IC 201P – Design Practicum

ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS)

Under the supervision of

Dr. Ashutosh Kumar

(ashutosh@iitmandi.ac.in)

Dr. Satyajit Thakor

(satyajit@iitmandi.ac.in)

Indian Institute of Technology, Mandi

Certificate

This is to certify that the work contained in the project report entitled **"ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS),"** submitted by Group 41 to the Indian Institute of Technology Mandi for the course IC 201P – Design Practicum, is a record of bonafide research works carried out by him under our direct supervision and guidance.

Dr. Ashutosh Kumar — Signature and Date

Dr. Satyajit Thakor — Signature and Date

Acknowledgments

We would like to express our gratitude to our Mentors, Dr. Satyajit Thakor, Dr. Ashutosh Kumar, and our T.A., Mr. Mayand Malik, who was a continual source of inspiration and guidance. They pushed us to think imaginatively and urged us to do this project without hesitation. Their vast knowledge, extensive experience, and professional competence enabled us to accomplish this project. This endeavor would not have been possible without their help and supervision.

Also, we would like to thank the Indian Institute of Technology, Mandi (IIT Mandi) administration for providing us with the opportunity to work on the **"ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS)."** The completion of this project would not have been possible without the participation and assistance of a lot of individuals and team members contributing to this project who guided us all along in this project.

Abstract

The role of **Advanced Driver Assistance System**(ADAS) is to prevent deaths and injuries by reducing the number of car accidents and the severe impact of those which cannot be avoided. Through a safe human-machine interface, ADAS increases car and road safety by using automated technology, such as LIDAR sensors and stereovision cameras, to detect nearby obstacles or driver errors and respond accordingly.

We are training the most popular object detection algorithm **YOLO (Y**ou **O**nly **L**ook **O**nce**)** on a huge and diverse dataset of speed breakers, potholes, speed limits and traffic lights so that if there is any chance of occurring damage or accident , the voice alert will be give to driver to prevent any kind of serious damage.

This project aimed to design a prototype of a completed advanced driver-assistance system targeting such cars that need to be equipped with or lack some driver-assistance functions. The implemented product should be a system with hardware and software to provide three main parts: (i) unmarked speed breaker and potholes detection, (ii) traffic light recognition (iii) sign detection for maximum speed limit signs and over-speed warnings.

The project's work is such that the road image is captured by the camera mounted on the car's dashboard. The object of interest is detected by a trained deep learning model on **CNN** based object detection algorithm YOLO on cost effective hardware of Jetson Nano, and appropriate voice alerts are provided to drivers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Background of Problem

Drivers often need help recognizing speed breakers and potholes on unknown roads and finding the speed limit on some particular roads. We are sure you think the problem is that someone cannot find a speed hump while driving. The driver needs to pay attention, but that's only sometimes true.

The commonly observed problems with Indian speed breakers are:

- No speed breaker sign boards.
- Fainted Paint/No paint
- Absence of solar-based cat eye.
- No Marking /Unmarked speed humps
- Absence of daylight or street lighting

The Ministry of Road Transport and Highways[1] showed that about 30 accidents occur daily, and nine people die in road accidents due to speed breakers and potholes.

Apart from crashes and deaths, there are many problems that the public faces during their day-to-day commute due to traveling through roads having non-standardized speed humps/bumps. These are given as follows:

- Spinal Problem and Injury
- Delay Emergency services
- Damages suspension when crossed above safer speeds
- Increased pollution(CO_2) due to frequent acceleration and deceleration
- Time and money wasted

For more news and articles regarding unmarked speed breaker problems refer to [2][3][4][5].

Similarly, drivers most of the time unknowingly overspeeding as he/she didn't pay attention to the speedometer and also the speed limit board and enjoy fast driving, which will cause a lot of problems:

- Causes severe accidents
- Let you pay thousands of rupees challan
- Loss of life

Over-speeding constituted the main traffic rule violation associated with accident-related deaths (69.3%) and injuries (73.4%) [6].

Scope of Problem

It is unrealistic to expect that all speed breakers in India, all made according to the IRC099[7] guidelines of the Indian Road Congress (IRC) in India, with proper planning and signboards, and all speed breakers are properly painted and properly maintained.

Also, who can deny that India has one of the largest road networks in the world, and with the increase in the number of vehicles on the road, the number of speed breakers and potholes has also increased.

Additionally, the absence of proper signage, inadequate lighting, and poor maintenance of speed breakers contribute to the problem. Speed breakers are often installed without proper markings or lighting, making them difficult to spot, especially at night.

The problem of speed breakers is particularly acute for emergency services such as ambulances and fire trucks, which require uninterrupted access to the road to provide timely assistance.

Therefore, there is significant scope for developing technologies and solutions that can detect speed breakers and potholes in real-time, provide information about their location, and alert drivers to slow down to prevent vehicle accidents and damage. This technology can also help emergency services to navigate roads safely and efficiently, reducing response times and potentially saving lives.

Design philosophy used in this report

Our design philosophy is to make it a complete package on its own and easy setup without any external attachments with the car so that users can find it easy to use and there is modularity. Also, we focus on developing an accurate, reliable, and efficient model that can quickly detect speed breakers and potholes and overspeeding in real-time. The system should also be able to operate under various environmental conditions, such as different lighting and weather conditions.

Additionally, the system must be able to communicate with the driver, providing alerts and warnings in real-time to prevent accidents and damage to vehicles.

Problem Statement

We aim to develop a system that can detect and alert drivers to the presence of speed breakers, potholes, and potential over-speeding situations in real-time.

The primary goal of this project is to improve road safety and reduce accidents caused by speed breakers, potholes, and over-speeding on Indian roads.

Beneficiaries (Intended Market)

Our Product will benefit all the car drivers who often face problems in recognizing the speed breakers and potholes and are unable to pay attention to speed limits. All the old and low-end cars that are not equipped with or lack some driver-assistance functions will be benefited from these products. This product will improve road safety and accidents by a significant amount.

Organization of this report

The whole report is divided into the following sections:

1. ***Market Research*****:** In Chapter 2, we discuss the existing products in the market and compare them with our product. We also briefly discuss how our intended part differs from the existing products.

2. ***Designing*****:** In Chapter 3, we discuss the conceptual design and in Chapter 4 discuss the detailed design of the product. The designing section can be divided into:

    - Product Architecture
    - System Level Design
    - Detailed Design
        - Electronics aspect
        - Software Aspect
        - Mechanical design

Chapter 2

Market Research

Existing Products In Market

For this market, technology companies also develop separate products to set up used car models or car models without integrated ADAS. This type of product, *MobileEye 630* is a popular device, which is developed by MobileEye, a subsidiary of Intel. MobileEye 630 provides intelligent features such as forward collision warning (FCW), lane departure warning (LDW), intelligent high beam control (IHC), speed limit indication (SLI), and traffic sign recognition (TSR). In Vietnam, WebVision is a company specializing in providing dashcam products with intelligent driver assistance technologies. *WebVision A69 AI* with camera recording function, lane departure warning, forward collision warning, and moving reminder when the traffic light turns green. **WebVision S8** [A][B], in addition to the dashcam function, also warns drivers when they go over speed.

Mobileye

WebVision

Comparison with current products

Although both products are intended to solve a similar problem, both work on different principles. Our product intends to solve incidents caused by unmarked speed breakers and potholes but products available in the market do not give such features in their driving assistance system and they are also not designed as per the road conditions present in India.

Problems with current alternatives

There are some problems with existing current alternatives such as:

- All current alternatives are not providing unmarked speed breaker and pothole detection which can lead to severe accidents and their driving assistance systems are also not designed as per the road conditions present in India.
- False alarms: ADAS systems may trigger false alarms, which can be annoying and distracting for drivers. For example, a lane departure warning may activate when a driver intentionally crosses a lane to avoid an obstacle or when driving on a road with faded lane markings.
- Lack of standardization: There is currently no standardization for ADAS systems, which means that different manufacturers may use different technology and terminology for similar features. This can lead to confusion for drivers and may affect the effectiveness of the systems.

Our Product vs Existing Alternatives

Our product is designed to adopt the road conditions present in India whereas existing alternatives are not designed as such. For the hardware of the proposed system, we choose NVIDIA Jetson Nano, a small, powerful computer that lets you run multiple neural networks in parallel to deploy the final system. Jetson Nano is suitable for this project as it is a powerful hardware architecture with a cheap price to deploy deep learning models. This will keep the production cost relatively low compared to other similar systems. We also attached a 5-inch screen and one small

speaker to build a user interface. Our product is cost-effective compared to the market prices of ADAS. We have thought of some alternative design choices that could further reduce the cost of the end product well up to the ***₹30,000*** mark while the current options will set you back somewhere around ***lacs*** without some major concern solutions and added comforts.

Chapter 3

Conceptual Design

How did we find our problem statement?

Initially, we had developed some Ideas and Problems statements that we discussed with mentors in the first two and three meets. The Ideas were:

- Library assistant system
- Anti-ACL sports Equipment
- Coin Counting Machine
- Speed Breaker Detection(ADAS).

Then, upon discussing with our mentors and considering the cost, the product's existence, Our team distribution among different branches, and the novelty of the idea, we decided to go with the Speed Breaker Detection.

The idea of speed breaker detection given by one of our group members comes to his mind as in winter holidays he learns to drive and while driving in the early morning and in the afternoon, he is unable to recognize unmarked speed breakers and potholes when he discussed and research about it in internet he came to know that this is a problem with lots of people, especially in India. (It is a true reason for proposing this idea)

The different problems that we initially encountered were mainly related to a particular set of people, for example, the Library assistant will only help library users Anti-ACL sports equipment only help football players while the coin counting machine only helps people who basically deal with coins example banks but speed breaker and potholes recognition will help a very large set of people. Also, we see that no product exists in the market which can detect speed breakers and potholes.
Also, Most of the projects which we had discussed are available in the market and can be easily implemented.
So after discussing with mentors and group members, we concluded that we would select Speed breaker and Potholes recognition as our project.

The basis on which we selected the current idea and rejected the other Ideas are

- Learning (0-5)
- Innovation (0-5)
- Cost of development(0-5)
- No. of people reached(0-5)

Table 3.1: Different projects considered

Criteria/ Ideas	Learning	Innovation	Cost to development(0-High cost 5-Least cost)	No. of People Reached (More people is better)	Total
Library Assistant system	3	2	5	2	12
Anti-ACL Sports Equipment	5	5	0	2	12
Coin Counting Machine	2	1	4	1	8
Speed Breaker Detection (ADAS)	4	5	3	5	17

Brainstorming and Idea Generation

We have lots of sessions (Initial 3-4 sessions) based on How, what are the things we need to decide, Which Material to use, which camera is required, which single board computer required and soon.
For the speed breaker part we take one complete session in Finding which method is best for finding the speed breaker. We went through almost all the articles

available on Google regarding speed breakers and finally selected one method/idea discussed in the next section.

