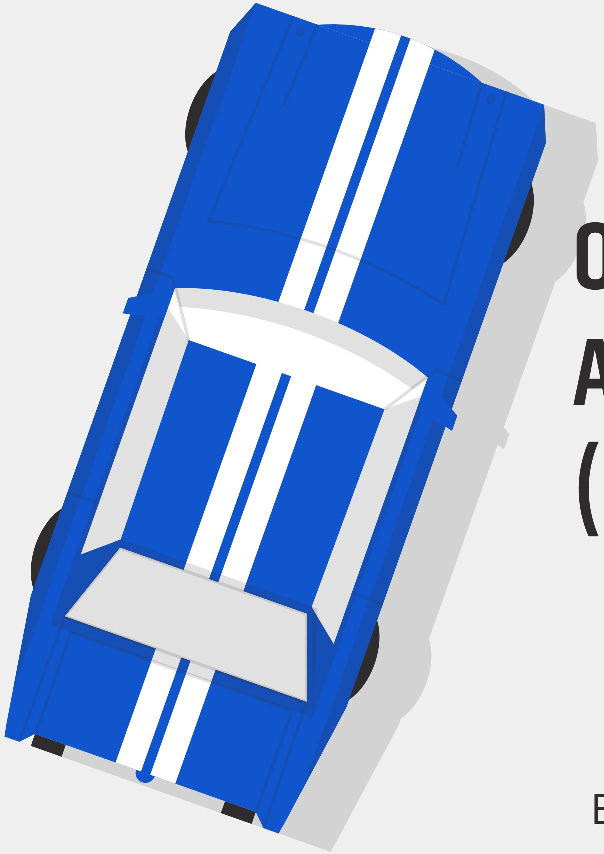
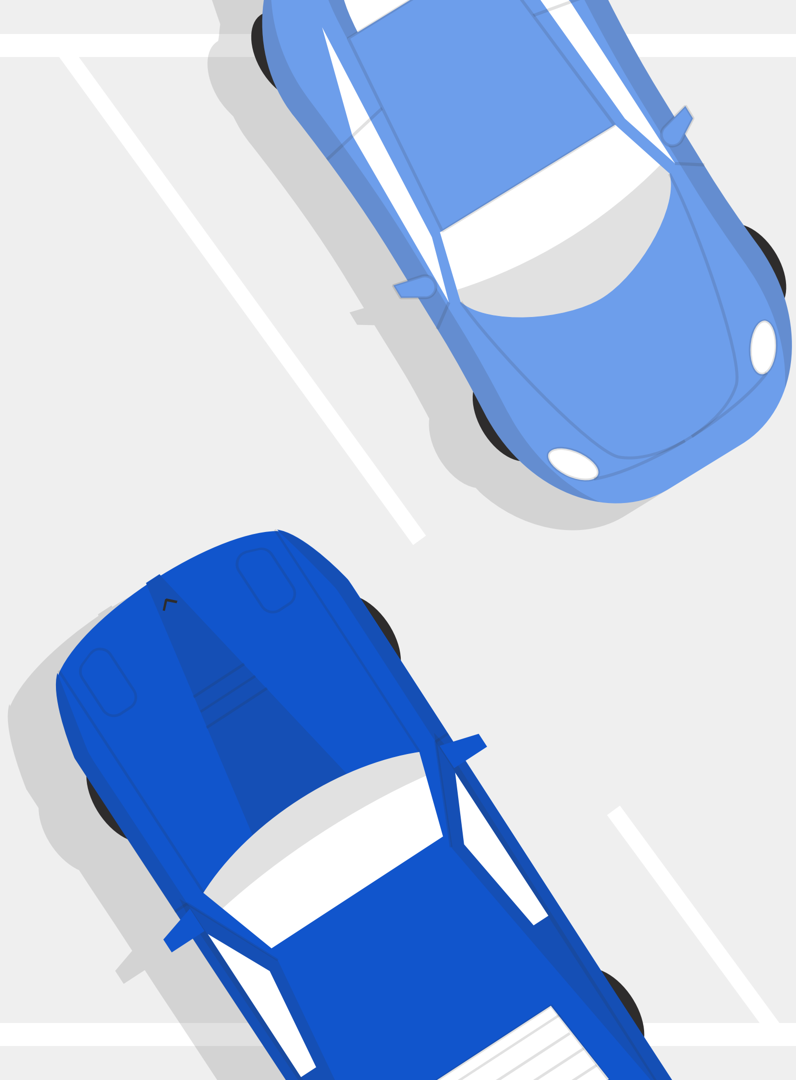


On-Edge Stop Sign Recognition and Alert System (Final Presentation)

Balaji Sathyanarayanan

Table of Contents



01

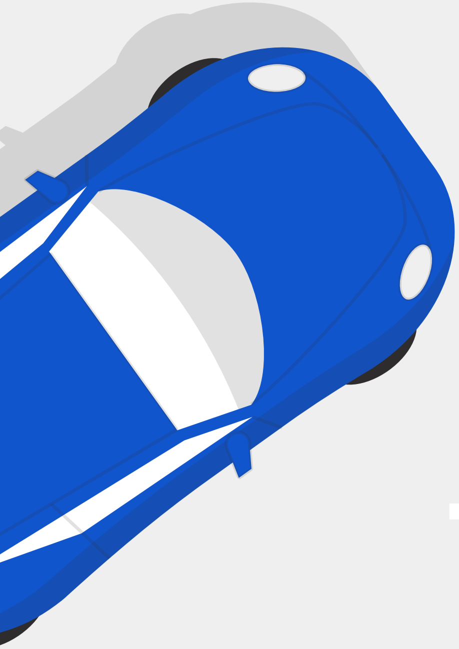
Project Overview

Motivation

- I enjoy driving my car both in India and in Pittsburgh.
- However, I have occasionally experienced challenges in detecting and responding to stop signs in a timely manner.
- This experience has motivated me to design a Stop Sign Recognition and Alert System.
- The goal of this system is to enhance driver awareness and improve road safety.

