**Automatic Garage Door Opener**

**BY TEAM NO’s:**

1. Chiluvollanka Sravan Kumar

2. Chimmili Jaya Sankar

3. Chinta Moulika

3. Chintapalli Prem Shainy Kumar

4. Chintha Voshal Jaya Surya

**Table of Contents:**

1. INTRODUCTION

a. Overview

b. Purpose

2. LITERATURE SURVEY

a. Existing problem

b. Proposed solution

3. THEORITICAL ANALYSIS

a. Block diagram

b. Hardware / Software designing

4. EXPERIMENTAL INVESTIGATIONS

5. FLOW CHART

6. RESULTS

7. ADVANTAGES & DISADVANTAGES

8. APPLICATIONS

9. CONCLUSIONS

10. FUTURE SCOPE

11. BIBILOGRAPHY

12. APPENDIXES

1. Source code
2. UI output

**1.Introduction**

**a. Overview:**

The project that we are designing will help us to provide an easy and secure access to garage door. We are designing this automatic door in order to understand the basic functions of the automatic door and also detailed working of the concepts behind this.

The goal of this project is to make a sustainable modification to the garage door in order to reduce the human effort while their opening and closing and also to increase the safety of the garages. This modification is not limited to only garage door but it can be applied to several doors which are of the sliding and rolling nature. With the application of this modification the doors are tend to be more secure. This modification is not only sustainable but is also cost effective. Rather customizable standard modifications are designed to be installed with existing doors in order to make them operational as well as more secure.

**b. Purpose:**

This automatic garage door is mainly designed to reduce the human efforts and also ensures the security of the garage. These doors can also be opened in domestic as well as in industrial area because the features that are provided will reduces efforts and provide a secure system.

**2.Literature Survey**

**a. Existing problems:**

The existing problem is while car entering to an area of garage or near to gates of garage or houses a person is required to open the doors of garage. It indicates that more man power and requirement of humans plays a major role to open the doors of garage.

**b. Proposed Solution:**

The proposed system, which consists of an automatic advanced technology, which consists of modification to the garage door in order to reduce human effort while the opening and closing were also increase the safety of garage. In this project we designed an automatic gate with a proper secure system. The aim of their project was to show simply how an automatic door works.

**3.Theoretical Analysis**

**a. Block diagram:**

After running the python code, camera will detect the car and then automatically door will open and lights turn ON.

**b. Hardware/software designing:**

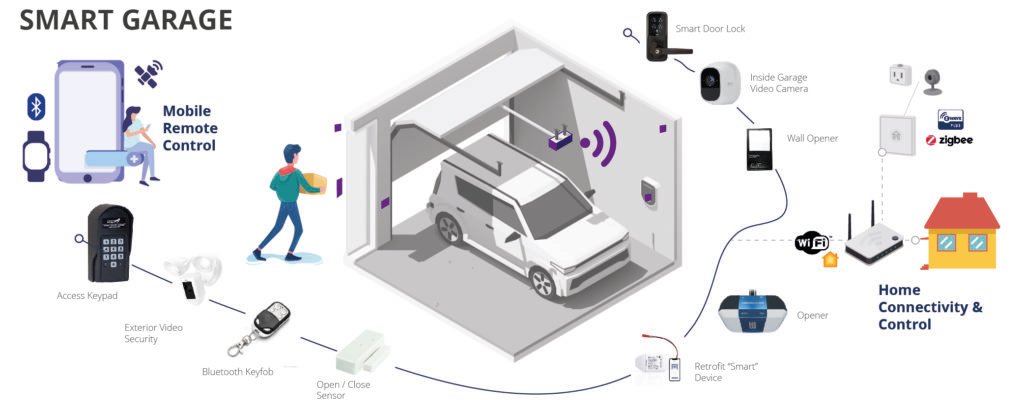
We follow a step-by-step procedure to set up all the interfaces required for our project and develop the code in python to automatically opens the door of garage. The following software is required:

* Python Idle (with specified packages installed)
* IBM cloud
* Node Red service

**4.Experimental Investigations:**

To successfully design this project, we implemented python code. Using this code camera will detect the car and then automatically door will open and lights turn ON. This design used to reduce the human efforts and also ensures the security of the garage.

