**INTRODUCTION**

A new approach towards automobile safety and security with autonomous region primarily based automatic automotive system is projected during this conception. We have atendency to propose 3 distinct however closely connected ideas viz. a Drowsy Driver Detection system and a traffic detection system with external vehicle intrusion dodging primarily based conception. In recent time's automobile fatigue connected crashes have very enlarged. so as to attenuate these problems, we've incorporated driver alert system by watching each the driver's eyes still as sensing still because the driver state of affairs based  primarily based native setting recognition based AI system is projected.

**ABSTRACT**

Nowadays, more and more professions require long-term concentration. Drivers must keep a close eye on the road, so they can react to sudden events immediately. Driver fatigue often becomes a direct cause of many traffic accidents. Therefore, there is a need to develop the systems that will detect and notify a driver of her/him bad psychophysical condition, which could significantly reduce the number of fatigue-related car accidents. However, the development of such systems encounters many difficulties related to fast and proper recognition of a driver’s fatigue symptoms. One of the technical possibilities to implement driver drowsiness detection systems is to use the vision-based approach. This article presents the currently used driver drowsiness detection systems. Here we are detecting the driver drowsiness by estimating vision system of him.

**INTRODUCTION**

Driver drowsiness detection is a car safety technology which prevents accidents when the driver is getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads. Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. Driver inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person’s attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, i.e., decreased driving performance, longer reaction time, and an increased risk of crash involvement. shows the block diagram of overall system. Based on Acquisition of video from the camera that is in front of driver perform real-time processing of an incoming video stream in order to infer the driver’s level of fatigue if the drowsiness is Estimated then it will give the alert by sensing the eyes.

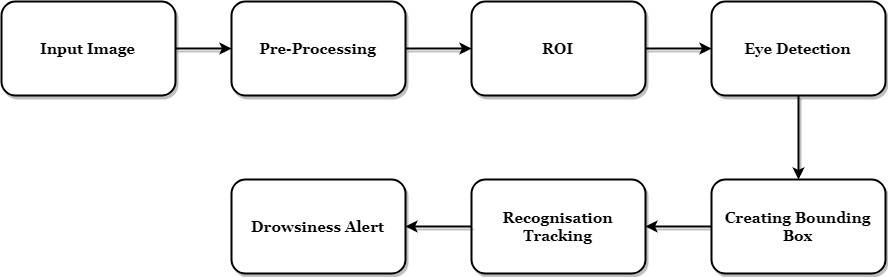
**EXISTING SYSTEM**

Here we are used SVM(support vector machine) to classify the components in the input video. While cropping the region of interest components in the video is not accurate. Sometimes it will show regions wrong. To sense the eyes first we have to create boundary boxes for that and a classification algorithm. The algorithm of SVM will not support.

**PROPOSED METHOD**

There are several different algorithms and methods for eye tracking, and monitoring. Most of them in some way relate to features of the eye (typically reflections from the eye) within a video image of the driver. The original aim of this project was to use the retinal reflection as a means to finding the eyes on the face, and then using the absence of this reflection as a way of detecting when the eyes are closed. Applying this algorithm on consecutive video frames may aid in the calculation of eye closure period. Eye closure period for drowsy drivers are longer than normal blinking. It is also very little longer time could result in severe crash. So we will warn the driver as soon as closed eye is detected.

**BLOCK DIAGRAM**

****

**ADVANTAGES**

* Region of interest is clear to identify
* Bounding box creation and tracking

**APPLICATIONS**

* Real time tracking applications
* Classification of small particles

**HARDWARE REQUIRMENTS**

* Camera
* Personal computer

**SOFTWARE REQUIREMENTS**

* Python 2.7 or above versions
* Anaconda software

**CONCLUSION**

The driver abnormality monitoring system developed is capable of detecting drowsiness, drunken and reckless behaviours of driver in a short time. The Drowsiness Detection System developed based on eye closure of the driver can differentiate normal eye blink and drowsiness and detect the drowsiness while driving. The proposed system can prevent the accidents due to the sleepiness while driving. The system works well even in case of drivers wearing spectacles and even under low light conditions if the camera delivers better output. Information about the head and eyes position is obtained through various self-developed image processing algorithms. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long, a warning signal is issued. processing judges the driver’s alertness level on the basis of continuous eye closures.

Summary

Our drowsiness detector hinged on two important computer vision techniques:

* Facial landmark detection
* Eye aspect ratio

[Facial landmark prediction](https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/) is the process of localizing key facial structures on a face, including the eyes, eyebrows, nose, mouth, and jawline.

Specifically, in the context of drowsiness detection, we only needed the eye regions (I provide more detail on [how to extract each facial structure from a face here](https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/)).

Once we have our eye regions, we can apply the *eye aspect ratio* to determine if the eyes are closed. If the eyes have been closed for a sufficiently long enough period of time, we can assume the user is at risk of falling asleep and sound an alarm to grab their attention. More details on the eye aspect ratio and how it was derived can be found in [my previous tutorial on blink detection](https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/).

# **Automatic License Plate Detection & Recognition using deep learning**

https://miro.medium.com/max/60/1*PUB8y2T2Qu__nQ3FQj3fmA.jpeg?q=20



***Introduction***

The massive integration of information technologies, under different aspects of the modern world, has led to the treatment of vehicles as conceptual resources in information systems. Since an autonomous information system has no meaning without any data, there is a need to reform vehicle information between reality and the information system.This can be achieved by human agents or by special intelligent equipment that will allow identification of vehicles by their registration plates in real environments. Among intelligent equipment, mention is made of the system of detection and recognition of the number plates of vehicles.The system of vehicle number plate detection and recognition is used to detect the plates then make the recognition of the plate that is to extract the text from an image and all that thanks to the calculation modules that use location algorithms, segmentation plate and character recognition.The detection and reading of license plates is a kind of intelligent system and it is considerable because of the potential applications in several sectors which are quoted:

- **Command force**: This system is used for the detection of stolen and searched vehicles. The detected plates are compared to those of the reported vehicles.

- **Parking management:** The management of car entrances and exits.

