within a potentially dangerous range or when a nearby vehicle is detected in a blind spot, the software activates the alert system. This includes illuminating red and orange LEDs on the corresponding side of the vehicle and triggering an auditory buzzer. These alerts will warn the driver against unsafe lane changes or signal the need for caution when approaching an animal ahead. If the detected hazard becomes critical, the software also triggers the vehicle's braking mechanism through a PWM control signal to safely slow or stop the car.

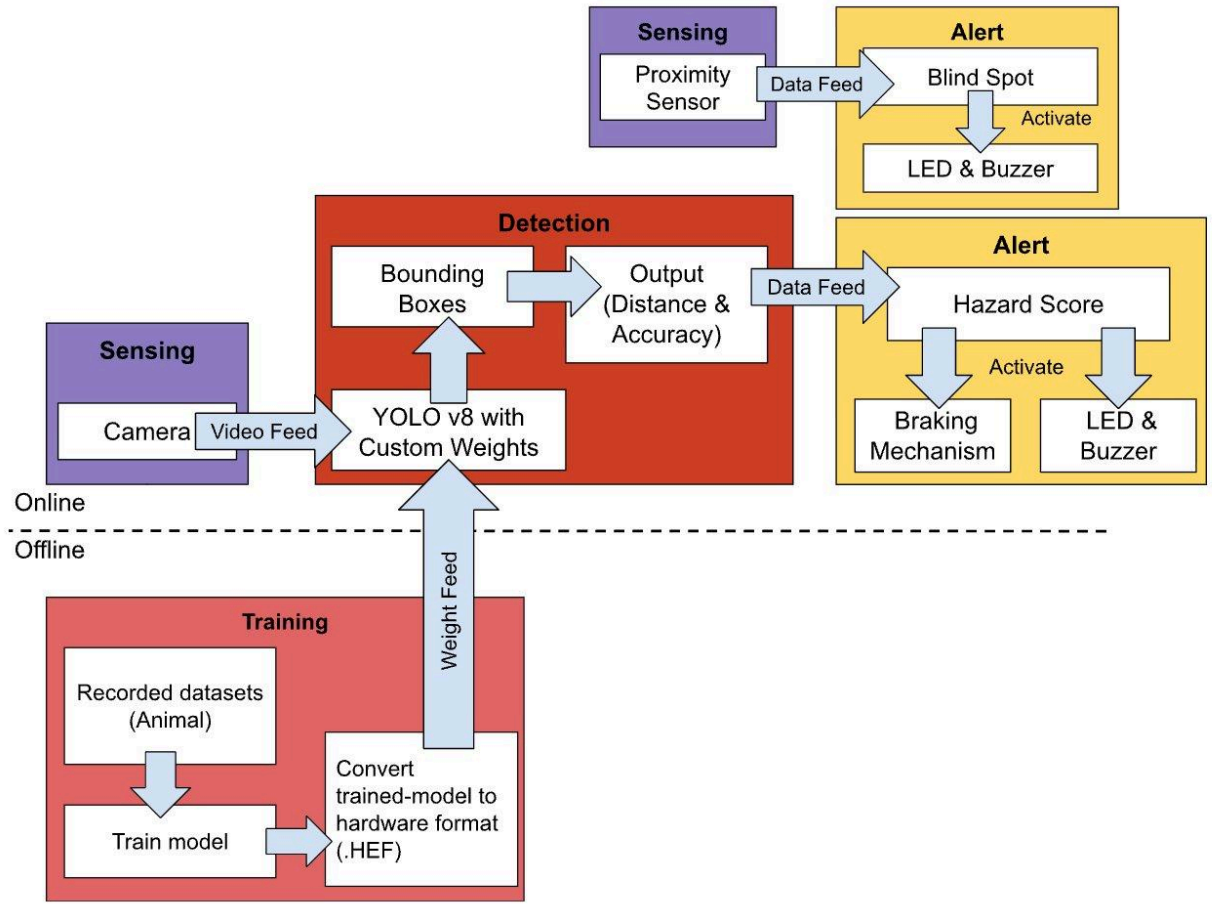

Figure 4: Software Block Diagram

The Prototype

I. *A. Prototype Overview*

The current prototype demonstrates a scaled proof-of-concept system designed to detect animals on the road and assist in preventing collisions. The system integrates an AI camera, proximity sensors, and a Raspberry Pi 5 for processing. The Pi runs a lightweight YOLOv8n detection model that identifies deer-sized animals in real time and outputs bounding boxes with confidence values. Each subsystem has been developed and demonstrated at MDR. The camera subsystem streams live frames to the Pi, the detection subsystem produces producing bounding boxes, detection confidence, and estimated animal distance, the alert subsystem which takes data from the side sensors for blind spot detection, and the camera for animal detection, drives the LEDs

