CSE1901 - Technical Answers to Real-World Problems (TARP)

Project Report

TITLE : SMART AI FOREST FIRE DETECTION SYSTEM

By

Name : Mourya Vardhan Reddy

Reg. No : 19BCE1533

B. Tech Computer Science and Engineering

Submitted to

Dr. D Rekha

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

DECLARATION

I hereby declare that the report titled " **SMART AI FOREST FIRE DETECTION SYSTEM** " submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. D Rekha** , School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of the Candidate

Mourya Vardhan Reddy

Reg. No. 19BCE1533

CERTIFICATE

Certified that this project report entitled " SMART AI FOREST FIRE DETECTION SYSTEM " is a bonafide work of Mourya Vardhan Reddy 19BCE1533 and they carried out the project work under my supervision and guidance for CSE1901 - Technical Answers to Real-World Problems (TARP).

Dr. D Rekha
SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. D Rekha,** School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to Dr. Nithyanandam P (Head of the Department (HoD), B.Tech Computer Science and Engineering . SCSE, VIT Chennai), Dr. Ganesan R(Dean of the School of Computer Science & Engineering, VIT Chennai), and Dr. Geetha S(Associate Dean of the School of Computer Science & Engineering, VIT Chennai), for extending the facilities of the School towards our project and for his unstinting support.

We also take this opportunity to thank all the faculty of the School for their support and the wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution

Signature of the Candidate

Mourya Vardhan Reddy

Reg. No. 19BCE1533

CONTENTS

❖ Abstract :

The advanced time of gadgets has seen a ton of improvement. Be that as it may, the advancement of electrical and electronic gadgets has prompted a developing pace of fire mishaps basically because of the indiscretion. The primary issue in the fire mishaps is that they are not having the option to recognise the fire as ahead of schedule as could really be expected. When we distinguish the fire it becomes hard tosecurely empty all or even lessen the harm caused.

Our aim is to detect the fire as early as possible. The main objective is to detect the fire as early as possible. This can be attained by capturing the fire and motions of the spreading fire by using CCTV system and processing it to produce an alert.

The processed image is compared with the pre fed image using ML and checked for fire accident. If the processed image matches the pre fed image then an alert message along with a buzzer is enabled.
In simple word, this fire monitoring system is an addition to the existing CCTV camerasto detect the fire effectively.

❖ Introduction :

In recent times, we have heard about a lot of disasters which involves forest fire. These fires are getting really common so our team has decided to come up with a solution. In this Smart Fire Detection Project, we will be using OpenCV to detect fire and code the program in Python. We will use datasets available on the internet to trainour model and when it detects fire, it will alert the nearby user and guards so that earlyaction can be taken. This software will help prevent spread of fire at a place when it's no longer under human attention.

❖ Scope of converting the project to a product/Research paper :

This project has a lot of capabilities as per us. This project enables us to detect the fire much faster than the traditional fire detector and can be a life saviour in many cases. Usually in case of a traditional fire detector, it detects the fire when the damage is done and nothing could be done about it but our fire detector on the other hand doesn't detect the fire by sensing the smoke but detects the fire by the colour gradient.

It can be converted into a product and can be installed at places where fire can spread easily thus alarming us before the onset of vast forest fire .

❖ Methodologies Available to detect :

- Fast R-CNN
- Faster R-CNN
- Histogram of Oriented Gradients (HOG)
- Region-based Convolutional Neural Networks (R-CNN)
- Region-based Fully Convolutional Network (R-FCN)
- Single Shot Detector (SSD)
- Spatial Pyramid Pooling (SPP-net)
- YOLO (You Only Look Once)
- Viola-Jones Algorithm

The methodology adopted by us in this project is “Viola-Jones Algorithm”for the face detection system and “faster R-CNN" because of how accurately its simplicity and how accurately it work.

❖ Creating the customised CNN Architecture :

We are going to use TensorFlow API Keras for building our model. Let’s first create our ImageDataGenerator for labelling our data. [1] and [2] datasets are used here for training and validation .

Finally, we will have 980 images for training and 239 images for validation. We are going to use data augmentation as well. We will use Adam as an optimiser witha learning rate of 0.0001. After training for 50 epochs, we get the training accuracy of 96.83 and validation accuracy of 94.98. The training and validation loss is 0.09 and 0.13 respectively.

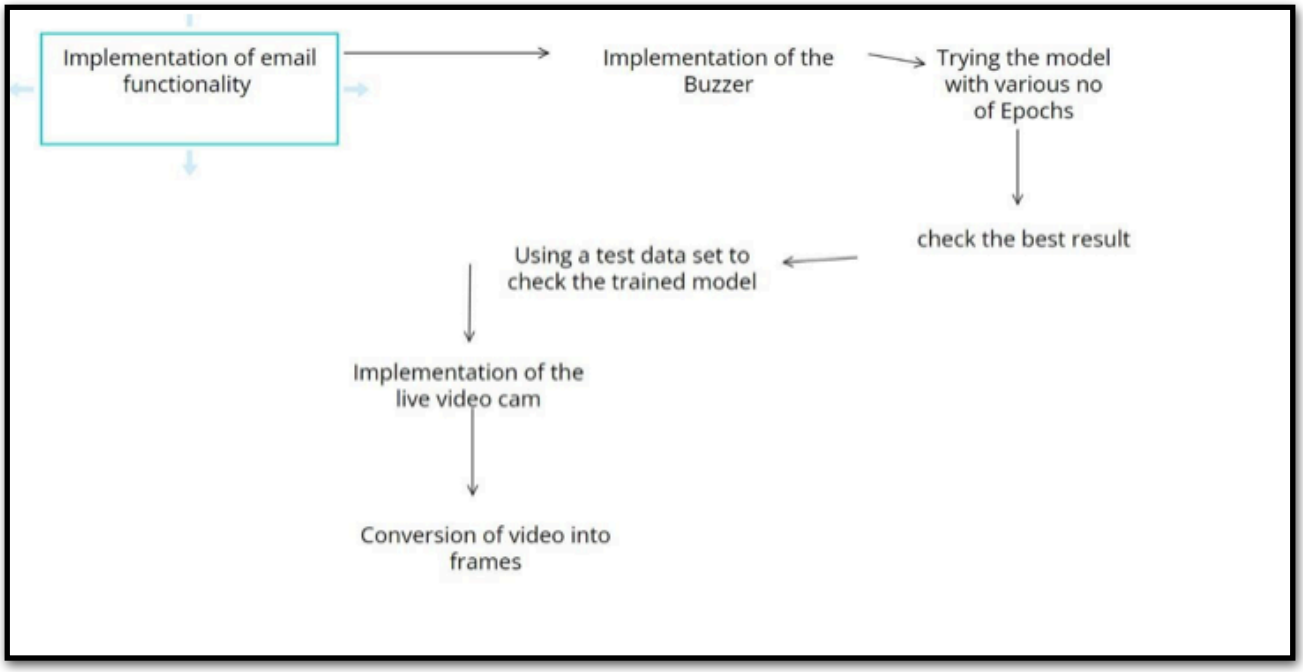
❖ Description of the project :

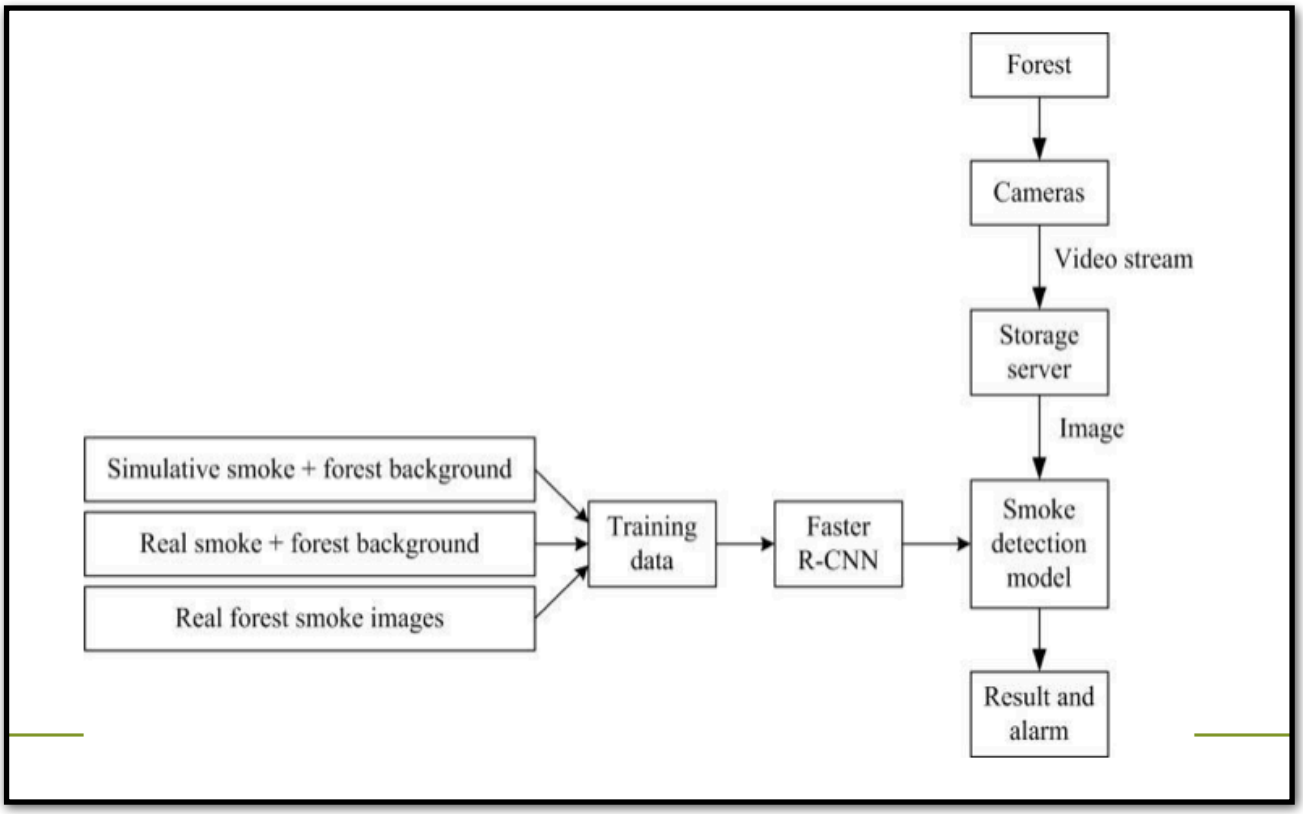
This smart fire detection system automatically detects the fire in real time and alert usthus limiting its further spread. we will train the system with positive and negative images thus all these data sets will help the system learn and analyse in real time that weather the image or the video they are feed with has a fire in it or not. The advanced time of gadgets has seen a ton of improvement. Be that as it may, the advancement of electrical and electronic gadgets has prompted a developing pace of fire mishaps basically because of the indiscretion. The primary issue in the fire mishaps is that they are not having the option to recognise the fire as ahead of schedule as could really be expected. When we distinguish the fire it becomes hard tosecurely empty all or even lessen the harm caused. Our aim is to detect the fire as early as possible.