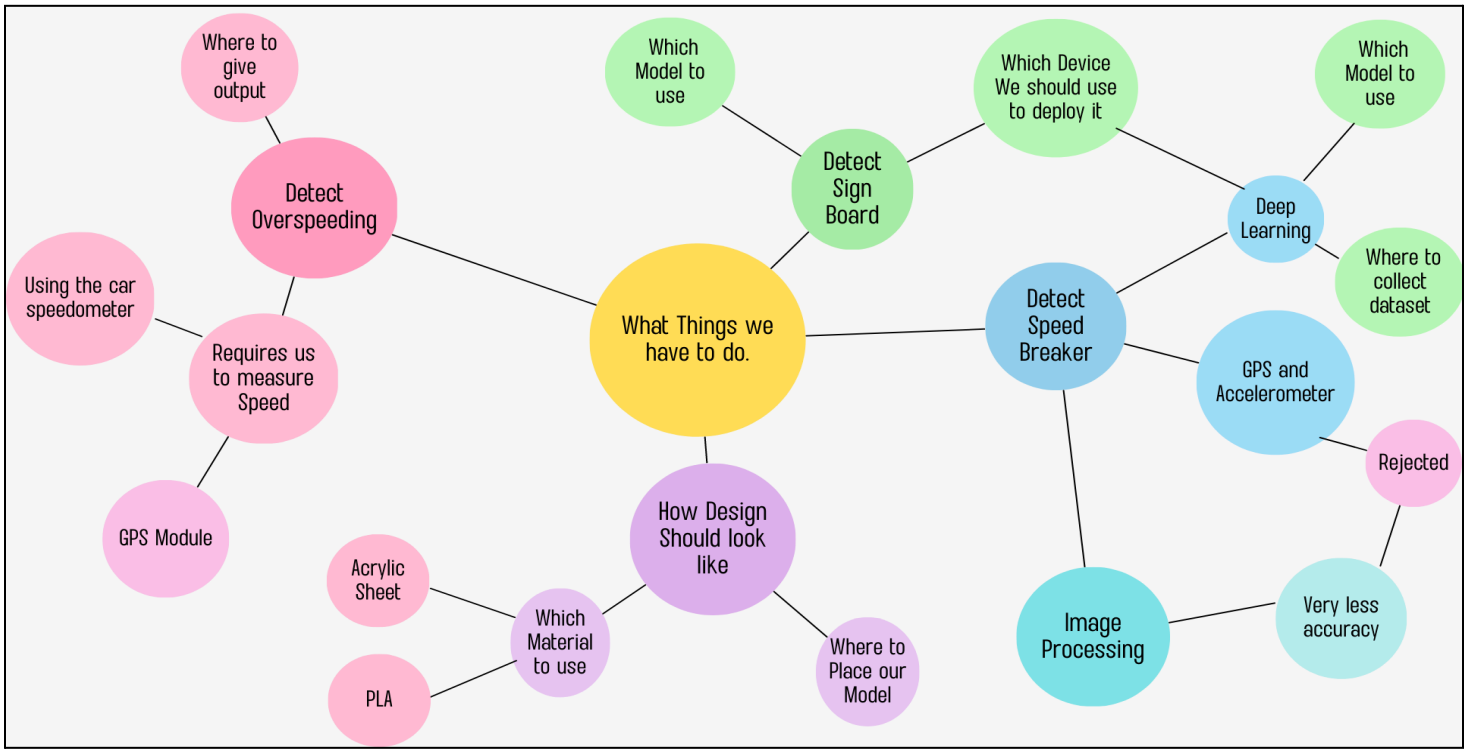


Figure 3.1: Mind Map

Selection of the most viable ideas proposed for Speed Breaker:

For finding speed breakers after researching we are able to find the following three methods of finding speed breakers shown in Fig 3.1.

Accelerometer, GPS-based Detection: The crowdsourcing method records the occurrence of a bump using an accelerometer and logs GPS information into a database for assisting commuters when they travel through the same route with navigation services. These accelerometer-based detections will not be able to alert the driver on the new road hump. It detects only after the vehicle passes through the hump based on thresholding and other time and frequency domain processing techniques applied to accelerometer data obtained.

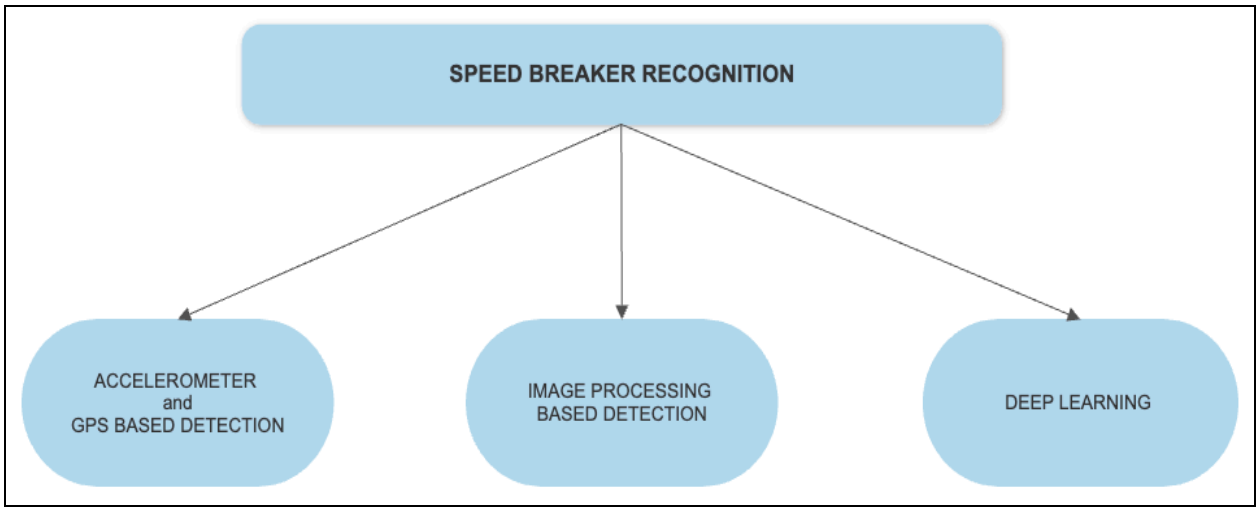


Figure 3.2: Methods of speed breaker recognition

Image Processing Based Detection: For speed breaker recognition using image processing-based detection we followed this paper.
This paper uses edge detection to find the speed breaker as we are trying to find horizontal edges in images and treat them as a speed breaker.

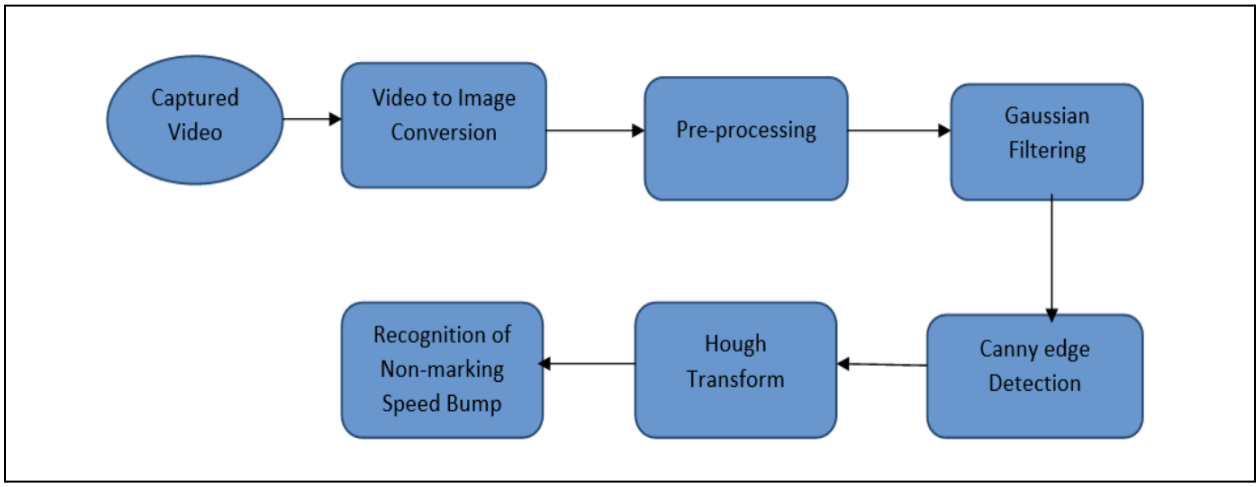


Figure 3.3: speed breaker recognition using Image processing

But this process failed very badly as our canny edge detector was unable to detect the speed breaker as an edge which resulted in an accuracy of 4% in the case of an unmarked speed breaker.

Hence at last we came to a deep learning-based approach because of the great success of deep learning in object detection. We decided to use the YoloV8 model (the latest YOLO model) because Yolo has very high accuracy in object detection. Initially, We wanted to go with the Image processing model as there is no proper dataset of Indian speed breakers available on the Internet but because of such a less accuracy, We moved to a Deep Learning based approach.

Also in this, we are initially doing speed breaker recognition and overspeeding with a feature that if someone does overspeeding or breaks the traffic signal then that data is forwarded to the authority and the authority can take required action but the **privacy issue will come** and that device will not be accepted as a product. Also, we feel if that will be the case then no one will purchase it so we drop this and move to the part that helps people who break the rules and are overspeeding unknowingly, of course, speed breakers are the main part of the project.

Decision Matrix:

The decision matrix for choosing to select the current idea of speed breaker detection and to reject all other ideas is given below:

The basis on which we selected the current idea and rejected the other Ideas are:

- detection in the new road (0-5)
- Feasible (0-5)
- Cost (0-5)
- Reliability(0-5)

Table 3.2: Different Ideas considered

Ideas	detection in new road	Feasible	Cost	Reliability	Total
Image Processing	3	5	4	0	12
GPS/Accelerometer	0	4	3	5	12
Deep Learning	4	4	3	4	15

Proposed Solution :

The **‘Proposed Solution’** for our project is that first the image of the road is detected by a wide-angle camera and in run time 2 deep learning models run on our image and do suitable detection. If a speed breaker and pothole is recognized then it will be displayed on the screen and also an audio warning is given to the driver. Parallelly speed of the car is calculated using a GPS module and if the image contains any speed limit board and if the speed of the car is greater than the speed limit driver gets a warning.

Chapter 4

Embodiment and Detailed Design

Product Architecture

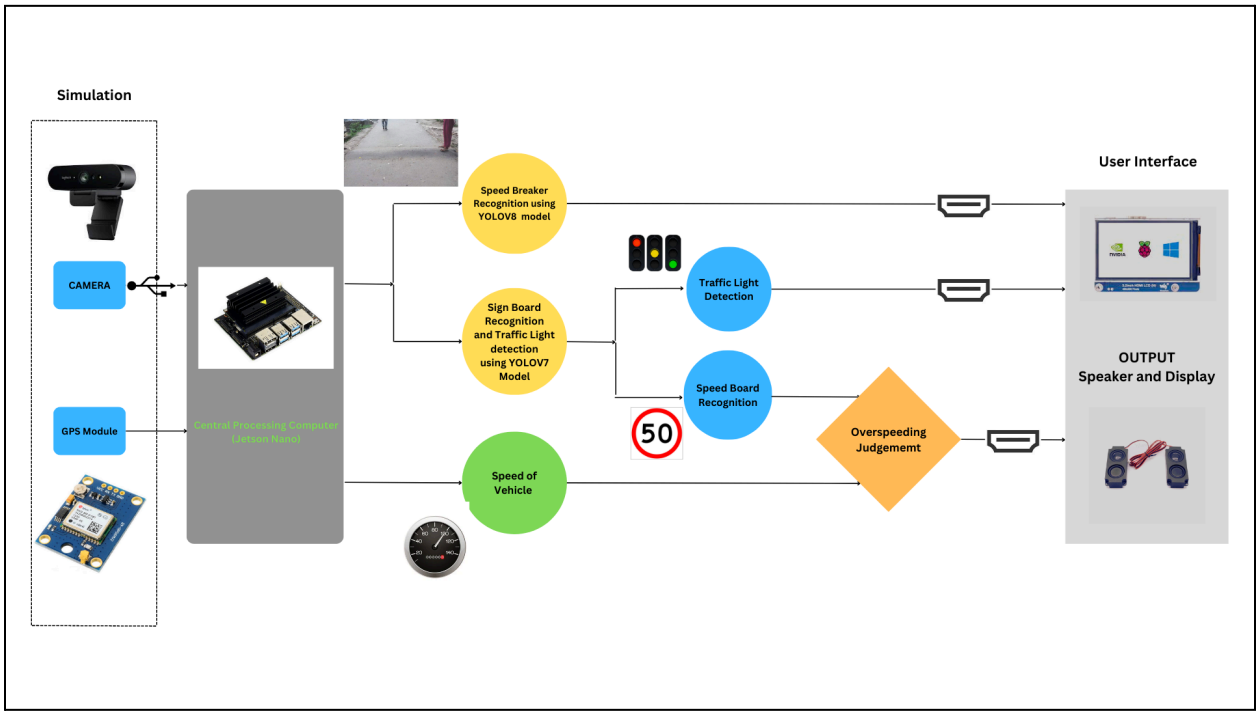


Figure 4.1: Flow Chart of Project Architecture

System-level Design

Components used:

1. Jetson Nano
2. USB GPS Receiver display
3. Wide angle camera
4. HDMI display
5. Speakers
6. Waveshare sound card

I. Speed Estimation using GPS receiver:

- **USB GPS Receiver:**

This GPS receiver offers highly accurate and fast positioning performance with a strong signal, outperforming traditional GPS receivers. It is compact, lightweight, and portable, with a built-in receiving antenna. The high-precision positioning chip and industrial-grade manufacturing process ensure that it can meet the positioning needs of both industrial and personal use. The GPS receiver has an internal battery to power the storage of satellite data, such as satellite signal status, final position, and time, which significantly improves the module's positioning speed at the next boot. It also has a USB interface, making it easy to use with main controllers like Raspberry Pi, NVIDIA, and LattePanda. It is suitable for a variety of applications, including vehicle navigation, handheld positioning, wearable devices, and more.

Figure 4.2: USB GPS Receiver

(Source: https://evelta.com/usb-gps-receiver-compatible-with-raspberry-pi/-lattepanda/-jetson-nano-/ **)**

Specifications:

Table 4.1: Specifications of USB GPS Receiver

Operating Temp (Celcius)	- 40 ~ +85
Dimensions (mm)	60x24x9mm
Weight (g)	50
Update Rate	1 Hz

Sensitivity	tracking -162dBm; Acquisition: 160dBm; cold start: -148dBm

II. Object Detection using Camera and Yolov7:

- ### Nvidia Jetson Nano:

Jetson Nano is a powerful platform for deploying machine learning algorithms. Because of a small size board with a quite strong GPU, it is suitable to be used as the central processing computer. One special feature of this computer in comparison with ones from other companies is that it can use **TensorRT**, an SDK (software development kit) for high-performance deep learning inference. This SDK includes a deep learning inference optimizer and runtime for low latency and high-throughput experience.
Two Yolov7 Models run on this Jetson Nano using the image output of the camera and the weights externally trained by us. These models predict the existence of speed breakers and potholes, the State of traffic lights, and traffic signs to signal overspeeding.

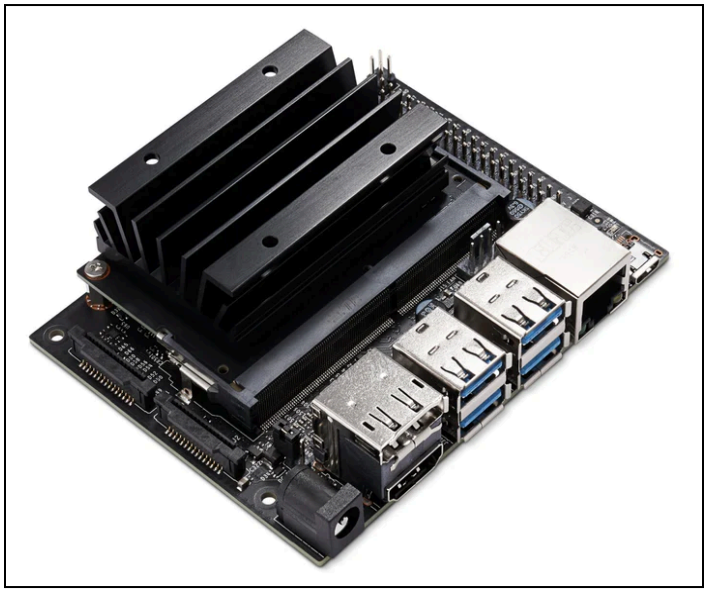


Figure 4.3: Jetson Nano

(Source: https://thinkrobotics.com/products/nvidia-jetson-nano-developer-kit-online)

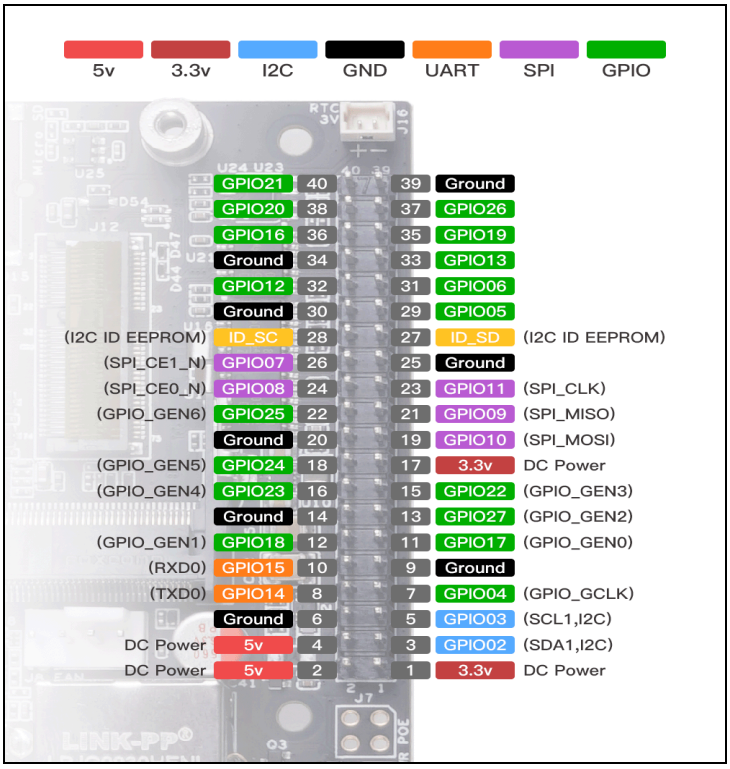


Figure 4.4: Pin configuration of Jetson Nano

(Source: https://wiki.seeedstudio.com/reComputer_Jetson_Series_GPIO_Grove/)

It should be added, Jetson Nano is also equipped with GPIO, I2C, I2S, SPI, and UART pins similar to those of Raspberry Pi for us to communicate with sensors and other components of the embedded system.

Specifications:

Table 4.2: Specifications of Jetson Nano

CPU	Quad Core ARM 1.43 GHz
GPU	128 Core Maxwell
RAM	4 GB
Input Voltage	5 V
Mechanical	100mm x 80mm x 29mm
Ports	HDMI, USB, Micro USB

- **Waveshare IMX477-160:**

  MIPI-CSI Camera, based on Sony IMX477 sensor, 1230M pixels. Supports Raspberry Pi Compute Module series and Jetson Nano developer Kit. Suitable for AI applications such as license plate number recognition, and facial gesture recognition.

Figure 4.5: Waveshare camera

(Source: https://www.waveshare.com/wiki/IMX477-160_12.3MP_Camera**)**

Advantages of Waveshare IMX477-160:

It is suitable for AI applications like: facial recognition, road mark recognition, license plate number recognition, and so on.Also It has wide FOV able to capture the whole road completely and also has high MegaPixel therefore give high quality images that helps in detecting the speed breaker from greater distances.

The 60 fps helps in real time detection of speed breaker, potholes, traffic lights and speed limit boards by reducing the inference time.

Specifications:

Table 4.3: Specifications of Camera

Model	IMX477
Resolution	12.3MP, 4056 × 3040
Pixel Size	1.55µm (H) × 1.55µm (V)
Mega Pixel	12.3 MP

CMOS diagonal length	7.9 mm
Operating voltage	3.3 V
Field of View	160°(D) 118°(H) 87°(V)
Frames per Second(FPS)	4K : 30 fps 1920 x 1080 : 60fps

- **Waveshare 3.2 inch HDMI LCD IPS Display:**

  For user interaction, Waveshare 3.2-inch non-touch is a good option for this project. It provides a large enough space for a comfortable user experience.

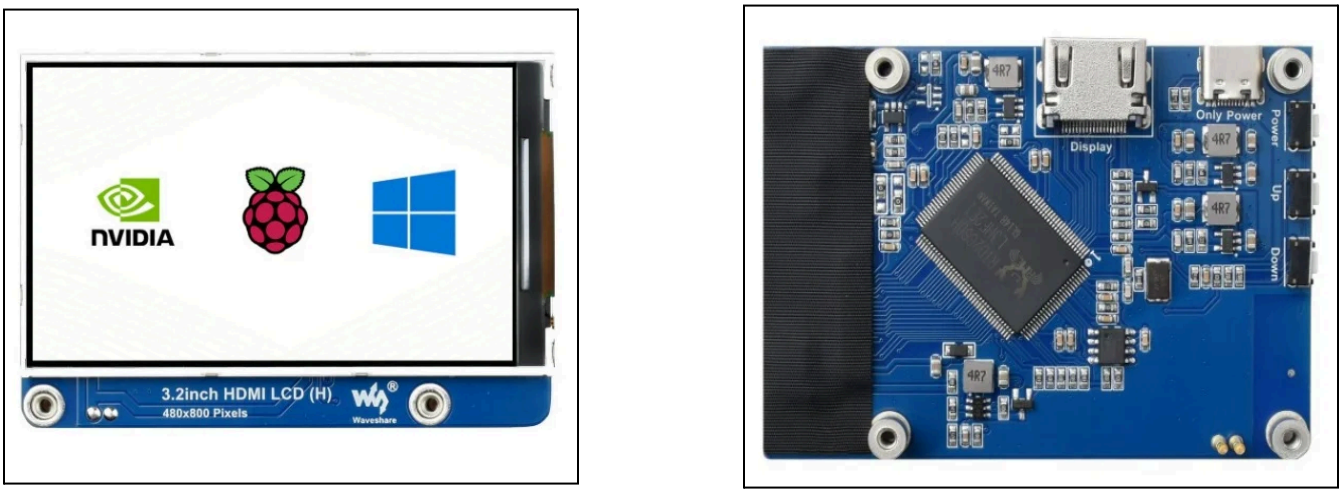


Figure 4.6: Waveshare Display

(Source: https://www.waveshare.com/wiki/3.2inch_HDMI_LCD_(H))

Specifications:

Table 4.4: Specifications of Display

Brand	Waveshare
Pixel Resolution	480 x 800

HDMI	1 port
Display Size	3.2 inch

- **8 Ohm 5 W Speakers:**

  We use 5Ω - 8W dual speaker from Wareshare to play warning and notification sounds in designed system.

Figure 4.7: Speakers
(Source: ThinkRobotics)

Specifications:

Table 4.5: Specifications of Speakers

Weight (g)	145
Dimensions (mm)	36 x 92

Design configuration:

Shape and general dimensions or sizes are established to the components that are defined in product architecture. It is largely dependent on the three-dimensional constraints that define the envelope in which the product operates and the product

architecture. This would be a preliminary selection of material, manufacturing process, modeling, sizing of parts, etc.

Detailed Design:

Software Part

The complete software portion of our project is divided into six different parts:

A. Research related to speed breaker and Pothole recognition is covered.
B. Covering the aspects of the signboard and traffic light recognition.
C. Initial setup for jetson nano.
D. Speed of vehicle using USB GPS receiver.
E. Deployment of models in Jetson Nano.
F. Creating a user interface and displaying output to the user and giving warnings from speakers to drivers using Python code in jetson nano.

A. Related to Speed Breaker and Pothole Recognition

In speed breaker recognition part using deep learning involves:

- Dataset
    - Dataset Collection
    - Preprocessing of data
- Model Selection
- Model Training and Testing

Dataset Collection:

We feel this is one of the toughest parts of the project as there were no open speed hump/bump datasets found available in the public domain.

Table 4.6: Dataset Description

Class	Train_data	Validation_data
Speed Breaker	23152	4109
Potholes	24514	4849
Negative Data	5977	1023

In the dataset collected, we observed the bumps which are having traffic signs and solar cat eyes are very few. Most of the humps are observed to have been poorly maintained with faded paints. The dataset collected consists of bumps, humps, rumble strips, shaded bumps due to trees/buildings, and bumps with faded paints.