and a buzzer, and the braking/deceleration subsystem slows or stops the robot car when an *Urgent* or *Caution* condition is triggered, with the help of the front sensor.

The prototype is built on a robot car chassis with PWM motor control for variable speed.

<u>During MDR the following subsystems were demonstrated:</u>

1.) AI Detection System

- Front-facing AI Raspberry Pi Camera captured live frames for real-time processing
- YOLOv8n model was used to detect replica animals within each frame
- Bounding boxes and confidence scores were generated for all detected animals , and distance to the animal was estimated based on bounding-box size
- System was not yet interfaced with the alert subsystem; instead, detection results were represented through a simulated hazard score, which will later be computed using combined inputs from the all sensor systems

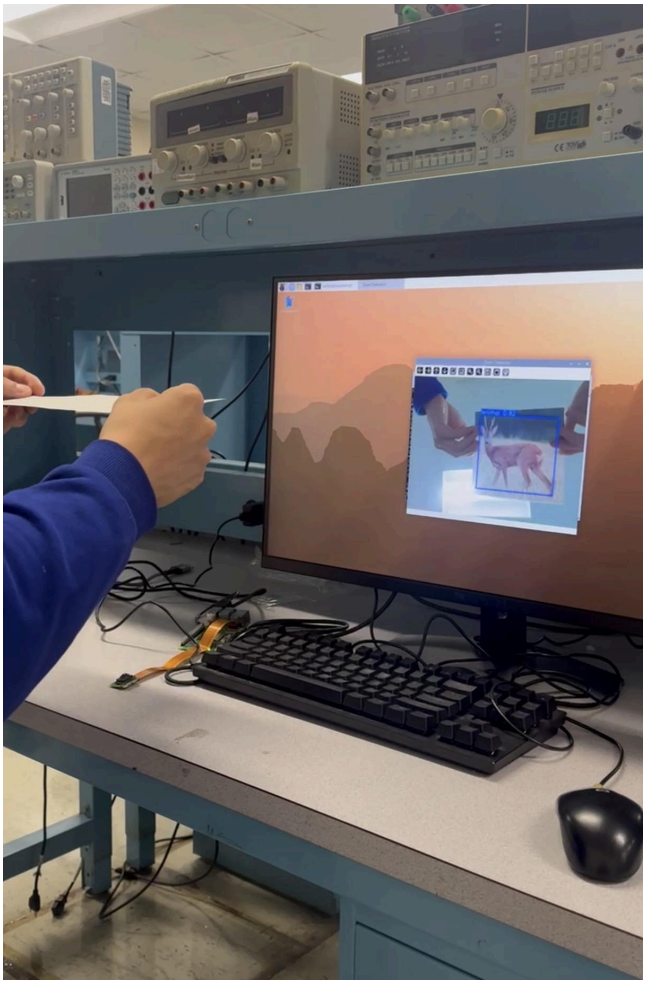

Figure 5. AI Detection System Prototype Setup for MDR

2.) Sensor and Alert System on Breadboard

- Sensor System consisted of 3 Ultrasonic HC-SR04 Sensors used as “simulation” of our real VL53L0X sensors
- Alert System consisted of 4 LEDs (two red, two orange) and a buzzer
- Proximity Sensing was demonstrated by simply placing object close to sensors, showing that the proper LED turns on the proper side and that the buzzer sounds
- System wasn’t interfaced to camera, so camera inputs were “simulated” with hazard score to alert system

