### **Smart Guest Identifier With Remote Access Management(OpenCV) Internship**

Category: Internet Of Things

Skills Required:  
Python,IOT Cloud Platform ,IOT Communication Protocols

Project Description:

Features:

Whenever someone arrives at your door this smart device will detect the face and captures the images

The captured images will be sent to the admins mobile app through the cloud platform

If the admin is not in the home and if he wants to open the door, he can open the door by pressing the button in the mobile application

If the person is recognized as the authorized person the door will be opened automatically

There is one emergency button if there are any emergencies the person can press the button to send alerts to the concerned persons.

Name :Tummala Sahithi

Date of join :5/05/2020

1.INTRODUCTION

• Wireless Home security and Home automation are the dual aspects of this project. The currently built prototype of the system sends alerts to the owner over voice calls using the Internet if any sort of human movement is sensed near the entrance of his house and raises an alarm optionally upon the user’s discretion. The provision for sending alert messages to concerned security personnel in case of critical situation is also built into the system. On the other hand if the owner identifies that the person entering his house is not an intruder but an unexpected guest of his then instead of triggering the security alarm, the user/owner can make arrangements such as opening the door, switching on various appliances inside the house, which are also connected and controlled by the micro-controller in the system to welcome his guest. The same can be done when the user himself enters the room and by virtue of the system he can make arrangements from his doorstep such that as soon as he enters his house he can make himself at full comfort without manually having to switch on the electrical appliances or his T.V. channel for an example. Thus using the same set of sensors the dual problems of home security favouriteand home automation can be solved on a complementary basis. The alerts and the status of the IoT system can be accessed by the user from anywhere even where Internet connectivity may not be readily available (since it is not necessary for the mobile phone to be connected to internet only board is required to have wi-fi

Abstract

•remotely connecting and monitoring real world objects (things)

•through the Internet . When it comes to our house, this

•concept can be aptly incorporated to make it smarter, safer and

•automated. This IoT project focuses on building a smart wireless

•home security system which sends alerts to the owner by using

•Internet in case of any trespass and raises an alarm optionally.

•Besides, the same can also be utilized for home automation by

•making use of the same set of sensors . The leverage obtained by

•preferingthis system over the similar kinds of existing systems

•is that the alerts and the status sent by the wi-ﬁ connected

•microcontroller managed system can be received by the user on

•his phone from any distance irrespective of whether his mobile

•phone is connected to the internet . The microcontroller used in

•the current prototype is the TI-CC3200 Launchpad board which

•comes with an embedded micro-controller and an onboard Wi-Fi

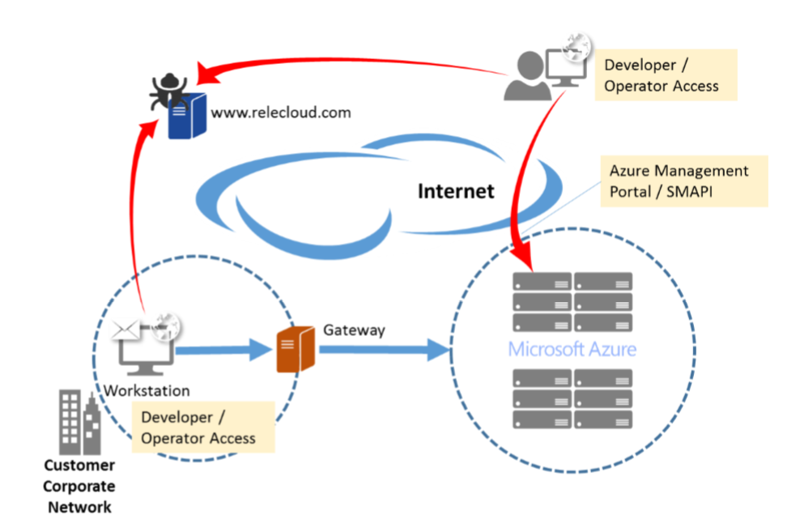
•shield making use of which all the electrical appliances inside the

•home can be controlled and managed.

2.1-Existing Problem

•The existing infra-red (IR) or Blue-tooth remote controls present in the market are in general appliance specific and the same cannot be used interchangeably. Electrical appliances connected through Bluetooth making use of Blue-tooth enabled smart phones cannot be managed from a distant location . Thus functions such as being able to turn on an air-conditioner while returning home cannot be done with such systems. In contrast, this work gives a cost effective and simple solution for wireless home automation and home security systems . The difficulty faced by current home security/surveillance systems in providing information pertaining to the situation to users while being away from home is tried to overcome in this project. The subsequent sections of the paper have been organized as follows: a comparative analysis between the proposed system and the existing solutions has been provided in section II featuring the benefits of the proposed system over the existing ones. Section III illustrates how the system has been implemented, while sections IV and V goes into greater detail about working of the individual components present in the system and the overall functioning.

