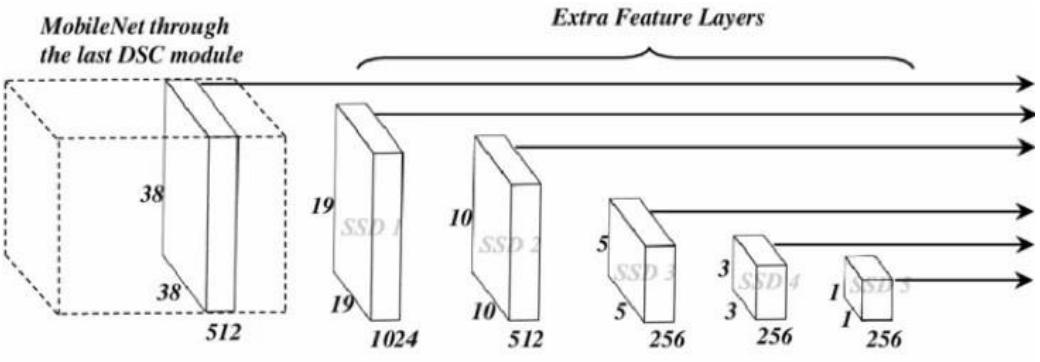
OBJECT DETECTION ALGORITHM(MoblieNetSSD):

ML and **DL** techniques empower many aspects of modern society, like recommendation systems, **text-to-speech devices, or objects identification in images**. Deep learning plays a major role in object detection to identify the class of the image feature of the image. Object detection is the technique of computer vision. Which identifies the labeled objects within the images,videos and live footage.

For object detection one of the models is **(Single Shot Multibox Detector) SSD MobileNet V2 FPN 640X640.**

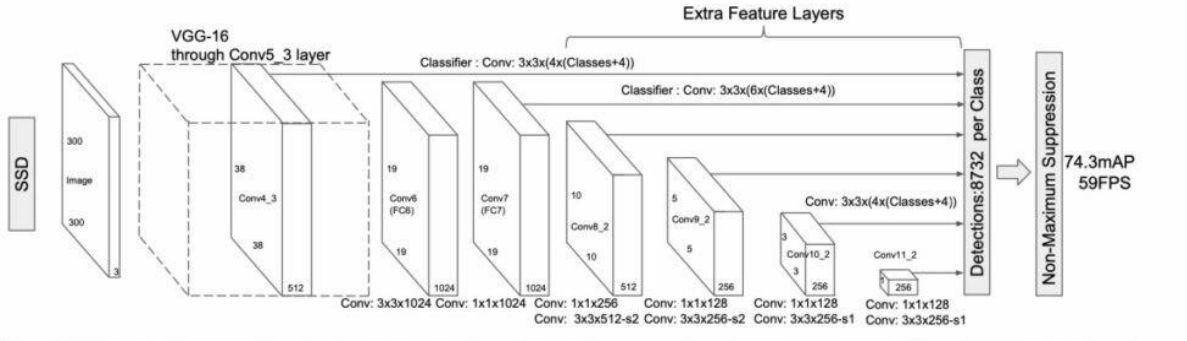
MobileNetSSD:



A lightweight deep neural network architecture called MobileNet was created for mobile devices and embedded vision applications.

MobileNet is based on depth-wise separable filters for its foundational layers. The exception is the first layer, which is a full convolution.