The displayed data i.e., image of car is stored in IBM cloudant and using node-red services, a node-red flow is created which retrieves data from IBM cloud.



**Flow chart:**

Run the python code

Camera opens

**NO**

Check the errors and run again

**YES**

Car detects

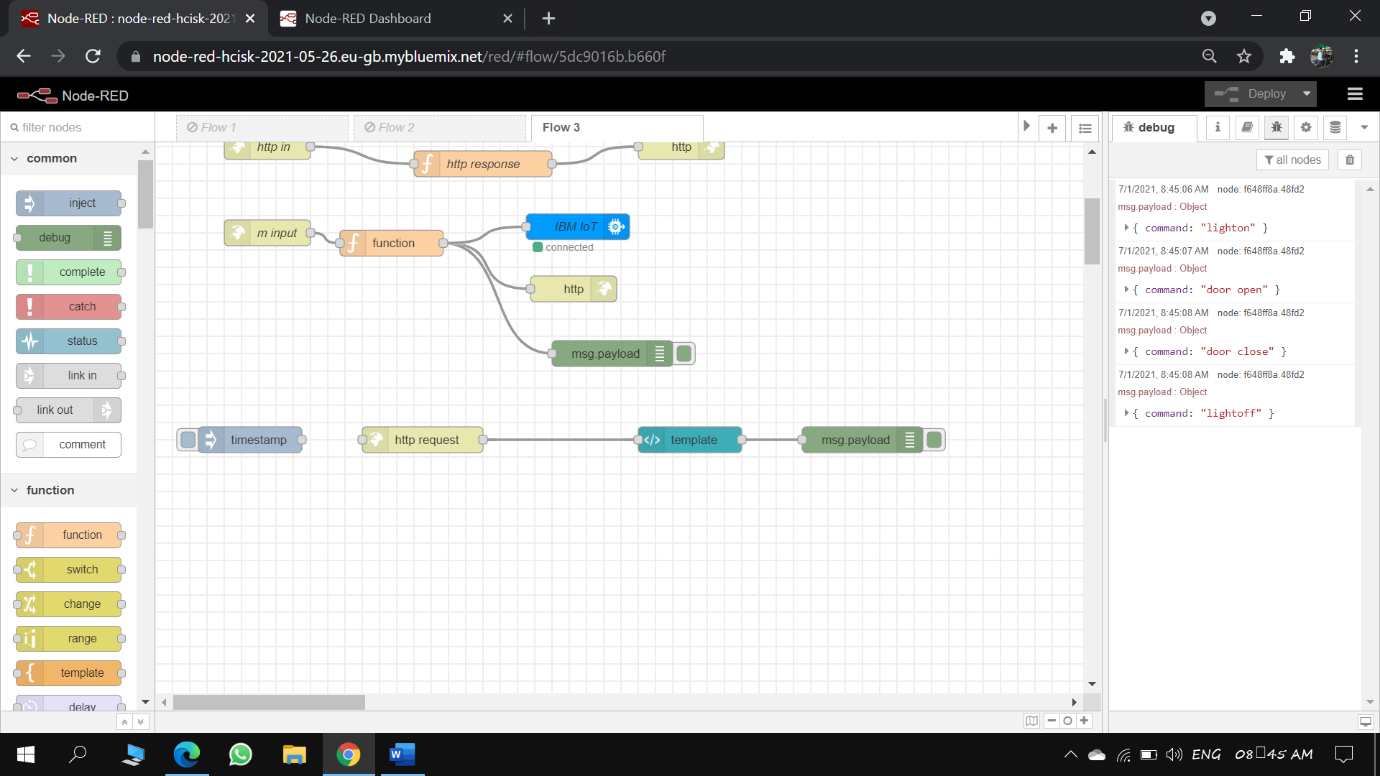
Save image in cloud object storage

Garage door opens &lights ON

Retrieve the data from Cloud ant to Node-Red using node-red services

Final result displays on Node-Red UI

**6.Result:**

****

**Explanation:**

The result of our project fulfils our requirements. We have successfully Executed the designed program and the results have been as expected. This will help to reduce human effort. The method developed proves that it is effective and efficient method to manage the time and reduce the complexity.