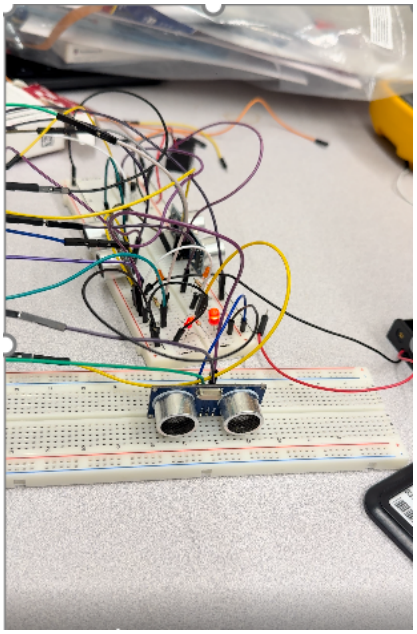

Figure 6. Alert System Prototype Setup for MDR

3.) Basic Braking Mechanism

- The ESC's PWM and GND was cut and reconnected to the Raspberry Pi using a breadboard, with the PWR wire remaining connected to the car's NiMH battery pack
- Using a 50 Hz standard RC control PWM signal, and changing pulse width using Python script, we implemented two basic functions, one for hard braking and one for slowing down/deceleration

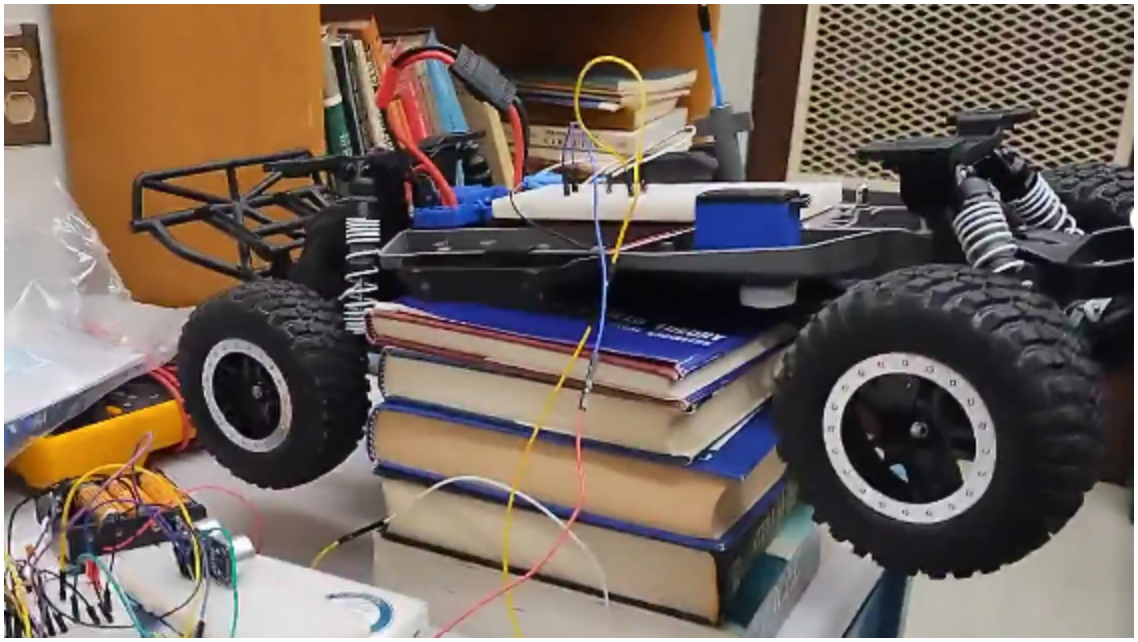

Figure 7. RC Car Speed Control Demo- Setup

The system will later combine these modules into a unified hazard-response pipeline during integration for the final demo.