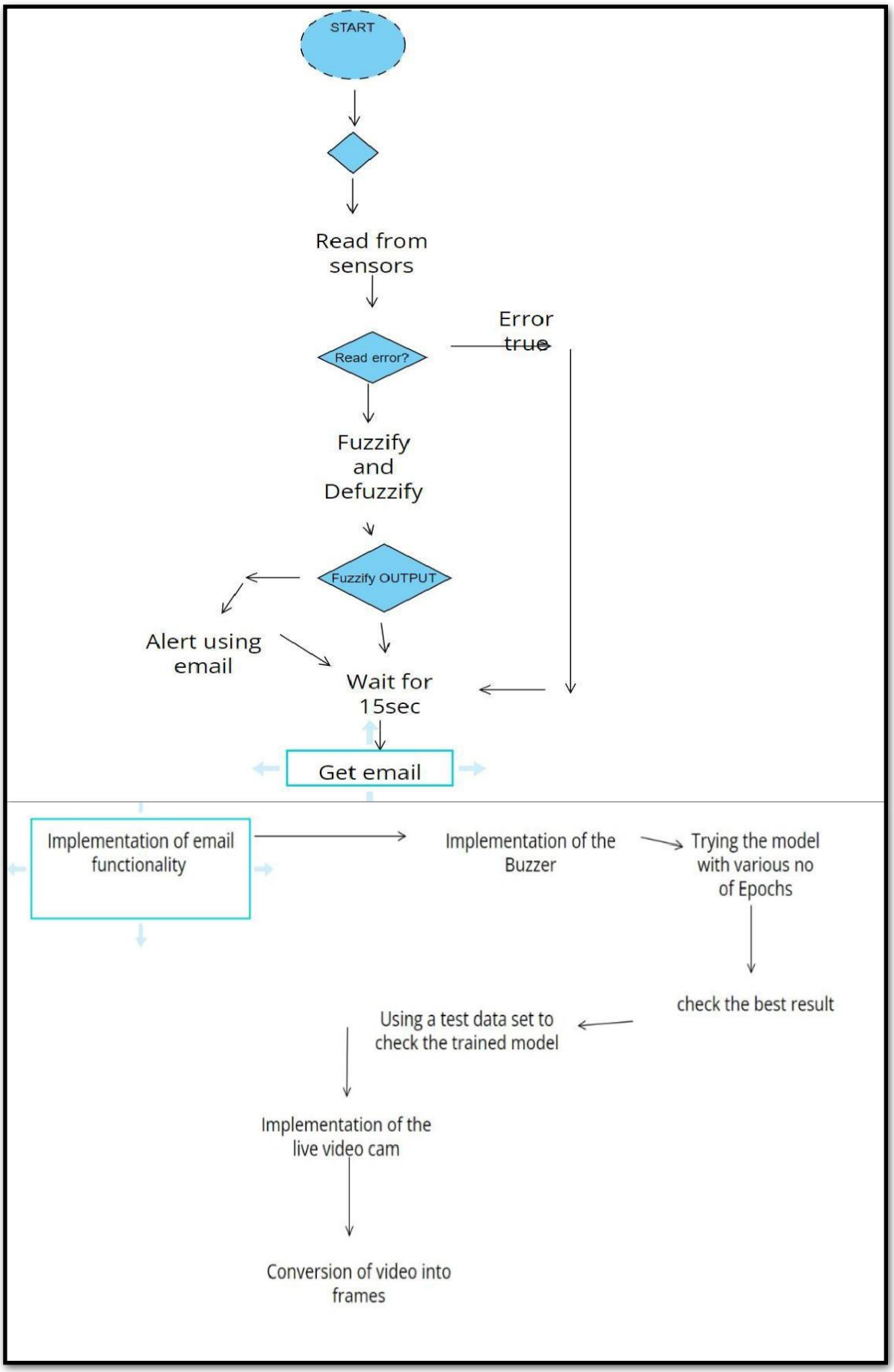
The main objective is to detect the fire as early as possible. This canbe attained by capturing the fire and motions of the spreading fire by using CCTV system and processing it to produce an alert.

The processed image is compared with the pre fed image using ML and checked for fire accident. If the processed image matches the pre fed image then an alert message along with a buzzer is enabled. In simple word, this fire monitoring system is an addition to the existing CCTV cameras to detect the fire effectively.

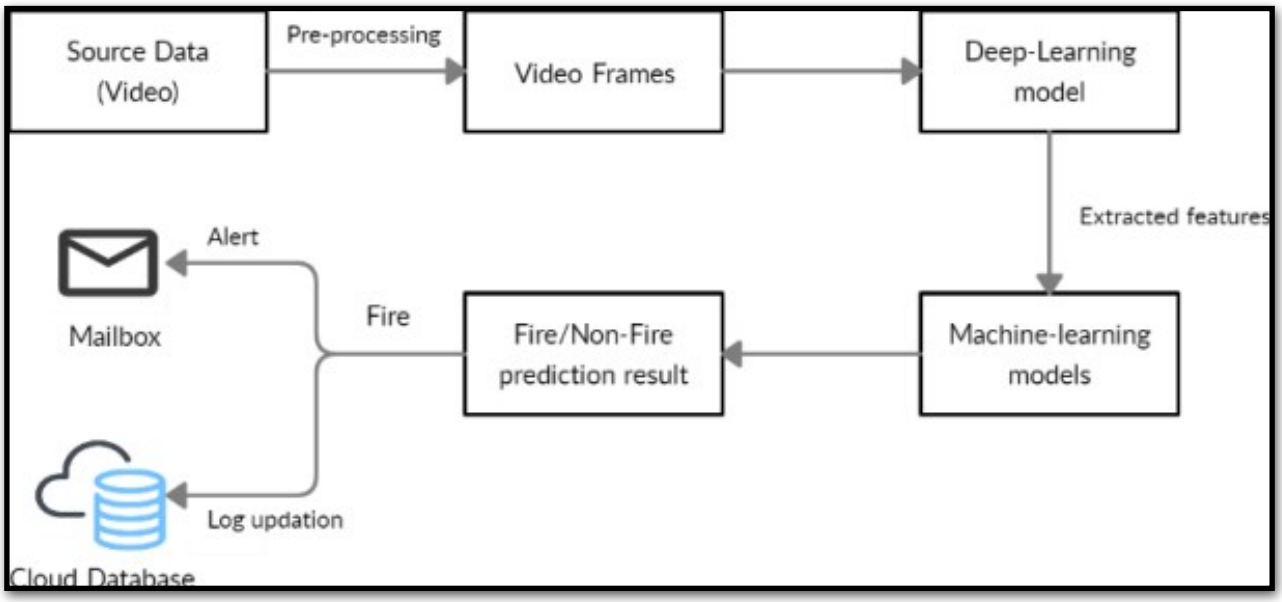
❖ Software Engineering Flow :






START
Read from sensors
Read error?
Error true
Fuzzify and Defuzzify
Fuzzify OUTPUT
Alert using email
Wait for 15sec
Get email
Implementation of email functionality
Implementation of the Buzzer
Trying the model with various no of Epochs
check the best result
Using a test data set to check the trained model
Implementation of the live video cam
Conversion of video into frames

❖ System Design :



❖ Comparison :

All the possible feature-extractor and classifier combinations were evaluated using stratified K-foldvalidation. Table 1 shows the obtained performance metric values for different combinations of deep learning networks and classifier algorithms.

The best performance was observed with ResNet50 as the deep learning feature extraction model and Support Vector Machine as the ML classifier and the Accuracy, Precision and Recall values for this combination was 97.8%,97.46% and97.66% respectively. Hence this would be used in our application.

Network	Algorithm	Accuracy	Precision	Recall
Resnet 50	Decision Tree	94.78%	93.98%	94.44%
	Naïve Bayes	93.17%	91.92%	92.98%
	Logistic Regression	97.73%	97.30%	97.66%
	SVM	97.80%	97.46%	97.66%
Inception-Net V2	Decision Tree	90.51%	89.59%	89.25%
	Naïve Bayes	94.32%	91.75%	95.97%
	Logistic Regression	96.81%	96.23%	96.71%
	SVM	96.25%	94.71%	97.07%
Inception V3	Decision Tree	92.58%	92.07%	91.37%
	Naïve Bayes	94.71%	92.63%	96.19%
	Logistic Regression	97.17%	96.53%	97.22%
	SVM	96.51%	95.36%	97.00%

❖ Applications Used in this Project :

- Jupiter notebook
- Anaconda navigator
- Tinker cad
- ThingSpeak
- Viola jones algorithm
- Fast R CNN
- G-MAIL application

Test Case Description	Expected Output	Actual Output	Test Status (P/F)
Video with visuals of mountains, pools, indoors, driving	Non-fire	Non-fire	P
Headlight Glare during night	Non-fire	Fire	F
Kid playing with toys	Non-fire	Non-fire	P
Man riding bicycle	Non-fire	Non-fire	P
Close up view of light through of fabric	Non-fire	Non-fire	P
Daytime Traffic	Non-fire	Non-fire	P
Camera focused on sun in a garden setting	Non-fire	Non-fire	P
Man walking with an orange bag in an office environment	Non-fire	Non-fire	P
Lake with yachts and windmill	Non-fire	Non-fire	P
Yellow heavy-duty vehicles moving on a flyover	Non-fire	Non-fire	P
People waiting at train terminal with a train approaching	Non-fire	Non-fire	P
People moving inside an airport, subway terminal	Non-fire	Non-fire	P

❖ Implementation of the project :

- To detect fire and for the machine to study we have collected good images (withoutfire) and bad images (with fire) sample.

- We started writing the python code for the project to establish and to develop afrontend.
- We also have stated OpenCV code so as to detect fire and identify it properly.
- We also used tinker cad to built a circuit of temperature detection sensor system.
- We used ThingSpeak for the better analytics and date representation for tinker cad.
- We used mail application to give and alerts and notifications to the base/camp.

❖ Details of all the experiments conducted so far :

Test Case Description	Expected Output	Actual Output	Test Status P/F
Bus on Fire	Fire	Fire	P
Kitchen sink fire	Fire	Fire	P
Twigs and leaves on fire	Fire	Fire	P
Fireplace	Fire	Fire	P
Automobile on Fire	Fire	Fire	P

❖ Solution of the problem :

The aim of our project was to develop to an application capable of detecting fire in videos and images, as well as live videos which is robust and works in any environment. In this regard, we haveexperimented with various deep learning models and classification models. An email alert feature has also been incorporated to our application to provide real time alerts to the concerned stakeholders along with a logging system, which is implemented using Firebase. The application performed exceptionally well during testing. It was able to identify fires in all of the twelve test fire videos but misclassified some instances of non-fire videos.

Piece of cloth set on fire by a firefighter	Fire	Fire	P
Trash can set on Fire	Fire	Fire	P
Close-up in view of fire	Fire	Fire	P
Huge Campfire	Fire	Fire	P
Forest Fire	Fire	Fire	P
Bush fire	Fire	Fire	P
Fire on mattress	Fire	Fire	P

❖ Impact of the project :

The purpose of this document is to present a detailed description of the Smart Fire Detection System. It will explain the purpose and features of the system and what thesystem will do. This document is intended for both the stakeholders and the developers of the system. Smart Fire Detection project will be applicable at various places such as institutions, homes, factories, etc. It will be more efficient and easier way to help prevent big fires causing loss of lives and huge damages.

Notifications ofpotential fires which user can easily access according to his rights makes it better as compared to the traditional ways of fire surveillance.

❖ Challenges faced during the project :

20. we faced difficulty while detecting the fire point or spot it use to detect any bright object as fire so we had to train it with more no of datasets.
21. we had problem connecting our code with the live webcam and had to go though and read about various directories.
22. overfitting of the data
23. Interconnecting all the applications and modules

❖ Literature Survey :

Design of intelligent fire alarm system based on GSM network

Author: Chun-yuan Lian

Changzhou Institute of Technology, School of Electronic Information & ElectricEngineering, China

Review:

An intelligent fire alarm system based on GSM network is designed in order to solvethe problem of complex cabling, misdeclaration and missing alarm of traditional firealarm systems. MSP430F149 is adopted as main control chip, and GSM module TC35I is used for remote alarming and data exchanging.

Result: this design has characters of real-time and good reliability, and will be widelyused.

Design and Realisation of Fire Alarm by Determining Probability Based on Multi-sensor Integrated

Authors: Xia Huanxiong, Sun Shuwen, Yao Yiwu, Guo Wenzeng, Zhang Shanshan,Wang Jie (College of Mechanical Engineering & Applied Electronics Technology, Beijing Univ. of Technology, Beijing 100124, China)

Due to the shortage of detecting characteristic parameters, a single detection signal based fire alarm usually causes leak-check, error and solidified conditions and parameters. In order to reduce such a situation, we design an intellectualised fire alarm system where C8051F040 MCU collects temperature, smoke and light intensity sensor signals, and uses continuous probability models of a comprehensive analysis and processing.

Result: The alarm can accurately and quickly respond to the probability of the fire process, and give early warning, alarm signals and act related firefighting equipment.

Fire Detection using Artificial Intelligence for Fire-Fighting Robots

Authors: Sreesruthi Ramasubramanian (School of Engineering and Physical Sciences,Heriot-Watt University, Dubai, United Arab Emirates), Senthil Arumugam Muthukumaraswamy (School of Engineering and Physical Sciences, Heriot-Watt University, Dubai, United Arab Emirates), A. Sasikala (Department of Electrical and Electronics Engineering, Srisairam Institute of Technology, Chennai, India)

Fire-fighting robots are used in indoor environments to detect fires and extinguish them. Sensors such as flame sensors are currently used to detect fire in fire-fighting robots. The disadvantage of using sensors is that fire beyond a threshold distance cannot be detected. Using artificial intelligence techniques, fire can be detected in a wider range. Haar Cascade Classifier is a machine-learning algorithm that was initially used for object detection. The results obtained using Haar Cascade Classifier were not very accurate, especially when multiple fires had to be detected. Transfer learning from a pre-trained YOLOv3 model was then used to train the model for fire detection to improve accuracy. The benefits and drawbacks of using deep learning forobject detection over machine learning are highlighted. The algorithm used to obtain the target location the robot must move to use bounding box coordinates is also discussed in this paper.

❖ **Code :**

In [1]:

```
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
import shutil
```

In [2]:

```
TRAINING_DIR = "/Users/mourya/Desktop/fire-detection-master/
Datasets 1-2/Training"

training_datagen = ImageDataGenerator(rescale = 1./255,
                                      horizontal_flip=True,
                                           rotation_range=30,

height_shift_range=0.2,
                                      fill_mode='nearest')

VALIDATION_DIR = "/Users/mourya/Desktop/fire-detection-
master/Datasets 1-2/Validation"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(224,224),
    class_mode='categorical',
  batch_size = 64
)
#print(train_generator)
validation_generator =
validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(224,224),
    class_mode='categorical',
  batch_size= 16
)
```

```
Found 980 images belonging to 2 classes.
Found 239 images belonging to 2 classes.
```

In [3]:

```
from tensorflow.keras.optimizers import RMSprop,Adam
model = tf.keras.models.Sequential([
          tf.keras.layers.Conv2D(96, (11,11), strides=(4,4),
activation='relu', input_shape=(224, 224, 3)),
```

```
          tf.keras.layers.MaxPooling2D(pool_size = (3,3),
strides=(2,2)),
          tf.keras.layers.Conv2D(256, (5,5),
activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3),
strides=(2,2)),
          tf.keras.layers.Conv2D(384, (5,5),
activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3),
strides=(2,2)),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2048, activation='relu'),
          tf.keras.layers.Dropout(0.25),
          tf.keras.layers.Dense(1024, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2, activation='softmax')
])
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.0001),
              metrics=['acc'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
#
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 26, 26, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 22, 22, 256)       614656

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 6, 6, 384)
2457984
```

```
 max_pooling2d_2 (MaxPooling  (None, 2, 2, 384)        0
 2D)

 flatten (Flatten)           (None, 1536)              0

 dropout (Dropout)           (None, 1536)              0

 dense (Dense)               (None, 2048)
3147776

 dropout_1 (Dropout)         (None, 2048)              0

 dense_1 (Dense)             (None, 1024)
2098176

 dropout_2 (Dropout)         (None, 1024)              0

 dense_2 (Dense)             (None, 2)                 2050

=================================================================
====
Total params: 8,355,586
Trainable params: 8,355,586
Non-trainable params: 0
_________________________________________________________________
____
```