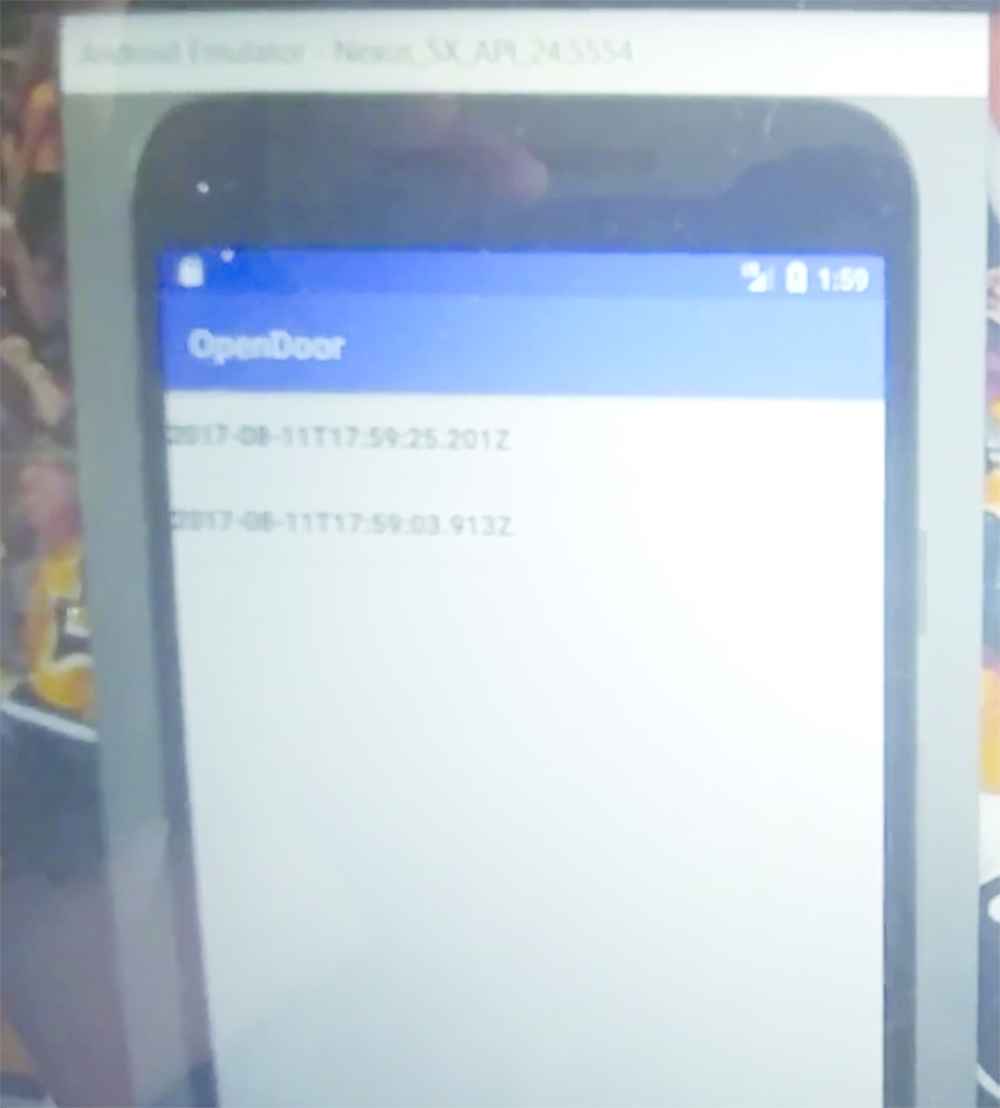
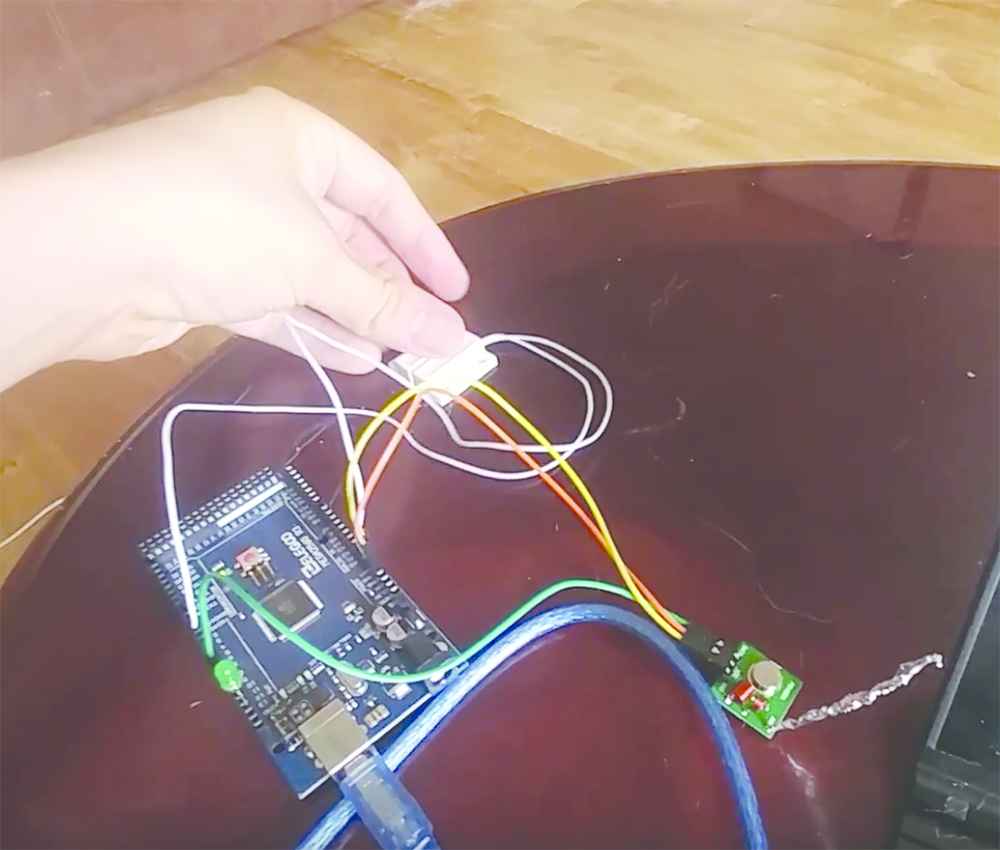