Features of Dataset:

- The dataset compiled contains pictures of speed bumps and potholes in various lighting conditions.
- Road data of all kinds and states is taken into consideration.
- Additionally taken into account are various perspective options and variable weather conditions.
- Images of speed breakers and potholes are captured from a greater distance.
- Negative dataset consists of pseudo speed breakers and potholes due to the shadows, stones and dashboard of the car also taken to reduce **False Positives**.

Train_Val Split:

The collected dataset is splitted into train data and validation data using 80-20 rule i.e 80% of total data is training data and 20% is validation data. For splitting into train data and validation, the following python code is used. We are using python inbuilt libraries **Shutil** and **os** for file related actions and **Numpy** for generating the random numbers.

Code for Train_Val Split:

```
# Importing the libraries
import os
import shutil
import numpy as np

#Assigning the Path
path='/Users/prakulkhurana/Desktop/potholes_dataset'
path_lab='/Users/prakulkhurana/Desktop/potholes_labels'
a=sorted(os.listdir(path))
b=sorted(os.listdir(path_lab))
length=len(a)
l1=len(b)
path1='/Users/prakulkhurana/Desktop/yolov7/data/train/images'
path2='/Users/prakulkhurana/Desktop/yolov7/data/train/labels'
path3='/Users/prakulkhurana/Desktop/yolov7/data/val/images'
path4='/Users/prakulkhurana/Desktop/yolov7/data/val/labels'
#selecting random 20% of images for validation dataset
q=np.random.randint(0,length,size=int(0.2*l1))

#creating the count of each Image:
n_train_i=0
n_train_l=0
n_val_i=0
n_val_l=0

#code for splitting the data and its corresponding labels into corresponding folders.
for i in range(len(b)):
    if i in q:
        shutil.copy(os.path.join(path_lab,b[i]), path4)
        n_val_l=n_val_l+1
        o=b[i].rfind('.')
        z=b[i][:o]
        for r in a:
            c=r.rfind('.')
            d=r[:c]
            if d==z:
                shutil.copy(os.path.join(path,r), path3)
                n_val_i=n_val_i+1
                break
    else:

        shutil.copy(os.path.join(path_lab,b[i]), path2)
        n_train_l=n_train_l+1
        o=b[i].rfind('.')
        z=b[i][:o]
        for r in a:
            c=r.rfind('.')
            d=r[:c]
            if d==z:
                shutil.copy(os.path.join(path,r), path1)
                n_train_i=n_train_i+1
                break
```

Preprocessing of Data:

In order to train an object detection model, we need image height, width, class, and bounding box parameters containing x_min, y_min, x_max, and y_max of the object. We obtain these by performing labeling. It is performed on our dataset using Label Img[12], an open-source, free tool that saves the bounding box parameters of each image into a corresponding .txt file.

YOLOv7 takes label data in the **text(.txt)** file and has the following format:

```
<object-class-id> <x> <y> <width> <height>
```

Our model has two classes, i.e., speed breakers and potholes.

- **Data Augmentation**

  The process is basically used to increase the number of images on which our speed breaker and potholes model will be trained. This is achieved by using the **image generator** function from the **Keras library**. From one image, about ten different images are generated through the process of random croppings and alteration in brightness range and rotations.

```
for img in os.listdir(path):
  j=1;
  #---convert the image to 3D array---
  img_array=cv2.imread(os.path.join(path,img))
  #---convert into a 4-D array of 1 element of 3D array representing
  # the image---
  images_data = np.expand_dims(img_array, axis=0)
  #---create image data augmentation generator---
  datagen =ImageDataGenerator(
          # width_shift_range=0.2,
          # height_shift_range=0.2,
          # shear_range=0.2,
          zoom_range=[1,1.4],
          horizontal_flip=True,
          fill_mode='nearest',
          brightness_range=[0.6,1])
```

```
  train_generator = datagen.flow(images_data, batch_size=1)
  for i in range(5):
    image_batch = train_generator.next()
    image = image_batch[0].astype('uint8')
    cv2.imwrite("/content/augmented_images_1/"+"%d"%i+img,image)
    j=j+1
print("done")
```

- **Reducing Image Size**
  All images are compressed in size for faster training on GPUs without losing information that is needed for the algorithm and compromising too much on accuracy.

  Code for Reducing Image Size:

```
# Importing Libraries
import cv2
import numpy as np

# Read the image using imread function
image = cv2.imread('image.jpg')
cv2.imshow('Original Image', image)

#downscale the image using new  width and height
down_width = 416
down_height = 416
down_points = (down_width, down_height)
resized_down = cv2.resize(image, down_points, interpolation= cv2.INTER_LINEAR)

# Display images
cv2.imshow('Resized Down by defining height and width', resized_down)
cv2.waitKey()
#press any key to close the windows
cv2.destroyAllWindows()
```

- **Creating Labels for Negative dataset**

  For improving the model accuracy we are adding the negative dataset and its labels are given by empty files with same name as of image which we created using python inbuilt libraries **OS.**

Code for creating labels for negative dataset:

```
import os
for f in os.listdir("/Users/prakulkhurana/Desktop/sb"):
        a=f.split('.')
        file_path = os.path.join("/Users/prakulkhurana/Desktop/sb",a[0]+'.txt')
        with open(file_path, "w+"):
            pass
```

Model Selection:

We decided to use the yolov7_tiny Model for all the detections like speed breakers, potholes, traffic lights and speed sign boards.

Table 4.7: YOLOv7 models FPS comparison

Model	Parameters (million)	FPS	AP test (%)
YOLO7-Tiny	6.2	286	38.7
YOLOv7	36.9	161	51.4
YOLOv7-X	71.3	114	53.1
YOLOv7-W6	70.04	84	54.9
YOLOv7-E6	**97.2**	**56**	**56.0**
YOLOv7-D6	154.7	44	56.6
YOLOv7-E6E	151.7	36	56.8

- YOLOv7-Tiny, YOLOv7, and YOLOv7-W6 are meant for edge GPU, normal (consumer) GPU, and cloud GPU, respectively.
- Since we are using **edge GPU(Jetson Nano)** for real-time speed breaker detection, i.e. why are we using the yolov7-tiny model? compare to other versions edge optimized yolov7-tiny uses the **leaky rectified linear unit** as an activation function while the other model uses the **sigmoid weighted linear unit** as activation function.

- YOLO v7 also has a higher resolution than the previous versions. It processes images at a resolution of 608 by 608 pixels, which is higher than the 416 by 416 resolution used in YOLO v3. This higher resolution allows YOLO v7 to detect smaller objects and to have a higher accuracy overall.
- One of the main advantages of YOLO v7 is its speed. It can process images at a rate of 155 frames per second, much faster than other state-of-the-art object detection algorithms
- This makes it suitable for sensitive real-time applications such as surveillance and self-driving cars, where higher processing speeds are crucial.

Custom Dataset Training:

We train around 52,000 datasets of speed breakers and potholes using the weight files **Yolov7.pt** and also using **Yolov7-tiny.pt** first using training config files, and then that custom dataset trained weight file is re-trained using the deploy config files as that will reparameterized the weights for the **Inference.** It focuses on optimizing the model's performance during inference by adjusting parameters like **confidence thresholds** and **non-maximum suppression.**

Script File for custom dataset training

```
#!/bin/bash
#PBS -q gpuq
#PBS -o out.o
#PBS -e out.e
#PBS -N yolov7
#PBS -j oe
#PBS -l nodes=n66.cluster.iitmandi.ac.in:ppn=1
#PBS -V

cd ${PBS_O_WORKDIR}
echo "Running on: "
cat ${PBS_NODEFILE}
cat $PBS_NODEFILE > machines.list
echo "Program Output begins: "
source $HOME/anaconda3/bin/activate dp
python $HOME/yolov7_potholes/train.py --workers 4 --device 0,1 --batch-size 64  --epochs 1 --img 416 416
--data data/custom_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg cfg/deploy/yolov7-custom.yaml
--name yolov7-custom --weights yolov7-tiny.pt
```

-- **img** = size of images on which model will train. We take 416 for training and 416 for test for faster training and inference.

-- **batch-size** = batch size used in training. We choose 64 here to make max use of the GPU we have.

-- **epochs** = number of training epochs. We choose here 50 according to our dataset and the accuracy we want

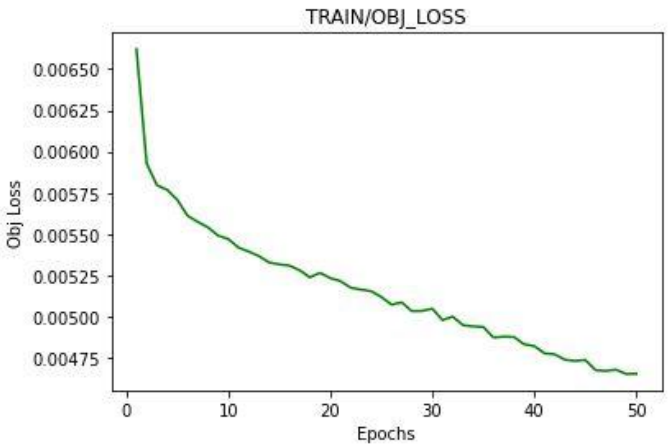
-- **data** = path of custom config file

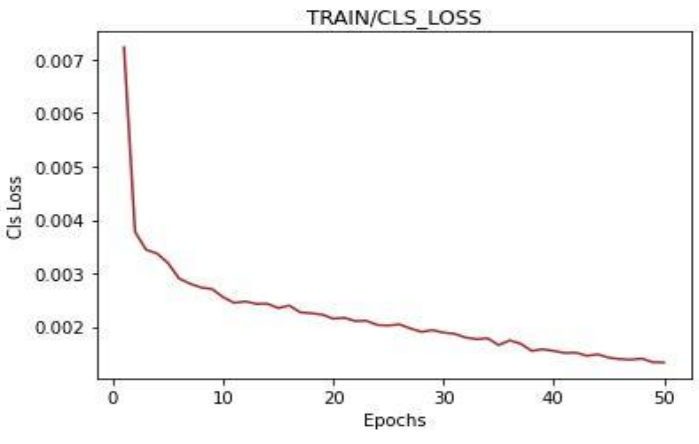
-- **weights** = pre-trained weights. We trained our model on both *yolov7.pt* and *yolov7-tiny.pt* and based on accuracy of training and inference time we chose one.

