SMART FIRE MANAGEMENT SYSTEM

Introduction

Overview:-

A fire can happen at any time at any place irrespective of its occupancy status. You can expect a fire at any structure, may be at your home or at your workplace or in a hospital or in public places like theatres, malls, etc… Fire in any occupancy has the potential to cause harm to its occupants and severe damage to property.

On an average, in India, every year, about 25,000 persons die due to fires and related causes. Female accounts for about 66% of those killed in fire accidents. It is estimated that about 42 females and 21 males die every day in India due to fire. According to the statistics released by the National Crime Records Bureau, fire accounts for about 5.9% (23,281) of the total deaths reported due to natural and un-natural causes during the year 2012. Probably many of these deaths could have been prevented, had we taken enough fire protection measures.

Purpose:-

The ultimate goal of the smart fire management is to integrate this wellspring of data into firefighting. According to a report co-authored by Grant, it will “revolutionize firefighting by collecting data globally, processing the information centrally and distributing the results locally.

This system is mainly designed on the purpose of minimizing the fire accidents and the losses people experiencing because of these accidents.

Literature Survey

Exsisting Problem:-

For mitigating a fire in any occupancy, whether it is a business house or in a factory or in a residential building, require a deep understanding about the problem.

A small fire in a residential building may be spread very fast and within a few minutes it can reach a stage beyond the control of its occupants and ultimately seek the help of fire brigade to carry out a major firefighting operation. During the last one decade there was a vibrant growth in the constructions activities in India, especially in High Rise buildings. Thousands of High Rise buildings have already constructed in metros and major cities in India, and thousands are under construction. Because of its peculiar nature, fire in residential buildings in particular, high rise buildings become more complex and the salvaging operations become more difficult and sometimes even resulting in many deaths and huge property losses.

In an era of highly competitive business environment any interruption due to fire can be catastrophic. A major fire can bring a business to halt. Restoring the damage done by fire is only part of the cost of fire. A fire may have serious consequences for the production capacity of a business and in the extreme, the time taken to restore production may be such that the business is forced to close down altogether.

Proposed Solution:-

For getting out of this problem,visual recognition plays a vital job.Integration of visual recognition with the cloud services and the devices like raspberrypi we can solve this problem within fraction of seconds.

Internet of Things (IoT) helps this situation to be solved.Whenever the camera at the location finds fire,it sends the message to the referred number and we can check the temperature,flame sensor values and gas sensor values aswell.

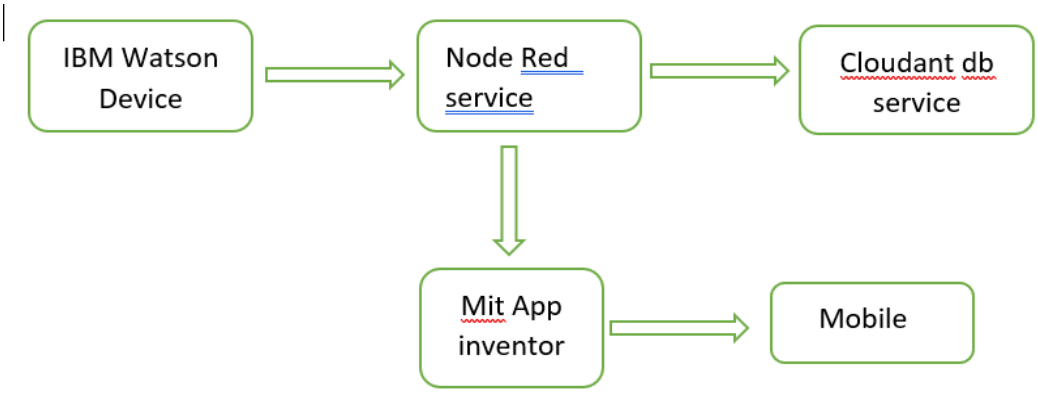
As soon as the message is notified on the screen we can turn on the sprinklers near the location using the app we designed for this purpose.

So this stabilizes the situation until the fire brigade arrives at the location.This solution

is much economical than any other safety measure we are presently taking to solve this issue.