A. B. List of Hardware and Software

Hardware

Traxxas Slash 1/10 2WD - $280

https://traxxas.com/58034-8-110-slash-xl-5-2wd-short-course-truck-w-tq-radio

Raspberry Pi 5 (8 GB) - $80

https://www.digikey.com/en/products/detail/raspberry-pi/SC1432/21658257?gclsrc=aw.ds&gad_source=1&gad_campaignid=20228387720&gbraid=0AAAADrbLlj75WZUgxC6MxfI4UXjdsuDt&gclid=Cj0KCQjwmYzIBhC6ARIsAHA3IkQdiJ2tZ4qoQTD63tLhYB6GWUdN79uPeMW-FZ8BHtRUyzH4rYT_FhEaAjNwEALw_wcB

Pi Camera Module 3 (standard, later compatible with NoIR) - $77 (Alternative: Wide-angle Camera Module Arducam 1080p - $40)

https://www.adafruit.com/product/6009?gad_source=1&gad_campaignid=21079227318&gbraid=0AAAADx9JvSEGXA8Y3aL-24SmtILgiOZ1&gclid=Cj0KCQjwmYzIBhC6ARIsAHA3IkRHX4WWqVXb91KcoREj-AQbOHy4Y_7St4L7IeYNlBSLlWq766JufEMaAok2EALw_wcB

VL53L0X Time-of-Flight laser ranging sensor break out boards (x5) - total around $15

https://www.amazon.com/Qoroos-VL53L0X-Breakout-GY-VL53L0XV2-Measurement/dp/B0DP6893DS/ref=sr_1_1_sspa?crid=1OXWTBGWXF700&dib=eyJ2IjoiMSJ9.mt2neMEiDHu3alHNS1n_YZumP1IPbZjHRKYYKDROjm0Q1N-4UxUlN1mfiMGfjXbu_jHRpwpnK5aIF3aG1fcKTkckhLC2UTrQXwOJZhSsESouo7209FeGWwprPs-YBY9rTkhNUblY13nYZdKJja9_J_kn06QplONAvoJibHCWI_GuzOu3Wa0ZfCMFlJEysWBfY6AL1C8np2-qiMaswe4bdD_C2GihEL51KF0J4fmdX-Y.vybkzVgUx17yLoeAh2mK_LhC_oobx5sIWUkr1I0uv-Y&dib_tag=se&keywords=vl53l0x&qid=1765483650&sprefix=vl53l0x%2Caps%2C124&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

- LEDs (Red/Orange) (x4) - using from M5
- Buzzer - using from M5
- MPS6515 (x1) - using from M5
- MCP1700 (x1) using from M5

- Custom PCB for interfacing sensors, LED, buzzer with raspberry pi, as well as power distribution for sensors

Wiring (per buzzer):

- Buzzer + → **Pi+3V3**
- Buzzer − → **MPS6515 Collector**
- GPIO → **MPS6515 Base**
- **MPS6515 Emitter - Pi GND**

Wiring (per Sensors)

- VL53L0X SCL/SDA → **Pi SCL/SDA I2C Bus**
- VL53L0X VCC → **Vout of MCP1700 regulator**
- VL53L0X GND → **Pi GND**
- **MCP1700 Vin → Pi +5V**
- VL53L0X GPIO1/XSHUT → **Pi GPIO**

Wiring (LEDs)