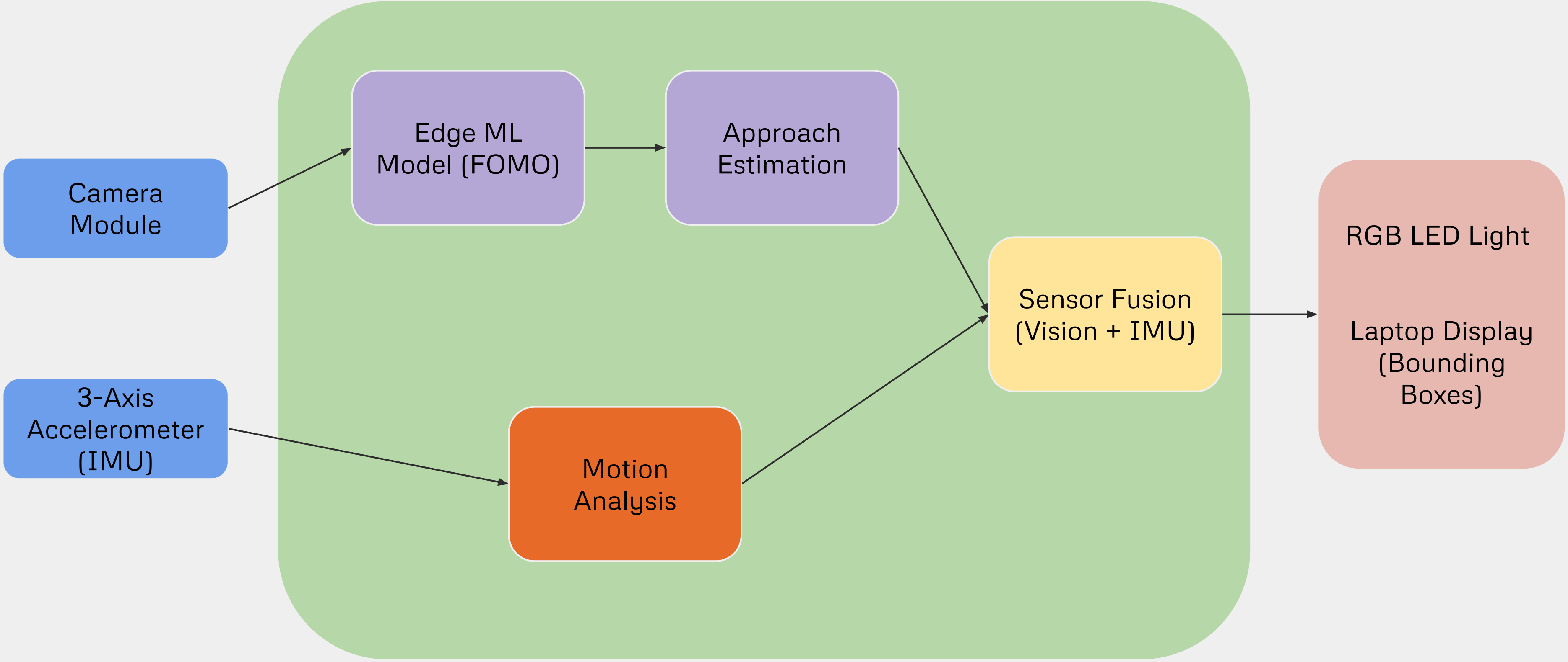
Project Summary

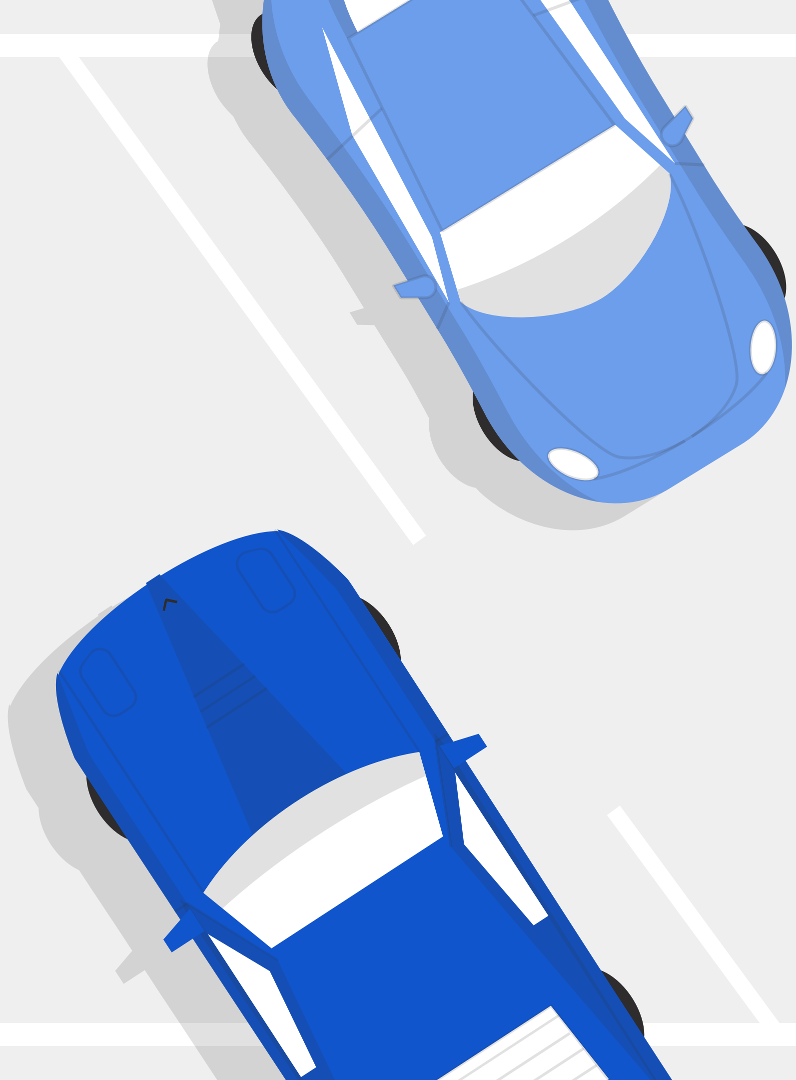
- The system performs real-time object detection and motion analysis directly on a low-power embedded device.
- The prototype first detects stop signs from camera input.
- Then, it uses accelerometer data to check if the vehicle is moving and accelerating/decelerating.
- Finally, the system alerts the user when the vehicle approaches a stop sign too quickly (based on sensor fusion logic).
- Potential stakeholders include automobile manufacturers, driver-assistance system developers, and road-safety organizations.