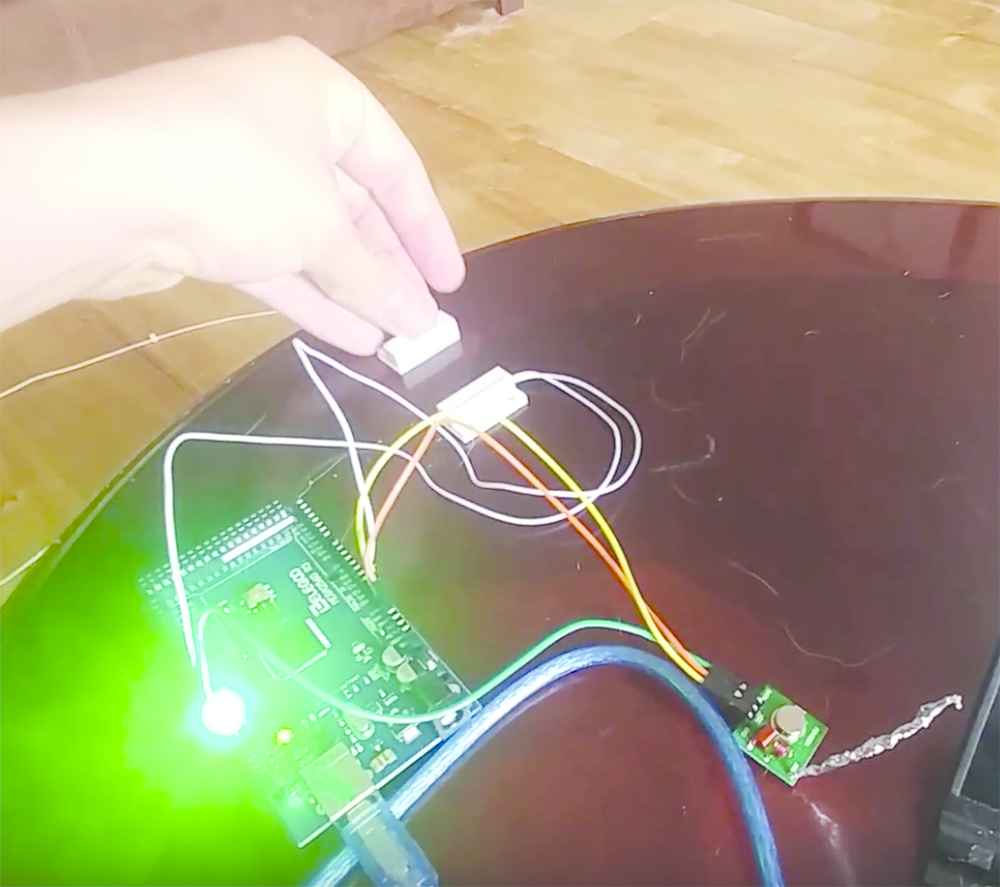