* **Road safety:** This system is used to detect license plates exceeding a certain speed, coupling the plate reading system with road radar, crossing wildfires …

Our project will be divised into 3 steps :

***Step1 : Licence plate detection***

In order to detect licence we will use Yolo ( You Only Look One ) deep learning object detection architecture based on convolution neural networks.

This architecture was introduced by Joseph Redmon , Ali Farhadi, Ross Girshick and Santosh Divvala first version in 2015 and later version 2 and 3.

Yolo v1 : Paper [link](https://arxiv.org/pdf/1506.02640.pdf).

Yolo v2 : Paper [link](https://arxiv.org/pdf/1612.08242.pdf).

Yolo v3 : Paper [link](https://arxiv.org/pdf/1804.02767.pdf).

Yolo is a single network trained end to end to perform a regression task predicting both object bounding box and object class.

This network is extremely fast, it processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second.

**Implementing YOLO V3:**

First, we prepared a dataset composed of 700 images of cars that contains Tunisian licence plate, for each image, we make an xml file ( Changed after that to text file that contains coordinates compatible with Darknet config file input. Darknet : project used to retrain YOLO pretrained models) using a desktop application called [LabelImg](https://github.com/tzutalin/labelImg" \t "_blank).

# First download Darknet project  
$ git clone <https://github.com/pjreddie/darknet.git># in "darknet/Makefile" put affect 1 to OpenCV, CUDNN and GPU if you # want to train with you GPU then time thos two commands  
$ cd darknet  
$ make# Load convert.py to change labels (xml files) into the appropriate # format that darknet understand and past it under darknet/  
 <https://github.com/GuiltyNeuron/ANPR># Unzip the dataset  
$ unzip dataset.zip# Create two folders, one for the images and the other for labels  
$ mkdir darknet/images  
$ mkdir darknet/labels# Convert labels format and create files with location of images  
# for the test and the training  
$ python convert.py# Create a folder under darknet/ that will contain your data  
$ mkdir darknet/custom# Move files train.txt and test.txt that contains data path to  
# custom folder  
$ mv train.txt custom/  
$ mv test.txt custom/# Create file to put licence plate class name "LP"  
$ touch darknet/custom/classes.names  
$ echo LP > classes.names# Create Backup folder to save weights  
$ mkdir custom/weights# Create a file contains information about data and cfg   
# files locations  
$ touch darknet/custom/darknet.data# in darknet/custom/darknet.data file paste those informations  
classes = 1  
train = custom/train.txt  
valid = custom/test.txt  
names = custom/classes.names  
backup = custom/weights/# Copy and paste yolo config file in "darknet/custom"  
$ cp darknet/cfg/yolov3.cfg darknet/custom# Open yolov3.cfg and change :  
# " filters=(classes + 5)\*3" just the ones before "Yolo"  
# in our case classes=1, so filters=18  
# change classes=... to classes=1# Download pretrained model  
$ wget https://pjreddie.com/media/files/darknet53.conv.74 -O ~/darknet/darknet53.conv.74# Let's train our model !!!!!!!!!!!!!!!!!!!!!  
$ ./darknet detector train custom/darknet.data custom/yolov3.cfg darknet53.conv.74

After finishing the training, to detectect u liscence plate from an image, choose the latest model from darknet/custom/weights , and put its path or name in file object\_detection\_yolo.py, also we will use yolov3.cfg file, just in this file put # before training so we desable training then run :

python object-detection\_yolo.py --image= image.jpg

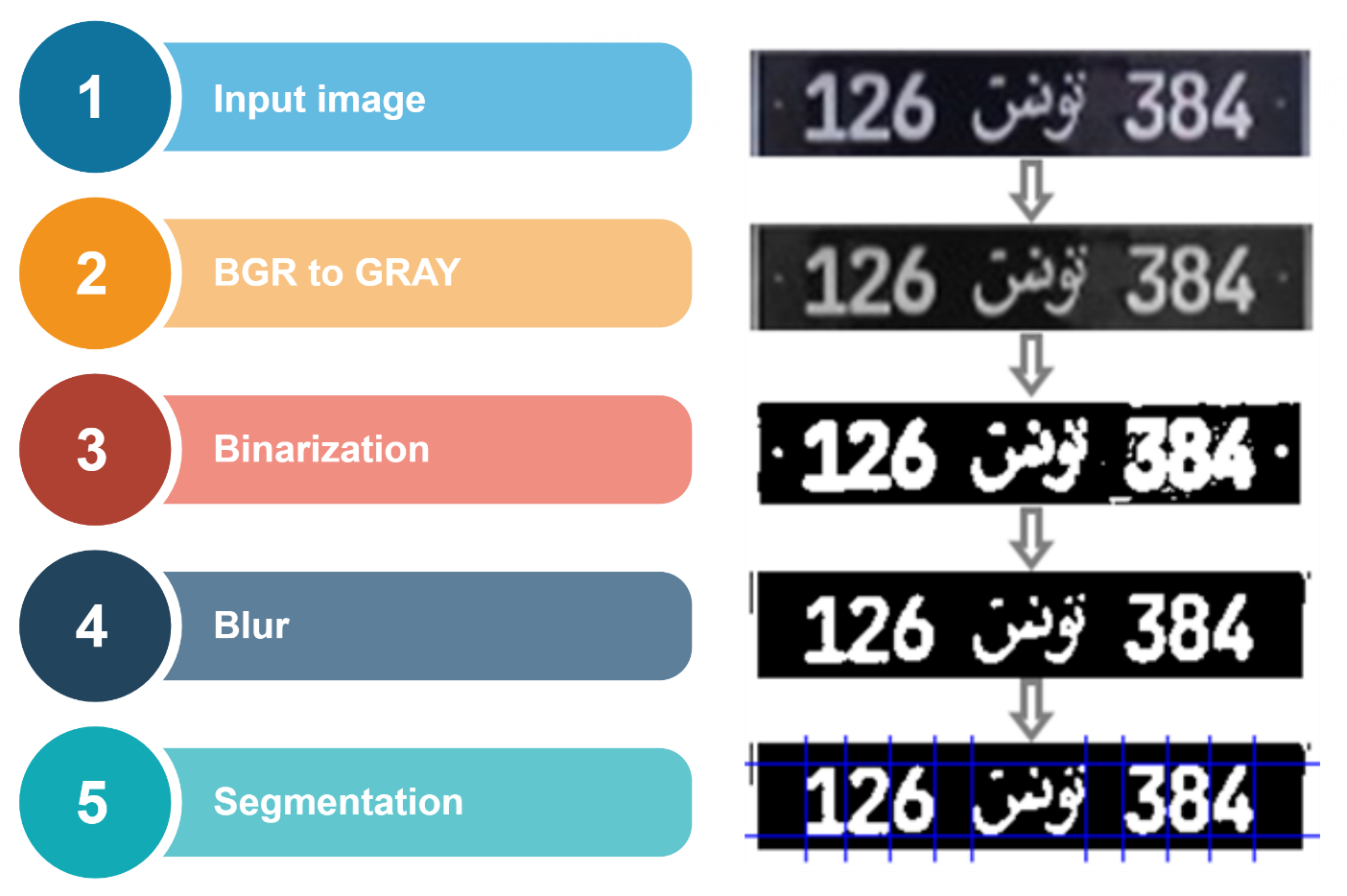
and this what we have as a result :