02

Block Diagram


Camera Module
3-Axis Accelerometer (IMU)
Edge ML Model (FOMO)
Approach Estimation
Motion Analysis
Sensor Fusion (Vision + IMU)
RGB LED Light
Laptop Display (Bounding Boxes)

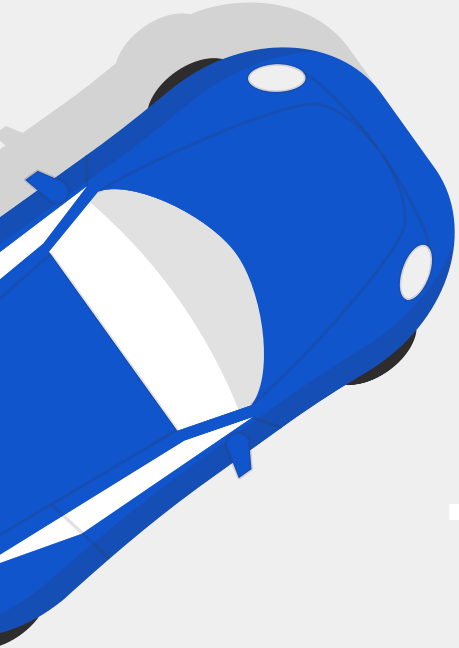


03

Hardware and Sensors

Edge Impulse with Arduino Board

- Board: Arduino Nicla Vision.
- A built-in camera module to capture visual data.
- Built-in accelerometer (IMU) to collect accelerometer readings.
- RGB LED Light to alert the user.
- A laptop with Edge Impulse and OpenMV IDE.



04

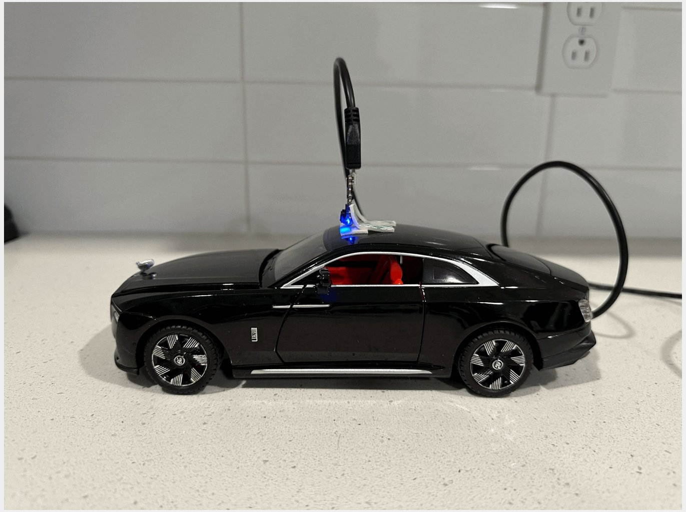
Simulated Environment

Simulated Environment

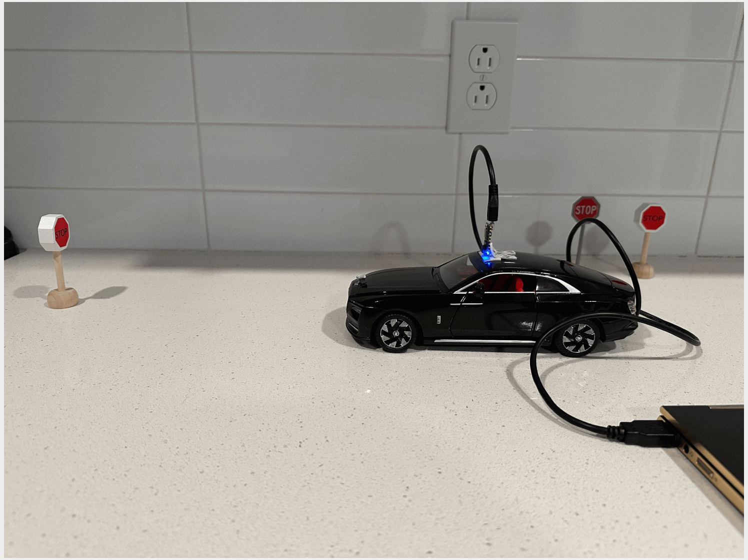
- Due to safety and logistical constraints, the system cannot be tested using a real vehicle.
- Hence, a miniature car and miniature stop signs are used to simulate the testing environment.
- The custom dataset is also collected using this simulated testbed.
- If proof-of-concept works well, the system can be tested in real-world autonomous driving conditions.