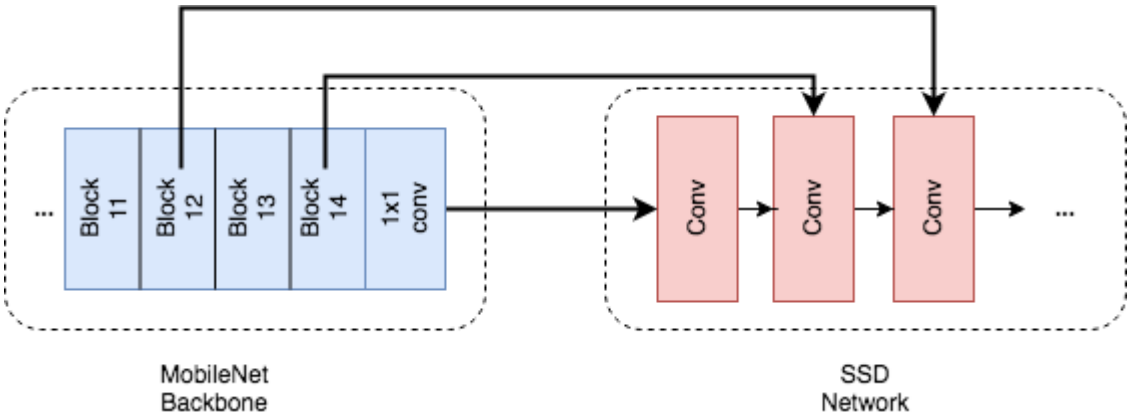
SSD Architecture :



SSD provides localization while mobilenet provides classification. Thus the combination of SSD and mobilenet can produce object detection. The image is taken from an SSD paper. The default classification network of SSD is **VGG-16.**

Single shot object detection or SSD takes one single shot to detect multiple objects within the image. The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes.

SSD is designed to be independent of the base network, and so it can run on top of any base networks, such as VGG, YOLO, or MobileNet.



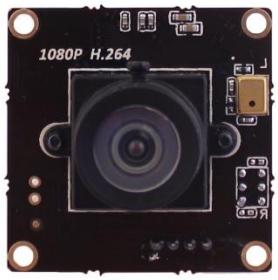
Purpose of JETSON ORIN AGX:

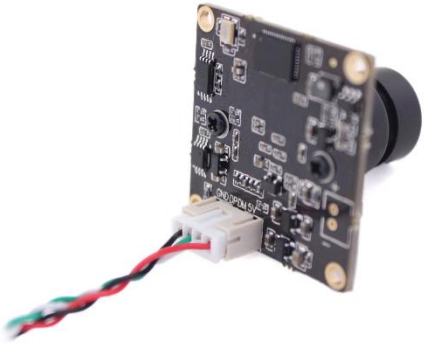
The **NVIDIA Jetson AGX Orin** has a computational speed of **275 trillion operations per second** (TOPS), which is more than **8 times** that of the **Jetson AGX Xavier**. On the Jetson AGX Orin development system, developers may launch their cutting-edge robotics and AI apps for smooth deployment across the whole spectrum of Orin-based production modules.

This is the high end GPU device of this period.This module is used for our training purpose only which trained the more epochs in the very small amount of time.