- Typical LED wiring to a GPIO, with/ resistor

Software

- Operating System: Raspberry Pi OS (64-bit)
- Programming Language: Python 3
- Detection Model: YOLOv8n for real-time animal detection and bounding-box extraction
- Computer Vision Library: OpenCV for camera frame capture, preprocessing, and visualization
- Distance Estimation: Python scripts compute approximate distance based on bounding-box size
- Sensor Integration: Built-in API for VL53L0X Time-of-Flight sensors for front distance measurement and side blind-spot detection
- Alert Control: Software implementation that triggers LEDs and buzzer based on fused hazard conditions from camera and sensors
- Braking Control: PWM signal generation from the Raspberry Pi for controlling deceleration or emergency braking
- Custom Scripts: Modules for detection handling, sensor processing, hazard scoring (future), and system coordination

II. *C. Custom Hardware*

A custom PCB is under design to distribute power to the sensors and to interface Raspberry PI GPIO and data lines with the sensors, LED, and buzzer.

- GPIO and I²C headers for sensors and alert devices.
- Separate conductive paths for data lines and power
- Board design was presented for MDR, not yet manufactured

D. Prototype Functionality

By MDR, Team 34 successfully demonstrated each major subsystem of the Animal-on-Road Detector operating **independently** on the robot car platform. These demonstrations established technical feasibility for sensing, detection, alerting, braking, and logging. By CDR, these subsystems will be integrated into a unified real-time pipeline (Camera → Detection → Hazard Score → Alert + Log → Braking).

Camera & Detection (Quan, Thinh)

The AI camera module was connected to the Raspberry Pi 5 and streamed real-time video frames to the system. A YOLOv8n-based detection model successfully processed test animal images, drawing bounding boxes, reporting confidence scores, and estimating distance based on bounding-box size. Detection accuracy on small test datasets met the MDR target (≥ 85%). Distance estimates from vision were demonstrated and validated for later use in hazard-score computation during the CDR phase.

Alert System (Arjun, Julia)

A fully functional distance-based alert system was demonstrated using real front, left, and right ultrasonic sensors. The system performs real-time hazard evaluation on the Raspberry Pi and activates four directional LEDs and a buzzer based on proximity thresholds:

- *Caution*: orange LED(s) corresponding to sensor direction

- *Urgent*: red LED(s) plus buzzer

  During MDR, the alert system operated independently of AI detection. Integration of AI-derived hazard scores with sensor data is planned for CDR.

Braking System (Arjun, Syed)

The braking and speed-control subsystem was demonstrated using PWM control of the RC car's ESC. Python functions (slowDown() and hardBrake()) were implemented to command deceleration or braking by adjusting pulse widths sent to the ESC. The system successfully demonstrated acceleration, power cutoff, slowing, and braking behavior. Full integration with AI-based hazard scoring and sensor fusion will occur during CDR.

Event Logging (Julia)

An event-logging framework was demonstrated that records timestamp, detection confidence, alert level, and sensor-measured distance. This logging pipeline verifies synchronization between sensing, alerting, and detection outputs and will be expanded during CDR to capture integrated system behavior.

E. Prototype Performance

During MDR bench and controlled testing, each subsystem operated stably and met preliminary performance goals. The vision subsystem demonstrated real-time processing with detection accuracy ≥ 85% on test datasets. The alert subsystem consistently triggered correct LED and buzzer patterns based on distance thresholds, with clear directional indication. The braking subsystem reliably responded to software commands to slow or brake the vehicle. Full end-to-end latency, false-positive rate, and braking-before-impact verification will be formally measured after subsystem integration during the CDR phase.

Conclusion

At MDR, Team 34 successfully demonstrated independent camera, detection, sensor, alert, braking, and logging subsystems, establishing the technical feasibility of the Animal-on-Road Detector. These demonstrations validate the system architecture and confirm readiness for

integration. By CDR, these modules will be combined into a unified real-time pipeline featuring AI-based hazard scoring, sensor fusion, directional alerts, event logging, and autonomous braking. The final system aims to provide a complete animal-detection and collision-avoidance solution that enhances driver awareness and actively reduces collision risk through timely alerts and controlled deceleration.