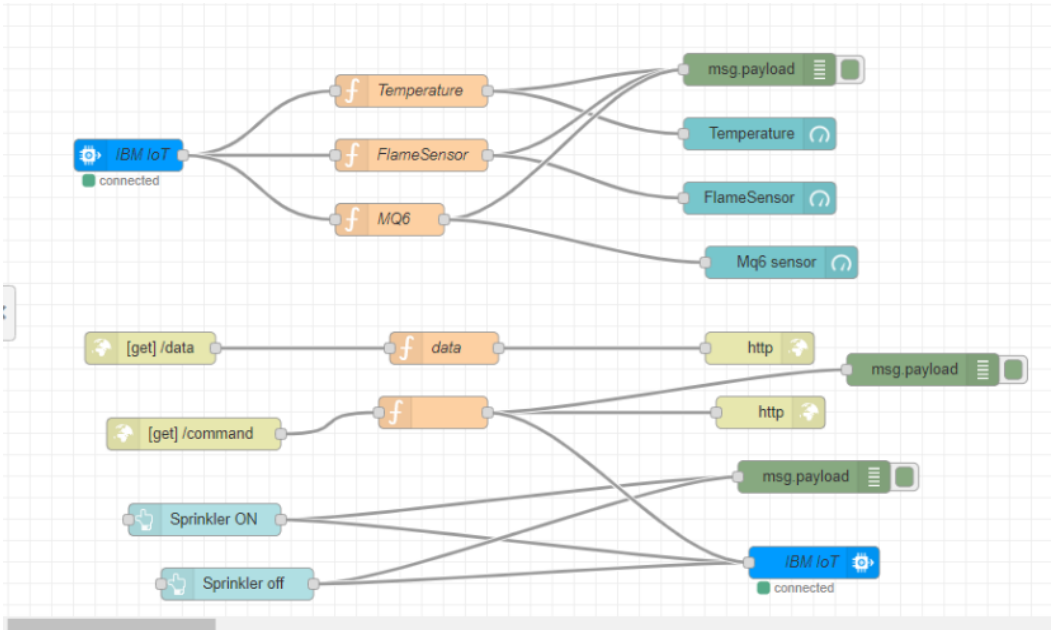
Theoretical analysis:-

Block Diagram:-

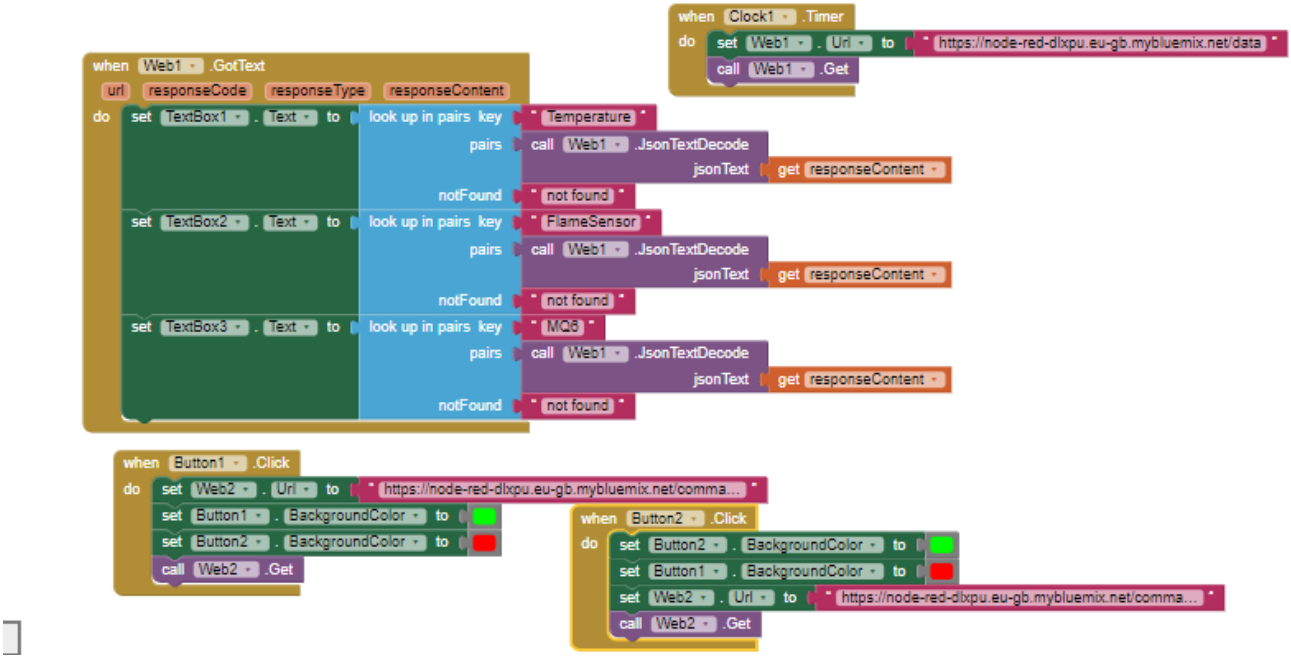


Software design:-

Node-Red Flows:-



Mit App Design:-



Experimental Investigations:-

IBM Device:-

This is used as the IoT device we require to complete the project.
In this project Raspberrypi is used virtually through this.

Node-Red services:-

Node-Red is used as the connection between the cloud and the IoT devices ,we can connect the required flows for the project and make them work accordingly.We can send http requests,create web UI,

Cloudantdb Service:-

