Real-Time Vehicle Speed Monitoring via YOLOv11 and ByteTrack

Group 24 : 1)Mohammad Hammad – 202311454

2)Nikunj – 20231161

3)Utpal Anand - 20231271

What is the use of Arduino/Microcontroller in this project?

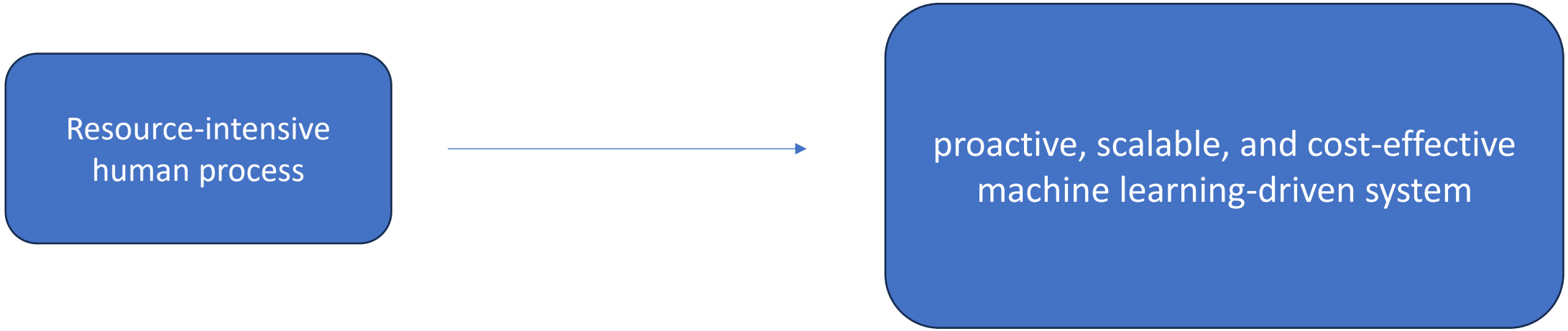
We are using ESP32-CAM (a microcontroller) for streaming video feed to the laptop/control center.

Motivation for the project:

1. Objective: To implement **real-time object detection** within defined areas for critical applications.

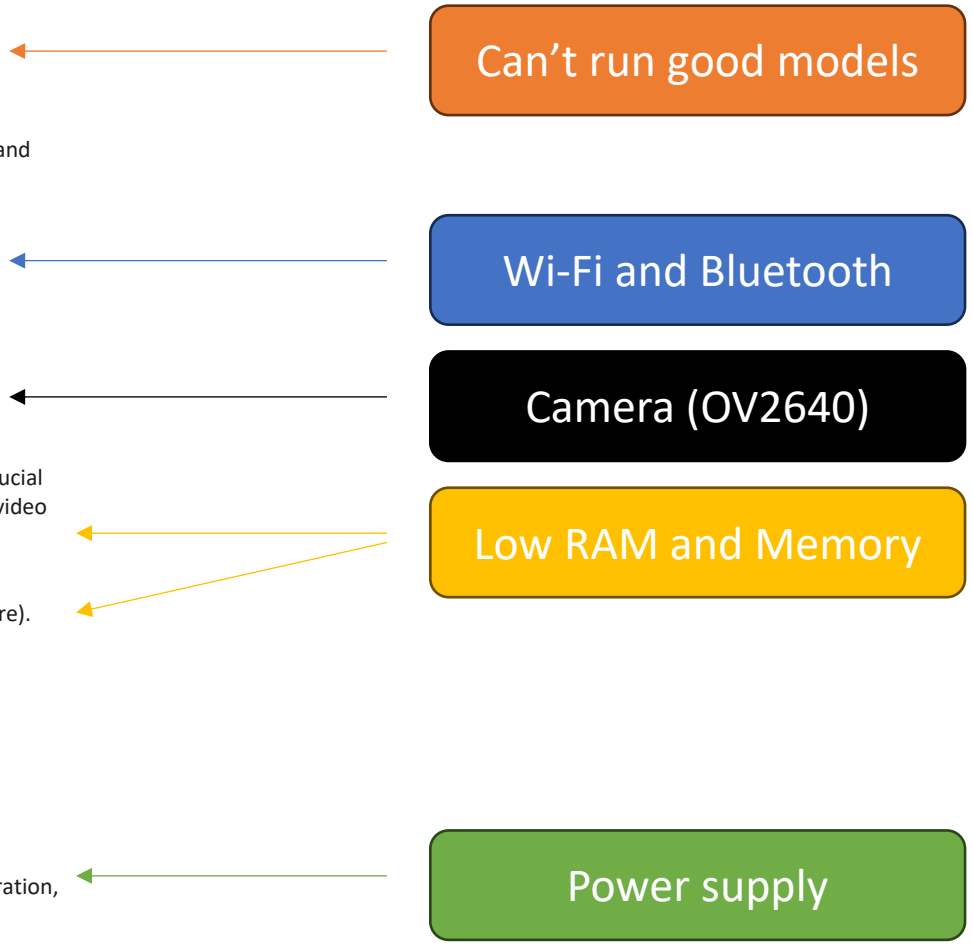
- Focus Areas: Enhancing **security and public safety**
- Focus Areas: Optimizing operations in **industrial and laboratory environments**