***Step2 : Licence plate segmentation***

Now we have to segment our plate number. The input is the image of the plate, we will have to be able to extract the unicharacter images. The result of this step, being used as input to the recognition phase, is of great importance. In a system of automatic reading of number plates.

Segmentation is one of the most important processes for the automatic identification of license plates, because any other step is based on it. If the segmentation fails, recognition phase will not be correct.To ensure proper segmentation, preliminary processing will have to be performed.



**The histogram of pixel projection** consists of finding the upper and lower limits, left and right of each character. We perform a horizontal projection to find the top and bottom positions of the characters. The value of a group of histograms is the sum of the white pixels along a particular line in the horizontal direction. When all the values ​​along all the lines in the horizontal direction are calculated, the horizontal projection histogram is obtained. The average value of the histogram is then used as a threshold to determine the upper and lower limits. The central area whose segment of the histogram is greater than the threshold is recorded as the area delimited by the upper and lower limits. Then in the same way we calculate the vertical projection histogram but by changing the rows by the columns of the image to have the two limits of each character (left and right).



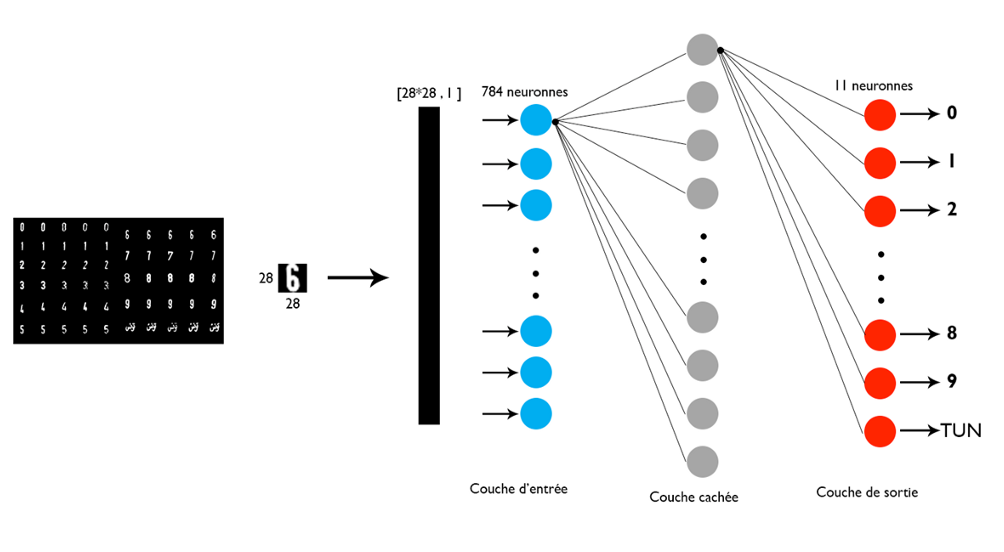
Another approach to extract digits from licence plate is to use open/close morphologye to make some sorte of connected region then use connected component algorith to extract connected regions.

***Step3 : Licence plate recognition***

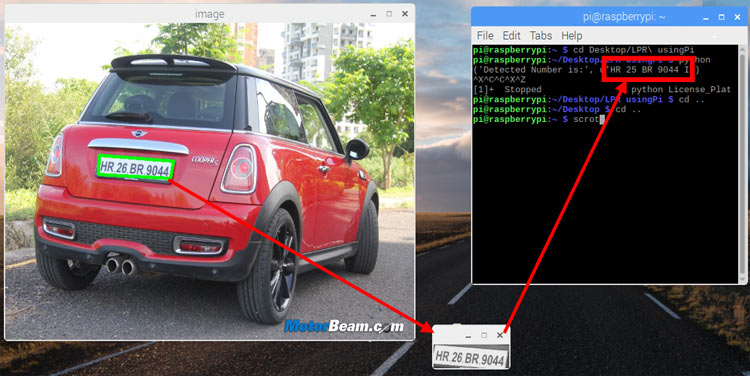
The recognition phase is the last step in the development of the automatic license plate reader system. Thus, it closes all the processes passing by the acquisition of the image, followed by the location of the plate until the segmentation. The recognition must make from the images characters obtained at the end of the segmentation phase. The learning model that will be used for this recognition must be able to read an image and to render the corresponding character.

In order to make the most of the data available for learning, we cut each character indivudually by resizing it in a square after applying the same image processing steps used before segmentation of the license plate. As a result, we obtained a set of of data composed of 11 classes and for each class we have 30–40 images of 28X28 pixel dimesion PNG formats; numbers from 0 to 9 and the word in Arabic (Tunisia).

Then, we made some researches based on scientific articles that compare the multilayer perceptron (MLP) and the classifier K nearest neighbors (KNN). And as a result we have found that: performance is increased if the number of hidden layer neurons is also increased when using the MLP classifier and if the nearest neighbor number is also increased when using the KNN. the ability to adjust the performance of the k-NN classifier is very limited here. But an adjustable number of hidden layers and adjustable MLP connection weights provides a greater opportunity to refine the decision regions. So as a result, we will choose the multilayer perceptron for this phase.