Using this service the data is linked to the object storage where the data to be stored.

The metadata is stored in the cloudant database and there will be a link, which directs it to the object Storage

Object Storage:-

Object Storage is used to store the data from the cloud that is sent from the device.This data is stored in the form of buckets.This buckets are accessed using api key.

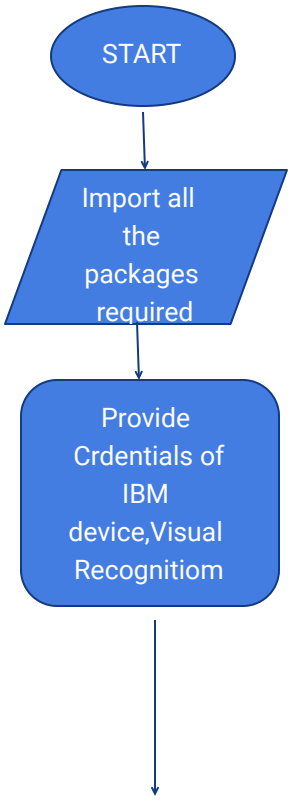
Mit App Inventor:-

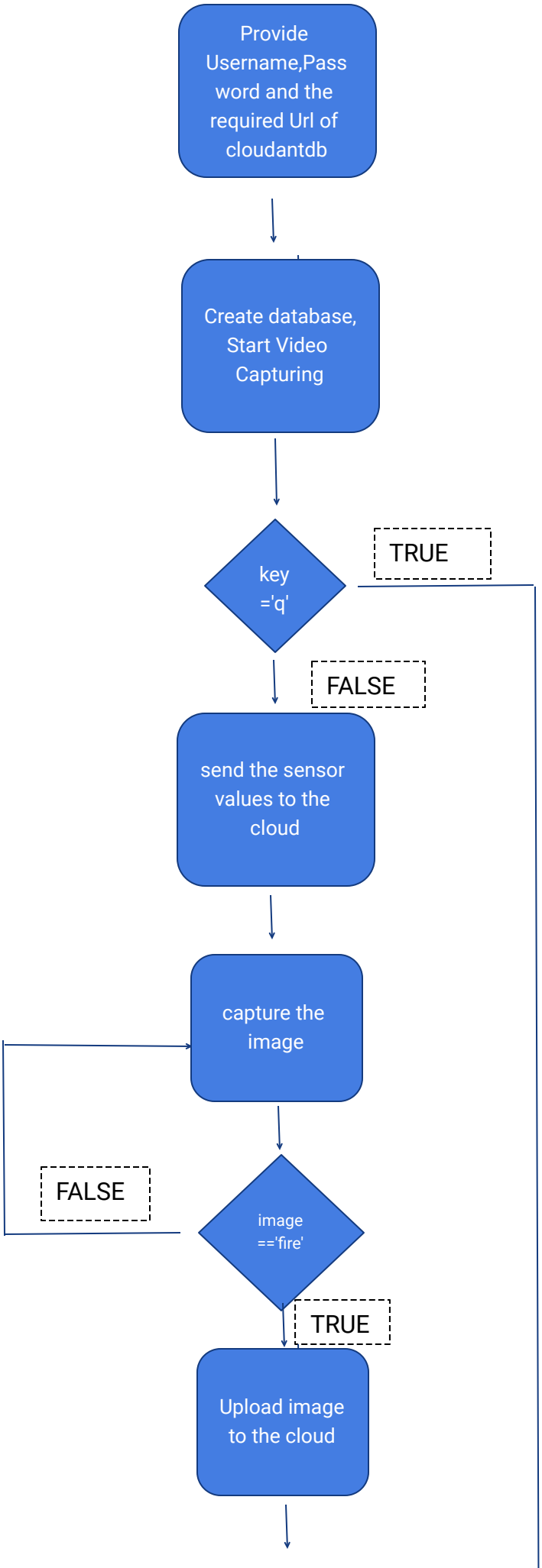
This Mit App is used for creating app required for the project,the backend code used to develop the app is easy.W e can connect this app to our mobile phone and use the app in it comfortably.