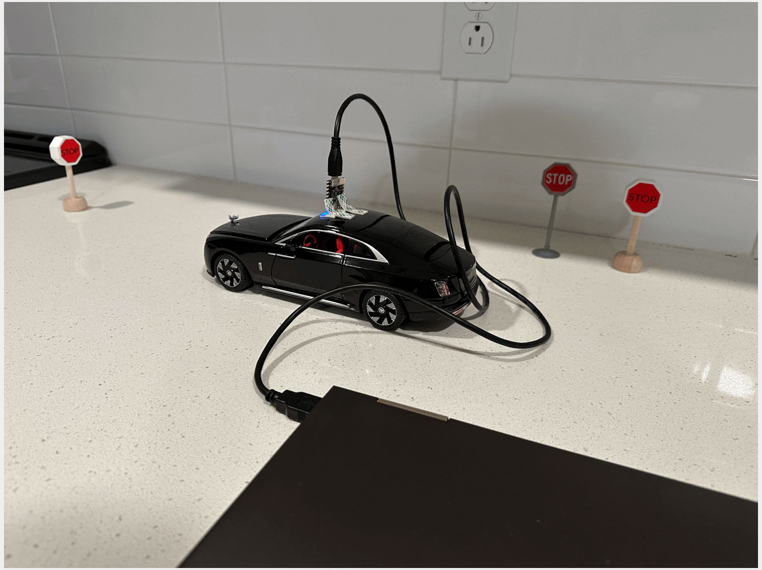
Miniature Stop Signs

Miniature Car

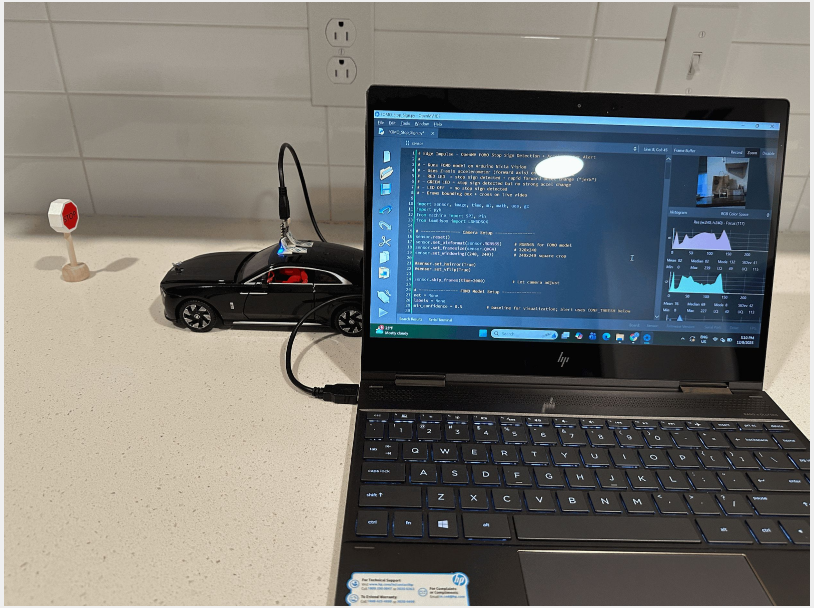


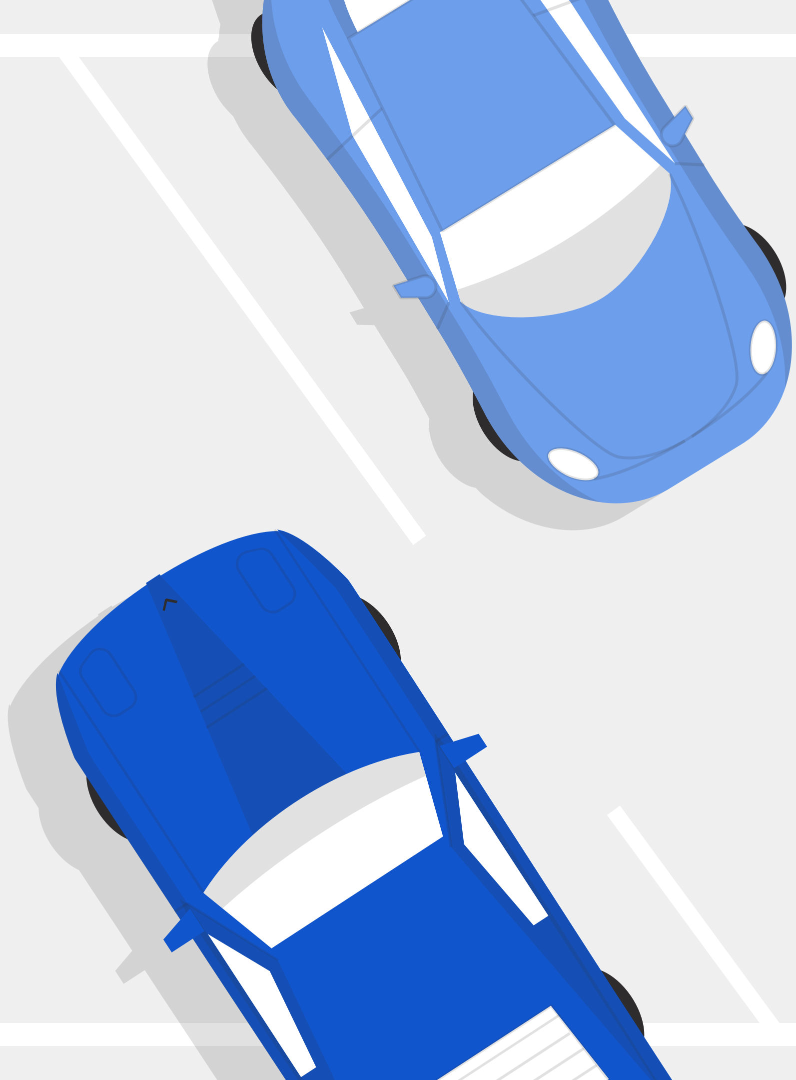
Environment





Environment





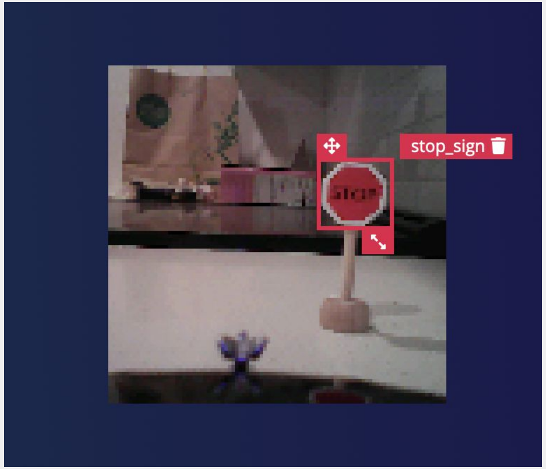
05

Data Sources

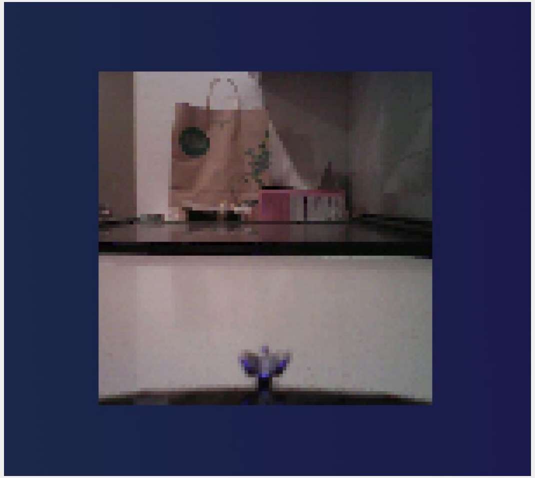
Datasets

- A custom dataset of 400 stop sign images was created. 200 positive images (where stop signs are present) and 200 negative images (background with no stop signs) were collected in the simulated environment.
- The dataset includes varied backgrounds, stop sign sizes, lighting, and environmental conditions.
- To diversify the dataset, 30 positive images from the LISA Traffic Sign Dataset were used.
- The images were annotated on Edge Impulse.
- The dataset was split 80/20 for training and testing respectively.

