Smart Bridge-RSIP

GARAGE DOOR OPENER-IOT

PROJECT

M Sai Lohith

K Eswar Sai Suhas Reddy

Pokuri Bhargav

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview

2.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 Proposed Solution

3. THEORITICAL ANALYSIS

3.1 Block Diagram

3.2Hardware / Software designing

4. FLOWCHART

5. RESULT

6. ADVANTAGES & DISADVANTAGES

7. APPLICATIONS

8. CONCLUSION

9. FUTURE SCOPE

10. BIBILOGARAPHY

APPAENDIX

1. Source code

1.INRODUCTION

1.1 Overview

Most modern houses have built-in garages that people use to store their belongings and park their cars. Oftentimes, garage doors accidentally get left open when people rush groceries in from the car or after they work outside in their yard. This poses a threat to the homeowner and leaves them vulnerable to the dangers of today’s world such as theft and vandalism of their belongings in the garage.

Up to 15% of home burglaries start from a thief entering the house via a garage Nearly half of these burglars enter through an open garage, making it clear that one of the easiest home security tips suggests keeping the garage door closed at all times the homeowner doesn’t need it open . An open garage door gives thieves an easy target; they get a clear glimpse at the unsecured belongings in the garage, making it easier for them to choose a house to rob. It seems like a simple concept to keep the garage door closed, but it becomes easy to forget once inside the house and the homeowner gets sidetracked and never thinks about the garage again until the next time they go outside.

1.2 Purpose

Building your own garage door remote is an easy and moderately priced first project for any one interested in the world of hackers and makers. Personally this modification is the most useful of the projects that I have attempted. Over the last few years it has been updated and improved so that the radio remote that came with the door is almost never used. Plus, with the added sensors, I can check the closed status of the door and open or close it from anywhere there is internet access

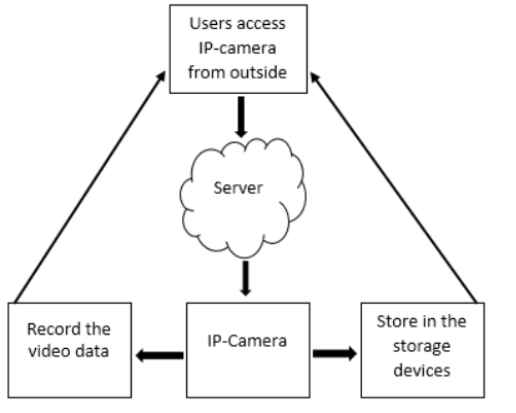
2. LITREATURE PROBLEMS

2.1 Existing Problems

* Classification of car images.
* Door Remote issues.
* Classification of Human Images.
* By using Remote we can control garage doors from inside and also from outside of the garage.
* Ensuring Vehicle Security.
  1. Proposed Solutions
* Classification of car images is done by using image classification using IBM cloud service.
* Door remote issues are solved by using NODE – REDFLOW and it is controlled by Application which is connected to Node red flow.
* Classification of Human images is done by image classification using IBM cloud.
* Ensuring of security is done by Application which shows the images of car and humans which helps the owner or user to authorize the garage.
* Remote controlling of garage can be done inside and outside of the garage when device is connected to internet.