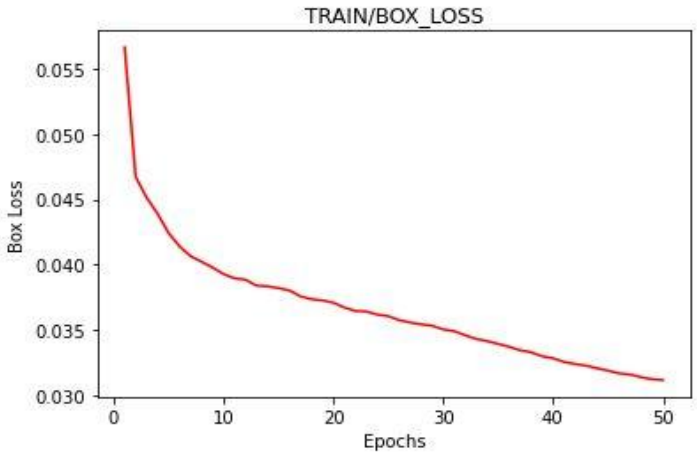
Result of yolov7-tiny Training:

Table 4.8: Result of YoloV7-tiny Model

Epochs	Precision	Recall	mAP
1	0.5444	0.3577	0.3546
10	0.7376	0.6207	0.6749
20	0.7524	0.7067	0.7423
30	0.7703	0.7435	0.768
40	0.7874	0.7486	0.7798
50	0.8111	0.745	0.7896







Loss curves

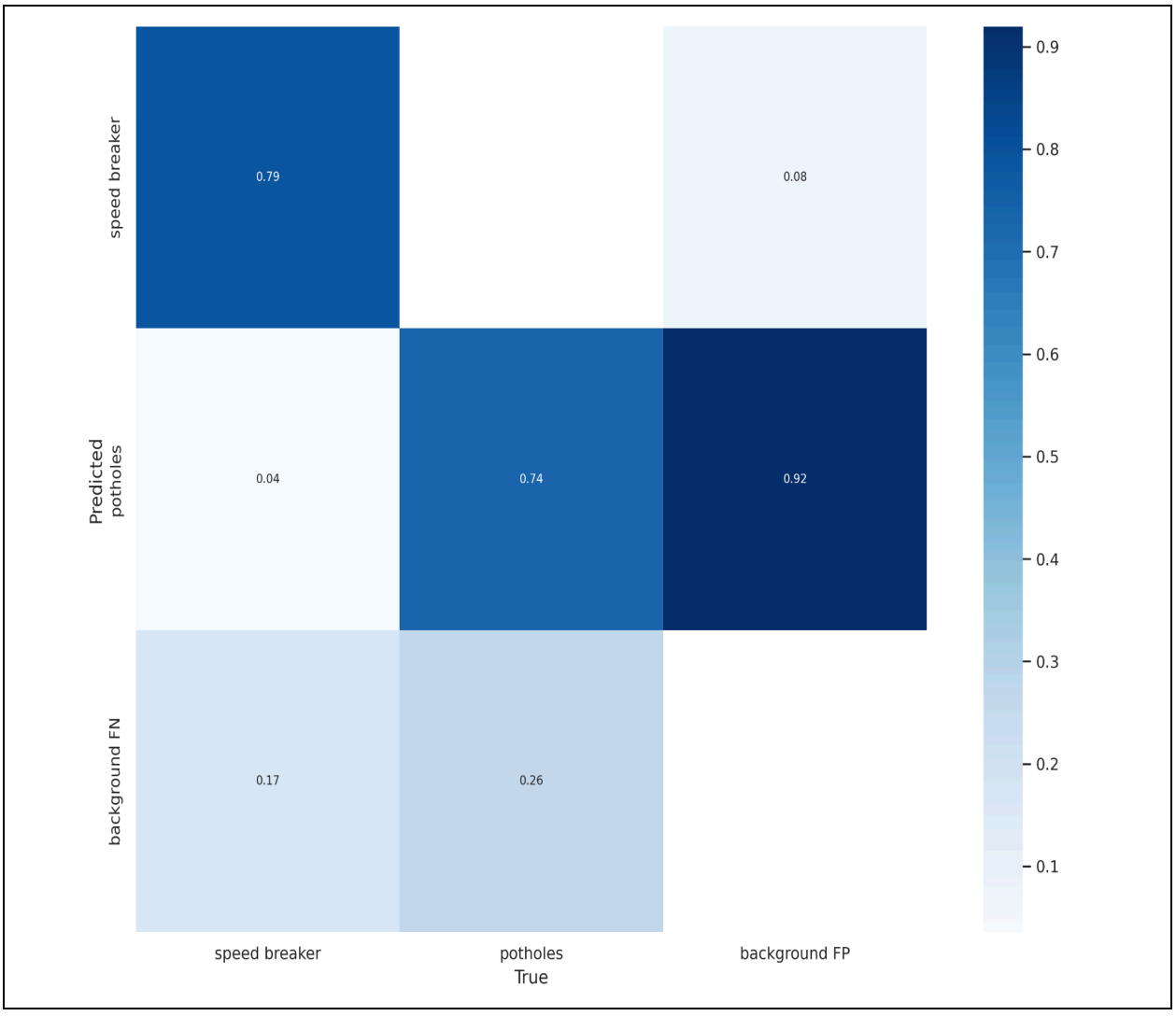


Fig 4.8: Confusion Matrix and loss curves

Inference with custom weights::

The image dataset consisted of about 25,000 images of speed breakers and 27,000 images of potholes for training. Some parts of the dataset were manually collected and some will be taken directly from the internet and they were augmented to increase the size of the dataset. The annotations were done on all the manually collected images. The final dataset containing both images and annotations are trained on the YOLOv7 algorithm using ***Yolov7-tiny weights*** file. The weight file produced after training the model is used to predict speed breakers and potholes for prediction in one image and also real time prediction.

Code for detection of speed breaker and Potholes

python3 detect.py --weights sb_50.pt --conf 0.1 --img-size 416 --source /Users/prakulkhurana/Downloads/IMG_8183.JPG --no-trace

Actual Image

Detected Image

python3 detect.py --weights sb_70.pt --conf 0.2 --img-size 608 --source /Users/prakulkhurana/Downloads/pothole_detect.JPG --no-trace

Actual Image

Detected Image

B. Related to the signboard and traffic light recognition:

We use YOLOv7 over a dataset containing the following 13 classes for Sign Board and Traffic Lights detection.

Green Light, Red Light, Yellow Light, Speed limit 100, Speed limit 120, Speed limit 20, Speed limit 30, Speed limit 40, Speed limit 50, Speed limit 60, Speed limit 70, Speed limit 80, Speed limit 90

I. Preprocessing and Dataset Generation

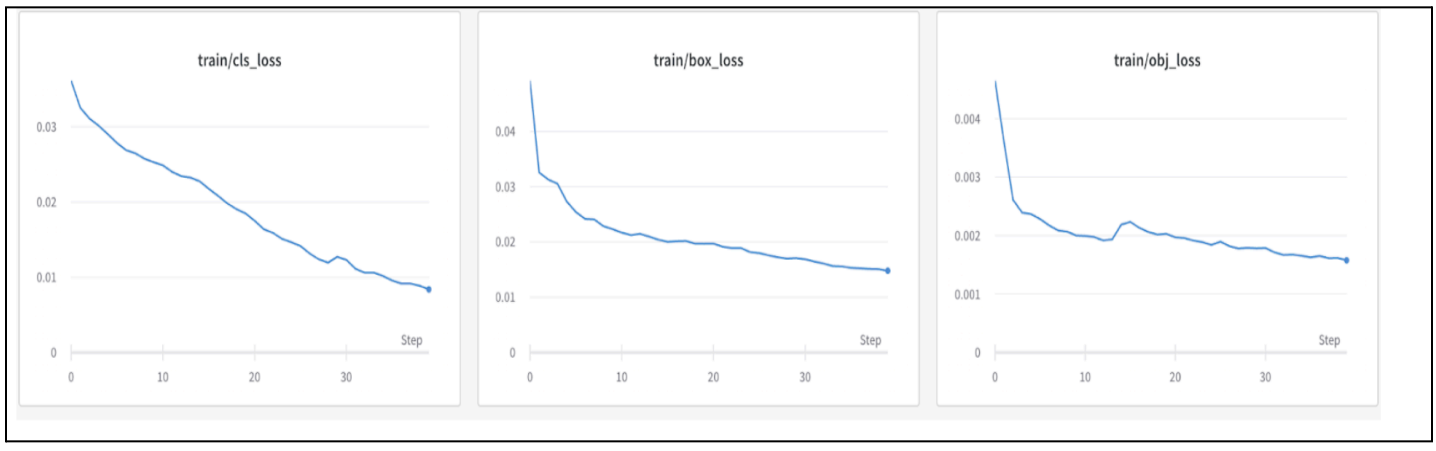
One of the crucial methods adopted to generate data for detecting Speed Limits was randomly placing speed limit signs of appropriate size above the dashboard and outside the road. These signs were also carefully chosen to match the environment's lighting conditions. Overall, they provide the model with real-life features and make it robust to false negatives.

These were generated for all Speed Limit Classes over different backgrounds randomly generated with the following Python script.

Code for generating speed limits dataset

```
bgs_set = set (np. random. randint (len (bgs_list)-1, size=1600))
for i in bs_set:
addon = cv2. imread ( 'Ful lIJCNN2013/00/'+sp_lim20 [0])
bg = cv2. imread ( 'background/bgwithout lights/'+bgs_list [i], cv2. IMREAD_UNCHANGED)
print (bg. shape, addon. shape)
copy = bg. copy ()
if bg. shape [0]=-960:
copy = copy :600,:,:1
temp = 600
else:
copy = copy [: 350,:, : 1
temp = 350
X = np. random. uniform (low=100, high= (temp*2-50))
Y = np. random. uniform (low=int (temp/6) , high=temp-100)
bg_h,bg_w = b. shape [0], bg. shape [1]
dim = (min (addon. shape [0] , addon. shape [1]) , min (addon. shape [0] , addon. shape [1]) )
if dim [0]>80:
dim = (int (dim[01/3) , int (dim [0]/3))
wid = dim[0]
hei = dim [0]
sta_x = int (X-wid/2)
sta_y = int (Y-hei/2)
print (sta_y,sta_y+hei,sta_x, sta_x+wid)
copy_ = copy [sta_y: sta_y+hei,sta_x:sta_x+wid,:]
resized = cv2. resize(addon, dim, interpolation=cv2. INTER_AREA)
mask = np.zeros (resized. shape [:21, dtype=p.uint8)
cv2. circle (mask, (int (dim [01/2) ,int (dim[0]/2)) ,int (dim[01/2-1), (255,255,255) ,-1)
masked_image = cv2.bitwise_and (resized, resized, mask=mask)
cv2. circle(copy_, (int (dim [0]/2) ,int (dim [0]/2)) ,int (dim[0]/2-1), (0,0,0), -1) print (masked_image. shape,
                                                                                            copy_. shape, dim)
new = cv2.bitwise_or (masked_image, copy_)
copy [sta_y: sta_y+hei,sta_x: sta_x+wid,:] = new
bg [:350, :,:] = copy
name = 'speedlimit20_'+bgs_list[i]
nametext = 'speedlimit20_'+bgs_list[i].split('. ') [0]+' .txt'
cv2.imwrite('speedlimits/images/'+name,bg)
file = open ('speedlimits/labels/'+nametext, 'w')
file.write(f'5 {X/bg_w} {Y/bg_h} {wid/bg_w} {hei/bg_h}')
```

II. Training



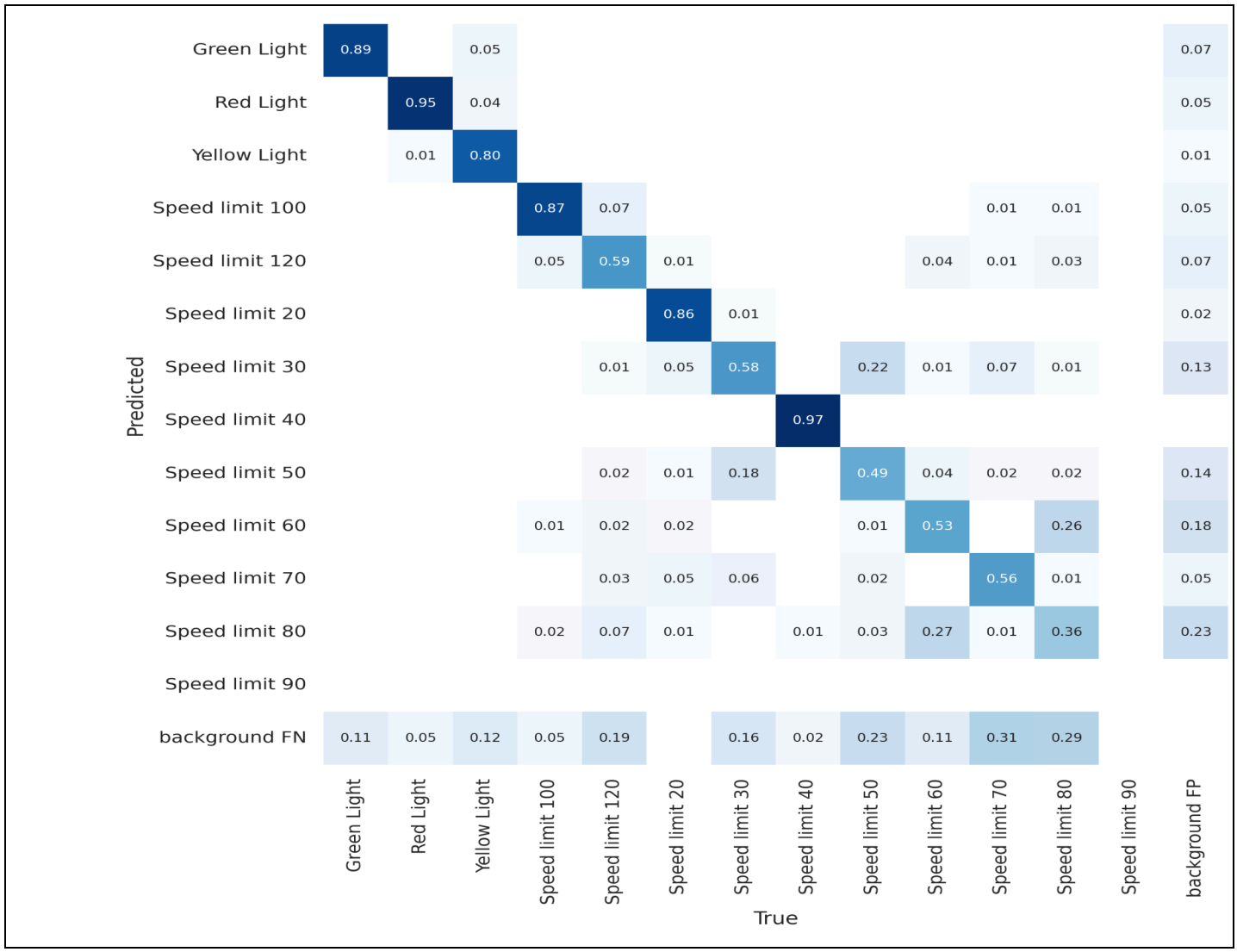


Figure 4.9: Training Losses and Confusion Matrix

The total number of images in the dataset is 22846, divided into 18306 of the training set and 4540 of the validation set. The training was done for 40 epochs on a YOLOv7-tiny model. The suffix 'tiny' means that this is optimized for Edge AI and deep learning workloads and is more lightweight to run ML on mobile computing devices or distributed edge servers and devices. Jetson Nano is one of these devices. The Image size is 416 x 416.

```
35/39      10.4G   0.01531  0.001629   0.00956    0.0265        40       416
           Class      Images      Labels           P           R      mAP@.5
             all        7640        8157       0.746       0.741       0.793       0.598
36/39      10.4G   0.01522  0.001653  0.009164   0.02604        43       416
           Class      Images      Labels           P           R      mAP@.5
             all        7640        8157       0.773       0.738       0.803       0.607
37/39      10.4G   0.01512  0.001615  0.009151   0.02589        50       416
           Class      Images      Labels           P           R      mAP@.5
             all        7640        8157       0.789       0.734       0.806       0.609
38/39      10.4G   0.01508   0.00162  0.008868   0.02556        58       416
           Class      Images      Labels           P           R      mAP@.5
             all        7640        8157       0.794       0.748       0.817       0.617
39/39      10.4G   0.01479  0.001579  0.008385   0.02475        41       416
           Class      Images      Labels           P           R      mAP@.5
             all        7640        8157       0.807        0.74       0.821        0.62
     Green Light        7640        1139       0.872       0.831       0.884       0.515
       Red Light        7640        1403       0.913       0.922       0.937        0.63
    Yellow Light        7640         115       0.797       0.786       0.806       0.519
 Speed limit 100        7640         663       0.905       0.905       0.949       0.738
 Speed limit 120        7640         691       0.778       0.482       0.723       0.529
  Speed limit 20        7640         610       0.906       0.989       0.993       0.791
  Speed limit 30        7640         718       0.834        0.67       0.799       0.649
  Speed limit 40        7640          86       0.913       0.977       0.986       0.928
  Speed limit 50        7640         643        0.73       0.583       0.724       0.571
  Speed limit 60        7640         690       0.617       0.788       0.751       0.609
  Speed limit 70        7640         648       0.871       0.532        0.76       0.561
  Speed limit 80        7640         751       0.551       0.417       0.537       0.396
```

Figure 4.10: Training Epochs

After 40 epochs of training, the model had a Precision, Recall, and mean Average Precision listed in the following table.

Table 4.9: Result of YoloV7 corresponding to Fig 4.9

Epochs	Precision	Recall	mAP
8	0.224	0.698	0.297
16	0.415	0.666	0.522
24	0.522	0.735	0.654
32	0.699	0.736	0.764
40	0.807	0.740	0.821

The following Image is the result of the model over a sample test Image with two-speed limit boards. The Model, along with the camera, would work to find out traffic signs and assist the driver in helping them make fewer mistakes.

Figure 4.11: Results

C. Initial Setup for Jetson Nano:

A. Power Setting

There are many ways to power the Jetson Nano, such as micro USB power, using circular power, or powering via GPIO. We completely power the circuit through the Micro USB port with a 5V-2A phone charger from the car.

B. Installing Operating System

- **Step 1**: Download “SD Card Image” from Nvidia.

- **Step 2**: Use the Etcher tool to burn the last image file to the Jetson Nano's memory card.
- **Step 3**: Insert the memory card into the Jetson Nano circuit and plug it in.

D. Speed of the vehicle using USB GPS Receiver:

To calculate speed, the receiver measures the change in position over time. It determines the speed by comparing the current and previous positions and the time elapsed between the two measurements.

```
import serial # Importing the serial module for communication with the serial port

#Initializing the serial connection
ser = serial.Serial('/dev/ttyACM0', 4800, timeout=1)

# Variables to store latitude and longitude
latitude = ''
longitude = ''

# Function to convert speed from knots to km/h
def knots_to_kmph(knots):
    return knots * 1.852  # Conversion factor from knots to km/h

while True:
    line = ser.readline().decode('utf-8', errors='replace')  # Reading data from the serial port

    if "$GPRMC" in line:  # Checking if the line contains the GPRMC data

        data = line.split(",")  # Splitting the line by comma to extract the data

        if len(data) >= 8 and data[2] == 'A':  # Checking if the data has the required fields and is valid
            speed_knots = float(data[7])  # Extracting the speed in knots from the data
            speed_kmph = knots_to_kmph(speed_knots)  # Converting speed from knots to km/h
            latitude = data[3]  # Extracting latitude from the data
            longitude = data[5]  # Extracting longitude from the data
            print(latitude, longitude, speed_kmph)  # Printing latitude, longitude, and speed in km/h
```

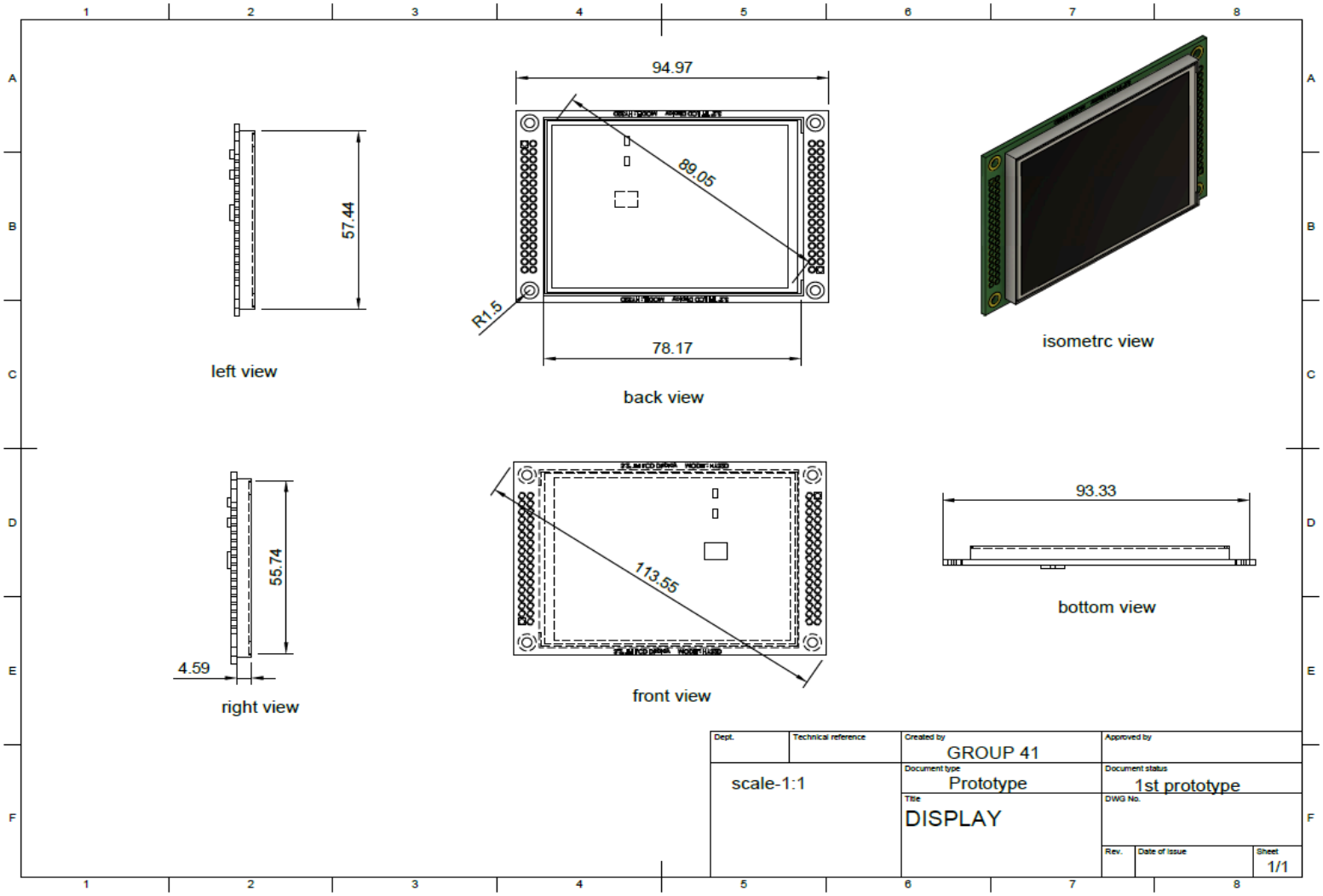
E. Deployment of Models in Jetson Nano :

- **Running Camera on Jetson Nano:**

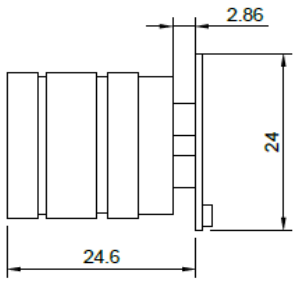
```
import cv2
import numpy as py
import os
def gstreamer_pipeline (capture_width=1280, capture_height=720, display_width=640, display_height=360, framerate=60,
                        flip_method=0) :
    return ('nvarguscamerasrc ! '
    'video/x-raw(memory:NVMM), '
    'width=(int)%d, height=(int)%d, '
    'format=(string)NV12, framerate=(fraction)%d/1 ! '
    'nvvidconv flip-method=%d ! '
    'video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! '
    'videoconvert ! '
    'video/x-raw, format=(string)BGR ! appsink'  % (capture_width,capture_height,framerate,flip_method,display_width,
                                                    display_height))

#print(gstreamer_pipeline(flip_method=0))
cap = cv2.VideoCapture(gstreamer_pipeline(flip_method=0), cv2.CAP_GSTREAMER)
#if cap.isOpened():
window_handle = cv2.namedWindow('CSI Camera', cv2.WINDOW_AUTOSIZE)
# Window
while cv2.getWindowProperty('CSI Camera',0) >= 0:
    _, img = cap.read()
    cv2.imshow('CSI Camera',img)
    if cv2.waitKey(16) & 0xff == 27:
        break
cap.release()
cv2.destroyAllWindows()
#else:
 #   print('Unable to open camera')
```

Mechanical Aspects:

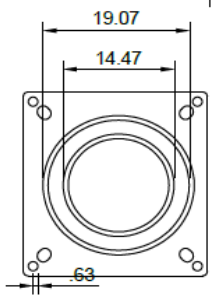
Drawing 1: display



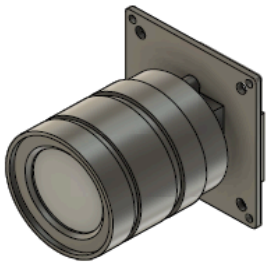
Drawing 2: Camera



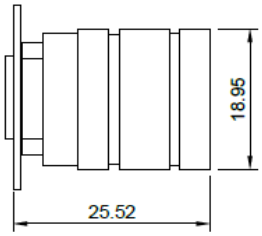
top view



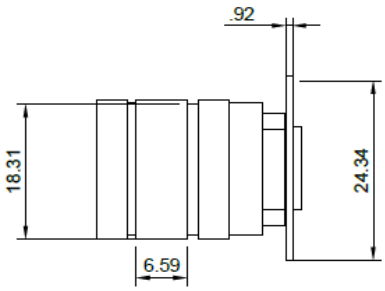
left view



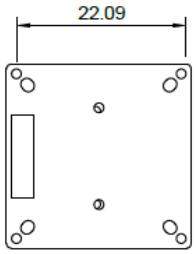
isometric view



back view



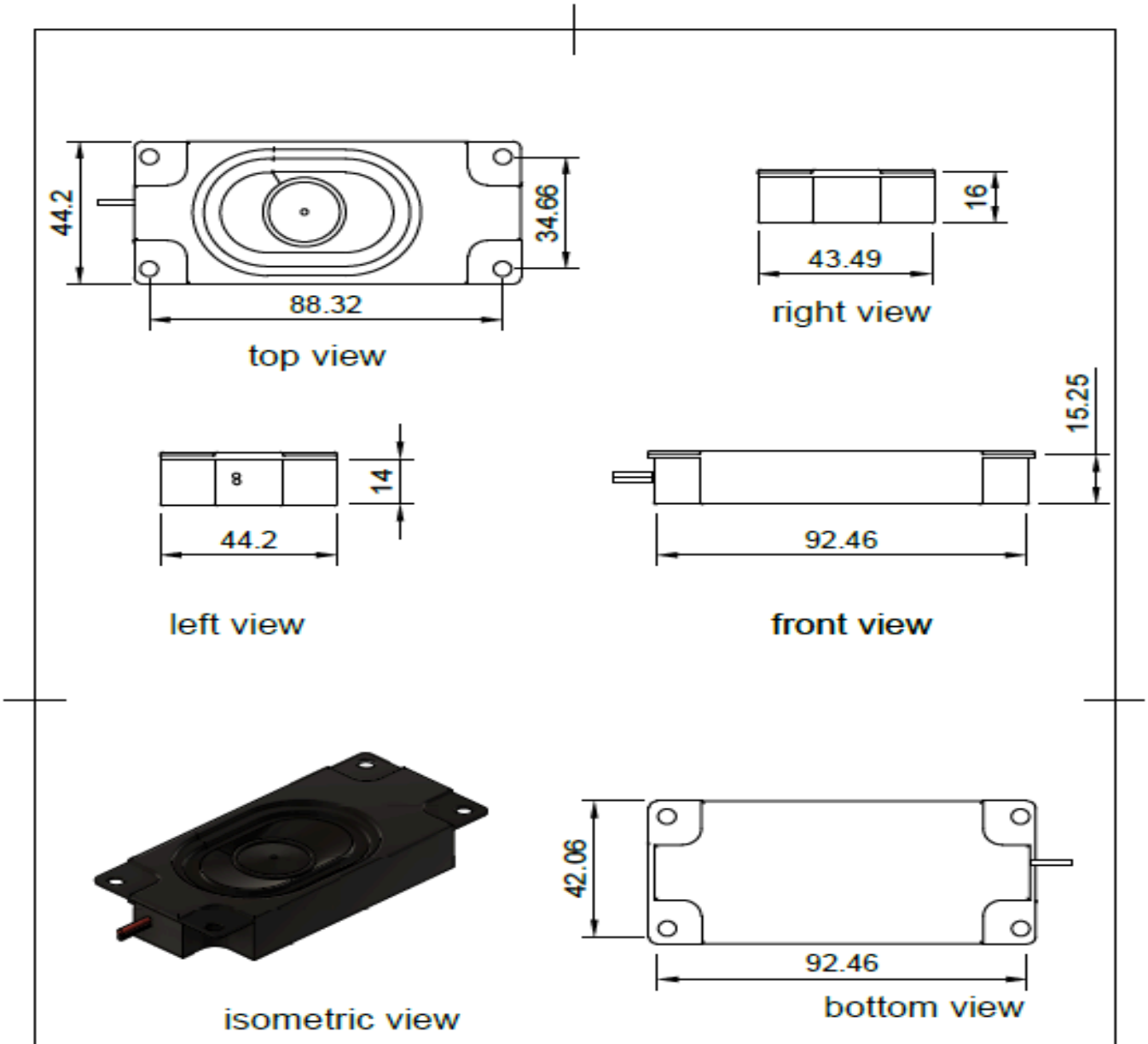
front view



right view

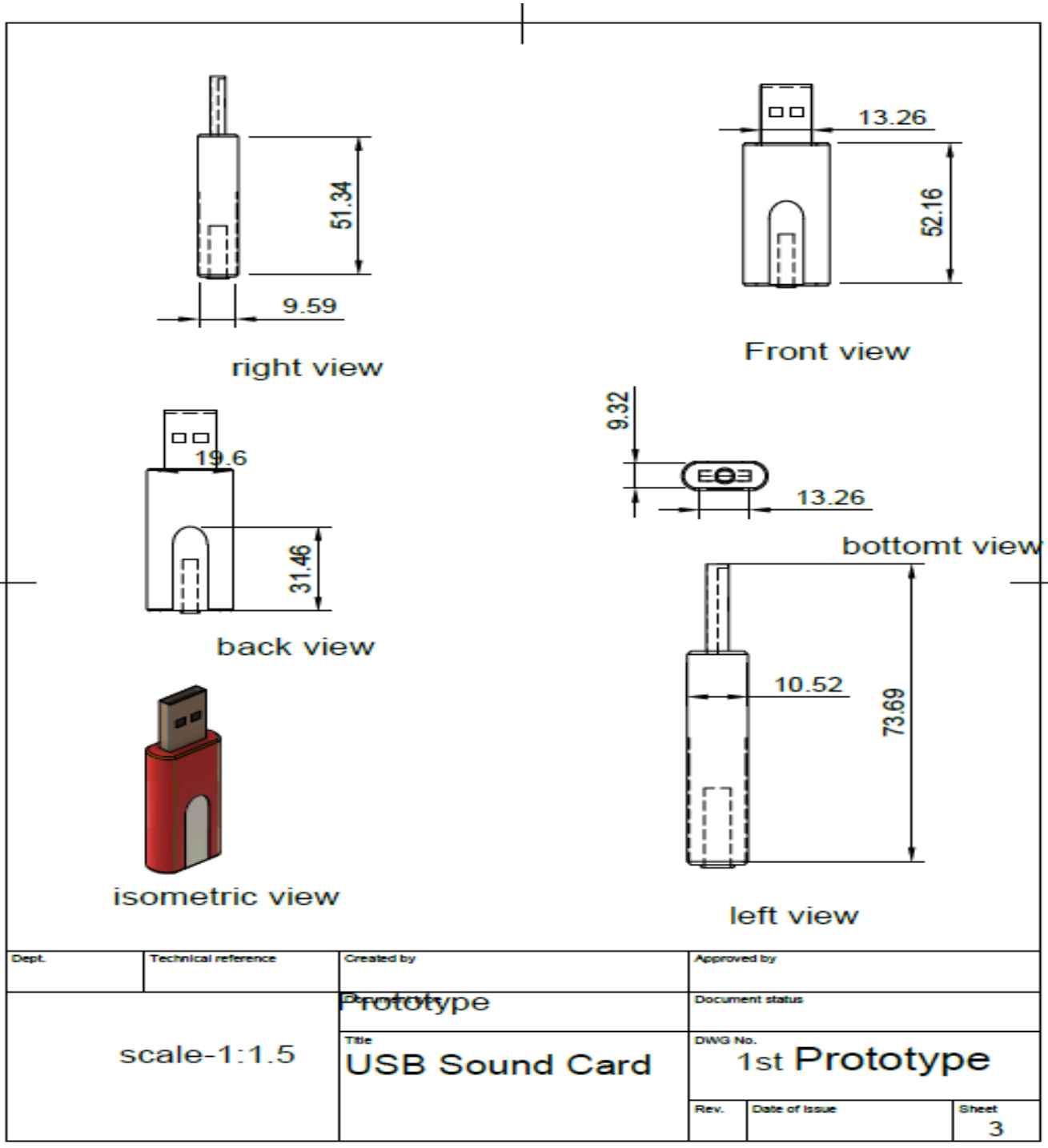
Dept. / Technical reference		Created by: GROUP 41	Approved by:
scale-2:1		Document type: prototype	Document status: 1ST prototype
		Title: IMX477 (12.3MP)	DWG No.
			Rev. / Date of issue / Sheet: 1/1

Drawing 3: Speaker



Dept.	Technical reference	Created by GROUP 41	Approved by
SCALE- 1:1.5		Document type prototype	Document status 1st prototype
		Title **Mini speaker 4ohm 3watt**	DWG No.
			Rev. / Date of issue / Sheet 1/1

Drawing 4: USB Sound Card



Drawing 5: Final model design

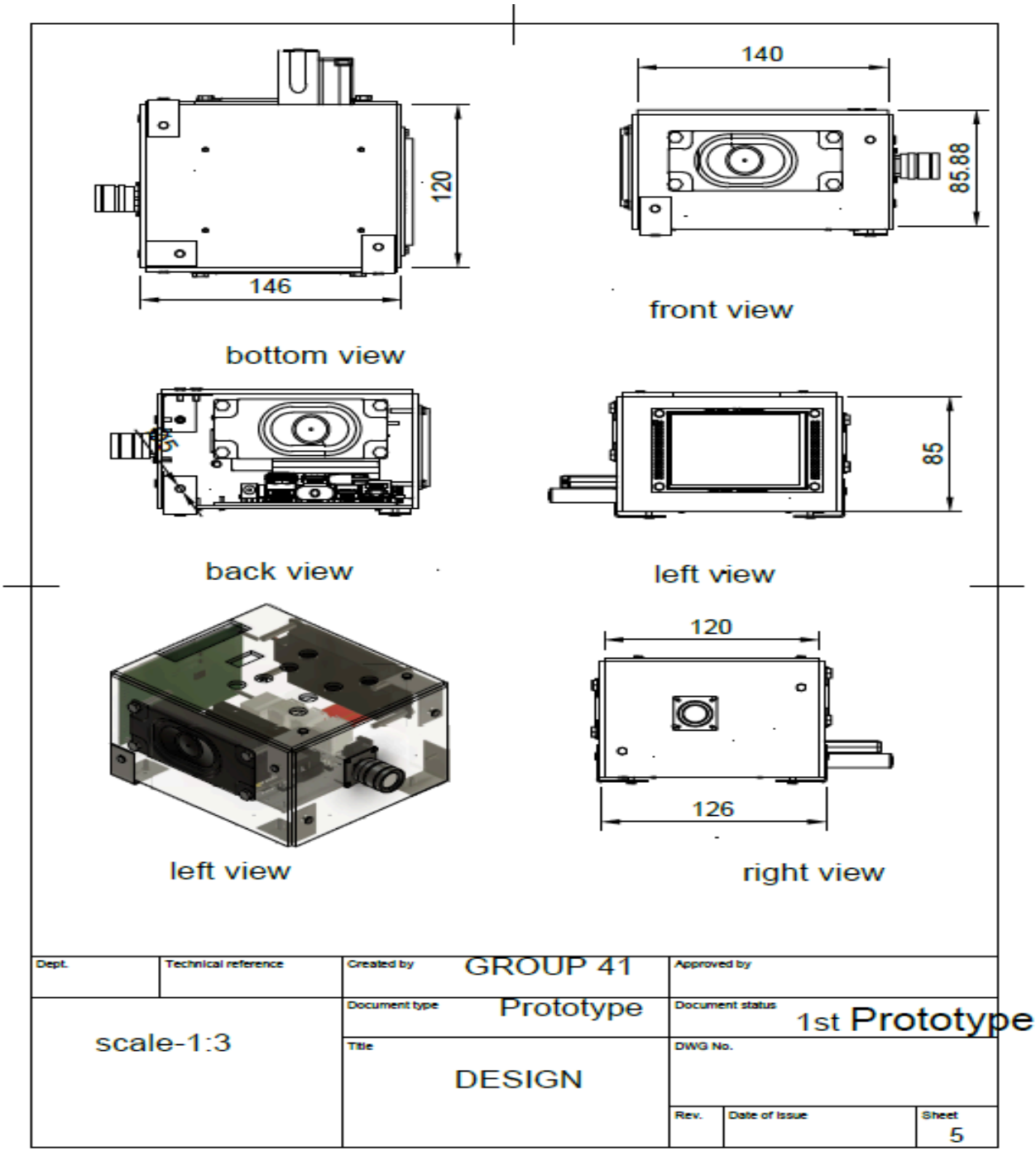


Figure 4.12: Mechanical Design

Chapter 5

Fabrication and Assembly

Bill of Materials:

This is the bill of materials that we ordered till now and further purchase we can update it in this report.