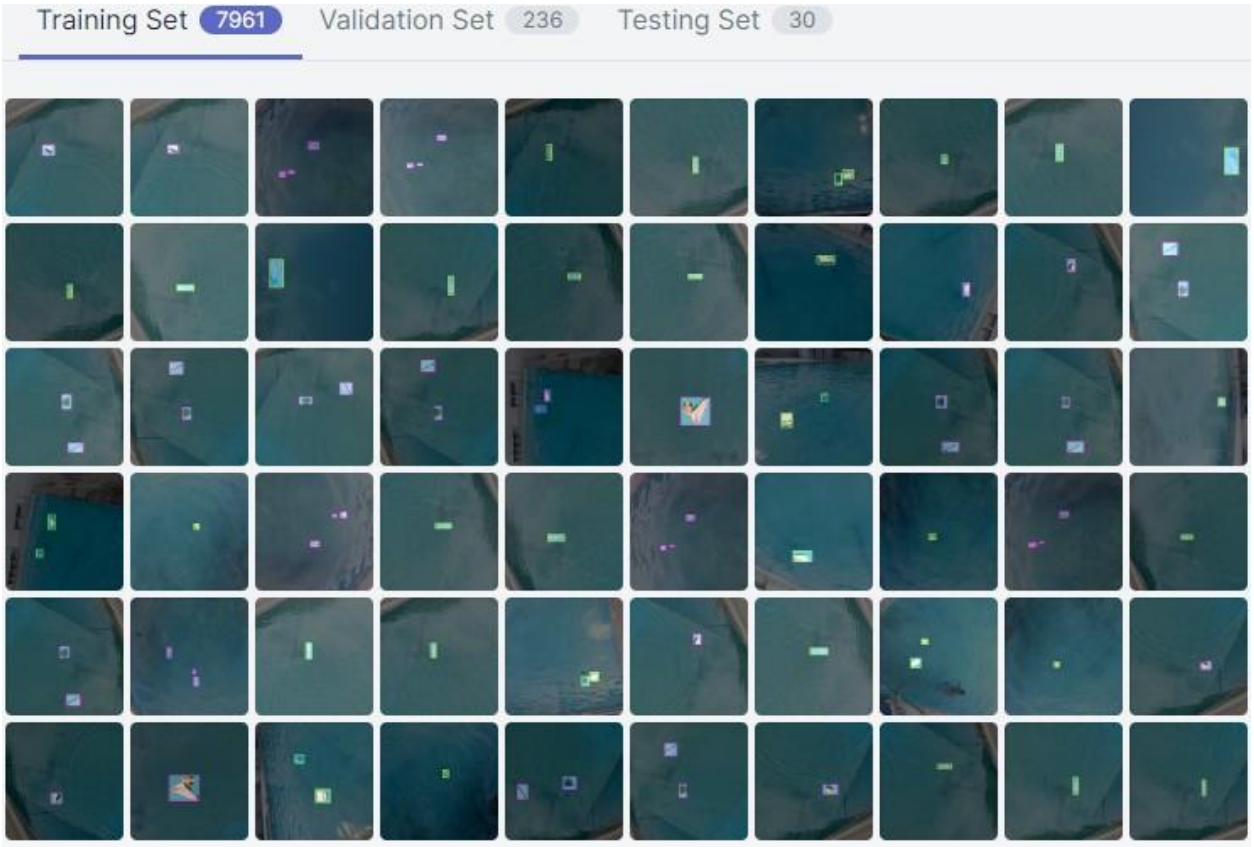
We worked on **Tensorflow** for this training .Our own work was compiled into a dataset. As we gathered more than **5000 pictures** of floating trash. We classified all of the photographs as garbage (biodegradable or non-biodegradable waste) using those images, and we then added further data, producing **8000 images.**