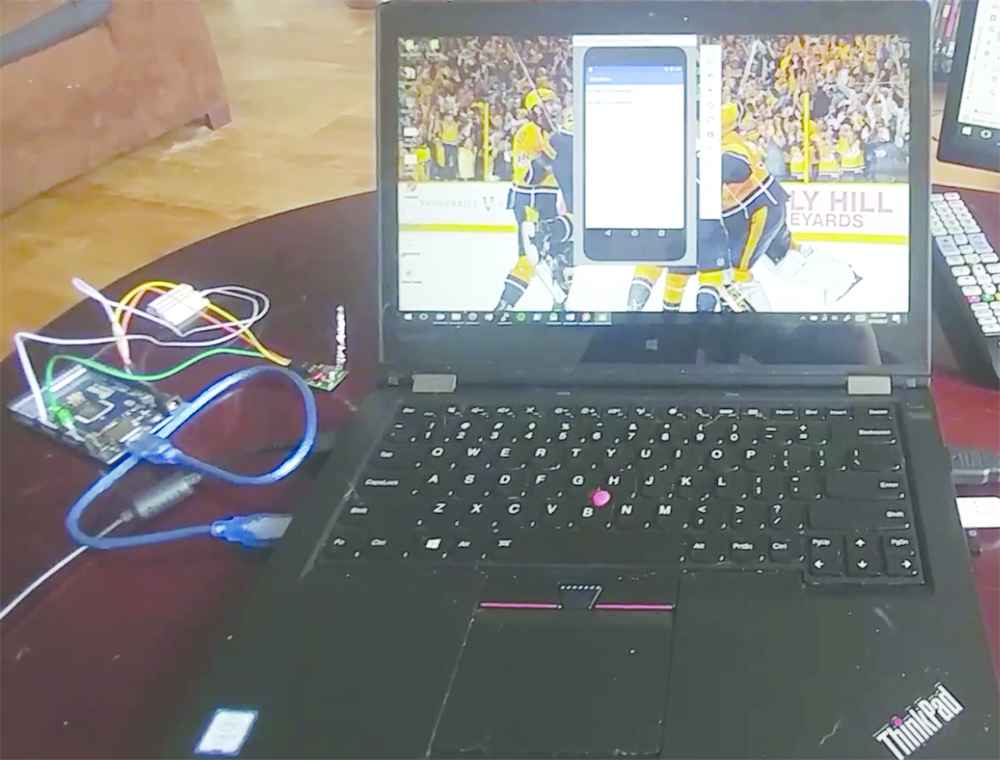
3.Theoreticalanalysis  
3.1 Block diagram