Table 5.1: Bill of Materials

S. No.	Name of Item	Cost	Quantity	Total
1.	Nvidia Jetson nano	**18356**	**1**	**18356**
2.	8Ω 5W Speakers	**400**	**1**	**400**
3.	3.2 Inch Waveshare display	**2548**	**1**	**2548**
4.	USB-GPS receiver	**1247**	**1**	**1247**
5.	Micro sd card	**479**	**1**	**479**
6.	Waveshare USB sound card	**799**	**1**	**799**
7.	Waveshare IMX477-160 12.3 MP camera	**7500**	**1**	**7500**

Total Bill: 31,329 /-

1. Manufacturing Process Description:

Step 1: We meticulously crafted Computer-Aided Design (CAD) blueprints for the various essential components that we had meticulously mapped out for the successful execution of our ambitious project.

Step 2: This model was specifically tailored to accurately represent our carefully chosen acrylic sheets, meticulously accounting for every minute measurement and dimension to ensure utmost precision and accuracy. We ensured that the CAD model impeccably depicted our acrylic sheets' desired specifications and dimensions.

Step 3: We strategically created openings or perforations at precisely designated positions within the designated components, meticulously ensuring that these holes were expertly crafted to accommodate the seamless assembly of various parts.

Step 4: We transitioned towards the pivotal phase of assembling the array of meticulously procured components and acrylic sheets, which comprised a diverse assortment, including the Jetson Nano camera, speaker, GPS module, sound card USB, and display. We methodically united these essential elements, employing a combination of L brackets and adhesive acrylic glue.

Step 5: These invaluable resources were obtained in the widely recognized and universally compatible DXF format. By successfully downloading each CAD model sheet in this preferred format, we fortified our repository of essential design assets equipped with the precise and detailed specifications necessary for our project's accurate visualization and implementation.

Step 6: Advancing diligently in our project's trajectory, we transitioned to a pivotal stage where we embarked on the meticulous process of laser cutting.

Step 7: Finally, we joined all our components through bolts, nuts, L brackets, and glue.

Step 8: Thus, our final model gets ready.

2. Assembly:

In our innovative project, we have incorporated the cutting-edge Jetson Nano microcontroller as a fundamental component. To provide power to this microcontroller, we have ingeniously connected a power bank via a USB cable, ensuring a reliable and portable energy source. Furthermore, we have skillfully integrated a camera and a visually captivating display into the model, enriching its functionality and enhancing its user experience. An essential module is seamlessly integrated into the model, further augmenting its capabilities.

We employed a combination of sturdy nuts, bolts, L brackets, and adhesive glue to achieve a robust and secure assembly. These meticulous attachment methods guarantee all the components' structural integrity and stability, ensuring a solid and reliable construction. With each component securely fastened and meticulously integrated, our model takes its final form, poised and ready for its intended purpose.

3. Limitations and Challenges:

- The project also aimed to measure the distance of the speed breakers and potholes, but that requires either a stereovision camera or a LIDAR sensor. These couldn’t be integrated due to the unavailability of cheaper options.
- Recognition from larger distances requires high-quality images, which requires a high-quality camera.
- Because of the Shortage of Raspberry Pi, we had to move to Jetson Nano which is, of course, more advanced than Raspberry Pi but costlier than Raspberry Pi.
- because of hardware limitations, the training of object detection models on 50k images was limited to 40 epochs.
- The dataset of speed breakers and potholes is very limited, so finding the dataset is one of the biggest challenges. Also, the real-time speed sign board dataset is not directly available on the internet, so we had to create that.

4. Scheduling plan:

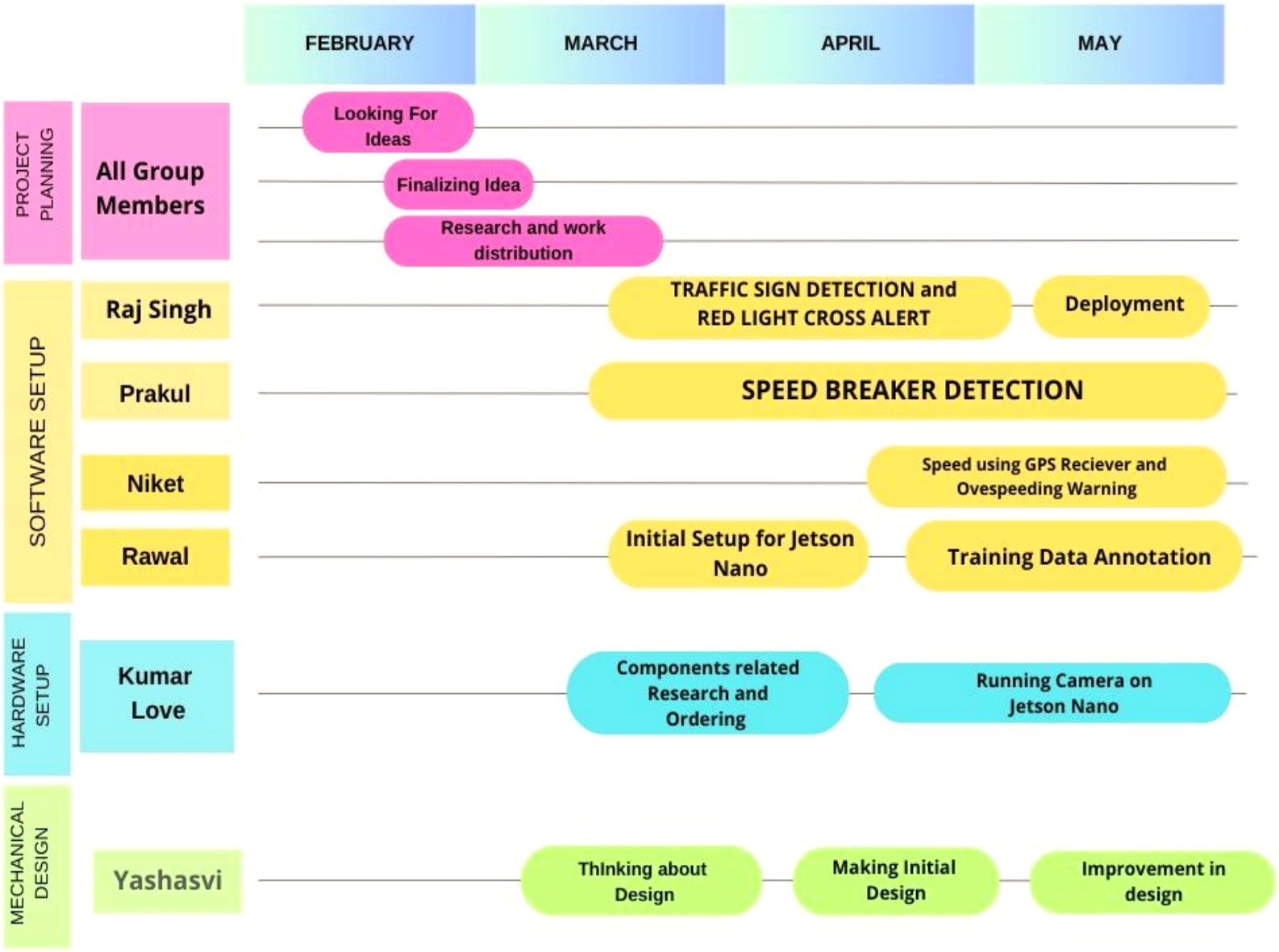


Fig. 5.1: Gantt Chart

5. Contribution:

The project's progress has been carried forward in the scheduled group meetings. From brainstorming over the initial idea to the Final Open House prototype, everyone contributed to their fullest and according to the task

assigned. Every team member **contributed equally** to every group discussion and had an equal role in making this project successful.

The contribution of each team member is as follows:

➔ ***Raj Singh Bani:***

- Yolo model to detect a car running a red light and detection of traffic signs for speeding limit.
- Deployment of Model in Jetson Nano

➔ ***Yashasvi:***

- AutoCAD model for the casing of the whole setup, which would be mounted on the vehicle dashboard.
- Laser Cutting
- Report making of Mechanical design part.
- Assembly

➔ ***Kumar Love:***

- Report Making
- Research regarding the design of the device.
- running camera on Jetson Nano.

➔ ***Prakul:***

- Speed Breaker recognition and potholes detection.
- Report Making.

➔ ***Niket:***

- Speed Calculation using USB GPS receiver.
- Giving warning when overspeeding, speed breaker, and red light is detected.

➔ ***Rawal Ram***

- Initial Setup of Jetson Nano
- Dataset Images annotation

6. Conclusion:

This project combines a number of elements and concepts in order to achieve its aim, which is to build a driver assistance system considering unmarked speed breakers and potholes. As the project's primary components, we utilize a camera to capture video and Jetson Nano using CNN based object detcrction algorithm YOLO. We are achieving accuracy of 92% in case of detection of marked speed breakers and 82.7% in case of unmarked speed breakers.

In future we are trying to improve the accuracy of our model by increasing the dataset and by trying the different deep learning algorithms. Also, we will try to integrate the LIDAR sensor in our model to give the distance of obstacles accurately and will make the design much efficient and compet.

All the work is distributed according to their respective branches as Yashasvi covered mechanical aspects, Raj covered the signboard and traffic light portion, Prakul covered the speed breaker and potholes recognition part, Love covered Jetson Nano and suitable camera + location in the vehicle, Niket covered GPS module, Rawal did electrical aspects.

References:

1. https://www.newindianexpress.com/nation/2017/jul/20/nine-lives-lost-daily-in-accidents-due-to-speed-breakers-1631348.html
2. https://www.thehindu.com/news/cities/Coimbatore/unmarked-speed-breakers-in-coimbatore-pose-a-threat-to-road-users/article66231036.ece
3. https://indianexpress.com/article/cities/pune/pune-unmarked-speed-breaker-police-file-case-against-dead-rider-not-negligent-officials-4928819/
4. https://newsband.in/?p=22739
5. https://www.cartoq.com/unmarked-speed-breaker-causes-massive-accident/
6. https://www.thehindu.com/news/national/over-speeding-caused-maximum-accident-related-deaths-in-2020-government-report/article65857044.ece
7. https://law.resource.org/pub/in/bis/irc/irc.gov.in.099.1988.pdf
8. https://ejmcm.com/article_7334_6bcb311788bdb09d029bcbd5b93535f3.pdf
9. https://www.kaggle.com/datasets/mitangshu11/indian-roads-dataset
10. https://universe.roboflow.com/veldi-aashrith/speed-breaker
11. https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9
12. https://github.com/tzutalin/labelImg
13. https://drive.google.com/drive/folders/1nt6q8Q6piGZscoaBZ6_PSzRNmRaEqV7V?usp=sharing