**7.Advantages:**

* Ease of use is at an all-time high when you choose automatic garage door openers as a simple touch of button will allow you access to your garage without having to set foot outside the vehicle; perfect for rainy days.
* Automatic garage doors do not need any added maintenance when compared to manual alternatives
* Ease of Life

**Disadvantages:**

* Cannot work without Power.
* Instalment cost is high.

**8.Applications:**

It is used at homes, offices for specified people.

**9.Conclusion:**

A project with secured system easily accessible. We come to know about programming of our code, and some essential processor and concepts. We need to do some reliability test so that the project can be used in future. We can also use the concept behind this project in toll collection and to identify a particular vehicle after some modification.

**10.Future Scope:**

* Automatic courtesy lights that turn on when the door opens and automatically.
* Door activation over the Internet to allow home owners to open their garage door from their office for deliveries.

**11.Bibliography:**

* <http://www.ijirset.com/upload/april/56_Apr_13.pdf>
* <https://www.pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/>

**12.Appendix**

**a. Source code:**

import cv2

import imutils

import numpy as np

import pytesseract

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import sys

import ibmiotf.application

import ibmiotf.device

import random

import time

from PIL import Image

pytesseract.pytesseract.tesseract\_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

img = cv2.imread('1.jpg',cv2.IMREAD\_COLOR)

img = imutils.resize(img, width=500 )

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

cnts,new = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

img1=img.copy()

cv2.drawContours(img1,cnts,-1,(0,255,0),3)

cv2.imshow("img1",img1)

cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:30]

screenCnt = None #will store the number plate contour

img2 = img.copy()

cv2.drawContours(img2,cnts,-1,(0,255,0),3)

cv2.imshow("img2",img2) #top 30 contours

count=0

idx=7

# loop over contours

for c in cnts:

# approximate the contour

peri = cv2.arcLength(c, True)

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

if len(approx) == 4: #chooses contours with 4 corners

screenCnt = approx

x,y,w,h = cv2.boundingRect(c) #finds co-ordinates of the plate

new\_img=img[y:y+h,x:x+w]

cv2.imwrite('./'+str(idx)+'.png',new\_img) #stores the new image

idx+=1

break

#draws the selected contour on original image

cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)

cv2.imshow("Final image with plate detected",img)

Cropped\_loc='./7.png' #the filename of cropped image

cv2.imshow("cropped",cv2.imread(Cropped\_loc))

pytesseract.pytesseract.tesseract\_cmd=r"C:\Program Files (x86)\Tesseract-OCR\tesseract.exe" #exe file for using ocr

text=pytesseract.image\_to\_string(Cropped\_loc,lang='eng') #converts image characters to string

print("Number is:" ,text)

cv2.waitKey(0)

cv2.destroyAllWindows()

COS\_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

#s3.us-south.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints

COS\_API\_KEY\_ID = "R3xLbCY7ncvxoAqNz924pPJC93I4Gat4Mn5ivXwxgM0b" # eg "W00YiRnLW4a3fTjMB-oiB-2ySfTrFBIQQWanc--P3byk"

COS\_AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"

COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/844ba91065754eaabbcf565c5c5b3d0e:36463f2e-ac42-4a1f-b611-dc8ff0060a0f::"

# 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

)

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:

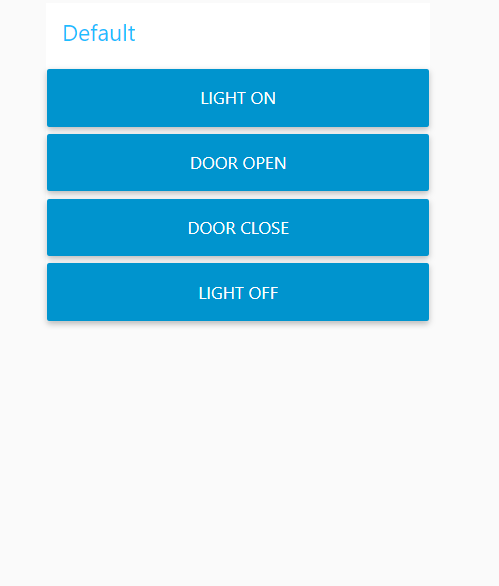
print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