Custom Dataset

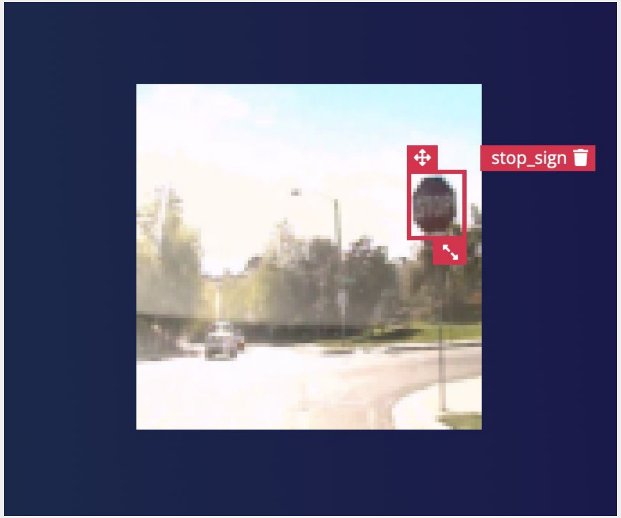


Positive sample

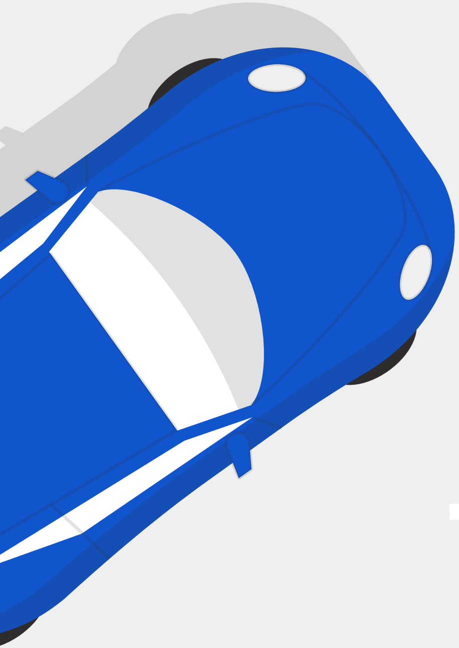


Negative sample

LISA Traffic Sign Dataset

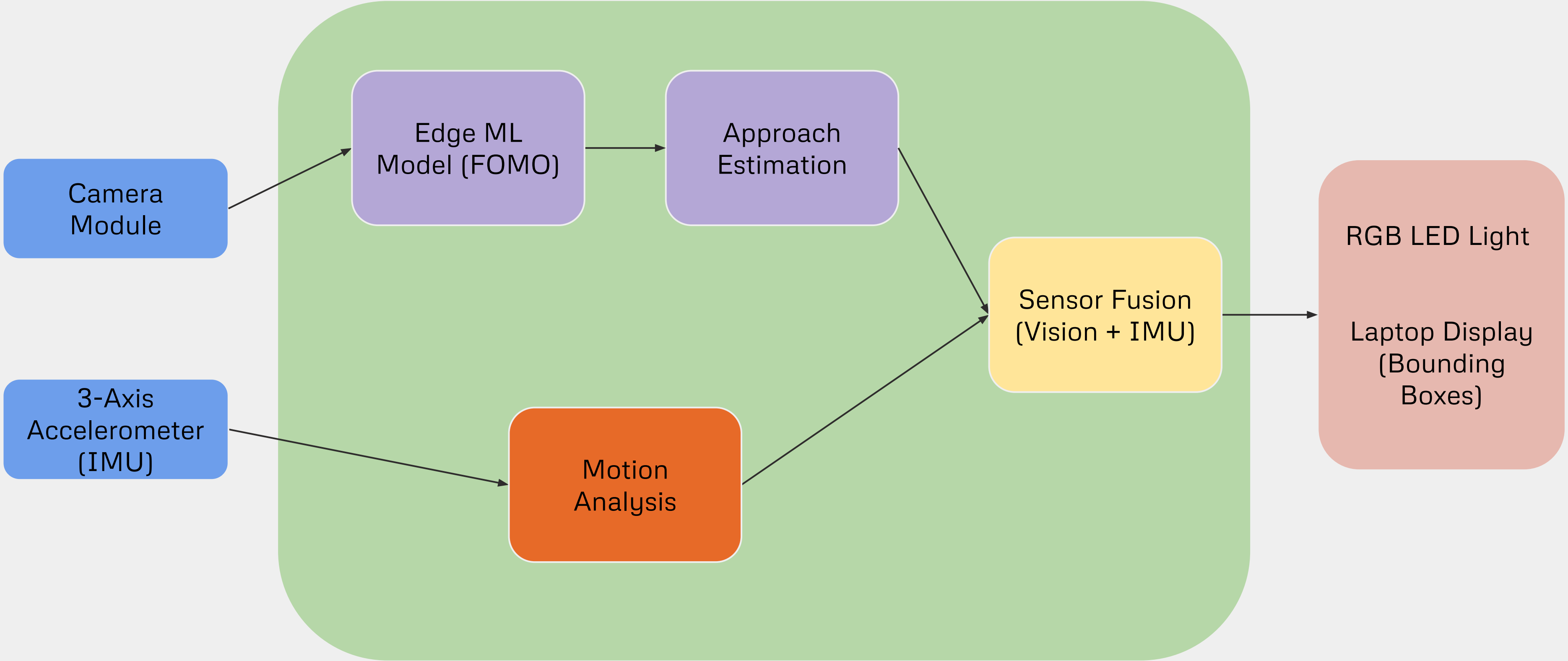


Positive sample



06

ML Model and Alert System



Camera Module
3-Axis Accelerometer (IMU)
Edge ML Model (FOMO)
Approach Estimation
Motion Analysis
Sensor Fusion (Vision + IMU)
RGB LED Light
Laptop Display (Bounding Boxes)

FOMO (Faster Objects, More Objects)

- The Edge Impulse pipeline converts camera frames into 96x96 grayscale images.
- FOMO convolutional neural network is used to extract visual features.
- FOMO is an object detection architecture optimized for microcontrollers.
- It is designed to detect object centroids directly from feature maps.
- It does not compute bounding boxes as in traditional detectors (e.g., YOLO or SSD).
- MCU sends centroid packets to the laptop and OpenMV IDE is used to overlay bounding boxes.

Object Detection (FOMO)

- FOMO with a pre-trained MobileNetV2 backbone was used for object detection.
- The model was trained on 430 images for 100 cycles with a 0.001 learning rate.
- The validation set size was 20% and batch size was 32.
- Input layer = 9216 features, output layer = 1 class.

Model Performance (Training)

Model — Model version: Quantized (int8)

Last training performance (validation set)

