BEAST CROSS

A report submitted in partial fulfilment of the Academic requirements for the award of the degree of

Bachelor of Technology

AVINASH SHARMA (19H51A0597)

IFFAT MARIA (19H51A05A2)

ROHAN KONDAM (19H51A05A6)

Y. ARCHANA (19H51A05C1)

V.BHUVANA SRI (19H51A05J2)

UNDER THE COURSE

SOCIAL INNOVATION IN PRACTICE

CENTRE FOR ENGINEERING EDUCATION RESEARCH

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)
(NAAC Accredited with 'A' Grade & NBA Accredited)
(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

2020-21

CENTRE FOR ENGINEERING EDUCATION RESEARCH

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)
(NAAC Accredited with 'A' Grade & NBA Accredited)
(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

CERTIFICATE

This is to certify that the report entitled **"BEAST CROSS"** is a bonafide work done by **AVINASH SHARMA(19H51A0597),IFFAT MARIA(19H51A05A2),ROHAN KONDAM(19H51A05A6), Y. ARCHANA(19H51A05C1), V.BHUVANA SRI(19H51A05J2)** of II B.Tech, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology, submitted to Centre for Engineering Education Research, CMR College of Engineering & Technology, Hyderabad during the Academic Year 2020-21.

(Names of the project coordinators)

1. Mr. B. Suresh Ram
2. Mrs. N. Suvarna

(Mrs.D. Sowjanya)

Head CEER

DECLARATION

We, the students of II B.Tech of Centre for Engineering Education Research , CMR COLLEGE OF ENGINEERING & TECHNOLOGY, Kandlakoya, Hyderabad, hereby declare, that under the supervision of our course coordinators, we have independently carried out the project titled **"Beast Cross"** and submitted the report in partial fulfilment of the requirement for the award of Bachelor of Technology in by the Jawaharlal Nehru Technological University, Hyderabad (JNTUH) during the academic year 2020-2021.

STUDENT NAME	ROLL NUMBER	SIGNATURE
AVINASH SHARMA	(19H51A0597)	
IFFAT MARIA	(19H51A05A2)	
ROHAN KONDAM	(19H51A05A6)	
Y. ARCHANA	(19H51A05C1)	
V.BHUVANA SRI	(19H51A05J2)	

ACKNOWLEDGEMENT

We are obliged and grateful to thank Mrs.D.Sowjanya, Head(CEER), CMRCET, for her cooperation in all respects during the course.

We would like to thank the Principal of CMRCET, Dr.V.A.Narayana, for his support in the course of this project work.

Finally, we thank all our faculty members and Lab Assistants for their valid support.

We own all our success to our beloved parents, whose vision, love and inspiration has made us reach out for these glories.

ABSTRACT

Every year thousands of animals die after being hit by trains. Besides animals, if the tracks are not cleared i.e. boulders on tracks, accidents tend to occur. This phenomenon becomes particularly dangerous in hilly areas.

A cute little elephant calf named *Bholu* is the mascot of Indian Railways and our country's economy is often referred to as the 'Elephant', implying a slow, yet powerful economy. Yet hundreds of elephants and other animals die on the railway tracks.

FIG 1. *Bholu*, mascot of Indian Railways

Trains cannot be stopped from moving on these tracks but something surely can be done. Our major aim is to reduce the demise of such magnificent creatures which in turn avoids damages to trains from such accidents.

Using machine learning and deep learning, a system can be built that provides alarm and helps avoid any unfortunate accidents.

TABLE OF CONTENTS:

1. INTRODUCTION:

Railway is one of the most convenient modes of passenger transport. Animals on train tracks are dangerous for both animals and the train as well.

Railway lines and other linear infrastructure projects which run through the forests for thousands of kilometres affect wildlife corridors. This causes the fragmentation of their habitat. Animals have a natural tendency to cross the tracks to reach nearby water bodies, or in search of food, and sometimes end up getting trapped.

It is known that due to advancement in technology, many tracks were converted from narrow gauge(from 1 ft 11⅝ in to 3 ft 6 in) to standard gauge(4 ft 8½ in), allowing high speed trains to run. The steep embankments alongside these tracks make it even more difficult for the bulky and slow-moving animals to escape when a train approaches and hence accidents occur.

We propose a method for reducing wildlife losses on railways by alerting the train driver with an alert message, relayed in the form of audio and also automatically slow down or stop the train, particularly in areas of high strike risk.

We use Raspberry pi 4, which plays a vital role to coordinate the devices used in the system. The Computer Vision defines a raw overlay of a frame that tells what kind of object is detected, any kind of moving object is clearly identified and recognised. If the animals are detected for at-least 10 consecutive frames, then the alert message is announced.

FIG 2. Hot spot for wildlife crossings to occur

2.LITERATURE REVIEW:

2.1. Existing Solutions:

1. Wildlife exclusion fencing:

The **Swedish Railways** have employed a system of including wildlife exclusion fencing alongside the railway track to avoid wildlife-train collisions.Their goal is to prevent collisions while still allowing animals to cross the railway when no trains are approaching.

According to this method, standard exclusion fences lead animals towards an opening about 50 m wide where movement detectors, thermal cameras, and video cameras monitor the presence and behaviour of animals and trigger the warning system when trains approach. Crushed stone or cattle guards will discourage wild animals from entering the fenced area. Human access to the crosswalk is prohibited.

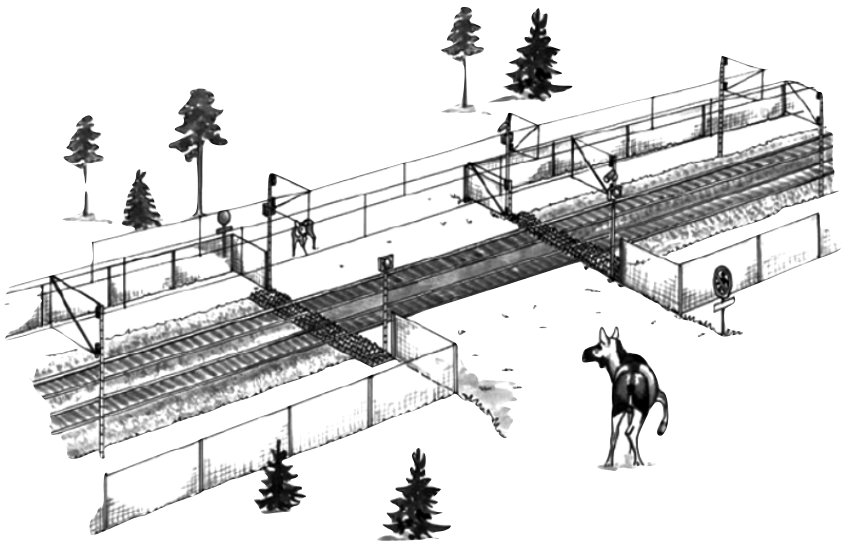


FIG 3. Crosswalk with wildlife exclusion fencing

DISADVANTAGES:

- Providing movement detectors, thermal cameras and continuous monitoring are highly expensive, estimated to be about 50,000 Euros(44,43,700 Rs) per Km.
- Besides, this existing solution to avoid wildlife-train collisions is nearly impossible to implement in India, as the wild life in our country is more diverse and the weather conditions tend to wear down the fencing.

2. The warning system:

Subrat Kar, a professor at IIT-Delhi, developed a device to curb elephant deaths due to train accidents.The Indian Railways extended a fund of Rs 30 Lakhs for this project.

Kar says a fully loaded train travelling at 60 kmph needs about 2 km to come to a halt. Thus, the presence of elephants on the railway track needs to be detected sufficiently ahead of time and at distances greater than 2.5 to 3 km of the moving train if it is to be brought to a halt.

The warning system comprises four sensors that detect feet vibration, infrared rays coming from the approaching animals, a laser detector and a camera to recognise the elephants. The sensors will relay the information back to the station nearest to the segment where the elephants were detected, and then send a signal to the train driver to apply brakes and come to a halt before the point where the elephants were seen.

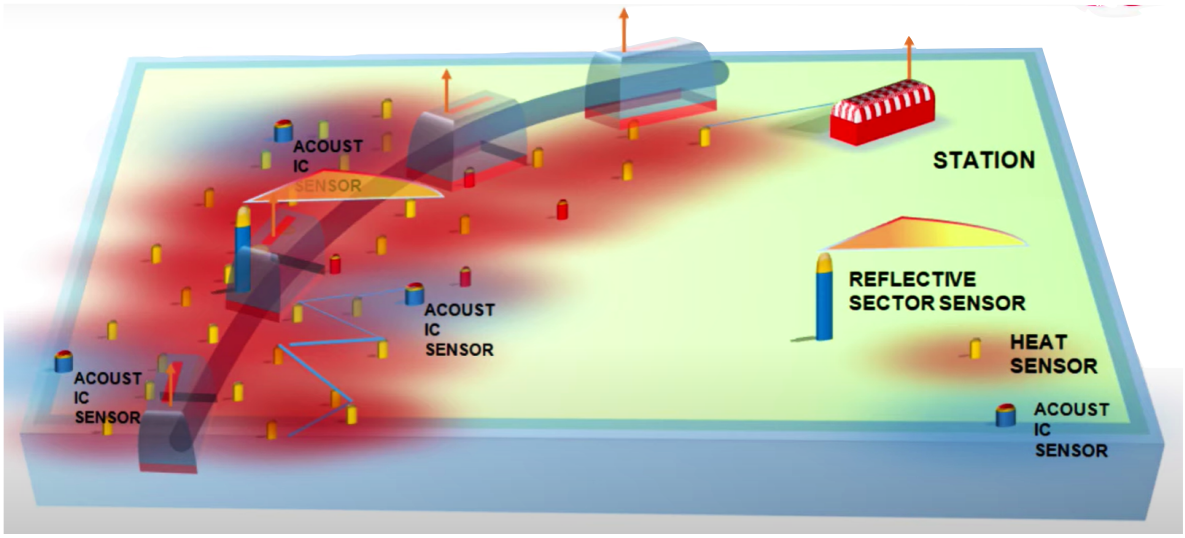


FIG 4. Working model of "The warning system"

DISADVANTAGES:

- Sensors can only be placed in areas anticipated as having high chances of a collision and not everywhere. This means, if the animals are crossing from other areas along the tracks they cant be identified.
- Multiple layers of sensors need to placed as a network to ensure the accurate location of animals.

3. Plan Bee:

The Northeast Frontier Railway (NFR) came with an innovative ‘ *plan bee*’ to prevent elephant deaths on the tracks.And the initiative involves keeping them away from the tracks by scaring them.

As strange as it sounds, elephants are afraid of bees and the buzz of beehives. The Railways included nearly 50 amplifying speaker systems that produce sound of honey bees to keep wild elephants away.

FIG 5. Speaker mounted on a nearby tree

DISADVANTAGES:

- The underlying problem is that when the warning sound is not followed by a real threat and is merely a bluff without consequence, animals will soon habituate and learn to ignore the signal.
- This system works only for elephants.
- The sound may be particularly disturbing for the locals if it is required to place the amplifying speakers over a large area.
- There are cases where the elephant herds traveled across non-migratory paths to cross the tracks, which is unexpected and a cause of chaos among the locals.