LO

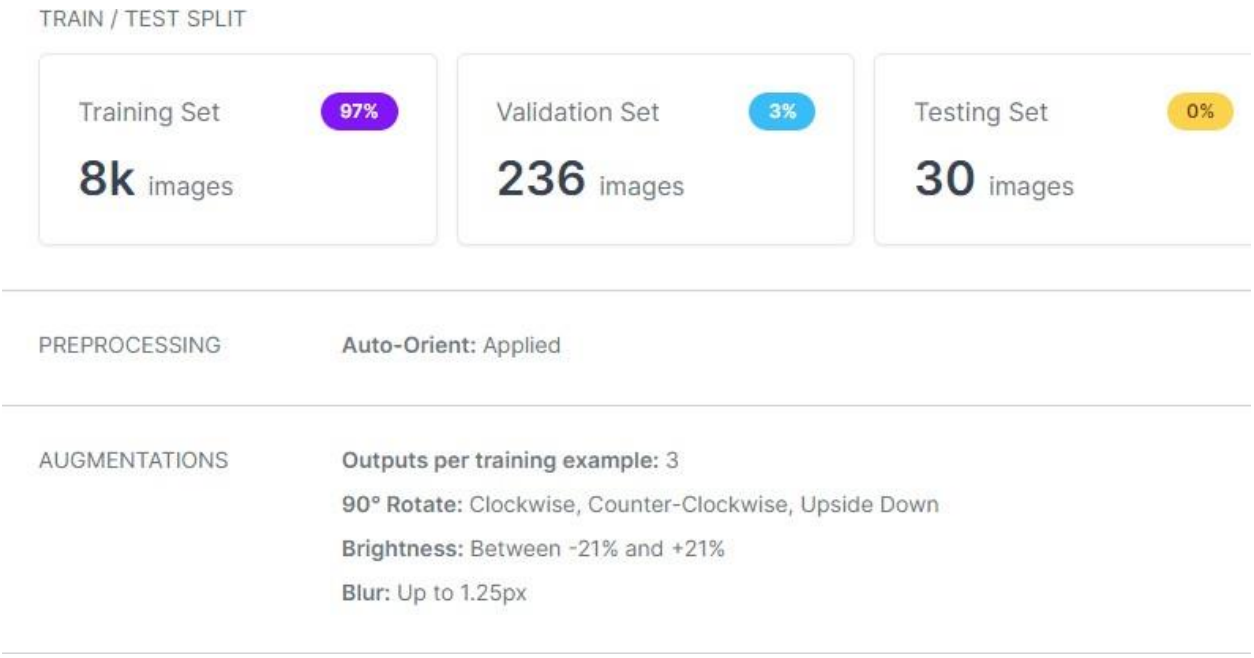




Low-light HD USB Camera is used for waste detection floating over the sea .This Low-Light HD USB Camera is suited to use underwater with excellent low-light performance, good color handling. It is embedded with onboard video compression. A specially-chosen wide-angle, low distortion lens provides excellent picture quality on the ROV, and a built-in microphone allows listening to sounds of the deep, and your vehicle.
Based on the Sony IMX322/323 sensor, this camera uses a large sensor (1/2.9") and a relatively low pixel count (2MP, 1080p). Large sensor meaning that the physical pixel size is large to allow maximum light sensitivity. At the deep sea around down at 300m depth, where we could see subtle bioluminescent creatures in the darkness!
This camera also has an onboard H.264 compression chip so that all of the video compression is done onboard . It doesn't place much load on the main computer. It also means that we will be able to support multiple cameras in the future for alternate views.



97% of the dataset was used for **training**, and **3%** was used for **testing and validation.**



DATASET :

Prepare the Dataset:

To get object detector up and running, we must first collect training photographs. To improve the accuracy of your final model , For waste detection we have took around 8000 images .We split the complete dataset into training dataset as 80% , validation dataset as 18% and test dataset as 2% .

Data Preprocessing:

For more reliability model we Augmented the dataset for into more brightness (Between -21% and +21%)

90° Rotate: Clockwise, Counter-Clockwise, Upside Down **Brightness: Blur:** Up to 1.25px. And label into two classes as Bio degradable and non bio degradable waste .

Annotating the Dataset:

Using **LabelImg tool** we label image and create a file with the same name as the image and the annotation text.

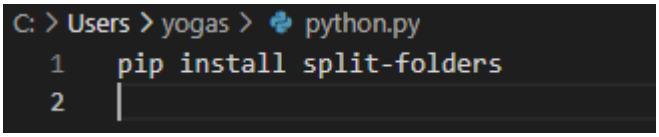
Prepare a set, for example, corresponding to

•images_0.jpg

•images_0.txt

Split Dataset :

The split ratio will be determined by the user, however the most common split is (80%-20%) percent, which implies that 80 percent of the data is utilized for training dataset and 20 percent for testing dataset. *The images and labels are stored in the stated folder architecture.



Using the command we split the dataset.

```
input/
    class1/
        img1.jpg
        img2.jpg
        ...
    class2/
        imgWhatever.jpg
        ...
    ...
```

```
output/
    train/
        class1/
            img1.jpg
            ...
        class2/
            imga.jpg
            ...
    val/
        class1/
            img2.jpg
            ...
        class2/
            imgb.jpg
            ...
    test/
        class1/
            img3.jpg
            ...
        class2/
            imgc.jpg
            ...
```

Create Custom Config File for Training :

Create a file with the name **"custom.yaml"** in the folder. In that file, paste the code below. Set the correct path to the dataset folder, alter the number of classes and their names, and then save it.