In [6]:

```
# class myCallback(tf.keras.callbacks.Callback):
#   def on_epoch_end(self, epoch, logs={}):
#     if(logs.get('val_acc')>=0.98):
#       print('\nReached ^98%')
#       self.model.stop_training = True
# callbacks = myCallback()

history = model.fit(
    train_generator,
    steps_per_epoch = 16,
    epochs = 55,
    validation_data = validation_generator,
    validation_steps = 15
```

```
    #callbacks=[callbacks]
)
```

```
Epoch 1/55
16/16 [==============================] - 107s 7s/step - loss:
0.5195 - acc: 0.7449 - val_loss: 0.3824 - val_acc: 0.8243
Epoch 2/55
16/16 [==============================] - 108s 7s/step - loss:
0.3257 - acc: 0.8694 - val_loss: 0.2365 - val_acc: 0.9163
Epoch 3/55
16/16 [==============================] - 95s 6s/step - loss:
0.3580 - acc: 0.8418 - val_loss: 0.3358 - val_acc: 0.8828
Epoch 4/55
16/16 [==============================] - 93s 6s/step - loss:
0.3061 - acc: 0.8857 - val_loss: 0.2719 - val_acc: 0.9163
Epoch 5/55
16/16 [==============================] - 88s 6s/step - loss:
0.2597 - acc: 0.9020 - val_loss: 0.2527 - val_acc: 0.9121
Epoch 6/55
16/16 [==============================] - 88s 5s/step - loss:
0.2608 - acc: 0.8898 - val_loss: 0.2140 - val_acc: 0.9289
Epoch 7/55
16/16 [==============================] - 88s 6s/step - loss:
0.2152 - acc: 0.9245 - val_loss: 0.2326 - val_acc: 0.9247
Epoch 8/55
16/16 [==============================] - 88s 5s/step - loss:
0.2096 - acc: 0.9245 - val_loss: 0.1949 - val_acc: 0.9372
Epoch 9/55
16/16 [==============================] - 88s 6s/step - loss:
0.2050 - acc: 0.9286 - val_loss: 0.1933 - val_acc: 0.9331
Epoch 10/55
16/16 [==============================] - 88s 5s/step - loss:
0.2171 - acc: 0.9112 - val_loss: 0.2318 - val_acc: 0.8954
Epoch 11/55
16/16 [==============================] - 87s 5s/step - loss:
0.2152 - acc: 0.9327 - val_loss: 0.1899 - val_acc: 0.9372
Epoch 12/55
16/16 [==============================] - 89s 6s/step - loss:
0.1901 - acc: 0.9327 - val_loss: 0.1905 - val_acc: 0.9456
Epoch 13/55
```

```
16/16 [==============================] - 87s 5s/step - loss:
0.1690 - acc: 0.9439 - val_loss: 0.1702 - val_acc: 0.9372
Epoch 14/55
16/16 [==============================] - 89s 6s/step - loss:
0.1715 - acc: 0.9367 - val_loss: 0.1741 - val_acc: 0.9372
Epoch 15/55
16/16 [==============================] - 91s 6s/step - loss:
0.1604 - acc: 0.9418 - val_loss: 0.1858 - val_acc: 0.9331
Epoch 16/55
16/16 [==============================] - 88s 5s/step - loss:
0.1835 - acc: 0.9357 - val_loss: 0.1468 - val_acc: 0.9456
Epoch 17/55
16/16 [==============================] - 88s 5s/step - loss:
0.1635 - acc: 0.9327 - val_loss: 0.1779 - val_acc: 0.9456
Epoch 18/55
16/16 [==============================] - 88s 5s/step - loss:
0.1553 - acc: 0.9429 - val_loss: 0.1645 - val_acc: 0.9540
Epoch 19/55
16/16 [==============================] - 88s 5s/step - loss:
0.1515 - acc: 0.9449 - val_loss: 0.1544 - val_acc: 0.9498
Epoch 20/55
16/16 [==============================] - 87s 5s/step - loss:
0.1442 - acc: 0.9469 - val_loss: 0.1516 - val_acc: 0.9582
Epoch 21/55
16/16 [==============================] - 87s 5s/step - loss:
0.1316 - acc: 0.9520 - val_loss: 0.1706 - val_acc: 0.9456
Epoch 22/55
16/16 [==============================] - 94s 6s/step - loss:
0.1368 - acc: 0.9500 - val_loss: 0.1238 - val_acc: 0.9498
Epoch 23/55
16/16 [==============================] - 89s 5s/step - loss:
0.1216 - acc: 0.9622 - val_loss: 0.1452 - val_acc: 0.9498
Epoch 24/55
16/16 [==============================] - 89s 6s/step - loss:
0.1185 - acc: 0.9582 - val_loss: 0.1306 - val_acc: 0.9582
Epoch 25/55
16/16 [==============================] - 88s 5s/step - loss:
0.1337 - acc: 0.9510 - val_loss: 0.1391 - val_acc: 0.9498
Epoch 26/55
```

```
16/16 [==============================] - 88s 5s/step - loss:
0.1293 - acc: 0.9531 - val_loss: 0.1434 - val_acc: 0.9582
Epoch 27/55
16/16 [==============================] - 87s 5s/step - loss:
0.1341 - acc: 0.9531 - val_loss: 0.1914 - val_acc: 0.9331
Epoch 28/55
16/16 [==============================] - 87s 5s/step - loss:
0.1396 - acc: 0.9490 - val_loss: 0.1362 - val_acc: 0.9414
Epoch 29/55
16/16 [==============================] - 87s 5s/step - loss:
0.1310 - acc: 0.9520 - val_loss: 0.1575 - val_acc: 0.9540
Epoch 30/55
16/16 [==============================] - 87s 5s/step - loss:
0.1201 - acc: 0.9602 - val_loss: 0.2001 - val_acc: 0.9205
Epoch 31/55
16/16 [==============================] - 86s 5s/step - loss:
0.1341 - acc: 0.9520 - val_loss: 0.1530 - val_acc: 0.9623
Epoch 32/55
16/16 [==============================] - 87s 5s/step - loss:
0.1157 - acc: 0.9551 - val_loss: 0.1348 - val_acc: 0.9456
Epoch 33/55
16/16 [==============================] - 87s 5s/step - loss:
0.1250 - acc: 0.9602 - val_loss: 0.1723 - val_acc: 0.9498
Epoch 34/55
16/16 [==============================] - 86s 5s/step - loss:
0.1143 - acc: 0.9653 - val_loss: 0.1324 - val_acc: 0.9456
Epoch 35/55
16/16 [==============================] - 90s 6s/step - loss:
0.1035 - acc: 0.9622 - val_loss: 0.1516 - val_acc: 0.9498
Epoch 36/55
16/16 [==============================] - 90s 6s/step - loss:
0.1394 - acc: 0.9541 - val_loss: 0.1321 - val_acc: 0.9540
Epoch 37/55
16/16 [==============================] - 89s 6s/step - loss:
0.1146 - acc: 0.9633 - val_loss: 0.1360 - val_acc: 0.9582
Epoch 38/55
16/16 [==============================] - 90s 6s/step - loss:
0.0885 - acc: 0.9704 - val_loss: 0.1409 - val_acc: 0.9540
Epoch 39/55
```

```
16/16 [==============================] - 90s 5s/step - loss:
0.1040 - acc: 0.9643 - val_loss: 0.1792 - val_acc: 0.9456
Epoch 40/55
16/16 [==============================] - 96s 6s/step - loss:
0.1115 - acc: 0.9592 - val_loss: 0.1559 - val_acc: 0.9623
Epoch 41/55
16/16 [==============================] - 87s 5s/step - loss:
0.1236 - acc: 0.9592 - val_loss: 0.1397 - val_acc: 0.9540
Epoch 42/55
16/16 [==============================] - 89s 6s/step - loss:
0.1062 - acc: 0.9673 - val_loss: 0.1353 - val_acc: 0.9456
Epoch 43/55
16/16 [==============================] - 93s 6s/step - loss:
0.1100 - acc: 0.9602 - val_loss: 0.1381 - val_acc: 0.9456
Epoch 44/55
16/16 [==============================] - 108s 7s/step - loss:
0.0938 - acc: 0.9673 - val_loss: 0.1783 - val_acc: 0.9540
Epoch 45/55
16/16 [==============================] - 88s 6s/step - loss:
0.1169 - acc: 0.9592 - val_loss: 0.1572 - val_acc: 0.9456
Epoch 46/55
16/16 [==============================] - 94s 6s/step - loss:
0.1074 - acc: 0.9622 - val_loss: 0.1380 - val_acc: 0.9540
Epoch 47/55
16/16 [==============================] - 103s 7s/step - loss:
0.0981 - acc: 0.9673 - val_loss: 0.1388 - val_acc: 0.9540
Epoch 48/55
16/16 [==============================] - 113s 7s/step - loss:
0.0881 - acc: 0.9633 - val_loss: 0.1375 - val_acc: 0.9623
Epoch 49/55
16/16 [==============================] - 110s 7s/step - loss:
0.0934 - acc: 0.9724 - val_loss: 0.1593 - val_acc: 0.9582
Epoch 50/55
16/16 [==============================] - 94s 6s/step - loss:
0.0870 - acc: 0.9663 - val_loss: 0.1476 - val_acc: 0.9540
Epoch 51/55
16/16 [==============================] - 89s 6s/step - loss:
0.1061 - acc: 0.9643 - val_loss: 0.1587 - val_acc: 0.9623
Epoch 52/55
```

```
16/16 [==============================] - 96s 6s/step - loss:
0.0933 - acc: 0.9673 - val_loss: 0.1350 - val_acc: 0.9582
Epoch 53/55
16/16 [==============================] - 95s 6s/step - loss:
0.0838 - acc: 0.9684 - val_loss: 0.1941 - val_acc: 0.9456
Epoch 54/55
16/16 [==============================] - 96s 6s/step - loss:
0.0947 - acc: 0.9633 - val_loss: 0.1669 - val_acc: 0.9498
Epoch 55/55
16/16 [==============================] - 98s 6s/step - loss:
0.0926 - acc: 0.9684 - val_loss: 0.2761 - val_acc: 0.9205
```

In [7]:

```
model.save('final.h5')
```

In [4]:

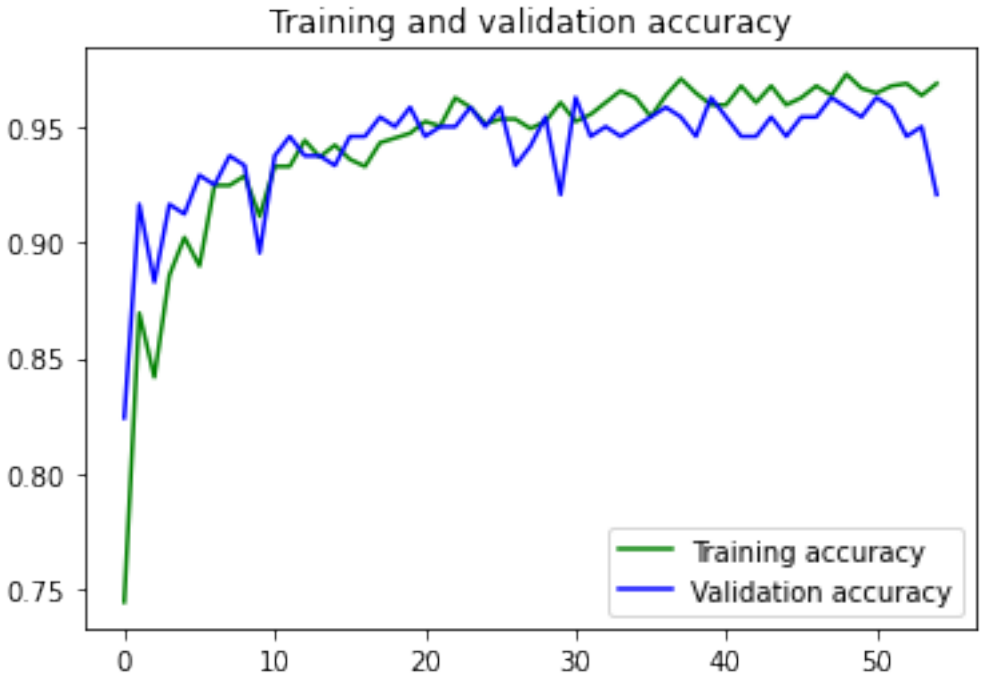
```
model=tf.keras.models.load_model('firemodelfinal.h5')
history=model
```

In [5]:

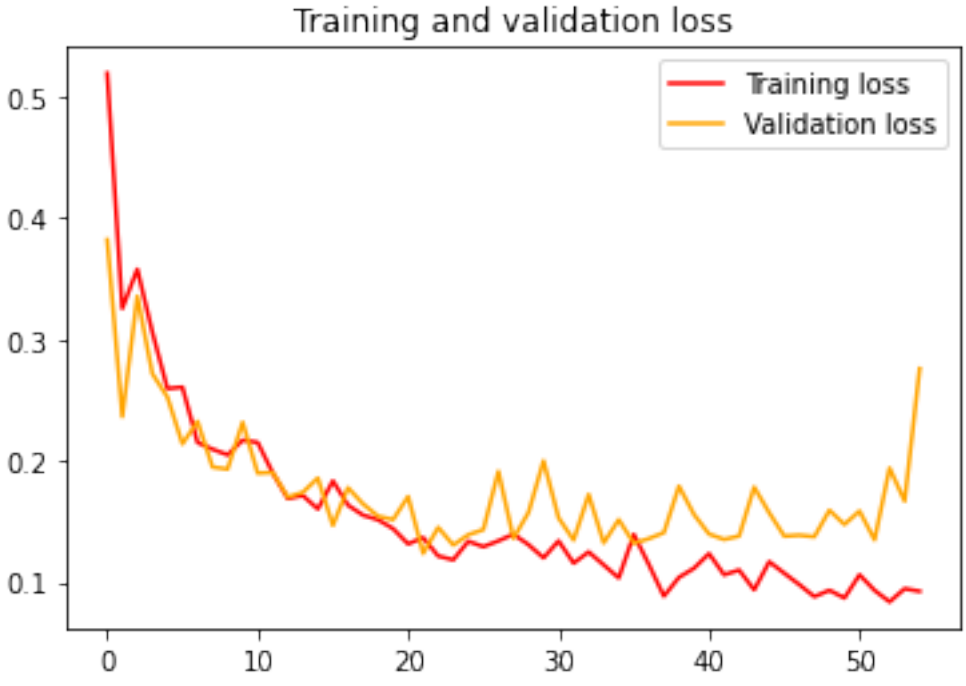
```
model=tf.keras.models.load_model('firemodelfinalnew.h5')
history=model
```

In [8]:

```
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'g', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'orange', label='Validation loss')
plt.title('Training and validation loss')

plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

In [6]:

```
import numpy as np
import files
from keras.preprocessing import image
import cv2
import smtplib
import requests



def send_mail_function(m):
    recipientEmail =
"mouryavardhan.reddy2019@vitstudent.ac.in"
    recipientEmail = recipientEmail.lower()

    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.ehlo()
        server.starttls()
        server.login("mourya.tarp@gmail.com",'Mouryatarp13')
        server.sendmail('mourya.tarp@gmail.com',
recipientEmail, m)
        print("sent to {}".format(recipientEmail))
        server.close()
    except Exception as e:
        print(e)



# Opens the camera
cap= cv2.VideoCapture(0)
i=0
counter=0
while(cap.isOpened()):
    msg=requests.get("https://api.thingspeak.com/channels/
1728567/feeds.json?api_key=U22W2EYH49SR8WO0&results=2")
    msg=msg.json()['feeds'][-1]['field1']
    #print("\nThe Message sent was: \n\n"+str(msg))
    ret, frame = cap.read()
    img = cv2.resize(frame, (224, 224))
    #if ret == False:
```

```
        # break
    #cv2.imwrite('fire'+str(i)+'.jpg',frame)

    #path ='fire'+str(i)+'.jpg'
    #img = image.load_img(path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0) /255
    classes = model.predict(x)
    #print(np.argmax(classes[0])==0, max(classes[0]))
    #print(classes)
    if(classes[0][0]>classes[0][1]):
        print('no fire')
        if(float(msg)>25.00):
                send_mail_function("Etremely High
Temperatures detcted. Chances of Fire.")
        image_path = r'C:\Users\psing\Music\python
projects\fire detection both\fire-detection-
master\outputimages\no fire'+str(i)+'.jpg'
        cv2.imwrite(image_path,frame)
    else:
        print('fire')
        counter=counter+1
        if(counter>6):

            if(float(msg)>25.00):
                send_mail_function("Warning A Fire Accident
has been reported.")
            else:
                send_mail_function("Alert! Chance of a fire
Reported.")
            print('sound playing'+ str(counter))
            counter=0


        image_path = r'C:\Users\psing\Music\python
projects\fire detection both\fire-detection-
master\outputimages\fire'+str(i)+'.jpg'
        cv2.imwrite(image_path,frame)
    i+=1
    cv2.imshow("output", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```

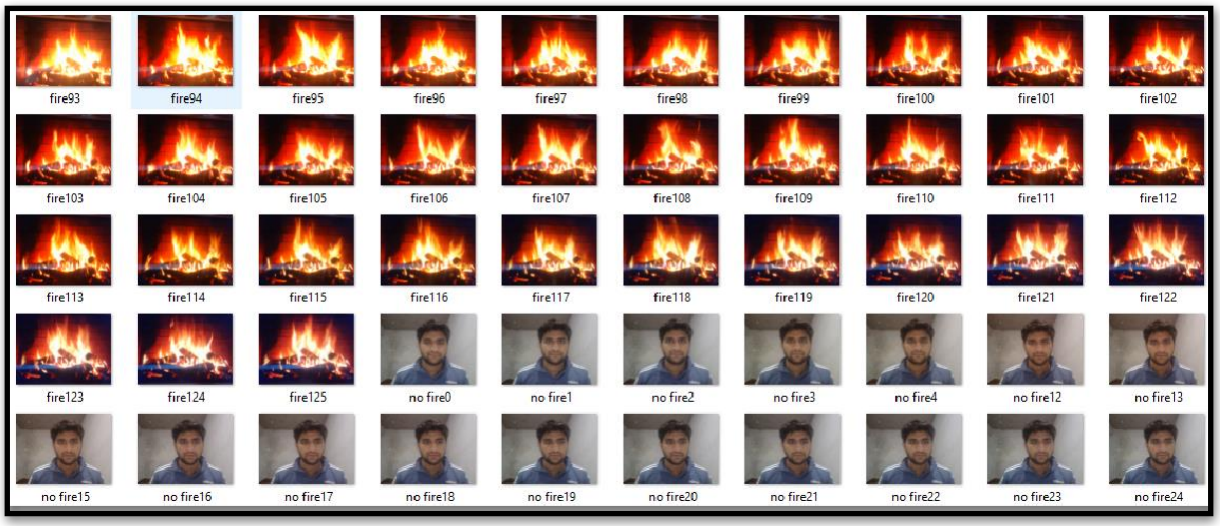
```
cv2.destroyAllWindows()
```

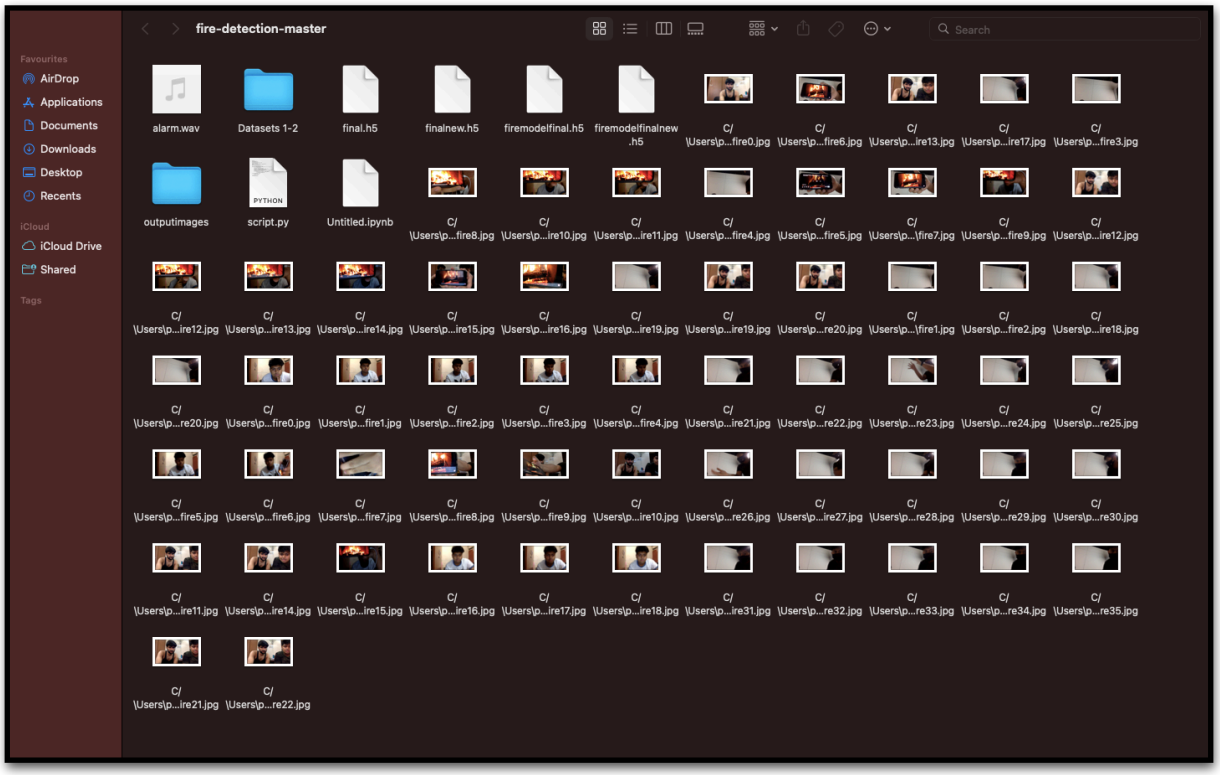
```
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
fire
fire
fire
fire
fire
fire
fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
sound playing7
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
```

In [ ]:

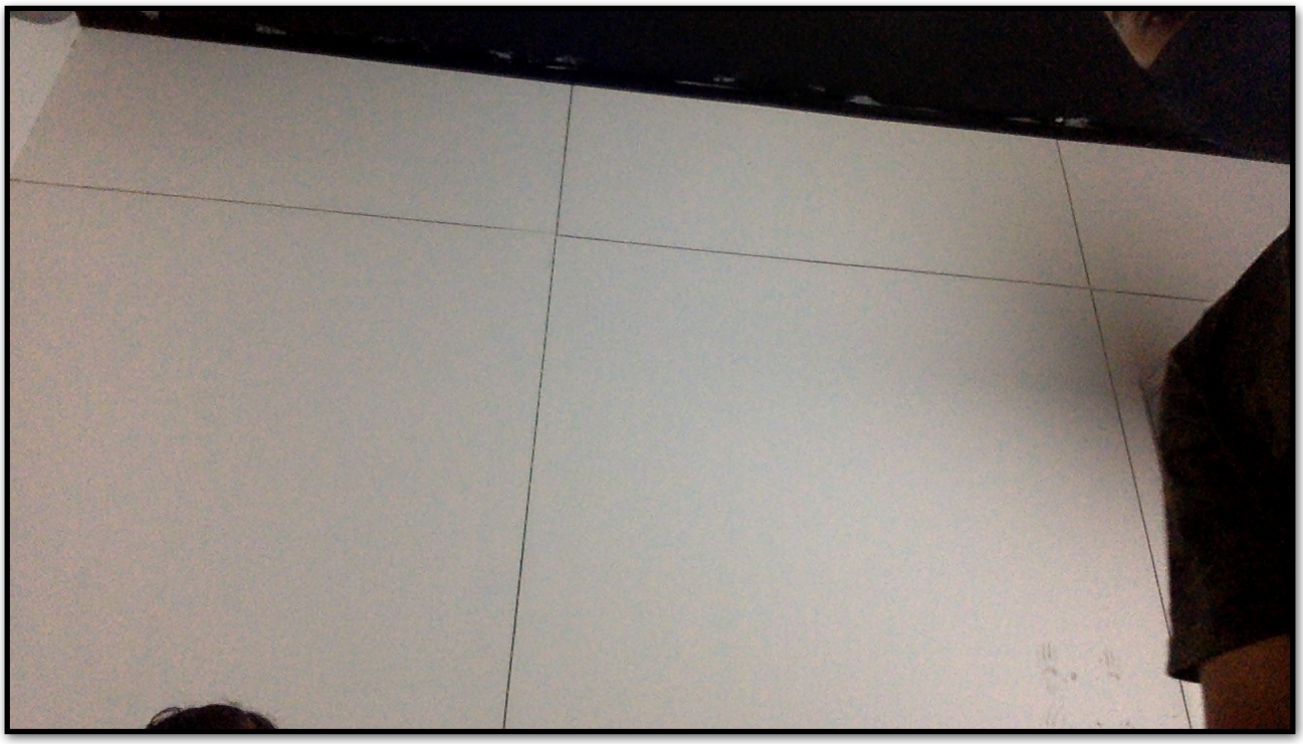
In [ ]:

❖ Results :


fire93
fire94
fire95
fire96
fire97
fire98
fire99
fire100
fire101
fire102
fire103
fire104
fire105
fire106
fire107
fire108
fire109
fire110
fire111
fire112
fire113
fire114
fire115
fire116
fire117
fire118
fire119
fire120
fire121
fire122
fire123
fire124
fire125
no fire0
no fire1
no fire2
no fire3
no fire4
no fire12
no fire13
no fire15
no fire16
no fire17
no fire18
no fire19
no fire20
no fire21
no fire22
no fire23
no fire24

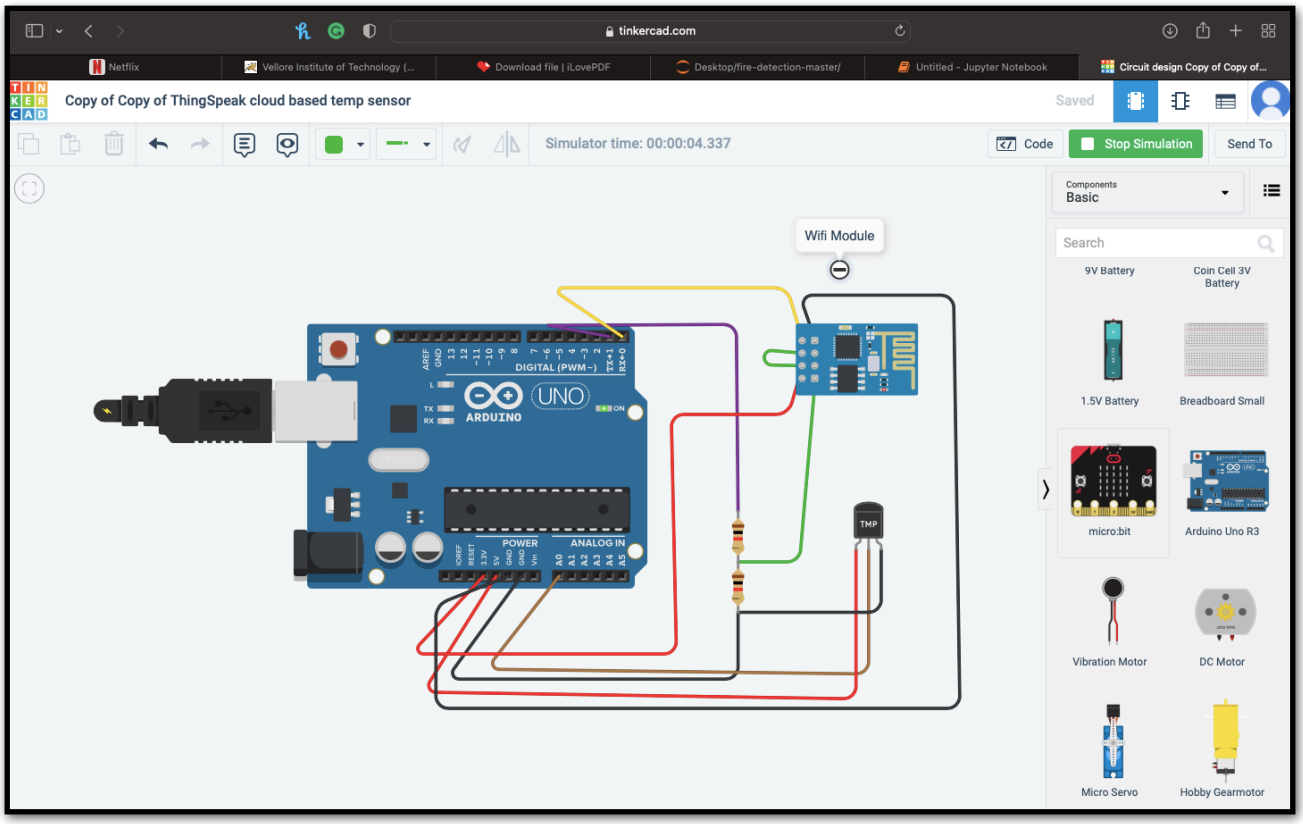

fire-detection-master
Favourites
AirDrop
Applications
Documents
Downloads
Desktop
Recents
iCloud
iCloud Drive
Shared
Tags
alarm.wav
Datasets 1-2
final.h5
finalnew.h5
firemodelfinal.h5
firemodelfinalnew.h5
outputimages
script.py
Untitled.ipynb

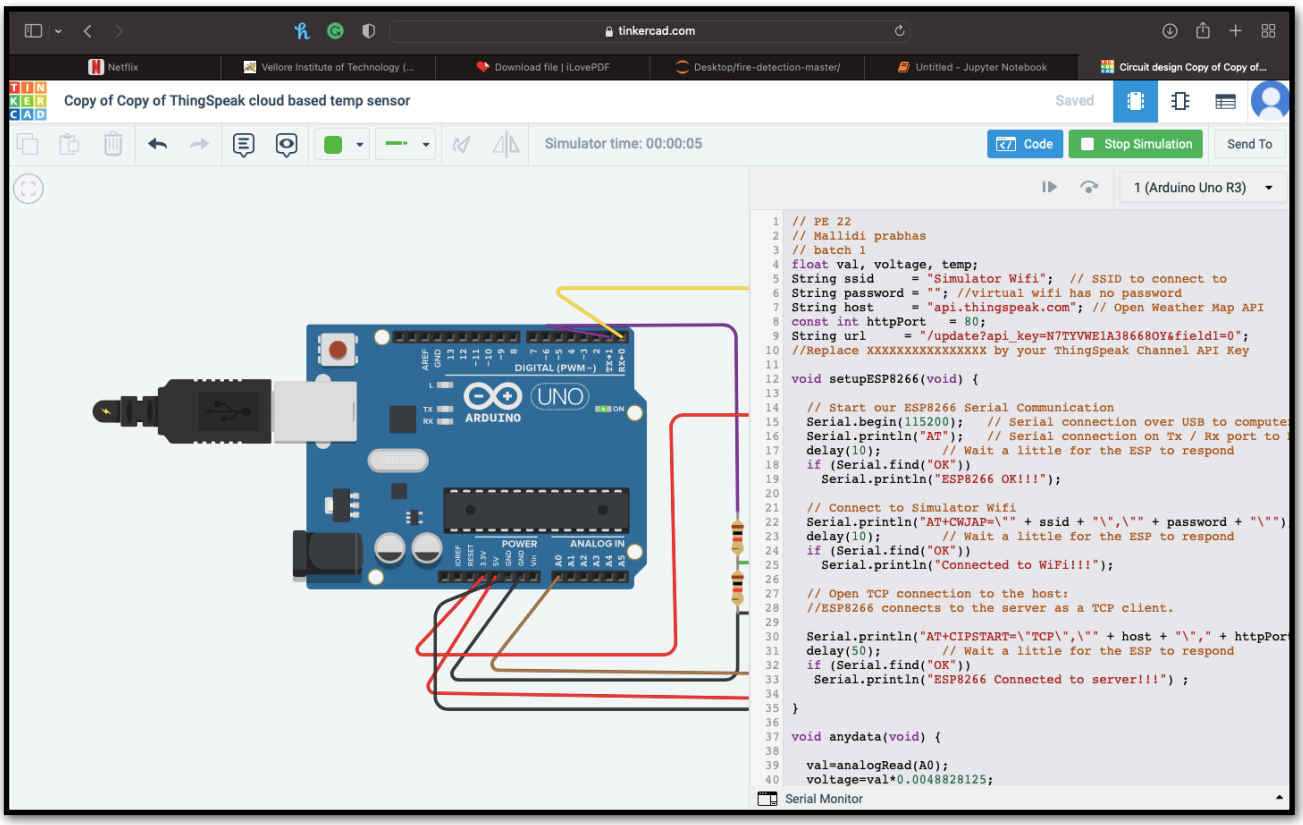
```
fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
sound playing7
fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
```



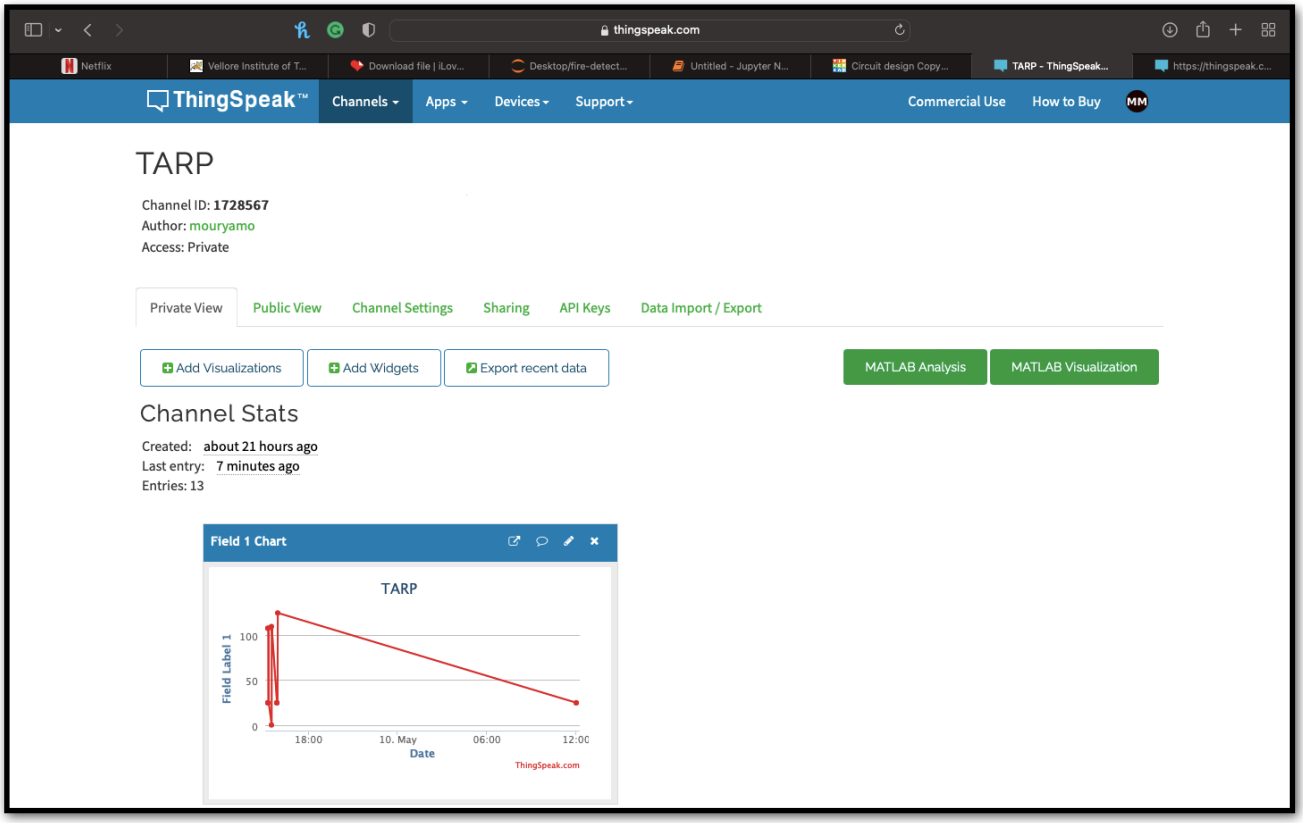
```
fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
no fire
sent to mouryavardhan.reddy2019@vitstudent.ac.in
```

❖ Tinkercad :

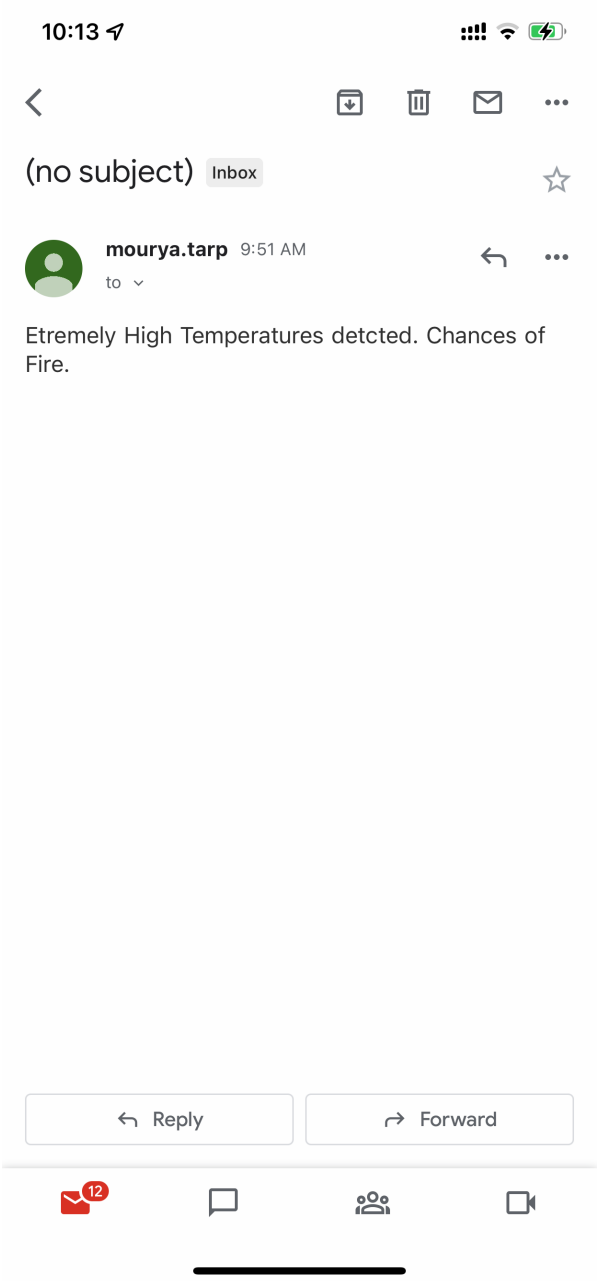


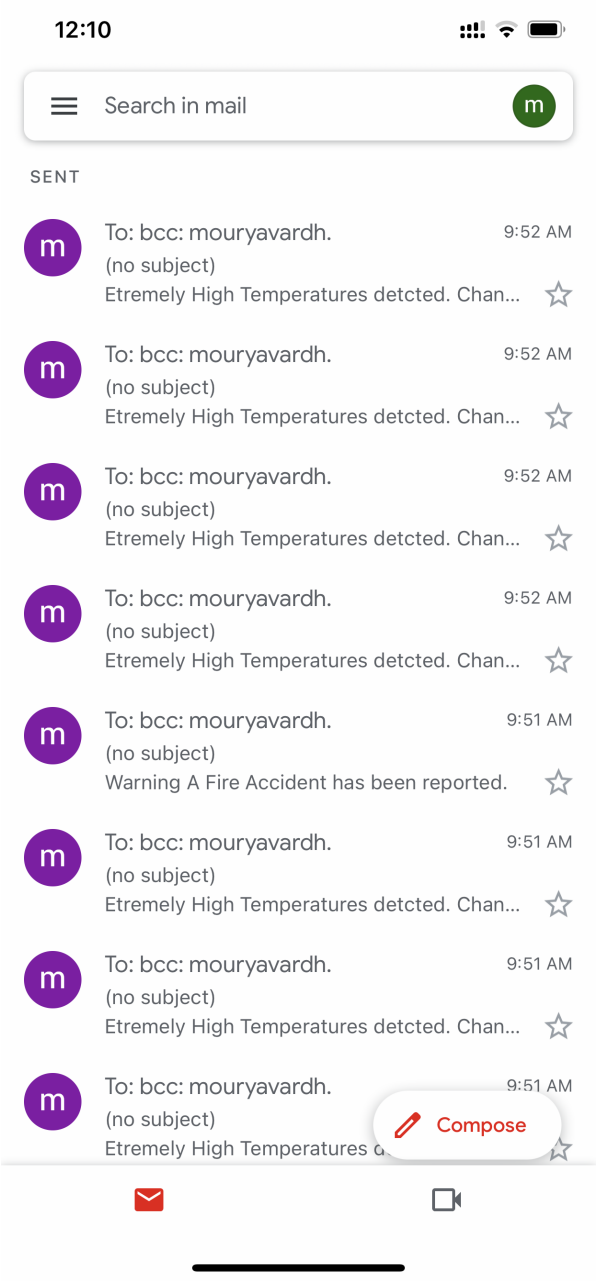

tinkercad.com
Copy of Copy of ThingSpeak cloud based temp sensor
Saved
Simulator time: 00:00:05
Code
Stop Simulation
Send To
1 (Arduino Uno R3)
// PE 22
// Mallidi prabhas
// batch 1
float val, voltage, temp;
String ssid = "Simulator Wifi"; // SSID to connect to
String password = ""; //virtual wifi has no password
String host = "api.thingspeak.com"; // Open Weather Map API
const int httpPort = 80;
String url = "/update?api_key=N7TYVWE1A38668OY&field1=0";
//Replace XXXXXXXXXXXXXXXX by your ThingSpeak Channel API Key
void setupESP8266(void) {
// Start our ESP8266 Serial Communication
Serial.begin(115200); // Serial connection over USB to compute
Serial.println("AT"); // Serial connection on Tx / Rx port to
delay(10); // Wait a little for the ESP to respond
if (Serial.find("OK"))
Serial.println("ESP8266 OK!!!");
// Connect to Simulator Wifi
Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"")
delay(10); // Wait a little for the ESP to respond
if (Serial.find("OK"))
Serial.println("Connected to WiFi!!!");
// Open TCP connection to the host:
//ESP8266 connects to the server as a TCP client.
Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPor
delay(50); // Wait a little for the ESP to respond
if (Serial.find("OK"))
Serial.println("ESP8266 Connected to server!!!") ;
}
void anydata(void) {
val=analogRead(A0);
voltage=val*0.0048828125;
Serial Monitor

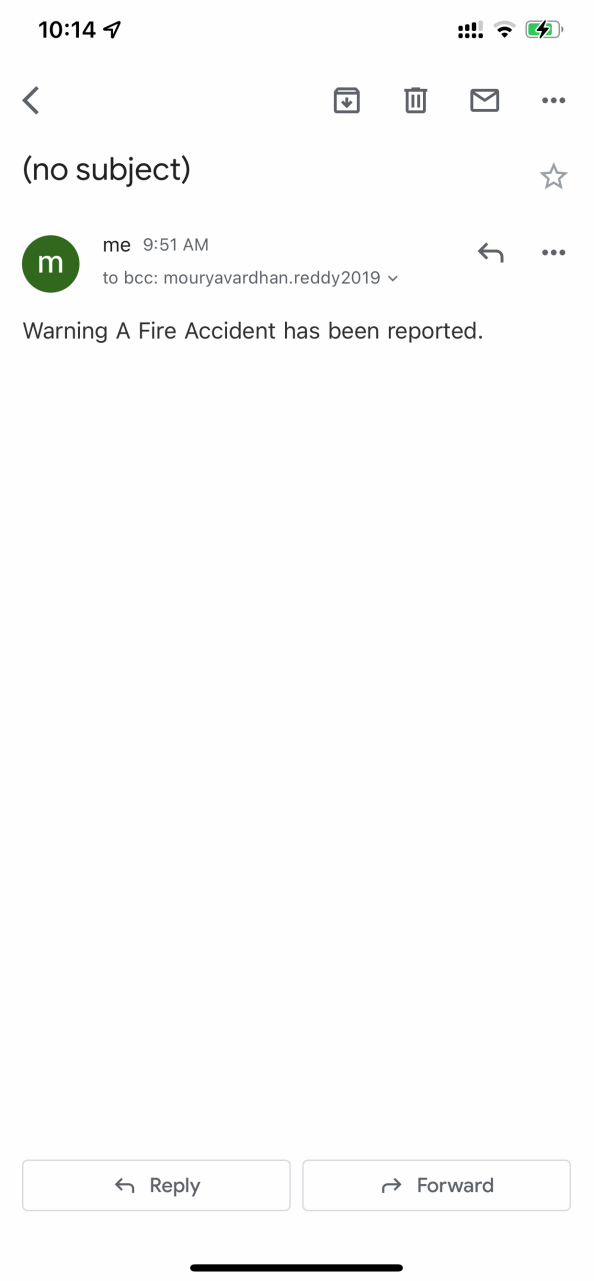
❖ ThingSpeak :


thingspeak.com
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
TARP
Channel ID: 1728567
Author: mouryamo
Access: Private
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Channel Stats
Created: about 21 hours ago
Last entry: 7 minutes ago
Entries: 13
Field 1 Chart
TARP
Field Label 1
100
50
0
18:00
10. May
06:00
12:00
Date
ThingSpeak.com

❖ Email application :




12:10
Search in mail
m
SENT
m
To: bcc: mouryavardh.
9:52 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:52 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:52 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:52 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:51 AM
(no subject)
Warning A Fire Accident has been reported.
m
To: bcc: mouryavardh.
9:51 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:51 AM
(no subject)
Etremely High Temperatures detcted. Chan...
m
To: bcc: mouryavardh.
9:51 AM
(no subject)
Etremely High Temperatures d
Compose


10:14
(no subject)
m
me 9:51 AM
to bcc: mouryavardhan.reddy2019
Warning A Fire Accident has been reported.
Reply
Forward

❖ References :

Gomes, P., Santana, P., & Barata, J. (2014). A vision-based approach to fire detection. International Journal of Advanced Robotic Systems, 11(9), 149.

Arul, A., Prakaash, R. H., Raja, R. G., Nandhalal, V., & Kumar, N. S. (2021, May). Fire Detection System Using Machine Learning. In Journalof Physics: Conference Series (Vol. 1916, No. 1, p. 012209). IOP Publishing.

Abdel-Zaher, M., Hisham, M., Yousri, R., & Darweesh, M. S. (2021, July). Light-weight convolutional neural network for fire detection. In 2021 International Conference on Electronic Engineering (ICEEM) (pp. 1-5). IEEE.