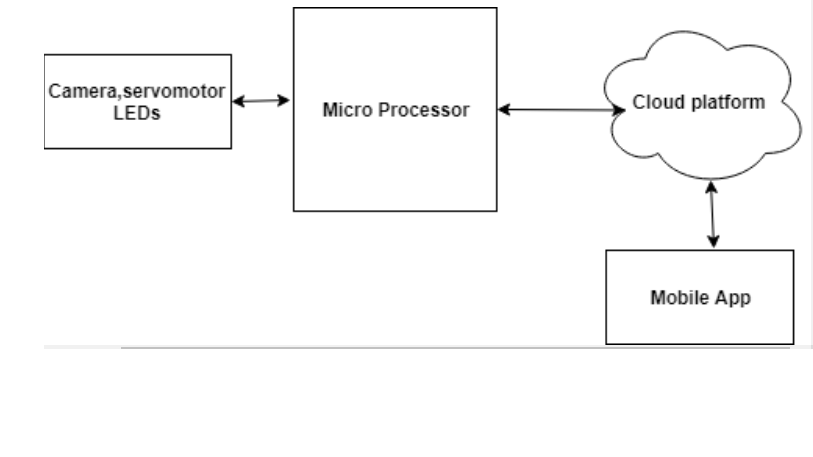
**3.**THEORITICAL ANALYSIS

**3.1**Block diagram

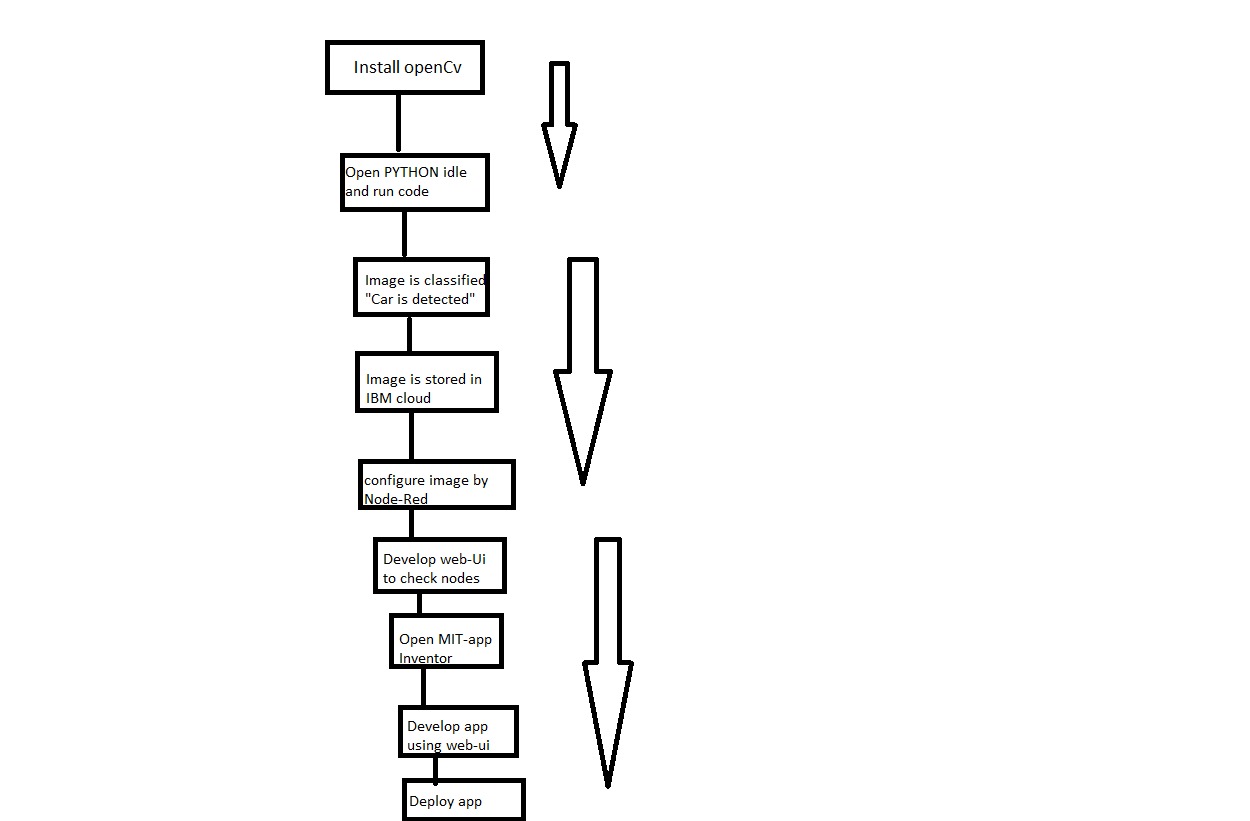


3.2 Hardware / Software Design

* Automatic opening of the garage doors by detecting the authorized vehicles.
* The garage door will be integrated with the camera which will detect the car and detect the number plate,if the number plate is recognized as the authorized one it will automatically open the garage doors to park the car.
* We can also train the authorized persons faces,based on the face recognition the garage doors will be opened whenever the person wants to take the vehicle out.
* He can also control the garage doors and lights using the mobile app.
* Everytime recognising the person or the number plate,images will be stored in the cloud along with the date and time.
* Through the mobile app admin can add the number plate details to the device.