Acknowledgment

Team 34 thanks Professor Eslami, Tessier, Pishro-Nik, and all Senior Design staff at the University of Massachusetts Amherst for their guidance and resources throughout the project. Special thanks to the ECE department for providing equipment support and to teammates who assisted with testing and data collection.

Bibliography

[1]“Bridging the gap: How the U.S. is starting to address wildlife-vehicle collisions - Smart Growth America,” *Smart Growth America*, 2024. https://www.smartgrowthamerica.org/knowledge-hub/news/bridging-the-gap-how-the-u-s-is-starting-to-address-wildlife-vehicle-collisions/

[2] S. Santhanam et al., “Animal Detection for Road Safety Using Deep Learning,” 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nov. 2021.

[3] E. Adams, “Volvo’s Large Animal Detection System Spots Moose, Deer, and Hits the Brakes,” WIRED, Jan. 27, 2017.

[4] Volvo Cars Toronto, “How Does the Volvo Animal Detection System Work?” Volvo Cars Toronto, 2022.

[5] “Detection of Obstacles with Assistance at Risk of Collision,” Volvocars.com, Oct. 17, 2024.

[6] “Pedestrian, Cyclist & Large Animal Detection,” Volvo Customer Help, accessed Oct. 2025.

References

[1] Ultralytics, “YOLOv8 Documentation,” https://docs.ultralytics.com/, accessed 2025.

[2] Raspberry Pi Ltd., “Raspberry Pi 5 Product Brief,” 2024.

[3] STMicroelectronics, “VL53L0X Time-of-Flight Distance Sensor Datasheet,” 2023.

[4] HC-SR04 Ultrasonic Sensor Datasheet, Elecfreaks / Generic Manufacturer.

[5] Traxxas, “XL-5 Electronic Speed Control User Guide,” Traxxas, 2023.

[6] OpenCV Documentation, https://docs.opencv.org/, accessed 2025.

Appendix

Design Alternatives

Before finalizing our design, the team explored several options for the main components of the system. Our goal was to balance performance, cost, and ease of integration on the robot car platform.

Processing Unit

- Alternatives: Raspberry Pi 4, NVIDIA Jetson Nano, Arduino Mega.
- Decision: Raspberry Pi 5 (8 GB) was chosen for its strong performance, USB-C power input, and ability to run real-time object detection.
- Reason: Faster than Pi 4, cheaper and simpler than Jetson Nano, and more capable than Arduino.

Object Detection

- Alternatives: Haar Cascades, MobileNet-SSD, YOLOv8n.
- Decision: YOLOv8n was selected for its good accuracy and real-time speed on the Pi 5.
- Reason: Easier to train and more accurate for animals than older models.

Sensor Choice

- Alternatives: LiDAR, IR sensor, Ultrasonic (HC-SR04), Laser ToF (VL53L0X).

- Decision:
  MDR: Ultrasonic sensors (HC-SR04) used for distance-based alert demonstrations.
  CDR: Laser time-of-flight (VL53L0X) sensors selected for improved accuracy and sensor fusion.

- Reason: Ultrasonic sensors enabled rapid MDR prototyping and validation of alert logic.
- VL53L0X sensors provide higher precision, compact size, and reliable I²C integration for final system deployment.

Alert and Braking

- Alternatives: Single alert, LED-only, Relay stop.
- Decision: Two-level alert (LED + buzzer) and PWM motor braking.
- Reason: Provides early warning and smooth slow-down, similar to real vehicle behavior.

Camera Module

- Alternatives: Pi NoIR camera, Standard Pi Camera Module 3.
- Decision: Standard Pi Camera Module 3 for daytime testing.
- Reason: Reliable and fits current goals; NoIR can be added later for night testing.