Make a file that specifies the training configuration. In **custom.yaml** file, write the following:

•Image_path
•Number_of_classes
•Classes_names_array

C: > Users > yogas > Downloads > yolov7-main > yolov7-main > data > ! coco.yaml

```
1
2   train: ./coco/train2017.txt
3   val: ./coco/val2017.txt
4   test: ./coco/test-dev2017.txt
5   # 2 classes
6   nc: 2
7
8   # class names
9   names: [ 'bio degradable waste', 'Non-bio degradable waste' ]
10
```

>[yolov7/runs/train/yolov7/weights/best.pt]

Training :

- **img** = size of images on which model will train; the default value is 640.
- **batch-size** = batch size used for custom dataset training.
- **epochs** = number of training epochs to get the best model
- **data** = custom config file path
- **weights** = pretrained yolov7 weights

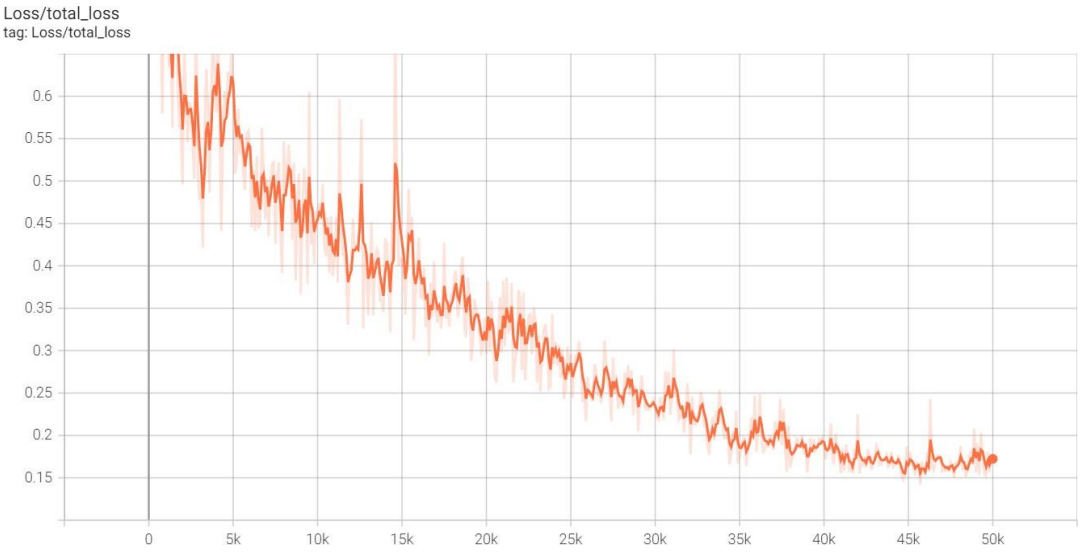
```
(base) C:\Users\yogas>python train.py --weights yolov7.pt --data "data/custom.yaml" --workers 4 --batch-size 4 --img 416
--cfg cfg/training/yolov7.yaml --name yolov7 --hyp data/hyp.scratch.p5.yaml
```

```
(base) C:\Users\yogas>[yolov7/runs/train/yolov7/weights/best.pt]
```

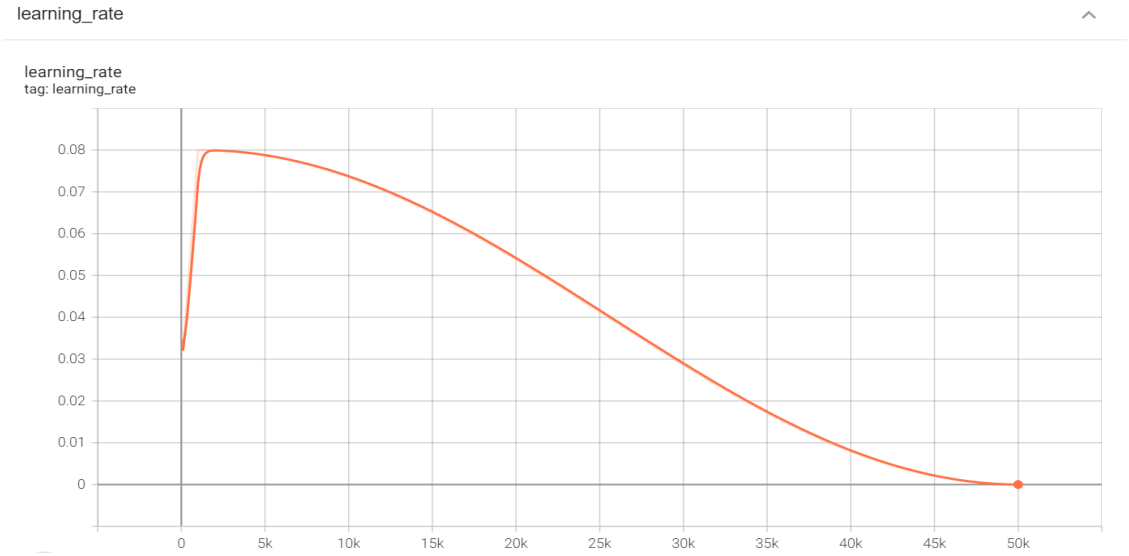
TENSORBOARD STATISTICS :