# [**License Plate Recognition using Raspberry Pi and OpenCV**](https://circuitdigest.com/microcontroller-projects/license-plate-recognition-using-raspberry-pi-and-opencv)

License Plate Recognition using Raspberry Pi and OpenCV

Security has always been a major concern for mankind. Today we have video surveillance cameras in schools, hospitals and every other public place to make us feel secured. According to a survey by HIS it is estimated that there were around 245 million security cameras installed and functioning back on 2014, which is like having one security camera for every 30 people on this planet. With the advancement in technology especially in Image processing and Machine Learning, it is possible to make these cameras smarter by training them to process information from the Video feed.

The video feed from these cameras can be used to perform [face recognition](https://circuitdigest.com/microcontroller-projects/raspberry-pi-and-opencv-based-face-recognition-system), pattern analysis, emotion analysis and much more which would really get it close to something like the “God’s Eye” shown in the FF7 movie. In fact, surveillance companies like Hikvision and many others have already started implementing these features in their products. We previously used [MATLAB Image processing to read the number plate](https://circuitdigest.com/tutorial/vehicle-number-plate-detection-using-matlab-and-image-processing), today in this article we will learn **how to recognize and read License Plate Number from Automobiles using Raspberry Pi and OpenCV**. We will use some random vehicle images from Google and write a program to recognize the number plate using [OpenCV Contour Detection](https://circuitdigest.com/tutorial/image-segmentation-using-opencv) and then read the number from the plate using Tesseract OCR. Sounds interesting right!, so Let’s get started.

### **Pre-requisites**

As told earlier we will be using the OpenCV Library to detect and recognize faces. So make sure to [install OpenCV Library on Raspberry Pi](https://circuitdigest.com/microcontroller-projects/how-to-install-python-opencv-on-raspberry-pi) before proceeding with this tutorial. Also Power your Pi with a 2A adapter and connect it to a display monitor for easier debugging.

This tutorial will not explain how exactly [OpenCV](https://circuitdigest.com/tags/opencv) works, if you are interested in learning Image processing then check out this [OpenCV basics](https://circuitdigest.com/tutorial/getting-started-with-opencv-image-processing) and [advanced Image processing tutorials](https://circuitdigest.com/tutorial/image-manipulation-in-python-opencv-part1). You can also learn about contours, Blob Detection etc in this [Image Segmentation tutorial using OpenCV](https://circuitdigest.com/tutorial/image-segmentation-using-opencv). We will be doing something similar to this to detect the license plate of the car from the image.

### **Steps involved in License Plate Recognition using Raspberry Pi**

License Plate Recognition or LPR for short, involves three major steps. The steps are as follows

**1. License Plate Detection:** The first step is to detect the License plate from the car. We will use the contour option in OpenCV to detect for rectangular objects to find the number plate. The accuracy can be improved if we know the exact size, color and approximate location of the number plate. Normally the detection algorithm is trained based on the position of camera and type of number plate used in that particular country. This gets trickier if the image does not even have a car, in this case we will an additional step to detect the car and then the license plate.

**2. Character Segmentation:** Once we have detected the License Plate we have to crop it out and save it as a new image. Again this can be done easily using OpenCV.

**3.** **Character Recognition:** Now, the new image that we obtained in the previous step is sure to have some characters (Numbers/Alphabets) written on it.  So, we can perform OCR (Optical Character Recognition) on it to detect the number. We already explained [Optical Character Recognition (OCR) using Raspberry Pi](https://circuitdigest.com/microcontroller-projects/optical-character-recognition-ocr-using-tesseract-on-raspberry-pi).

### **1. License Plate Detection**

The first step in this **Raspberry Pi License Plate Reader** is to detect the License Plate. Let’s take a sample image of a car and start with detecting the License Plate on that car. We will then use the same image for Character Segmentation and Character Recognition as well. If you want to jump straight into the code without explanation then you can scroll down to the bottom of this page, where the complete code is provided. The test image that I am using for this tutorial is shown below.



**Step 1:** **Resize the image to the required size and then grayscale it**. The code for the same is given below

**img = cv2.resize(img, (620,480) )**

**gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) #convert to grey scale**

Resizing we help us to avoid any problems with bigger resolution images, make sure the number plate still remains in the frame after resizing. Gray scaling is common in all image processing steps. This speeds up other following process sine we no longer have to deal with the color details when processing an image. The image would be transformed something like this when this step is done



**Step 2:**Every image will have useful and useless information, in this case for us only the license plate is the useful information the rest are pretty much useless for our program. This useless information is called noise. Normally **using a bilateral filter (Bluring) will remove the unwanted details from an image**. The code for the same is

**gray = cv2.bilateralFilter(gray, 11, 17, 17)**

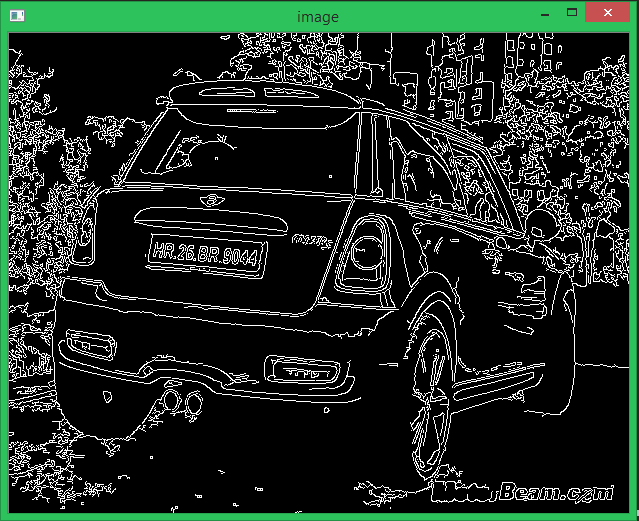
Syntax is destination\_image = cv2.bilateralFilter(source\_image, diameter of pixel, sigmaColor, sigmaSpace). You can increase the sigma color and sigma space from 17 to higher values to blur out more background information, but be careful that the useful part does not get blurred. The output image is shown below, as you can see the background details (tree and building) are blurred in this image. This way we can avoid the program from concentrating on these regions later.



**Step 3:** The next step is interesting where we perform **edge detection**. There are many ways to do it, the most easy and popular way is to use the **canny edge method from OpenCV.** The line to do the same is shown below

**edged = cv2.Canny(gray, 30, 200) #Perform Edge detection**

The syntax will be destination\_image = cv2.Canny(source\_image, thresholdValue 1, thresholdValue 2). The Threshold Vale 1 and Threshold Value 2 are the minimum and maximum threshold values. Only the edges that have an intensity gradient more than the minimum threshold value and less than the maximum threshold value will be displayed. The resulting image is shown below



**Step 4:**Now we can start **looking for contours on our image**, we have already learned about [how to find contours using OpenCV](https://circuitdigest.com/tutorial/image-segmentation-using-opencv) in our previous tutorial so we just proceed like the same.

**nts = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)**

**cnts = imutils.grab\_contours(cnts)**

**cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]**

**screenCnt = None**

**Once the counters have been detected we sort them from big to small** and consider only the first 10 results ignoring the others. In our image the counter could be anything that has a closed surface but of all the obtained results the license plate number will also be there since it is also a closed surface.

To filter the license plate image among the obtained results, we will loop though all the results and check which has a rectangle shape contour with four sides and closed figure. Since a license plate would definitely be a rectangle four sided figure.

**# loop over our contours**

**for c in cnts:**

**# approximate the contour**

**peri = cv2.arcLength(c, True)**

**approx = cv2.approxPolyDP(c, 0.018 \* peri, True)**

**# if our approximated contour has four points, then**

**# we can assume that we have found our screen**

**if len(approx) == 4:**

**screenCnt = approx**

**break**

The value 0.018 is an experimental value; you can play around it to check which works best for you. Or take it to next level by using machine learning to train based on car images and then use the right value there. Once we have found the right counter we save it in a variable called screenCnt and then draw a rectangle box around it to make sure we have detected the license plate correctly.

[](https://circuitdigest.com/fullimage?i=inlineimages/u1/Contour-Detection-of-Image-for-Raspberry-Pi-License-Plate-Reader.jpg)

**Step 5:**Now that we know where the number plate is, the remaining information is pretty much useless for us. So we can **proceed with masking the entire picture except for the place where the number plate is.** The code to do the same is shown below

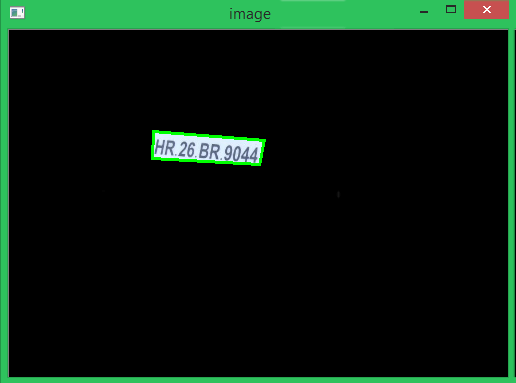
**# Masking the part other than the number plate**

**mask = np.zeros(gray.shape,np.uint8)**

**new\_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)**

**new\_image = cv2.bitwise\_and(img,img,mask=mask)**

The masked new image will appear something like below



### **2. Character Segmentation**

The next step in **Raspberry Pi Number Plate Recognition** is to segment the license plate out of the image by **cropping it and saving it as a new image.**We can then use this image to detect the character in it. The code to crop the roi (Region of interest) image form the main image is shown below

**# Now crop**

**(x, y) = np.where(mask == 255)**

**(topx, topy) = (np.min(x), np.min(y))**

**(bottomx, bottomy) = (np.max(x), np.max(y))**

**Cropped = gray[topx:bottomx+1, topy:bottomy+1]**

The resulting image is shown below. Normally added to cropping the image, we can also gray it and edge it if required. This is done to improve the character recognition in next step. However I found that it works fine even with the original image.



### **3. Character Recognition**

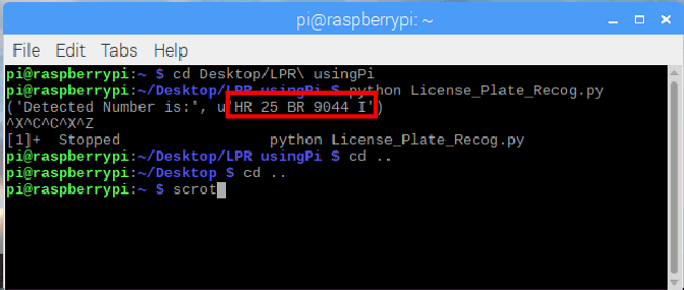
The Final step in this **Raspberry Pi Number Plate Recognition** is to actually **read the number plate information from the segmented image**. We will use the**pytesseract** package to read characters from image, just like we did in previous tutorial. The code for the same is given below

**#Read the number plate**

**text = pytesseract.image\_to\_string(Cropped, config='--psm 11')**

**print("Detected Number is:",text)**

We have already explained [how to configure a Tesseract engine](https://circuitdigest.com/microcontroller-projects/optical-character-recognition-ocr-using-tesseract-on-raspberry-pi), so here again if needed we can configure the Tesseract OCR to obtain better results if required. The detected character is then printed on the console. When compiled the result is shown as below

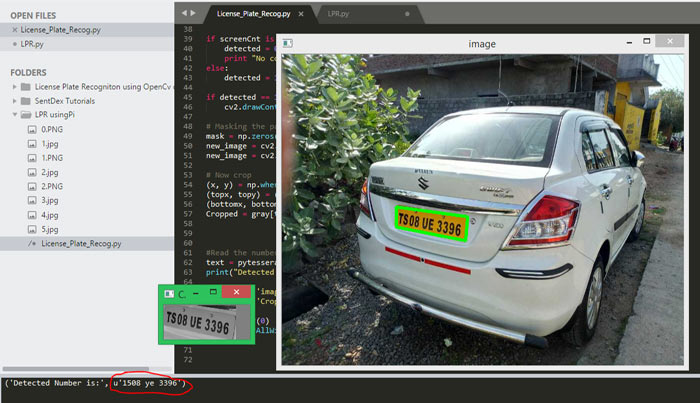


As you can see the original image had the number “HR 25 BR9044” on it and our program has detected it printed the same value on screen.