Technical Standards

This project follows a design approach that prioritizes the use of standardized hardware, software, and communication protocols to ensure technical reliability, interoperability, and safety.

A. A desirable characteristic of prototype design is to use standardized hardware and software for technical reliability and soundness; e.g., WiFi, Bluetooth and Unix are common standards. Together with numerous worldwide standards associations (for example, FAA, FCC and OSHA), IEEE has their own https://standards.ieee.org/ - a partial listing of IEEE standards are provided at https://en.wikipedia.org/wiki/IEEE_Standards_Association .

B. To demonstrate your awareness of engineering standards:

This project follows a design approach that prioritizes the use of standardized hardware, software, and communication protocols to ensure technical reliability, interoperability, and safety.

Standardized Hardware and Corresponding IEEE Standards

Hardware / Interface	Standard / Organization	Description and Relevance
Raspberry Pi 5 (8 GB)	IEEE 802.3 (Ethernet), IEEE 802.11 (Wi-Fi)	The Raspberry Pi 5 supports Ethernet and Wi-Fi communication under IEEE standards for local network connectivity and data transmission during model training or remote logging.
I²C & GPIO Interfaces	NXP I²C-Bus Specification	Used for communication with ultrasonic sensors (MDR) and VL53L0X sensors (CDR).
Pi Camera Module 3 & USB Interfaces	MIPI CSI-2 Standard and USB 2.0 Specification (IEEE Std 1394 family)	Provides high-speed camera data transfer to the Raspberry Pi and standardized video streaming and device enumeration between the Pi and camera.
Custom PCB Design Standards	IPC-2221 (Generic Standard on Printed Board Design)	Ensures trace width, spacing, and current capacity meet industry standards for safe operation.
RC Car Motor Control	IEEE 1675 (Control and	The PWM signal used for

(PWM)	Measurement Standard)	motor control is consistent with IEEE guidelines for control signal generation.

Standardized Software and Frameworks

Software / Framework	Standard / Compliance	Purpose and Justification
Raspberry Pi OS (64-bit Linux)	POSIX (IEEE 1003 Standard)	Ensures cross-platform compatibility and reliable file I/O, process control, and scripting.
Python 3 & OpenCV Libraries	IEEE 829 (Software Test Documentation Standard)	Used for image processing and model testing with structured testing documentation.
YOLOv8 Model (Ultralytics)	Conforms to IEEE 1012 (Software Verification and Validation)	Machine-learning modules were trained and verified according to IEEE best practices for data validation and testing accuracy.

Testing Methods

1. Camera and Detection Module

- Purpose: Validate that YOLOv8n detects animal targets at expected distances and sizes.

- Method: Printed or digital animal images are presented to the camera at scaled distances consistent with the robot car platform. Bounding-box size, confidence, and estimated distance are logged per frame.
- Block Tested: Detection (Computer Vision).
- Data Collection: CSV log with timestamp, bounding-box height, confidence, and distance.
- Expected Outcome: Accuracy ≥ 85%, latency ≤ 250 ms, false positives < 5 per hour.

2. Sensor Subsystem

- Purpose: Confirm distance accuracy and sensor threshold response.
- Method: MDR testing used HC-SR04 ultrasonic sensors placed 5–100 cm from obstacles. CDR testing will replace ultrasonic sensors with VL53L0X ToF sensors.
- Block Tested: Proximity Sensor (VL53L0X).
- Analysis: Linear fit and error calculation; error tolerance ≤ ±2 cm.

3. Alert System

- Purpose: Validate two-level alert responses (Caution and Urgent).
- Method: Simulate detections with different hazard scores. Monitor LED and buzzer timing.
- Expected Result: Caution alert = yellow LED + slow beeps; Urgent = red LED + fast beeps within 100 ms of detection.

4. Braking and Deceleration

- Purpose: Verify motor response under Urgent trigger.
- Method: Run car toward target; measure braking distance and time with stopwatch and video analysis. PWM signals generated by the Raspberry Pi control the ESC. Pulse widths: ~1400 µs → slowDown(); ~1300 µs → hardBrake()
- Block Tested: Motor Control via XL-5 ESC.
- Expected Performance: Car slows or stops before collision when object ≤ 2 × car length.

5. Event Logging System

- Purpose: Ensure data synchronization between visual feed and log file.
- Method: Compare on-screen bounding-box events to CSV entries for time alignment.
- Block Tested: Data Logging and System Integration.
- Expected Result: 100% timestamp consistency between visual and log output.

Future Verification (Integration Phase)

Full-system tests will combine all modules to validate:

- Frame rate ≥ 10 FPS under real conditions.
- End-to-end reaction time ≤ 250 ms (from detection to brake signal).
- False alert rate < 5 events per hour.
- Reliable logging of distance, alert level, and braking events for data review.

Project Expenditures

Item	Cost	Anticipated Time of Acquirement
Raspberry Pi AI Camera	$70	Arrived
Raspberry pi 5	$80	Arrived
Pi Cooler	$5	Arrived
Pi SD card	$13	Arrived
Rc Car	$280	Arrived
Unnecessary purchases made*	$32	Arrived
Future Purchases (PCB + VL53L0X breakout boards)	$14 + ??	
TOTAL	$494 + ??	

We need confirmation from Wouter on whether some unnecessary purchases can be returned and refunded.

Project Management

Team 34 is organized into clear, role-based responsibilities, balancing the workload between hardware, software, and system integration tasks. The team collaborates through weekly meetings and a shared GitHub repository for version control. Communication occurs via Slack for day-to-day coordination and Google Drive for document sharing and collaboration.

Team Roles

- **Arjun Rajapur: Project Manager, Sensors & PCB, System Coordination**
  Leads overall project planning, scheduling, and milestone tracking. Oversees sensor selection and PCB design for alert and sensing subsystems, ensuring reliable power distribution and communication between hardware components. Coordinates integration efforts across hardware and software teams.
- **Julia Le: Hardware Integration, Camera & Alert System, Data Collection**
  Responsible for physical system integration on the RC car platform, including camera mounting, sensor placement, LEDs, and buzzer hardware. Develops and tests the alert subsystem, manages data-collection experiments, and interfaces sensors with the Raspberry Pi. Ensures compact, reliable hardware design suitable for real-time testing.
- **Thinh Truong: Machine Learning & Software Integration**
  Leads machine-learning model development and deployment. Trains and fine-tunes the YOLOv8-based animal detection model, implements distance estimation and hazard-score logic, and integrates AI outputs with alert and braking software pipelines for real-time operation.
- **Quan Do: System Testing, Validation & Performance Analysis**
  Designs and executes subsystem and system-level tests to verify accuracy, latency, and reliability. Focuses on debugging, performance evaluation, and validation of detection, alerting, and braking behavior. Supports ML tuning and analyzes logged data to confirm system requirements are met.
- **Syed Abbas: Braking & Vehicle Control, System Integration, Budget Management**
  Designs and tests the braking and deceleration mechanism using PWM control of the RC car’s electronic speed controller (ESC). Implements software functions for slowing and

braking behavior, supports final system integration, and manages project expenditures to ensure compliance with budget constraints.

Beyond the Classroom

This project has pushed each team member to develop practical engineering skills beyond classroom concepts:

- Technical Skills: Machine learning deployment, Python GPIO programming, PCB design in KiCad, and Linux system management.
- Professional Skills: Team coordination, documentation discipline, and iterative testing strategies similar to industry automation projects.
- Resources Used: Ultralytics documentation, Raspberry Pi forums, IEEE Xplore papers on vehicle automation, and faculty office hours.

The team has gained insight into the real-world intersection of hardware integration and AI vision systems, a foundation relevant to fields such as autonomous vehicles, industrial automation, and embedded AI design.