4. Warning signals triggered by trains:

This method reduces losses of wildlife on railways by providing animals with warning signals that are triggered by approaching trains, particularly in areas of high strike risk. This system is comparable to the warning signals provided for people at road–rail crossings, it usually emits flashes of light and bell sounds approximately 20 s before train arrival at the location where the system is deployed.

It basically works on the theory that animals will associate these warning signals with train arrival if the warning signal (conditioned stimulus) consistently precedes train arrival (unconditioned stimulus).

The designed model detected passing trains with vibration and magnetic sensors and relayed the signals to warning devices. The cost of this system is around US $225(approximately 16,500 Rs) per unit.

FIG 6. Abstract working of warning signals

DISADVANTAGES:

- This model works on batteries, which need to be replaced frequently.
- If there is a false alarm, animals get used to it, thereby making it ineffective.
- If the alarms are just given 20 seconds before the train arrives, slow moving animals might die.

GAPS IN EXISTING SOLUTIONS:

All the existing solutions had various disadvantages. The following are the key disadvantages that caught our eye:

- Use of bulky components, which are expensive.
- Huge amounts of physical labour is needed to install these models.
- Existing solutions are fixed only to certain hot spot areas, leaving the rest of the tracks.
- Existing models are complex, and require tremendous amounts of research.
- Scaring the animals to avoid accidents is an ancient method and won't be effective.
- Climatic changes may damage the devices since they are left on the tracks.
- High maintenance is required.
- Continuous monitoring and surveillance is required.
- Any technical error may result in the death of animals.

PROPOSED SOLUTION:

As a part of getting the need statement, we have performed a literature survey so that we can have a clear picture of what our prototype must contain, the kind of features it should possess. While doing so we have identified the following constraints:

- It must be compact.
- It must be cost-efficient.
- It must automatically slow down or stop the train, when an obstacle(animal, person, etc.) poses threat to itself and the train.
- It must not require heavy maintenance.

PAPER MODEL:

FIG 7. Elephant before crossing the railway track

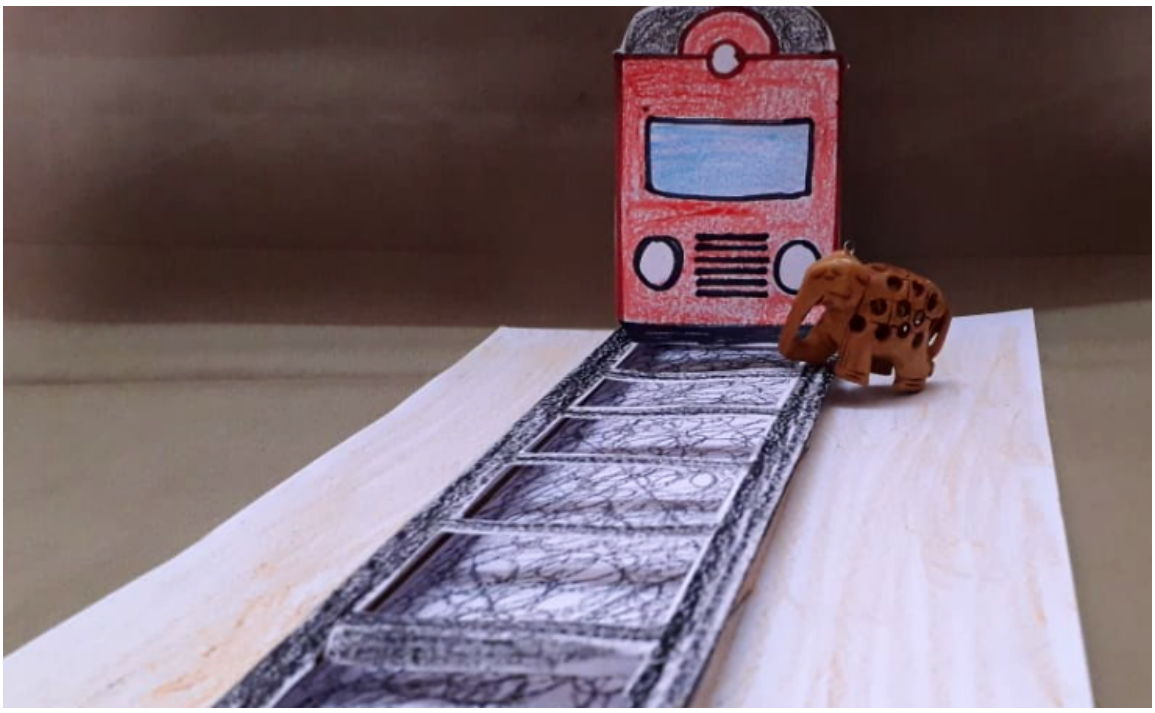
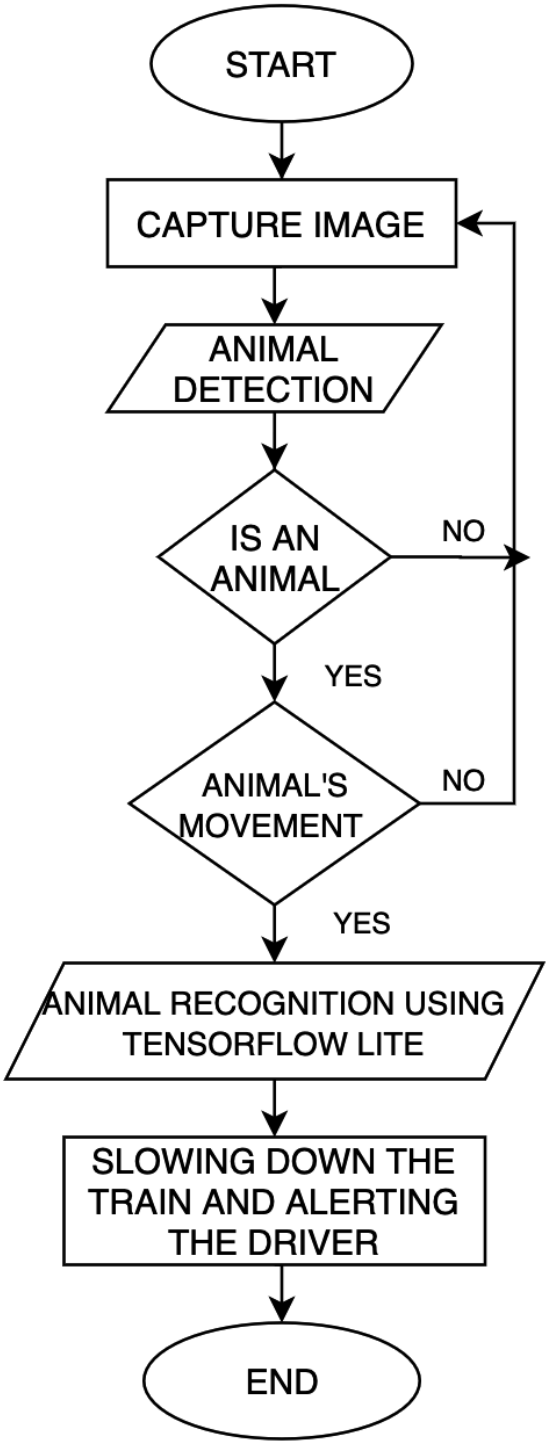