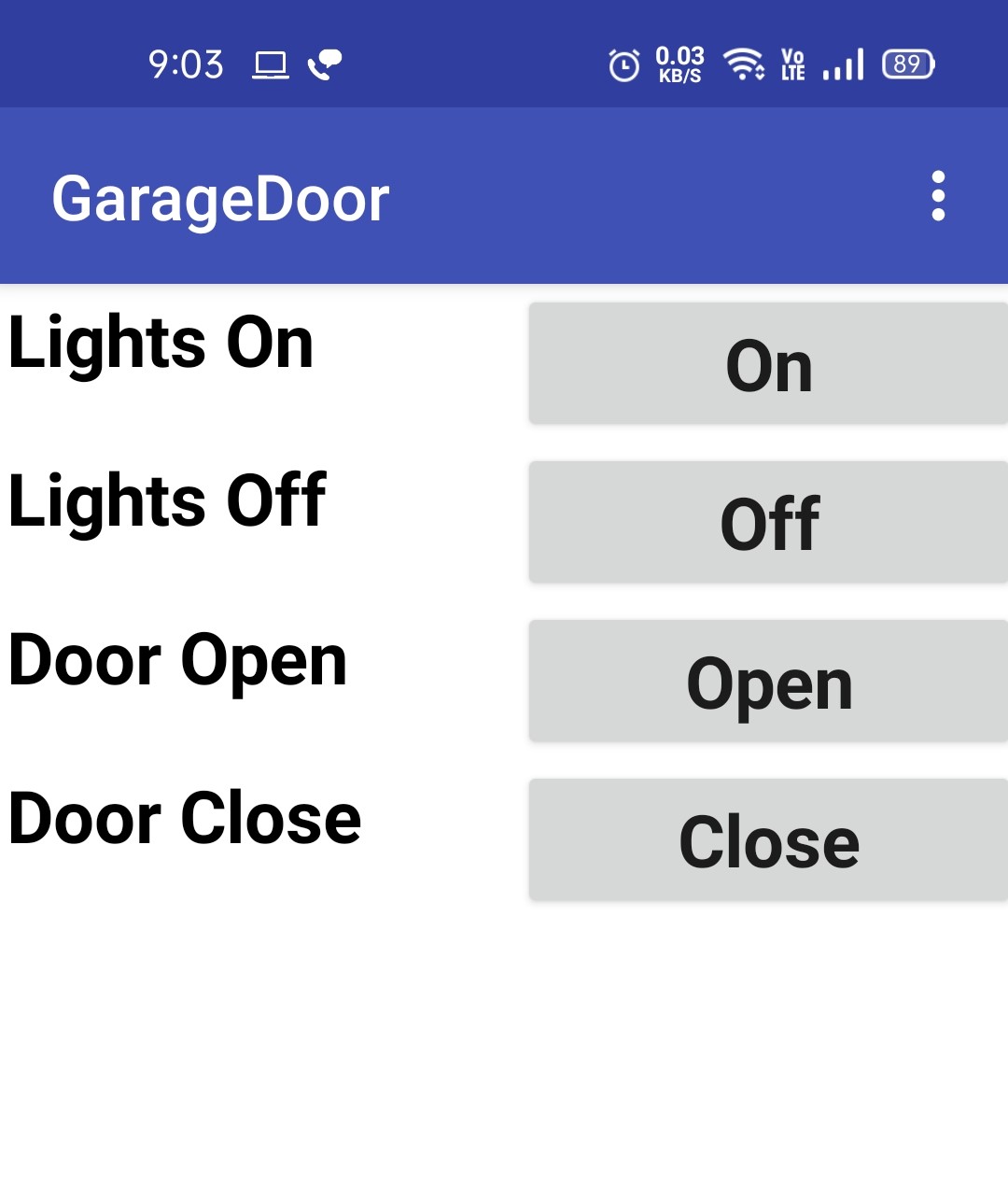
print("Unable to complete multi-part upload: {0}".format(e))

multi\_part\_upload("voshal123","car.jpg","C:/Users/vosha/Desktop/IOT/SmartInternz-IoT-Externship-2021-master/Smart Home security codes/1.jpg")

1. **UI output:**



**Mobile Application:**

****

|  |
| --- |
|  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |