```
OpenCV packages for Python
import cv2
# Python plotting package
import matplotlib.pyplot as plt
# Fork of argparse to add features and simplify its code
import argparse
# functions to make basic image processing functions
import imutils
# this for add math function
import math
import time
# package for array computing with Python
import pandas as pd
from numpy import asarray as pn
from sklearn.linear_model import LinearRegression
from imutils.perspective import four_point_transform
from imutils import paths
from sklearn.metrics import mean_squared_error

# capture frames from a camera
cap = cv2.VideoCapture(0)

cap.set(3, 640)
cap.set(4, 480)
count = 0
height = []

flag = 0

# reads frames from a camera
ret, frame = cap.read()
cv2.imwrite("testimage.jpg", frame)
im = cv2.imread("testimage.jpg")

r = cv2.selectROI(img=im, windowName="test")

t = time.localtime()
current_time = time.strftime("%H:%M:%S", t)

# loop runs if capturing has been initialized
while (1):

    ret, frame = cap.read()

    if frame is None:
        break

    # Crop image
```

```
    frame = frame[int(r[1]):int(r[1] + r[3]), int(r[0]):int(r[0] + r[2])]
    # Convert the img to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # Apply edge detection method on the image
    edges = cv2.Canny(gray, 100, 120)
    # Run Hough on edge detected image
    lines = cv2.HoughLinesP(edges, 1, math.pi/180, 20, None, 20, 480)
    dot1 = (lines[0][0][0], lines[0][0][1])
    dot2 = (lines[0][0][2], lines[0][0][3])
    slope = ((lines[0][0][3] - lines[0][0][1])/(lines[0][0][2] - lines[0][0][0]))
    #cv2.line draws a line in img from dot1 to dot2
    # (255,0,0) denotes the colour of the line to be drawn
    if 0 <= slope <= 0.15:
        cv2.line(frame, dot1, dot2, (255, 0, 0), 3)
        length = 150 - lines[0][0][3]
        print(length)
        height.append(length)

        cv2.imshow("Detected Line", frame)
        # finds edges in the input video and
        # marks them in the output map edges
        edged_frame = cv2.Canny(frame, 1, 100)
        cv2.imshow('Edged Frame', edged_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

x = []
y = []
file = open("Saved.txt","a")
for i in range(len(height)):
            x.append(i)
            y.append(height[i])
            file.write(str(x[i-1])+","+str(y[i-1])+"\n")

X=np(x)
Y=np(y)

X = X.reshape(len(X),1)
Y = Y.reshape(len(Y),1)

model = LinearRegression()
model.fit(X,Y)

model = LinearRegression().fit(X,Y)
r_sq = model.score(X,Y)

y_pred = model.predict(X)
```

```
y_pred = model.intercept_+ model.coef_*X

print('Predicted Response:', y_pred, sep='\n')
print('Start :', current_time)
print('Coefficient of Determination:', r_sq)
print('Intercept:', model.intercept_)

accuracy = mean_squared_error(y, y_pred)
print('Accuracy :', accuracy)

t = time.localtime()
current_time2 = time.strftime("%H:%M:%S", t)
print('Stop  :', current_time2)

plt.plot(X,Y,'.',color='black')


cap.release()
cv2.destroyAllWindows()

plt.plot(X,y_pred)

plt.title('Test Data')
plt.xlabel('Time')
plt.ylabel('Height')
plt.show()
```

As we all know that Flood is one of the major well known Natural Disasters. When water level suddenly rises in dams, river beds etc. Alot of Destruction happens at surrounding places. It causes a huge amount of loss to our environment and living beings as well. So in these case, it is very important to get emergency alerts of the water level situation in different conditions in the river bed.

The purpose of this project is to sense the water level in river beds and check if they are in normal condition. If they reach beyond the limit, then it alerts people through LED signals and buzzer sound. Also it alerts people through Sms and Emails alerts when the water level reaches beyond the limit.