F1 SCORE
90.1%

Confusion matrix (validation set)

	BACKGROUND	STOP_SIGN
BACKGROUND	100.0%	0.0%
STOP_SIGN	3.0%	97.0%
F1 SCORE	1.00	0.90

Metrics (validation set)

METRIC	VALUE
Precision (non-background)	0.84
Recall (non-background)	0.97
F1 Score (non-background)	0.90

On-device performance — Engine: EON™ Compiler (RAM optimized)

INFERENCING ...
67 ms.

PEAK RAM USA...
119.4K

FLASH USAGE
81.0K

Test Performance

Results

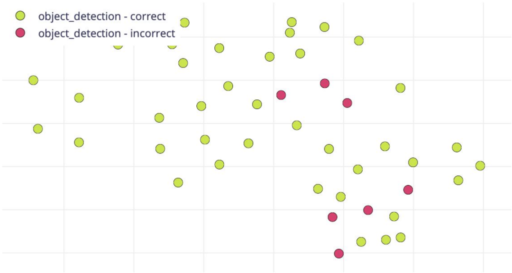
Model version: Unoptimized (float32)

ACCURACY
86.05%

Metrics for Object detection

METRIC	VALUE
Precision (non-background)	0.87
Recall (non-background)	0.93
F1 Score (non-background)	0.90

Feature explorer



Approach Estimation

```python
while True:
    img = sensor.snapshot()
    results = fomo_inference(img)  # EI-generated function

    # Pick best detection, get cx_cell, cy_cell, confidence, active_cells
    cx_cell, cy_cell, det_conf, active_cells = parse_fomo_results(results)

    # ---- APPROACH ESTIMATION ----
    S_t = active_cells                             # size proxy
    g_t = (S_t - S_prev) / dt
    TTC_t = S_t / max(eps, g_t)
    S_prev = S_t
```

Motion Analysis

```
ax, ay, az = read_accel()
ax_filt = ema(ax_filt, ax)

moving_flag = abs(ax_filt) > move_thresh
accelerating_flag = ax_filt > acc_thresh
braking_flag = ax_filt < brake_thresh
```