The **loss rat**e graph for our trained model is shown here.

- At early stage training model the loss rate is high ,
  Over the period it gradually reduced .



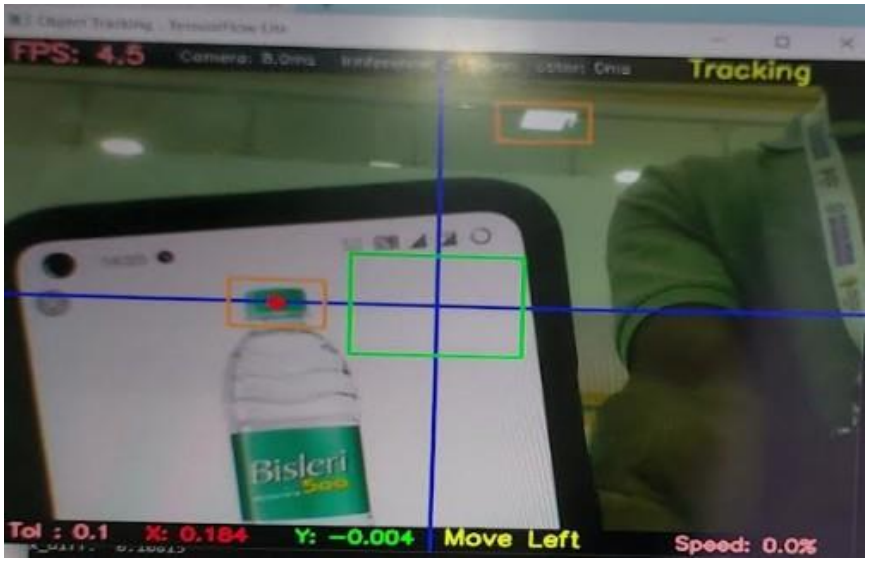
The **learning rate** graph for our trained model is shown here.



OBJECT DETECTION:

As soon as the training is successfully finished, we continue to test the trained model. The model's accuracy is really high, and it accurately predicted the waste. Additionally, we added more characteristics and used a trained model to further refine this. The boat is instructed to trust that side because that is where the rubbish is located.
This was the exact way that our model operated.

```
python detect.py --weights runs/train/yolov7/weights/best.pt --source "path to your testing image"
```



OBJECT DETECTION AND TRACKING WORKS ON FIVE STEPS:

Object tracking allows us to apply a unique ID to each tracked object, making it possible for us to count unique objects in a video. Object tracking is paramount to building a person counter. This object tracking algorithm is called centroid tracking as it relies on the Euclidean distance between (1) existing object centroids and (2) new object centroids between subsequent frames in a video.

STEP :1 Bounding box coordinates and compute centroids

The centroid tracking algorithm is a multi-step process. The centroid tracking algorithm assumes that we are passing in a set of bounding box *(x, y)*-coordinates for each detected waste in each frame.

These bounding boxes can be produced by the SSD algorithm . They are computed for every frame in the video.

Once we have the bounding box coordinates we must compute the “centroid”, or more simply, *the center (x, y)-coordinates* of the bounding box.

STEP 2: Compute Euclidean distance between center and detected wastes:

For every subsequent frame in our video stream we apply **Step #1** of computing object centroids; however, instead of assigning a new unique ID to each detected object (which would defeat the purpose of object tracking), we first need to determine if we can *associate* the *new* object centroids (yellow) with the *old* object centroids (purple). To accomplish this process, we compute the Euclidean distance (highlighted with green arrows) between each pair of existing object centroids and input object centroids.

STEP :3 Update *(x, y)*-coordinates of existing objects

The primary assumption of the centroid tracking algorithm is that a given object will potentially move in between subsequent frames. It continuously updates the position of the waste .

STEP :4 Detect the new waste:

In the event that there are more objects being tracked, we need to register the new object. “Registering” simply means that we are adding the new object to our list of tracked objects by:

1.Assigning it a new object ID’s

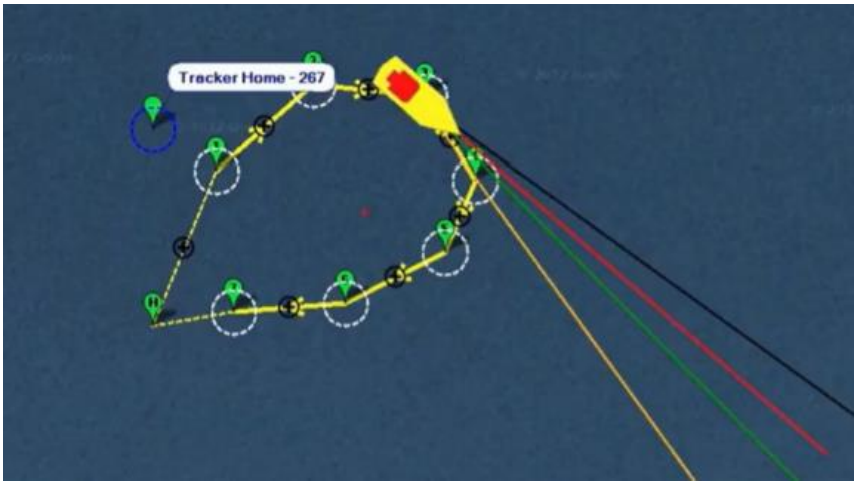
2.Storing the centroid of the bounding box coordinates for that new objects

STEP 5: Deregister old waste:

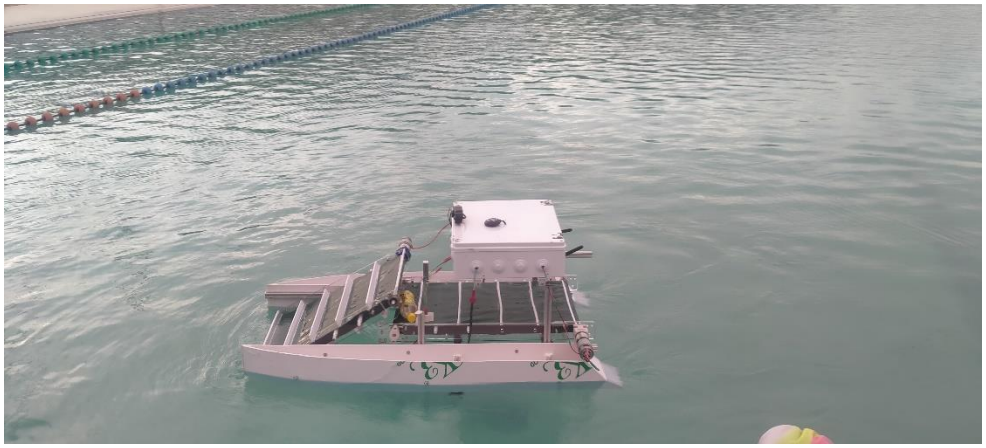
Any reasonable object tracking algorithm needs to be able to handle when an object(waste) has been lost, disappeared, or left the field of view.

Exactly how you handle these situations is really dependent on where your object tracker is meant to be deployed, but for this implementation, we will deregister old objects(waste) when they are not existing in *N* subsequent frames.

AUTONOMOUS BOAT NAVIGATION:



Once the positions of the detected waste objects is stored in a array. It sent to the pixhawk ,using GPS(Global Positioning System) it set wave point over the detected waste .So finally the boat moves through the wave point and collects the waste using the inclined conveyor and stores in flat conveyor.



THANK YOU !!!