FIG 8. Elephant crossing the track while the train is approaching

FLOW CHART


START
CAPTURE IMAGE
ANIMAL DETECTION
IS AN ANIMAL
NO
YES
ANIMAL'S MOVEMENT
NO
YES
ANIMAL RECOGNITION USING TENSORFLOW LITE
SLOWING DOWN THE TRAIN AND ALERTING THE DRIVER
END

3. PROBLEM DEFINITION:

3.1. Community interaction with the concerned project team:

As part of community visit organised by our college, we visited a nearby village . There we interacted with many locals and discussed about the problems they faced in their daily as well as professional lives, hereby shedding light on various issues.

We happened to meet a middle aged man, who worked in the railways. When asked about incidents that were unforgettable, he mentioned about an incident which caused a lot of chaos in the station he worked at. There was once an accident on the tracks where an elephant was hit by the train and this situation led to a huge delay at that particular station. So we decided to take this incident as the need statement of our course project and design a system which would be a solution to decrease these animal deaths.

3.2. Problem Statement:

Animals on tracks are a threat to both the animal and train as well. Besides animals, if the tracks are not cleared, accidents tend to occur. These accidents lead to loss of lives and damage the train. Trains cannot be stopped from moving on these tracks but any accident can be prevented. Using machine learning and deep learning, a system can be built that alerts the driver and automatically slows down the train helps to avoid any unfortunate accidents.

3.3. Objective:

1. To detect animals and reduce human effort.
2. To develop a simple mechanism to perform the operation.
3. The animals present on the track must be detected and distinguishably identified.
4. When the obstacle is detected, the train driver must be alerted with an audio message, which announces the type of obstacle(animal, person, etc).
5. If the animal is detected the train must be slowed down or eventually stop the train.
6. If the animal is not moving the train must be stopped.

3.4. Requirement Analysis :

Hardware components:

(1) Raspberry Pi 4 :

FIG 9. Raspberry Pi 4

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

The Raspberry Pi module is a capable device that enables people to learn how to program in languages like Scratch and Python. It serves as the main component in our model as it connects all the components together and efficiently finishes the task.

(2) RPi 5MP Camera Module:

FIG 10. Pi camera

A camera module is an image sensor integrated with a lens, control electronics, and an interface like CSI, Ethernet or plain raw low-voltage differential signalling.

In our model, the camera module's task is to detect any impeding danger (obstacles approaching the train). When the corresponding obstacle is caught for 10 consecutive frames, the camera passes on the control to the Raspberry pi for further computing.

(3) Servo Motor SG-90 :

FIG 11. Servo motor

It is a compact motor, with high output power. This servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. Using any servo code, hardware or library to control these servos. It comes with a 3 horns (arms) and hardware.

In our model, it corresponds to the lever of the train which is lowered down as a reflex response automatically to slow down the train and eventually comes to a halt. This mechanism portrays that of the train's.

(4) Ultrasonic distance sensor - HC-SR04 :

FIG 12. Ultrasonic sensors

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.

In our model we have used it for identification of any obstacle on the tracks, while the camera does the identification job. When an object is detected, the response is generated in the form of movement in the servo motor blades.

(5) Jumper wires :

FIG 13. Jumper wires

A jumper wire is an electrical wire, or group of them in a cable, with a connector or pin at each end . We are using jumper wires to efficiently connect all the components together without actually soldering the connections. According to the requirements, male-female (or) female-female jumper wires are used.

Software Used:

(1) OpenCV (Open Source Computer Vision Library):

It is a library of programming functions, mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source Apache 2 License. OpenCV features GPU acceleration for real-time operations.

In our model we use this OpenCV to detect images and clearly distinguish them.

(2)TensorFlow lite:

TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

Tensorflow lite is similar to tensorflow, it takes input as a multi-dimensional array, also known as tensors.We can construct a flowchart of operations that you want to perform on that input. We used Tensorflow lite in our prototype for object detection.

3.5. Methodology :

Our main motive is to reduce the loss of lives on railway tracks due to unnoticed movement along them. Wild animals which move in herds do not have quick reflexes to move away from the tracks when the train approaches at high speed. So, we designed this system which automatically slows down the train and stops it, hereby saving lives off the animals on that track.

4.1. Conceptual Design :

Our design includes mainly two steps: object detection and movement off the lever, i.e lowering the lever as the object gets closer.

FIG 15. Object detection using OpenCV and Tensorflow lite

Prototype:

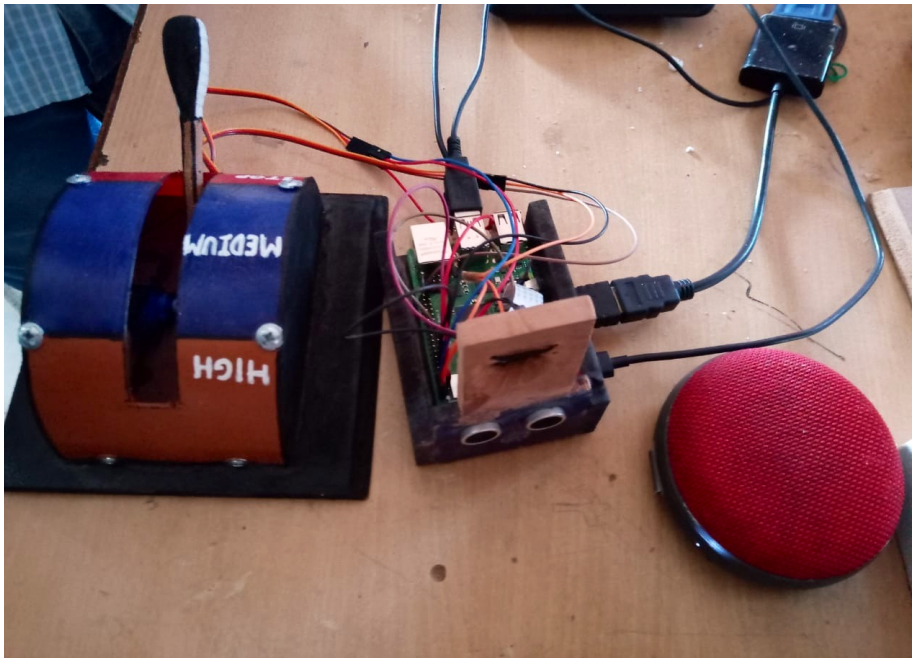
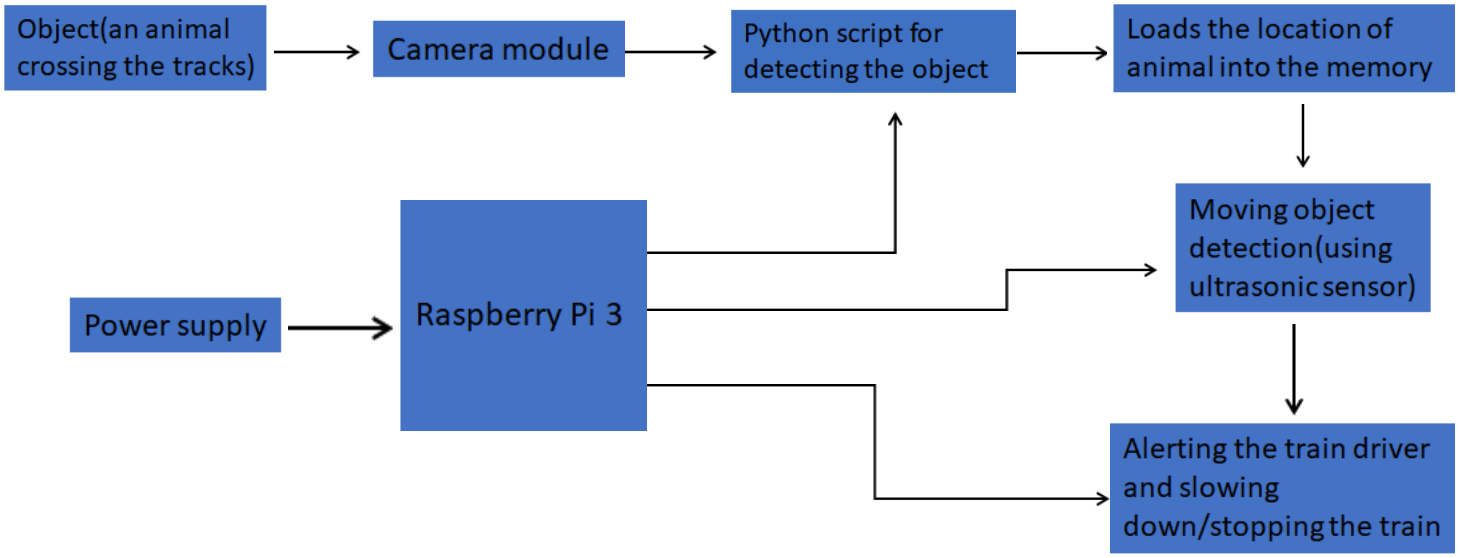


FIG 16. Experimental setup of the prototype

Here we use Raspberry Pi 3 as an intermediate that connects the real world to the computing side of our model. The camera module, ultrasonic sensor, IR sensor and servo motor together form a system that detects any obstacle on the tracks and automatically lowers the train lever to stop the train.

4.2. Block Diagram :



4.3. Design Description :

Since we are all aware of the fact that, animals crossing the train tracks is dangerous for animals as well as the train, this causes considerable loss on both the sides, we have come up with an idea of animal detection and automatic lever(brakes) control so that when there is an obstacle on the track, the train slows down and eventually stops.

- The Raspberry Pi serves as an intermediate between the real world and the computing part of our model.

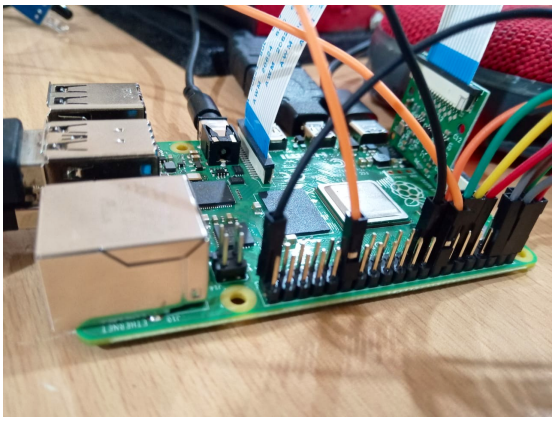


FIG 17. RaspberryPi with all the connections

- The camera module is placed on the train Pilot (in front of the train) and identifies any animal on the tracks.

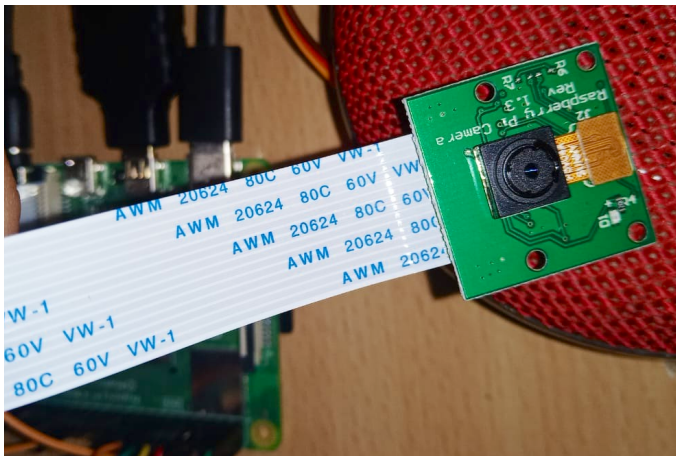


FIG 18. Pi camera to identify the obstacle on tracks

In our prototype the ultrasonic sensor detects any obstacle on the tracks and the camera module identifies the object that is on the tracks.

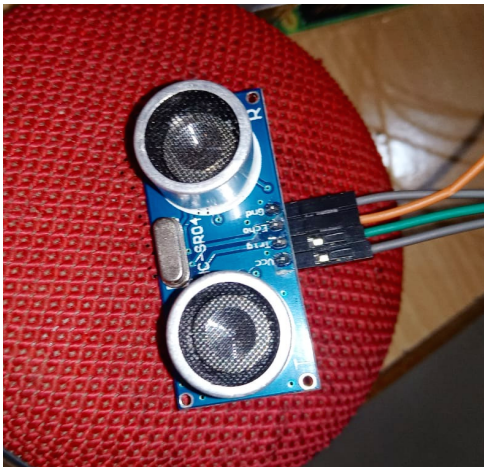


FIG 19.IR sensor and Ultrasonic sensor

It in turn relays an audio message to the train driver, announcing the obstacle on the tracks. The servo motors, control the movement of the train lever as the object distance with the train decreases.

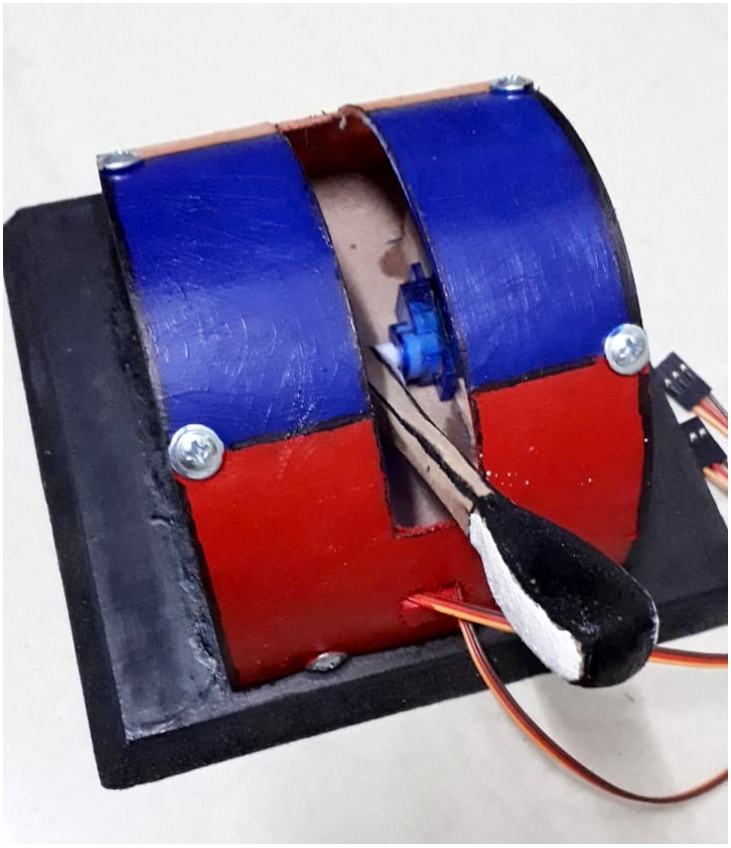


FIG 20. Lever to automatically control the brakes.

5.IMPLEMENTATION:

5.1 .Results and Discussions:

1. Automation of the brakes for the train is completely feasible.
2. Less manpower is required compared to other models.
3. Not only animals, but any kind of obstacle can be detected easily and be brought to the notice of the train driver immediately.

Our prototype can be used mostly on Trains as its sole purpose is to alert the driver of any dangers on the tracks and automatically control the lever to stop the train.

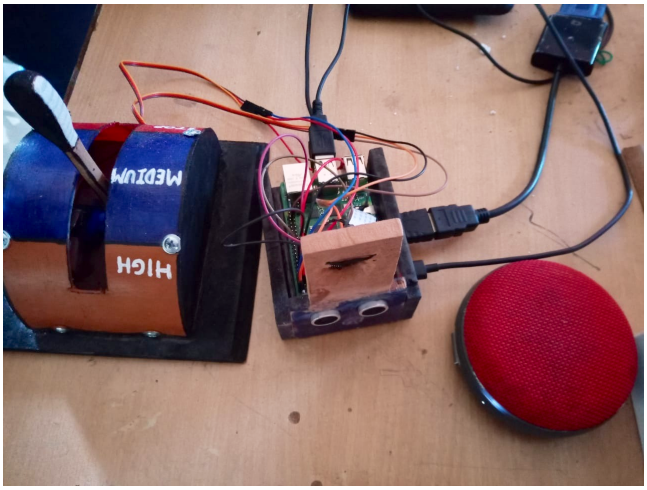


FIG 21. Experimental setup of the prototype

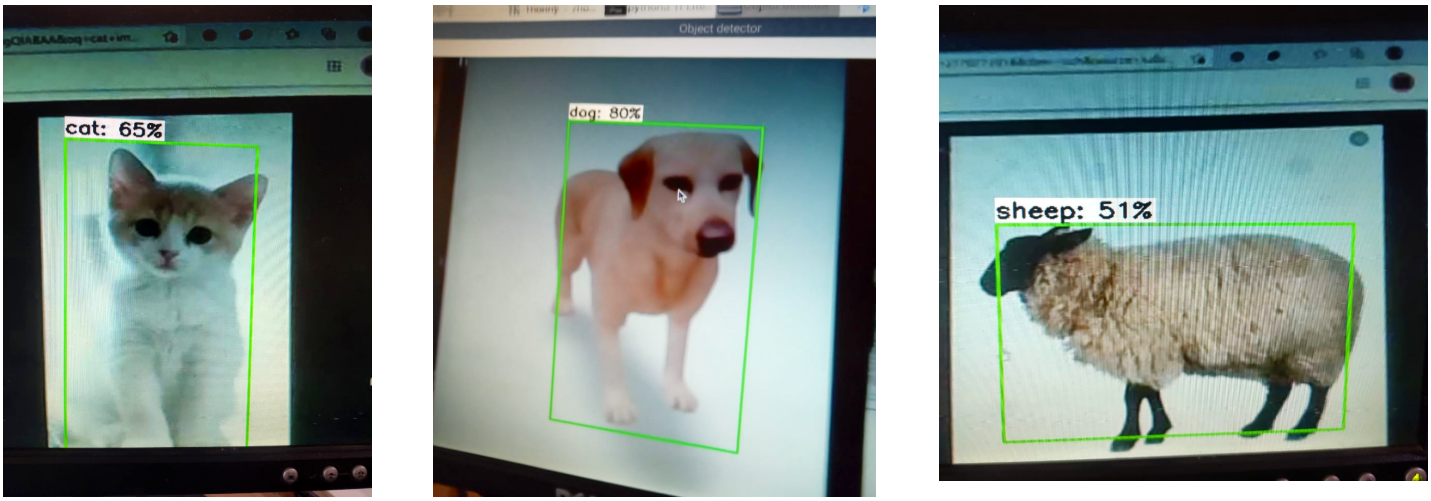


FIG 22. Animals being accurately recognised

5.2 .Conclusion:

We hereby conclude that our prototype, "Beast cross" helps avert dangerous train accidents and helps save the lives of wild-animals, whose habitat is perforated by railway tracks. It alerts the driver of the train and immediately slows down and eventually stops the train, it is cost efficient compared to the existing solutions to curb this problem and also works pretty effectively.

6.1 APPENDIX:

https://www.scoopwhoop.com/inothernews/elephants-killed-on-railway-tracks/

https://www.researchgate.net/publication/319893322_Wildlife_Deterrent_Methods_for_Railways-An_Experimental_Study

https://www.researchgate.net/publication/286712967_Wildlife_conservation_and_rail_track_monitoring_using_wireless_sensor_networks

6.2 REFERENCES:

https://www.newindianexpress.com/cities/bhubaneswar/2021/jan/03/iit-profs-answer-to-jumbodeaths-in-train-collision-2244698.html#:~:text=reports%20Diana%20Sahu-,An%20Odia%20professor%27s%20innovation%20may%20turn%20out%20to%20be%20a,elephants%20on%20the%20railway%20tracks.

https://www.electronicshub.org/raspberry-pi-servo-motor-interface-tutorial/

SERVO MOTER

```
import RPi.GPIO as GPIO
from time import sleep
from multiprocessing import Process
import sys
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
pwm=GPIO.PWM(3, 50)
pwm=GPIO.PWM(11, 50)
pwm.start(30)
duty = 0
try:
    def SetAngle1(angle):
        global duty
        duty= angle/18+2
        GPIO.output(3, True)
        pwm.ChangeDutyCycle(duty)
        GPIO.output(3, False)

    def SetAngle2(angle):
        angle = 180 - angle
        global duty
        duty = angle/18+2
        GPIO.output(11,True)
        pwm.ChangeDutyCycle(duty)
        GPIO.output(11, False)
    def Main(ang):
        if __name__=='__main__':
            p1 = Process(target=SetAngle1(ang))
            p1.start()
            p2 = Process(target=SetAngle2(ang))
            p2.start()
except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

ULTRASONIC SENSOR

```
from servo import Main
import time
import RPi.GPIO as GPIO
from horn import th
from ir import ir
GPIO. setmode (GPIO. BOARD)
trig = 11
echo = 13
GPIO. setup (echo, GPIO .IN)
GPIO. setup (trig, GPIO. OUT)
try:
    ang=30
    while True:
        GPIO.output (trig, True)
        time.sleep (0.00001)
        GPIO.output (trig, False)
        while GPIO.input(echo) == 0:
            pass
            start = time. time ()
```

```
        while GPIO.input(echo) == 1:
            pass
            end = time. time ()
        distance = int(((end - start) * 34300) / 2)
        print("distance:", distance, "cm")
        if distance <50 and ir():
            th(7)
            Main(30)
        elif distance >=50 and distance <=400:
            th(2)
            Main(90)
        else:
            Main(160)
        time.sleep (0.5)
except KeyboardInterrupt:
    GPIO.cleanup()
```

TEXT TO SPEECH

```
from gtts import gTTS
import time
import os
from playsound import playsound
try:
    def tts():
        mytext = input()
        myobj = gTTS(text=mytext, lang='en', slow=False)
        myobj.save("output.mp3")
        for i in range(0,3):
            playsound('output.mp3')
            time.sleep(0.7)
except:
    print("An exception occurred")

import os
from playsound import playsound
import time
try:
    def th(n):
        for i in range(0,n):
            playsound('train_horn.mp3')
            time.sleep(0.7)
except:
    print("An exception occurred")
```

HORN

```
from gtts import gTTS
import time
import os
from playsound import playsound
try:
    def tts(speech):
        mytext = speech
        language = 'en'
        myobj = gTTS(text=mytext, lang=language, slow=False)
        myobj.save("output.mp3")
        for i in range(0,3):
            playsound('output.mp3')
            time.sleep(0.7)
```

```
except:
    print("An exception occurred")
tts('ant')
```

OBJECT DETECTION

```
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

# Define VideoStream class to handle streaming of video from webcam in separate processing
thread
# Source - Adrian Rosebrock, PyImageSearch: https://www.pyimagesearch.com/2015/12/28/
increasing-raspberry-pi-fps-with-python-and-opencv/
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

            # Variable to control when the camera is stopped
        self.stopped = False

    def start(self):
            # Start the thread that reads frames from the video stream
        Thread(target=self.update,args=()).start()
        return self

    def update(self):
        # Keep looping indefinitely until the thread is stopped
        while True:
            # If the camera is stopped, stop the thread
            if self.stopped:
                # Close camera resources
                self.stream.release()
                return

            # Otherwise, grab the next frame from the stream
            (self.grabbed, self.frame) = self.stream.read()

    def read(self):
            # Return the most recent frame
        return self.frame

    def stop(self):
            # Indicate that the camera and thread should be stopped
        self.stopped = True

# Define and parse input arguments
```

```
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected
objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does
not support the resolution entered, errors may occur.',
                    default='1280x720')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

# Import TensorFlow libraries
# If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular
tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tflite_runtime')
if pkg:
    from tflite_runtime.interpreter import Interpreter
    if use_TPU:
        from tflite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwise use default
'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object detection
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]
```

```
# Have to do a weird fix for label map if using the COCO "starter model" from
# https://www.tensorflow.org/lite/models/object_detection/overview
# First label is '???', which has to be removed.
if labels[0] == '???':
    del(labels[0])

# Load the Tensorflow Lite model.
# If using Edge TPU, use special load_delegate argument
if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                              experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()

# Initialize video stream
videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
time.sleep(1)

#for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
while True:

    # Start timer (for calculating frame rate)
    t1 = cv2.getTickCount()

    # Grab frame from video stream
    frame1 = videostream.read()

    # Acquire frame and resize to expected shape [1xHxWx3]
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    # Perform the actual detection by running the model with the image as input
    interpreter.set_tensor(input_details[0]['index'],input_data)
    interpreter.invoke()

    # Retrieve detection results
```

```
    boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of
detected objects
    classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
    scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
    #num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected objects
(inaccurate and not needed)

    # Loop over all detections and draw detection box if confidence is above minimum threshold
    for i in range(len(scores)):
        if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

            # Get bounding box coordinates and draw box
            # Interpreter can return coordinates that are outside of image dimensions, need to force
them to be within image using max() and min()
            ymin = int(max(1,(boxes[i][0] * imH)))
            xmin = int(max(1,(boxes[i][1] * imW)))
            ymax = int(min(imH,(boxes[i][2] * imH)))
            xmax = int(min(imW,(boxes[i][3] * imW)))

            cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

            # Draw label
            object_name = labels[int(classes[i])] # Look up object name from "labels" array using class
index
            label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
            labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get
font size
            label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of
window
            cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0],
label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
            cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0,
0), 2) # Draw label text

    cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),
(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

    # All the results have been drawn on the frame, so it's time to display it.
    cv2.imshow('Object detector', frame)

    # Calculate framerate
    t2 = cv2.getTickCount()
    time1 = (t2-t1)/freq
    frame_rate_calc= 1/time1

    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break

# Clean up
cv2.destroyAllWindows()
videostream.stop()
```