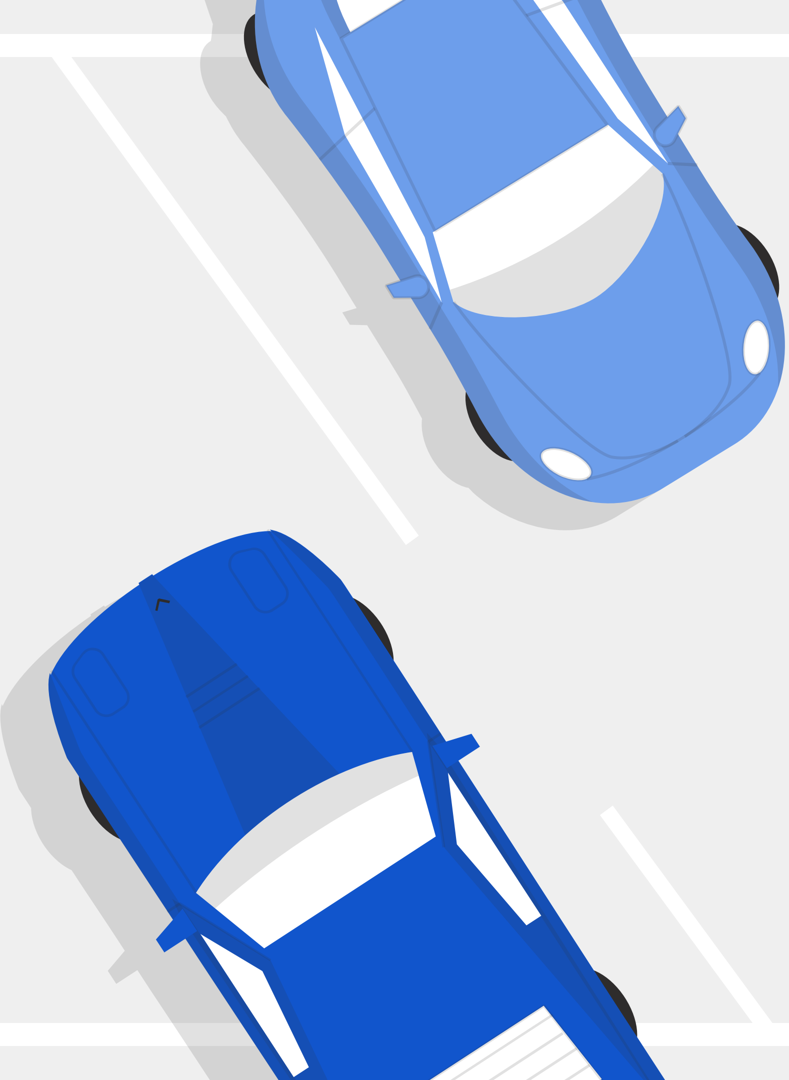
Sensor Fusion

```
cond_det = det_conf >= conf_thresh
cond_ttc = TTC_t <= ttc_thresh
cond_imu = moving_flag and (accelerating_flag or not braking_flag)

if cond_det and cond_ttc and cond_imu:
    danger_count += 1
else:
    danger_count = max(0, danger_count - 1)

if danger_count >= N_consensus and cooldown_ok():
    alert_flag = True
    trigger_led_alert()
else:
    alert_flag = False
```

System Output

- FOMO detects object centroids and the active cells around the centroids directly from feature maps.
- Using this information, bounding boxes were drawn around detected objects in the OpenMV IDE.
- The FOMO model was trained on Edge Impulse and deployed to the Arduino Nicla Vision using the OpenMV library/firmware.
- The code for model inference, bounding box overlay, approach estimation, motion analysis, sensor fusion, and system output were written in MicroPython using the OpenMV IDE.
- The system has two outputs:

1. Live video with bounding boxes.
2. RGB LED Light on the Nicla Vision (No light - no stop sign detected; Green light - stop sign detected and the vehicle is moving at a normal speed; Red light - stop sign detected and the vehicle is approaching the stop sign quickly).

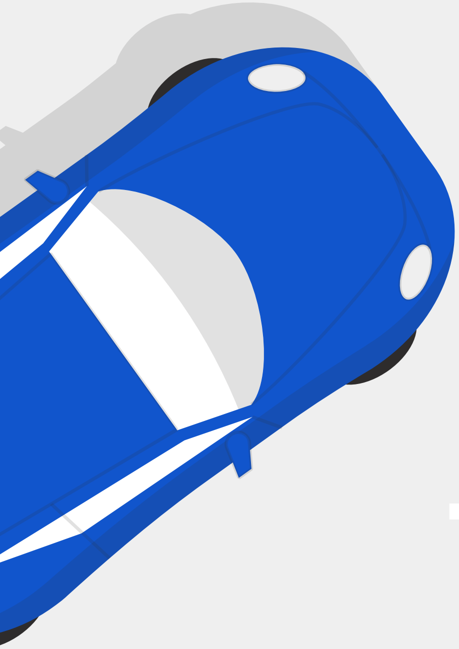


07

Challenges

Challenges

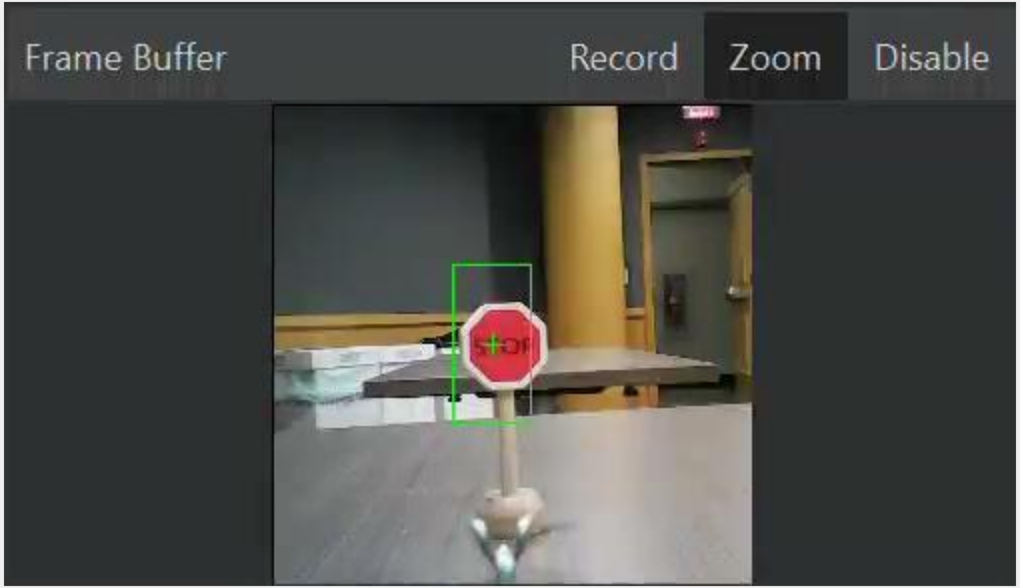
- Data collected using a simulated environment will not exactly match real-world data.
- Hence, the system might not generalize well to real-world traffic scenes.
- Anomalies in accelerometer and visual data are likely to occur.
- A vehicle and a miniature car have different motion profiles.



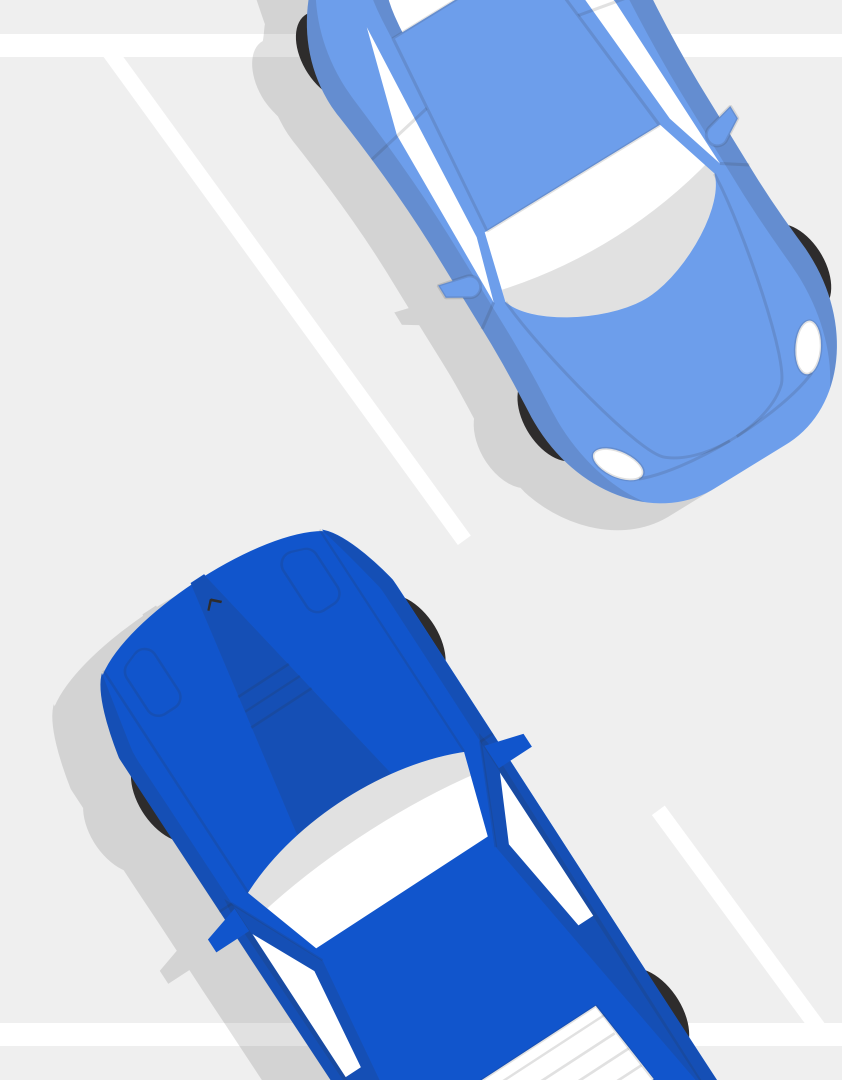
08

Final Demo

Live Video with Bounding Boxes (OpenMV IDE)



Demo

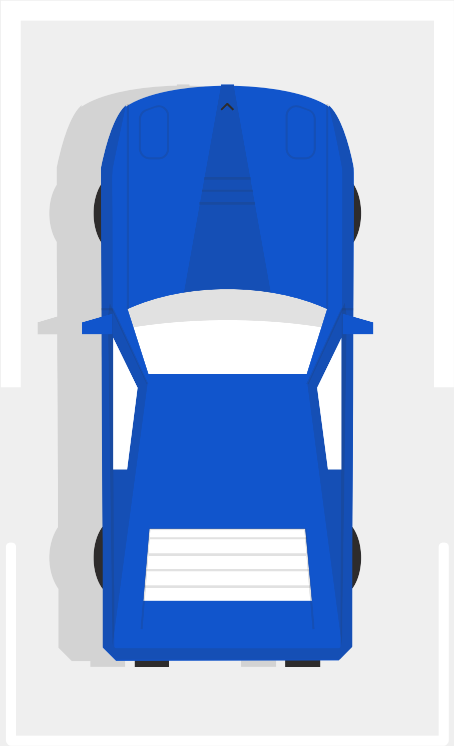


09

Future Work

Future Work

- The system can be tested and deployed in a real-world autonomous vehicle environment.
- The custom dataset can be collected using a real vehicle.
- The size of the training dataset can be increased to around 2000 images.
- More powerful hardware like the Raspberry Pi module can be used.
- Models like YOLO and SSD can be trained and deployed.



Thank You