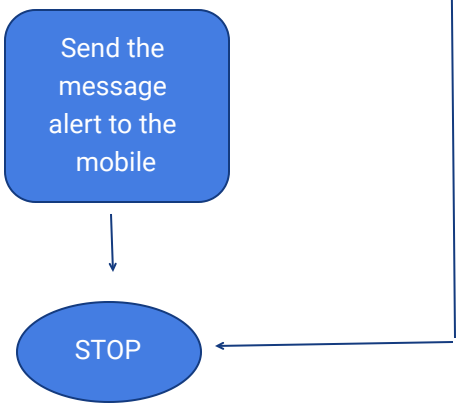
Visual Recognition Service:-

This service in IBM cloud services is used for recognizing the objects or living things infront of the camera.We can train the model by adding the required images and make use of this model.The data can be perfectly categorized with help of this service.

FLOW CHART:-




Provide Username,Password and the required Url of cloudantdb
Create database, Start Video Capturing
key ='q'
TRUE
FALSE
send the sensor values to the cloud
capture the image
FALSE
image =='fire'
TRUE
Upload image to the cloud



Result:-

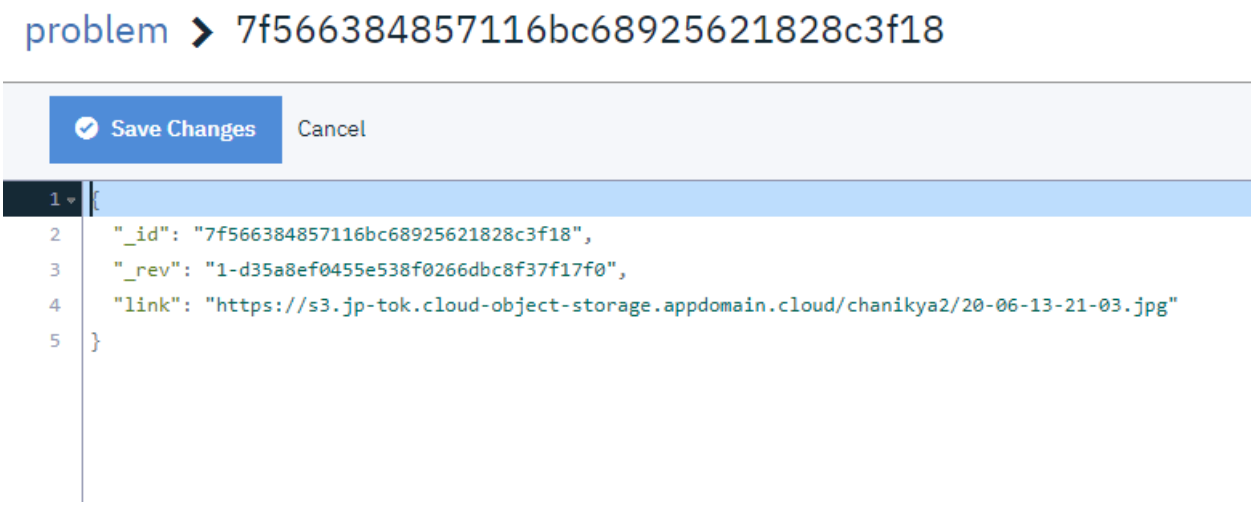
Output :-

```
'problem' successfully created.
2020-06-16 10:37:32,164   ibmiotf.device.Client      INFO    Connected successfu
lly: d:0l3mtb:raspberrypi:123456
mobile
Published Temperature = 51 C FlameSensor = 989 nm Mq6=2208 ppm to IBM Watson
fire
Published Temperature = 55 C FlameSensor = 845 nm Mq6=3273 ppm to IBM Watson
Starting file transfer for 20-06-16-10-37.jpg to bucket: chanikya2

Transfer for 20-06-16-10-37.jpg Complete!

Document successfully created.
200
people
Published Temperature = 66 C FlameSensor = 786 nm Mq6=6772 ppm to IBM Watson
```