4.FLOWCHART

5. Result

* Classification of car images is done by using image classification using IBM cloud service.
* Door remote issues are solved by using NODE – REDFLOW and it is controlled by Application which is connected to Node red flow.
* Classification of Human images is done by image classification using IBM cloud.
* Ensuring of security is done by Application which shows the images of car and humans which helps the owner or user to authorize the garage.
* Remote controlling of garage can be done inside and outside of the garage when device is connected to internet.

6.Advantages and Disadvantages

Convenience is the most obvious benefit of having a smart garage door opener. Once your opener is enabled, you never have to worry about losing your garage door remote or having it run out of batteries at an inopportune moment. You will always be able to open your garage door as long as you have access to Wi-Fi and your digital device.

An even greater benefit is security. You can set the door to open just before you get home, so there’s no time spent waiting outside for the garage door to open. As soon as you get home you can drive right in. You will also always know when your garage door is opened, even if you aren’t home, thanks to an automatic alert on your device.

You can open and close your door remotely from anywhere, so if you arrive at work or are headed out on vacation and suddenly realize you left the garage door open, you don’t have to return to close it. You can also use this feature to allow people access to your garage or home without giving them a key or passcode

There are not many disadvantages to having a smart garage door opener. There is the time and expense it takes to have it installed, but these are minimal compared to the benefits such a device can bring.

You also should not toss away your regular garage door remote entirely, as most smart garage door opener technology is designed with the idea that your primary method of opening and closing the garage door will still be the remote opener — unless you have it set on an automatic opening and closing schedule.

If you can tolerate these issues, you may want to take advantage of smart garage door opener technology for your home.

7.Applications

* Security and Safety of the car is ensured
* Authorizations of the persons or car to accesses the car can be done remotely
* It is easy to open car garage using garage opener Application
* The cost of installation is very low
* Classifications based on cars and humans are done to ensure safety
* Garage can be opened or closed from any where because it is designed base on the Internet

8. Conclusion

The Garage Door Security System, in its current state, accomplishes the goals we set and functions as required. The system is able to connect to the user’s home Wi-Fi network and send updates to a web server from the transmitter, and the receiver is able to connect to the Wi-Fi and get the updated information and display it wirelessly in the house. The transmitter is powered by a wall adapter so the user doesn’t have to worry about replacing batteries, and the receiver is powered by batteries so that it can easily be moved throughout the house.

This only needs to be done once, however the system currently doesn’t have a method for users to directly enter their IBM IOT credentials into the system.

9. Future Scope

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

One can enhance the project scope by interfacing a GSM module with it. Opening the garage door with a text message. You can also add voice recognition and speech recognition to the current soft which helps the software to increase the human interaction.Temperature control can also be added to the system.Again, we will make sure that general people can easily use this solution in their garage system . We will establish our application and make it free accessible.

10. BIBILOGRAPHY

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

* <https://appinventor.mit.edu/explore/resources>
* <https://www.ibm.com/in-en/internet-of-things?p1=Search&p4=43700053546521095&p5=b&cm_mmc=Search_Google-_-1S_1S-_-AS_IN-_-%2Bibm%20%2Biot_b&cm_mmca7=71700000066635546&cm_mmca8=kwd-303430198928&cm_mmca9=CjwKCAjw5vz2BRAtEiwAbcVILwC3Oc5BPqWAcIANacrHa5EAOx4aLCGCvxWQJA-0C-cC6RvnGRTYDxoCE9EQAvD_BwE&cm_mmca10=434009985075&cm_mmca11=b&gclid=CjwKCAjw5vz2BRAtEiwAbcVILwC3Oc5BPqWAcIANacrHa5EAOx4aLCGCvxWQJA-0C-cC6RvnGRTYDxoCE9EQAvD_BwE&gclsrc=aw.ds>
* <https://nodered.org/docs/getting-started/ibmcloud>
* <https://developer.ibm.com/recipes/tutorials/getting-started-with-watson-iot-platform-using-node-red/>
* <https://www.hackster.io/gregp228/smartphone-garage-door-remote-f9aa2c>
* <https://www.aagaragedoor.com/10-benefits-of-an-automatic-garage-door-opener/>
* <https://www.ibm.com/in-en/garage?p1=Search&p4=43700052660379186&p5=e&cm_mmc=Search_Google-_-1S_1S-_-AS_IN-_-ibm%20garage_e&cm_mmca7=71700000065068235&cm_mmca8=kwd-427574989857&cm_mmca9=CjwKCAjw5vz2BRAtEiwAbcVIL1s2RkHFg5c_qAM439jBOomIaIofFpFlK9JjOCKEQIwb1k6HwkS0zxoCvSoQAvD_BwE&cm_mmca10=428141526418&cm_mmca11=e&gclid=CjwKCAjw5vz2BRAtEiwAbcVIL1s2RkHFg5c_qAM439jBOomIaIofFpFlK9JjOCKEQIwb1k6HwkS0zxoCvSoQAvD_BwE&gclsrc=aw.ds>