### **Fail Cases in Number Plate Recognition**

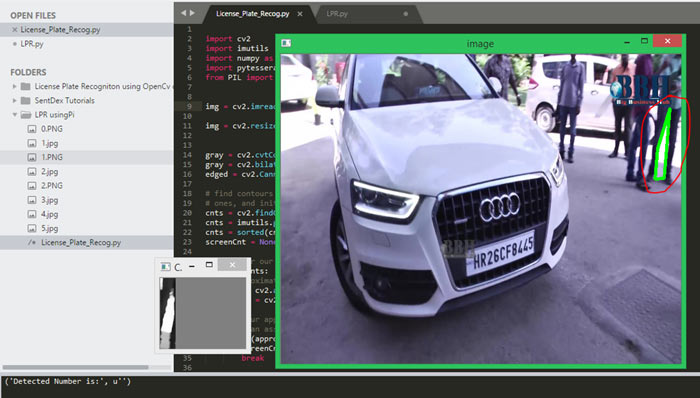
The complete project file this **Raspberry Pi License Plate Recognition** can be [downloaded from here](https://circuitdigest.com/sites/default/files/License-Plate-Recognition-using-Raspberry-Pi.zip), it contains the program and the test images that we used to check our program. Without being said, it is to be remembered that the results from this method will not be accurate**. The accuracy depends on the clarity of image, orientation, light exposure etc**. To get better results you can try implementing Machine learning algorithms along with this.

To get an idea, let’s look at another example where the car is not facing the camera directly.



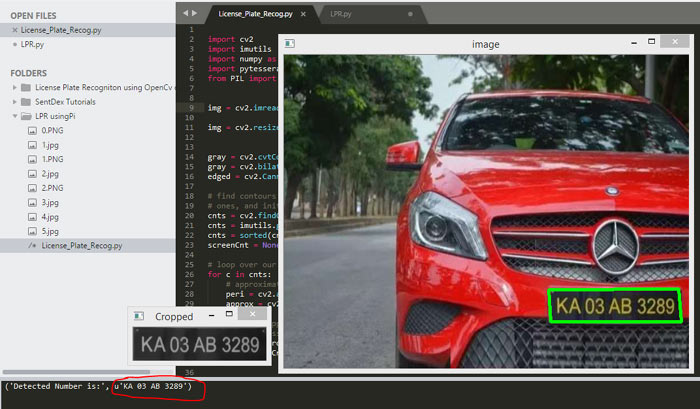
As you can see, our program was able to detect the license plate correctly and crop it. But the Tesseract library has failed to recognize the characters properly. Instead of the actual “TS 08 UE 3396” the OCR has recognized it to be “1508 ye 3396”. **Problems like this can be corrected by either using better orientation images or by configuring the Tesseract engine**.

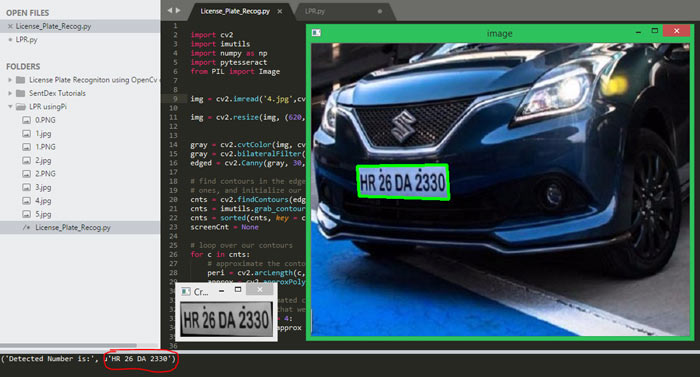
**Another worst case scenario is where the contour fails to detect the license plate correctly.** The below image has too much background information and bad lighting that the program has even failed to identify the license plate from the number. In this case we have to again relay on Machine learning or improve the quality of the picture.



### **Other Successful Examples**

Most of the times of the image quality and orientation is correct, the program was able to identify the license plate and read the number from it. The below snap shots show few of the successful results obtained. Again all the test images and the code used here will be available in the [ZIP file provided here](https://circuitdigest.com/sites/default/files/License-Plate-Recognition-using-Raspberry-Pi.zip).

Car Image Source: [Daily Hunt News](https://www.dailyhunt.in/)

Car Image Source: [ICN](https://www.indiacarnews.com/)

Hope you understood **Automatic Number Plate Recognition using Raspberry Pi** and enjoyed building something cool on your own. What else do you think can be done with **OpenCV and Tesseract**?, let me know your thoughts in the comment section. If you have any questions regarding this article please feel free to leave them in the comment section below or use the forums for other technical queries.

Code

import cv2  
import imutils  
import numpy as np  
import pytesseract  
from PIL import Image

img = cv2.imread('4.jpg',cv2.IMREAD\_COLOR)

img = cv2.resize(img, (620,480) )

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) #convert to grey scale  
gray = cv2.bilateralFilter(gray, 11, 17, 17) #Blur to reduce noise  
edged = cv2.Canny(gray, 30, 200) #Perform Edge detection

# find contours in the edged image, keep only the largest  
# ones, and initialize our screen contour  
cnts = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
cnts = imutils.grab\_contours(cnts)  
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]  
screenCnt = None

# loop over our contours  
for c in cnts:  
 # approximate the contour  
 peri = cv2.arcLength(c, True)  
 approx = cv2.approxPolyDP(c, 0.018 \* peri, True)  
   
 # if our approximated contour has four points, then  
 # we can assume that we have found our screen  
 if len(approx) == 4:  
  screenCnt = approx  
  break

if screenCnt is None:  
 detected = 0  
 print "No contour detected"  
else:  
 detected = 1

if detected == 1:  
 cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)

# Masking the part other than the number plate  
mask = np.zeros(gray.shape,np.uint8)  
new\_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)  
new\_image = cv2.bitwise\_and(img,img,mask=mask)

# Now crop  
(x, y) = np.where(mask == 255)  
(topx, topy) = (np.min(x), np.min(y))  
(bottomx, bottomy) = (np.max(x), np.max(y))  
Cropped = gray[topx:bottomx+1, topy:bottomy+1]

#Read the number plate  
text = pytesseract.image\_to\_string(Cropped, config='--psm 11')  
print("Detected Number is:",text)

cv2.imshow('image',img)  
cv2.imshow('Cropped',Cropped)

cv2.waitKey(0)  
cv2.destroyAllWindows()

SMART HELMET

ABSTRACT The impact when a motorcyclist involves in a high-speed accident without wearing a helmet is very dangerous and can cause fatality. Wearing a helmet can reduce shock from the impact and may save a life. There are many countries enforcing a regulation that requires the motorcycle's rider to wear a helmet when riding on their motorcycle, Malaysia is an example. A smart helmet is a special idea which makes motorcycle driving safer than before. This is implemented using GSM and GPS technology. The working of this smart helmet is very simple, vibration sensors are placed in different places of helmet where the probability of hitting is more which are connected to microcontroller board. So when the rider crashes and the helmet hit the ground, these sensors sense and gives to the microcontroller board, then controller extract GPS data using the GPS module that is interfaced to it. When the data exceeds minimum stress limit then GSM module automatically sends message to ambulance or family members. It also has an alcohol detector sensor which detects whether the person is drunk and switches off the engine if the sensor output is high.

INTRODUCTION Now a days most of the countries are enforcing their citizens to wear helmet while riding bike and not to ride a bike when the person is under the influence of alcohol, but still rules are being violated. 1.1. Motivation The motivation of this project comes from the real-world challenges that we face daily on the roads. Road accidents are on the rise day by day and in countries like India where bikes are more prevalent many people die to carelessness carried in wearing helmets. In present day scenario we encounter numerous cases of death due to two-wheeler road accidents. Despite of the fact that helmets are available everywhere, people are not wearing them. In the event of road accidents, the message is sent to the emergency contact through GSM. 1.2. Problem Definition As the bikers in our country are increasing, the road mishaps are also increasing day by day, due to which many deaths occur, most of them are caused due to most common negligence of not wearing helmets, also many deaths occur due to lack of prompt medical attention needed by the injured person. The project aims at the security and safety of the bikers against road accidents. 1.3. Objective of the Project The objectives of this project are to design the circuit that can improve safety of motorcyclists, to develop a smart safety helmet for complete rider. 1.4. Limitations of the Project • Person rides the bike even in the areas where mobile network lacks, so GSM network is required for sending SMS. It is expensive. • When the helmet is dropped down accidentally, the system treats it as accident.

2.1. Introduction There has been a sharp rise in the total number of deaths that occur due to road accidents in the past few years. Reckless driving, ignorance of traffic rules and absence of protective shield have been some of the most important reasons for these deaths. Most of the accidents occur due to drinking and then driving bikes and death caused due to neglecting helmets. Government adopted few measures like helmet and alcohol checking by traffic police but they are hardly useful. 2.2. Existing System The existing project basically has a wireless telecommunication and is connected to a smart phone. This prototype uses sensors to detect a crash or accidents and the communication hardware is used to automatically dial a predefined emergency contact. The other existing system is to control the speed in which the biker is going in. The helmet is fixed with all the components and sensors that read the speed of the bike and accordingly instruct the rider to reduce or increase the speed based on the obstacles ahead the bike. This has following disadvantages: • Rider does not wear helmet in regions where traffic checking is not done. • Testing alcohol content present in blood in each individual rider in big countries like India is impossible. • Difficulty of implementation of traffic rules by traffic police. 2.3. Proposed System The helmet checks if the rider is drunk and driving. If the rider is drunk then the ignition of the bike is avoided and the hence not letting the rider to ride the bike. In this system we use an Arduino microcontroller interfaced with alcohol sensor and it is used to monitor user’s breath and constantly sends signals to microcontroller. The microcontroller on encountering alcohol signal from sensor and send the data to motor using RF transmitter and we connect a RF receiver to the motor driver which stops dc motor to demonstrate as engine locking. The system need push button to start the engine. If the alcohol is detected the system locks the engine. The system also sends a message stating “Accident occurred” including the latitude and longitude location of the incident using GSM and GPS. It uses a vibration sensor to detect an accident. It also has a temperature sensor which notifies when the helmet gets heated up to avoid the circuit damage.

2.4. Software Requirement Specification 2.4.1. Purpose As all the road accident fatalities involve motorcyclists, we see many accidents in the day to day life due to alcohol consumption so we are trying to implement a project that can avoid the accidents. 2.4.2. Scope The scope of the project is the safe two-wheeler journey is possible which should decrease the head injuries during accidents and also reduce the accident rate due to driving bike after consuming alcohol. 2.4.3. Overall Description Road accidents are on the rise day by day and in countries like India where bikes are more prevalent, many people die due to carelessness carried in wearing helmets. Smart helmet has a feature of detecting accidents with the help of knock sensor and informs to specific people. It also includes another feature, if the driver is drunk and tries to drive, the system detects the alcohol presence in his breath and locks the engine, so that the vehicle fails to start. If the driver is not drunk while he starts the vehicle and engine is started but he drinks while driving the sensor still detects the alcohol in his breath and stops the engine so that the vehicle could not accelerate any further.

III. MODULES 3.1. Module Description Microcontroller (ARDUINO UNO) The Arduino Uno is a microcontroller board based on the ATmega328. (It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller, simply connect it to computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

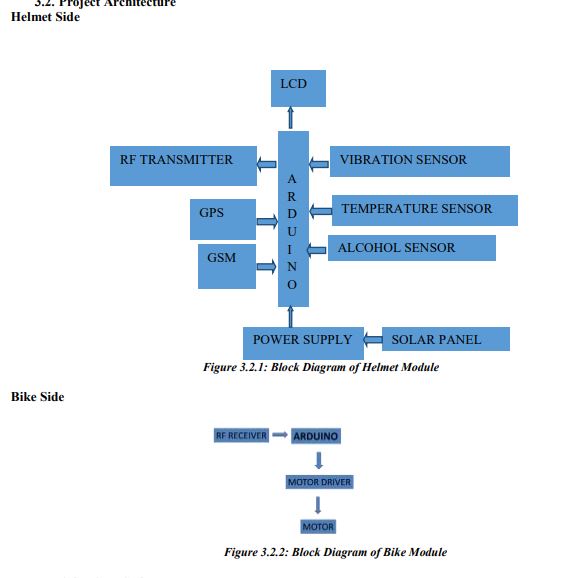
GPS Technology The Global Positioning System (GPS) is a satellite-based navigation system that sends and receives radio signals. A GPS receiver acquires these signals and provides the user with information. Using GPS technology, one can determine location, velocity and time, 24 hours a day, in any weather conditions anywhere in the world for free.GPS was formally known as the NAVSTAR (Navigation Satellite Timing and Ranging). Global Positioning System was originally developed for military. Because of its popular navigation capabilities and because GPS technology can be accessed using small, inexpensive equipment, the government made the system available for civilian use. The USA owns GPS technology and the Department of Defense maintains it. How GPS works • GPS provides specially coded satellite signals that can be processed in a GPS receiver, enabling the receiver to compute position, velocity and time. • Four GPS satellite signals are used to compute positions in three dimensions and the time offset in the receiver clock. • GPS is funded by and controlled by the U. S. Department of Defense (DOD). While there are many thousands of civil users of GPS worldwide, the system was designed for and is operated by the U. S. military. The architectural components of GPS are typically referred to as the control segment (ground stations), the space segment (satellites) and the user segment (receivers).

Determining Position Upon taking in all available satellite signals, the receiver compares the time that the satellite sent the signal to the time it was received for each of the available signals. Trilateralization (similar to triangulation) then calculates the position by comparing the difference among the signals. Figure 3.1.2: GPS Module GSM TechnologyDefinition of GSM GSM (Global System for Mobile communications) is an open, digital cellular technology used for transmitting mobile voice and data services. GSM (Global System for Mobile communication) is a digital mobile telephone system that is widely used in Europe and other parts of the world. GSM uses a variation of Time Division Multiple Access (TDMA) and is the most widely used of the three digital wireless telephone technologies (TDMA, GSM, and CDMA). GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot. It operates at either the 900 MHz or 1,800 MHz frequency band. It supports voice calls and data transfer speeds of up to 9.6 Kbit/s, together with the transmission of SMS (Short Message Service).

Computers use AT commands to control modems. Both GSM modems and dial-up modems support a common set of standard AT commands. In addition to the standard AT commands, GSM modems support an extended set of AT commands. These extended AT commands are defined in the GSM standards. With the extended AT commands, the following operations can be performed: • Reading, writing and deleting SMS messages. • Sending SMS messages. • Monitoring the signal strength. • Reading, writing and searching phone book entries

Motor Electric motors involve rotating coils of wire which are driven by the magnetic force exerted by a magnetic field on an electric current. They transform electrical energy into mechanical energy. There are several types of electric motors available today. The following outline gives an overview of several popular ones. There are two main classes of motors: AC and DC. AC motors require an alternating current or voltage source (like the power coming out of the wall outlets in your house) to make them work. DC motors require a direct current or voltage source (like the voltage coming out of batteries) to make them work. Universal motors can work on either type of power. Not only is the construction of the motors different, but the means used to control the speed and torque created by each of these motors also varies, although the principles of power conversion are common to both. RF Technology Radio frequency (RF) is a frequency or rate of oscillation within the range of about 3 Hz to 300 GHz. This range corresponds to frequency of alternating current electrical signals used to produce and detect radio waves. Since mostof this range is beyond the vibration rate that most mechanical systems can respond to, RFusually refers to oscillations in electrical circuits or electromagnetic radiation. RF Advantages • No line of sight is needed. • Not blocked by common materials: It can penetrate most solids and pass through walls. • Longer range. • It is not sensitive to the light. • It is not much sensitive to the environmental changes and weather conditions. The data is received by the RF receiver from the antenna pin and this data is available on the data pins. Two Data pins are provided in the receiver module. Thus, this data can be used for further applications. Alcohol Sensor (MQ2) This module is useful for gas leakage detection (in home and industry). It is suitablefor detecting H2, LPG, CH4, CO, Alcohol, Smoke orPropane. Dueto its high sensitivity and fast response time, measurement can be taken as soon as possible. The sensitivity of the sensor can be adjusted by potentiometer. Features • Wide detecting scope • Stable and long life • Fast response and High sensitivity

Vibration Sensor A piezoelectric sensor is a device that uses the piezoelectriceffect, to measure changes in pressure, acceleration, temperature, strain, or force by converting them to an electrical charge. The prefix piezo- is Greek for 'press' or 'squeeze'. Piezoelectric sensors are versatile tools for the measurement of various processes. They are used for quality assurance, process control, and for research and development in many industries. Pierre Curie discovered the piezoelectric effect in 1880, but only in the 1950s did manufacturers begin to use the piezoelectric effect in industrial sensing applications. Since then, this measuring principle has been increasingly used, and has become a mature technology with excellent inherent reliability. They have been successfully used in various applications, such as in medical, aerospace, nuclear instrumentation, and as a tilt sensor in consumer electronics or a pressure sensor in the touch pads of mobile phones. In the automotive industry, piezoelectric elements are used to monitor combustion when developing internal combustion engines. The sensors are either directly mounted into additional holes into the cylinder head or the spark/glow plug is equipped with a built-in miniature piezoelectric sensor.



CONCLUSION The developed project efficiently ensures: • Rider is wearing helmet throughout the ride. • Rider should not be under the influence of alcohol. • Accident detection. By implementing this project, a safe two-wheeler journey is possible which would decrease the head injuries during accidents and also reduce the accident rate due to driving bike after consuming alcohol. The helmet may not be a 100% foolproof but is definitely the first line of defense for the rider in case of an accident to prevent fatal injuries.