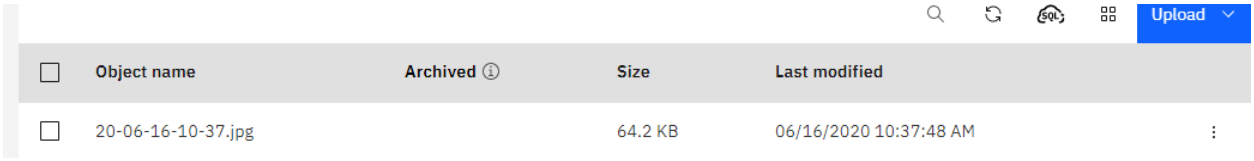
IBM Device Output:-

Event	Value	Format	Last Received
Weather	{"Temperature":66,"FlameSensor":738,"MQ6":5...	json	a few seconds ago
Weather	{"Temperature":53,"FlameSensor":819,"MQ6":3...	json	a few seconds ago
Weather	{"Temperature":70,"FlameSensor":838,"MQ6":6...	json	a few seconds ago
Weather	{"Temperature":38,"FlameSensor":745,"MQ6":8...	json	a few seconds ago
Weather	{"Temperature":37,"FlameSensor":846,"MQ6":8...	json	a few seconds ago

Cloudant db output:-

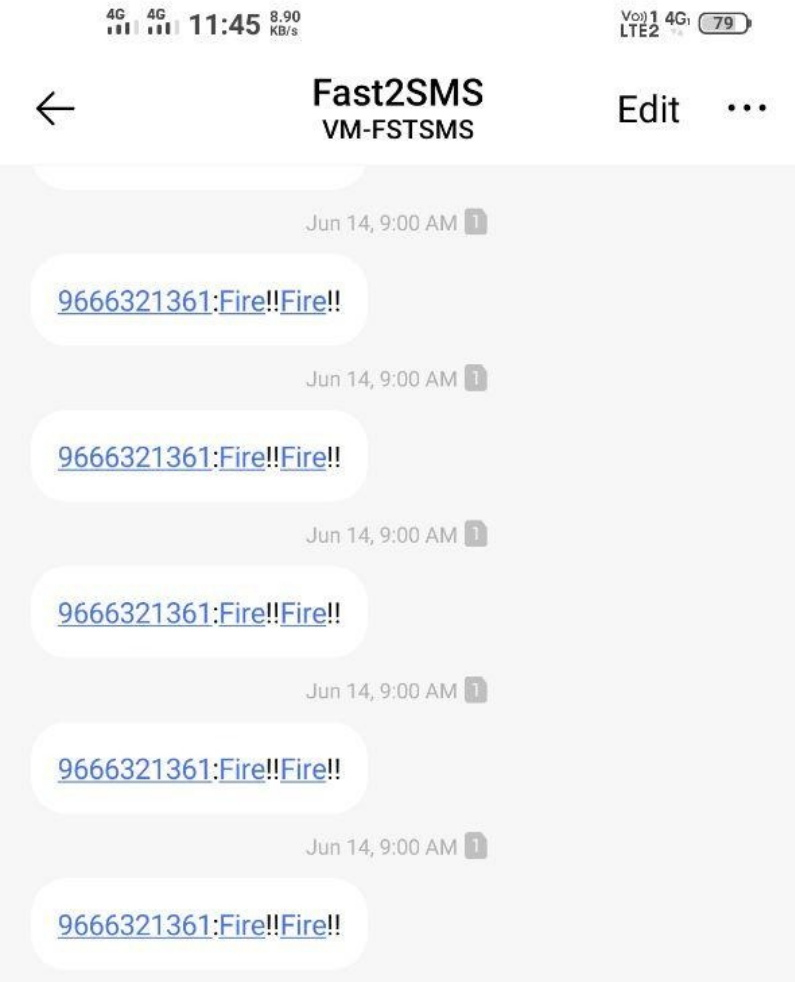


Object storage output:-



Sprinkler ON/Off Output:-

Message alert:-



Advantages:-

- The problem can be solved as fast as possible
- Economical way to solve this issue
- Hardware is also simple to construct as the most of the process is on the internet.

Disadvantages:-

- Security issues might trouble this system.
- It must be kept securely without exposing the credentials .
- Hackers are the main problem with this devices.

Applications:-

- Industries
- Hospitals
- Crackers manufacturing factories
- labs

Conclusion:-

In this internet era, taking the utmost advantage of the IoT devices can help us solve many problems.So I suggest using this fire safety management system in all the required areas and help the living from getting rid this problems.

Future Scope:-

In future there will definitely more advancements in this field of IoT and Visual-Recognition.This project will be more upgraded further moth more security patches.