2. Scope & Impact: To demonstrate the architecture of modern global surveillance systems and highlight how **high-security sectors leverage Machine Learning (ML)** for automated, efficient detection, thereby **significantly reducing reliance on continuous human monitoring.**

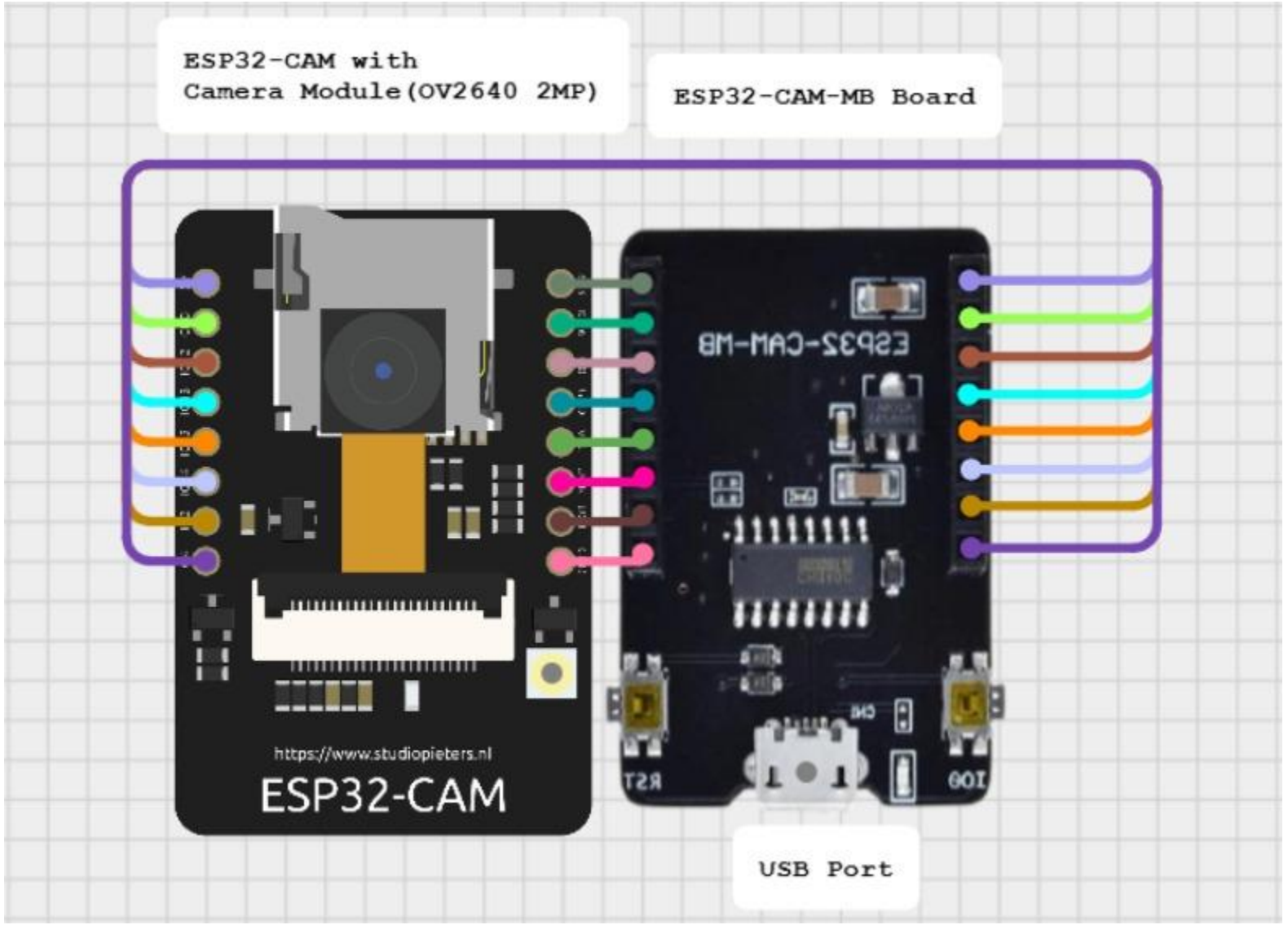


Details about ESP32-CAM and module used

Feature	Detail (AI-Thinker Version)	Notes
Microcontroller	**ESP32-S** (or ESP32-D0WD)	Dual-core, 32-bit LX6 processor.
CPU Clock Speed	Up to **240 MHz**	Sufficient power for video streaming and basic image processing.
Connectivity	**Wi-Fi** (802.11 b/g/n) & **Bluetooth 4.2 LE**	Enables wireless streaming and communication.
Camera Module	**OV2640**	**2 Megapixel** (UXGA - 1600x1200 max resolution). Supports JPEG, BMP, and GRAYSCALE output.
Memory	**520 KB SRAM + 4 MB PSRAM** (External)	The 4MB PSRAM (Pseudo-SRAM) is crucial for buffering high-resolution images/video frames.
Flash Memory	**4 MB** (32Mbit)	Used for storing the program (firmware).
Storage	**MicroSD Card Slot**	Supports up to a 4GB TF card for local image/video storage.
Onboard Peripherals	**Flash LED** (connected to GPIO 4)	Used as a camera flash or general illumination.
Power Supply	**5V** (Recommended) or 3.3V	5V is generally advised for stable operation, especially when using the flash.