APPENDIX

1.Source Code:

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import cv2

import numpy as np

import sys

from ibm\_watson import VisualRecognitionV3

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

import ibmiotf.application

import ibmiotf.device

import random

import time

import json

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey

#Provide your IBM Watson Device Credentials

organization = "x865gr"

deviceType = "rsip"

deviceId = "1001"

authMethod = "token"

authToken = "1234567890"

authenticator = IAMAuthenticator('uzbettbRqM8UJMbtvIeoo7YgWa8RtvMx6Gf0NwpiHC6h')

visual\_recognition = VisualRecognitionV3(

version='2018-03-19',

authenticator=authenticator

)

visual\_recognition.set\_service\_url('https://api.us-south.visual-recognition.watson.cloud.ibm.com/instances/c63250f3-2cef-47f8-b893-482f5475b713')

##def myCommandCallback(cmd):

## print("Command received: %s" % cmd.data)

## print(cmd.data['command'])

##

## if(cmd.data['command']=="open"):

## print("door open")

##

## if(cmd.data['command']=="close"):

## print("door close")

##

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()

#It will read the first frame/image of the video

video=cv2.VideoCapture(0)

print("videoooo")

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 = "zvX7TxWyLk-tjhH6SM1YGLqUVv\_nbUDvtxYA7F\_qDGQs" # 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/a0a6e53707df41969e9420072953f316:a940ce0b-ef90-4c36-b8f7-f1f6b6bcb09b::"

picname="sample2.jpg"

client = Cloudant("ea565ab2-700a-4181-b79d-ceeb00531bb1-bluemix", "3d20a995e83a0d24f4226afd6dfc6dcff2164191283ab002b3388a89e8255b81", url="https://ea565ab2-700a-4181-b79d-ceeb00531bb1-bluemix:3d20a995e83a0d24f4226afd6dfc6dcff2164191283ab002b3388a89e8255b81@ea565ab2-700a-4181-b79d-ceeb00531bb1-bluemix.cloudantnosqldb.appdomain.cloud")

client.connect()

database\_name = "doorbell"

# 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))

while True:

#capture the first frame

check,frame=video.read()

cv2.imshow('Video Streaming', frame)

picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")

picname=picname+".jpg"

pic=picname

cv2.imwrite(picname,frame)

picname="sam2.jpeg"

with open(picname,'rb') as images\_file:

classes = visual\_recognition.classify(

images\_file=images\_file,

threshold='0.6').get\_result()

print(json.dumps(classes, indent=2))

for i in classes['images'][0]['classifiers'][0]['classes']:

if i['class']=='car':

print("car is detected")

person=1

my\_database = client.create\_database(database\_name)

multi\_part\_upload("cloud-object-storage-dsx-cos-standard-s33",picname,picname)

if my\_database.exists():

print("'{database\_name}' successfully created.")

json\_document = {

"\_id": pic,

"link":COS\_ENDPOINT+"/cloud-object-storage-dsx-cos-standard-s33/"+picname

}

new\_document = my\_database.create\_document(json\_document)

if new\_document.exists():

print("Document '{new\_document}' successfully created.")

time.sleep(1)

t=34

h=45

data = {"d":{ 'temperature' : t, 'humidity': h, 'person': person}}

#print data

## def myOnPublishCallback():

## print ("Published data to IBM Watson")

##

## success = deviceCli.publishEvent("Data", "json", data, qos=0, on\_publish=myOnPublishCallback)

## if not success:

## print("Not connected to IoTF")

## time.sleep(1)

## deviceCli.commandCallback = myCommandCallback

person=0

#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

##deviceCli.disconnect()