APPENDIX:-

Source Code :-

```
import cv2
import numpy as np
import datetime

#ObjectStorage
import ibm_boto3
from ibm_botocore.client import Config, ClientError

#CloudantDB
from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
import requests

import json
from watson_developer_cloud import VisualRecognitionV3

import time
import sys
import ibmiotf.application
import ibmiotf.device
```

```
import random
#Provide your IBM Watson Device Credentials
organization = "0l3mtb"
deviceType = "raspberrypi"
deviceId = "123456"
authMethod = "token"
authToken = "12345678"

visual_recognition = VisualRecognitionV3(
   '2018-03-19',
   iam_apikey='41ufY2T9ZhQUxZg7FuNCCsGLYstvYU_lZCb9ESc_FzSZ')


# Constants for IBM COS values
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list avaiable at
https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "Qo2Za_6450oKGeIuPLU-TdJjgPCXjlo9D1RaACbgg2qi" # eg
"W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN =
"crn:v1:bluemix:public:cloud-object-storage:global:a/95c966e28a7346a4a248f295fbaecccb:0a7a79b6-f03b-4dd9-b6fc-2306f601053
7::" # eg
"crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d639
03::"
# Create resource
cos = ibm_boto3.resource("s3",
   ibm_api_key_id=COS_API_KEY_ID,
   ibm_service_instance_id=COS_RESOURCE_CRN,
   ibm_auth_endpoint=COS_AUTH_ENDPOINT,
   config=Config(signature_version="oauth"),
   endpoint_url=COS_ENDPOINT
)

#Provide CloudantDB credentials such as username,password and url

client = Cloudant("4eb3d1ab-4cde-44e2-ae43-59ca49f07d9e-bluemix",
"41199fcc219b18f411e5f5b5fb2d0a0ef3f89e6f0dc0316efa8c50f29e697574",
url="https://4eb3d1ab-4cde-44e2-ae43-59ca49f07d9e-bluemix:41199fcc219b18f411e5f5b5fb2d0a0ef3f89e6f0dc0316efa8c50f29e
697574@4eb3d1ab-4cde-44e2-ae43-59ca49f07d9e-bluemix.cloudantnosqldb.appdomain.cloud")
client.connect()

#Provide your database name

database_name = "problem"

my_database = client.create_database(database_name)

if my_database.exists():
  print(f"'{database_name}' successfully created.")

def myCommandCallback(cmd):
      print("Command received: %s" % cmd.data)#Commands
      print(type(cmd.data))
      i=cmd.data['command']
      if i=='Sprinkler ON':
            print("Sprinkler is ON")
      elif i=='Sprinkler OFF':
```

```
            print("Sprinkler is OFF")

try:
                                                                                deviceOptions = {"org":
organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
                                                                                deviceCli =
ibmiotf.device.Client(deviceOptions)
                                                                                #.....................................

except Exception as e:
                                                                                print("Caught exception
connecting device: %s" % str(e))
                                                                                sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect()


def vis(a):
  with open(a, 'rb') as images_file:
   a = visual_recognition.classify(