I/O Ports	Approx. **9 available GPIO pins**	Many of the ESP32's pins are internally connected to the camera and PSRAM.
Size	Approx. 40.5mm x 27mm	Very small form factor.



For uploading we used ESP32-CAM-MB Board



ML (Deep Learning) Model used:

Model: YOLOV11m

Medium Variant

Version Suffix	Name	Key Feature	Best Suited For
n	Nano	Smallest, highest speed, lowest accuracy.	Edge devices, real-time video on low-power hardware.
s	Small	Good balance of speed and accuracy.	Mobile/Embedded systems.
m	Medium	General-purpose sweet spot for accuracy and speed.	Desktop, cloud applications.
l	Large	Higher accuracy, more parameters, slower speed.	High-performance servers, complex scenarios.
x	Extra Large	Highest accuracy, largest model size.	High-end servers, maximum performance tasks.

Scalability and use cases!

- **Task Variant used:** Detection

Object Detection Variant

Task Variant	Suffix	Description
Detection	None	Standard object localization and classification.
Segmentation	-seg	Localizes objects and outlines them at the pixel level (Instance Segmentation).
Pose	-pose	Detects keypoints on objects (e.g., human joints for Pose Estimation).
OBB	-obb	Uses Oriented Bounding Boxes for objects at an angle (common for aerial imagery).
Classification	-cls	Model for image classification (identifying the entire image content).

Fine Tuning YOLOv11m for traffic

Dataset of around 5000 images

YOLOv11m
20 million parameter
Trained over 20 Epochs
using A30 GPU

Fine tuned YOLOv11m

Details of the training:

Metric	IoU Threshold	Localization Requirement
mAP50	**0.50** (Single value)	Forgiving (at least 50% overlap)
mAP50-95	**0.50** to **0.95** (Averaged over 10 values)	Strict (demands highly precise box placement)

Mean Average Precision (mAP):

a **True Positive (TP)** for mAP50, the IoU must be **0.50 or greater**

```
YOLO11m summary (fused): 125 layers, 20,039,284 parameters, 0 gradients, 67.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        966      13450      0.495      0.564      0.446      0.316
               big bus        210        273      0.751      0.392      0.628      0.465
             big truck        404       1162      0.816      0.456      0.637       0.42
                 bus-l-          8          8     0.0458       0.75     0.0428     0.0181
                 bus-s-         12         12      0.256       0.75      0.372       0.28
                   car        927       8537      0.865      0.699      0.814      0.504
             mid truck        118        257       0.67      0.405      0.399      0.304
             small bus         43         49      0.143      0.143     0.0724     0.0515
           small truck        517       1721       0.72      0.483      0.595      0.385
              truck-l-        266        433      0.461      0.672      0.512      0.393
              truck-m-        331        629      0.395      0.707      0.401      0.303
              truck-s-        147        221      0.337      0.555      0.257      0.172
             truck-xl-        110        148      0.475       0.75      0.626      0.492
Speed: 0.1ms preprocess, 0.8ms inference, 0.0ms loss, 3.1ms postprocess per image
```

Pipeline

MJPEG Stream over HTTP

Power Supply

Speed of the object

Using bytetrack

Object Detected

Object Not Detected

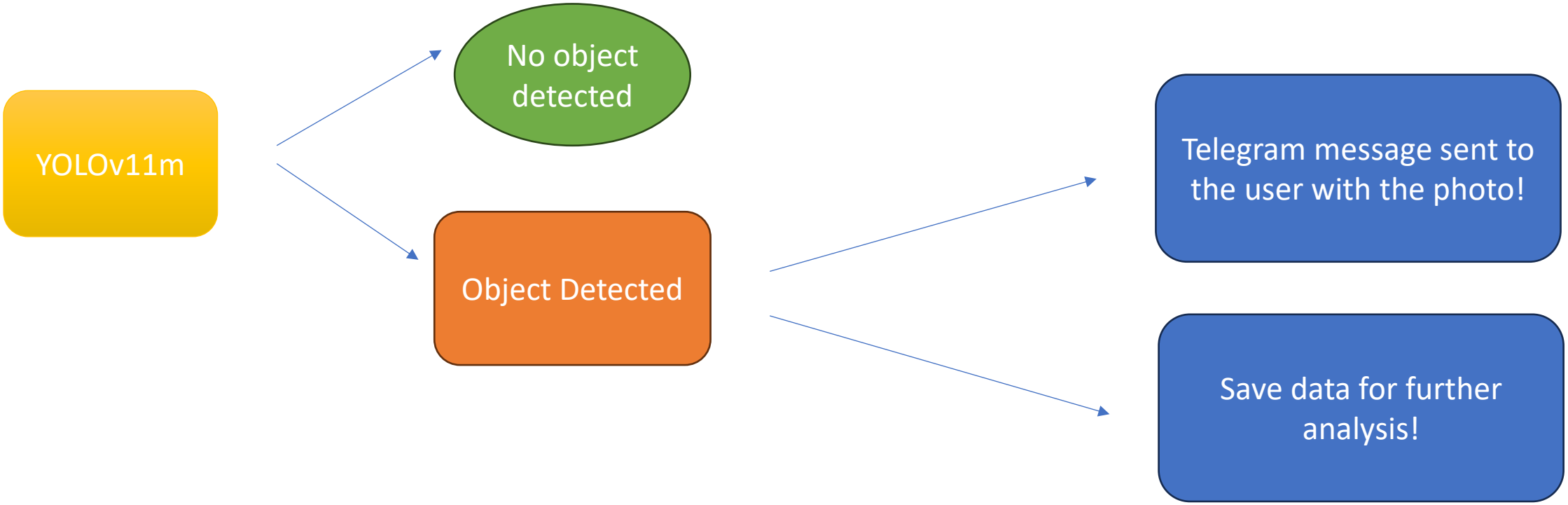
Inference using YOLOv11m

Bytetrack for object tracking after detection:

Aspect	Description
Goal	Robust **Multi-Object Tracking (MOT)** by maximizing the use of detections.
Core Principle	**Two-Stage Association** using both high-score >0.5 and low-score < 0.5 detection boxes.
High-Score Stage	Standard **IoU matching** between existing tracks and high-confidence detections.
Low-Score Stage	Matches remaining **unmatched tracks** to low-confidence detections to recover occluded/blurry objects (reduces ID switches).
Key Advantage	Superior performance in **crowded scenes** and during **occlusions** compared to traditional methods (e.g., DeepSORT) because it re-uses low-score boxes.
Metric Impact	Major improvements in **MOTA** (Tracking Accuracy) and reduction in **ID Switches**.

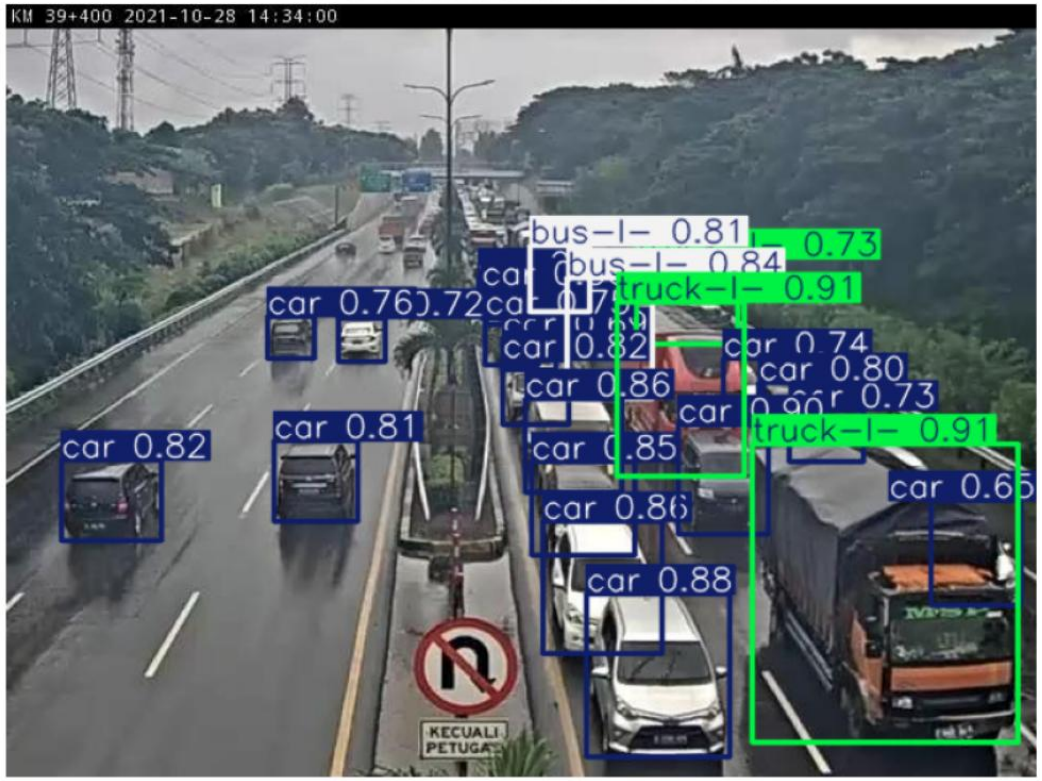
In case of message integration:

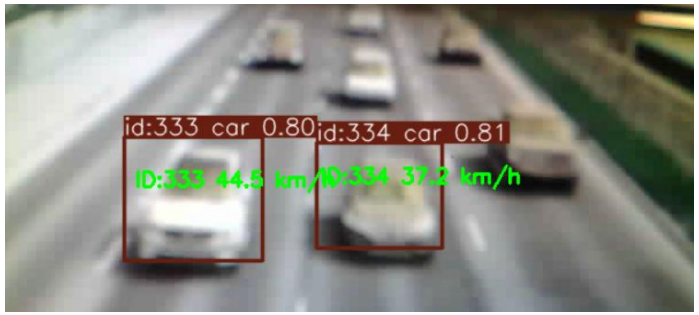
In case of object detection, we can also trigger a message! Or, we save the data in excel files for further analysis!



Some Results

YOLOv11 Prediction Result (Not Saved to Disk)





Thanks!

Rest details will be explained in demonstration!