      images_file,
      threshold='0.6',

classifier_ids='chanikya_2040677521').get_result()
  #print(json.dumps(a, indent=2))
  b=a["images"][0]["classifiers"][0]["classes"][0]["class"]
  return b


def multi_part_upload(bucket_name, item_name, file_path):
   try:
      print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
      # set 5 MB chunks
      part_size = 1024 * 1024 * 5

      # set threadhold to 15 MB
      file_threshold = 1024 * 1024 * 15

      # set the transfer threshold and chunk size
      transfer_config = ibm_boto3.s3.transfer.TransferConfig(
         multipart_threshold=file_threshold,
         multipart_chunksize=part_size
      )

      # the upload_fileobj method will automatically execute a multi-part upload
      # in 5 MB chunks for all files over 15 MB
      with open(file_path, "rb") as file_data:
         cos.Object(bucket_name, item_name).upload_fileobj(
            Fileobj=file_data,
            Config=transfer_config
         )

      print("Transfer for {0} Complete!\n".format(item_name))
   except ClientError as be:
      print("CLIENT ERROR: {0}\n".format(be))
   except Exception as e:
```

```
        print("Unable to complete multi-part upload: {0}".format(e))
#It will read the first frame/image of the video
video=cv2.VideoCapture(0)

while True:
  #capture the first frame
  check,frame=video.read()
  gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
  cv2.imshow('Face detection', frame)
  picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
  cv2.imwrite(picname+".jpg",frame)
  a=vis(picname+".jpg")
  print(a)
  temp =random.randint(30, 80)
  #Send Temperature & Humidity to IBM Watson
  flamesensor=random.randint(700,1000)#units in nm
  #print(firesensor)
  Mq6=random.randint(100,10000)#units in ppm isobutane
  #print(Mq6)
  data = { 'Temperature' : temp, 'FlameSensor': flamesensor,'MQ6': Mq6}
  #print (data)
  def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "FlameSensor = %s nm" % flamesensor,"Mq6=%s ppm"% Mq6, "to IBM Watson")
  success = deviceCli.publishEvent("Weather", "json", data, qos=0, on_publish=myOnPublishCallback)
  if not success:
    print("Not connected to IoTF")
  time.sleep(2)
  deviceCli.commandCallback = myCommandCallback#subscription
  if a=="fire":
    multi_part_upload("chanikya2", picname+".jpg", picname+".jpg")
    json_document={"link":COS_ENDPOINT+"/"+"chanikya2"+"/"+picname+".jpg"}
    new_document = my_database.create_document(json_document)
    # Check that the document exists in the database.
    if new_document.exists():
      print(f"Document successfully created.")
    r =
requests.get('https://www.fast2sms.com/dev/bulk?authorization=FEcl3Tw5s069IPiedqHkXLgbKYRxuVJnQ8vfO4mANUr2hjyBWz9b
4juKklE5MvfDT2pHGBQhtXnRac8x&sender_id=FSTSMS&message=Fire!!Fire!!&language=english&route=p&numbers=9666321361')
    print(r.status_code)

  #waitKey(1)- for every 1 millisecond new frame will be captured
  Key=cv2.waitKey(1)
  if Key==ord('q'):
    #release the camera
    video.release()
    #destroy all windows
    cv2.destroyAllWindows()
    break
```