3.2 Hardware and software designing

•Software development of smart guest identifier take place through IBM cloud and it needs cloudant and object storage resources . When we execute the python code the object gets stored in object storage and a database is created . Our pictures are saved in object storage . After executing the node red flow ,we get the ui connection and we are enabled to watch who is in front of out house. Now the cloud to mobile app and enable a button that needs to be pushed in order to open the door for people if we aren’t there

•Thehardware development goes in this way through MIT app inventor we connect





4.Experimental investigations

•Prior work in IoT-enabled home security system has proposed architectures that focused on the use of low-cost open-source hardware components like the Arduino and Raspberry Pi MCU boards and a combination of sensors. Passive Infrared (PIR) sensors are used to detect motion and can work in sync with a webcam that captures images to alert users of trespassing.

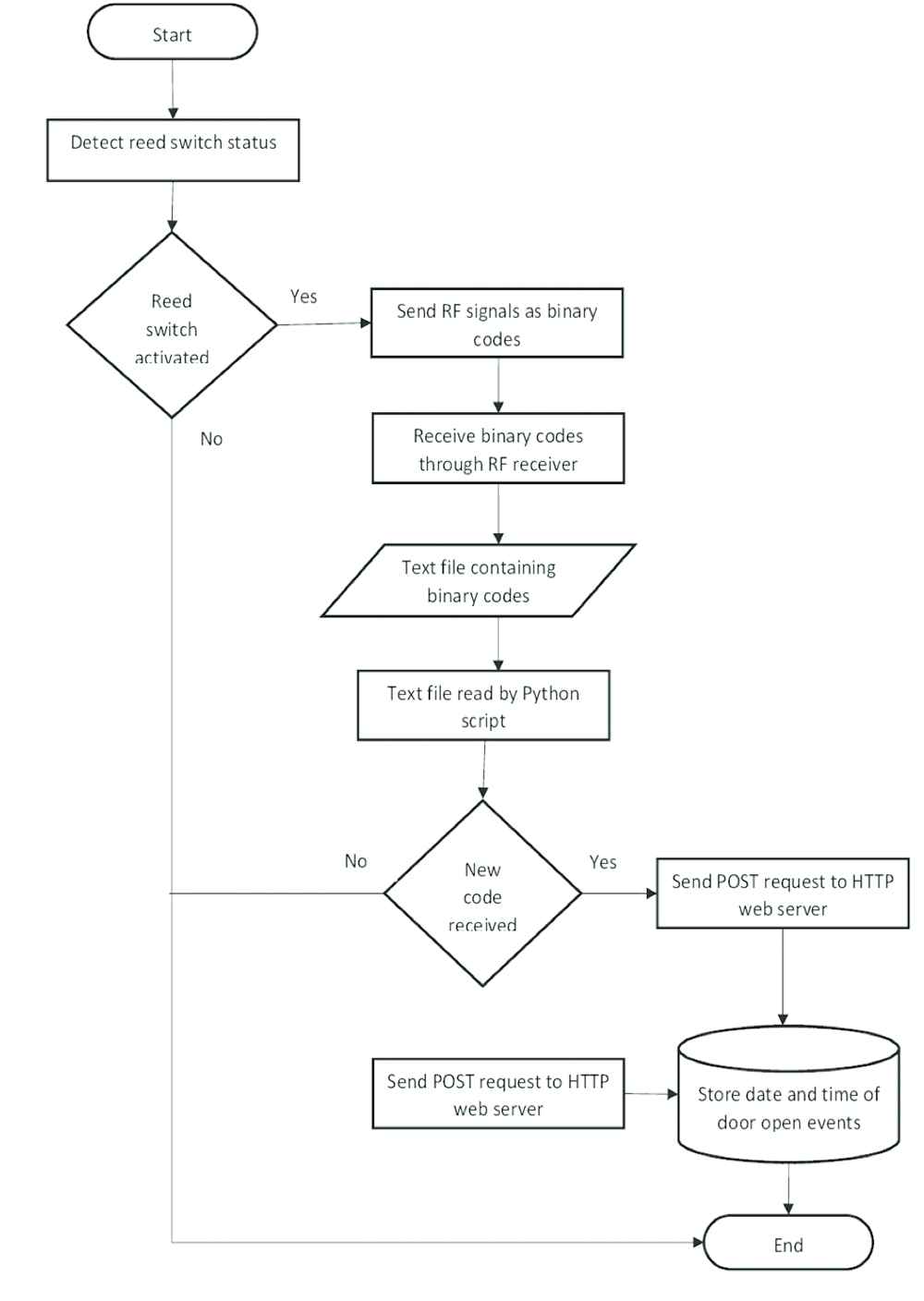
•Kodali et al. describe a cost-effective wireless home security and automation system based on the TI-CC3200 LaunchPad: a battery-powered Microcontroller Unit (MCU) with built-in Wi-Fi connectivity. PIR motion sensors are placed at the entrances to a building and connect to a digital input–output pin of the MCU. The MCU is programmed using Energia Integrated Development Environment (IDE) and Wi-Fi enabled. Kodali et al.’s configuration allows mobile phones without Internet connectivity to receive security alerts and control IoT devices connected to the microcontroller. Tanwar et al. describe an inexpensive home security system that implements a real-time email alert system. The system uses a PIR module and a Raspberry Pi MCU. Security cameras and PIR sensors are connected to the Raspberry Pi via USB ports and general purpose input/output pins respectively. The system assumes that homes have Internet access; it uses the Internet to send e-mails to the resident in real-time. The system’s intrusion detection logic identifies motion by comparing signal inputs from the PIR sensors with their previous values When current and previous signals differ, the security camera captures an image that is stored

•Gupta and Chhabra describe a cost-effective Ethernet-based smart home system for monitoring energy consumption, smoke and temperature levels and detecting trespassing. This system uses the Arduino-certified Intel Galileo 2nd generation microcontroller board. Temperature, smoke and PIR sensors are connected directly to the microcontroller, while four 220 V devices are connected via a relay module. An android based mobile app that connects to the Intel Galileo-based server over the Internet allows users to toggle switching devices by tap-to-touch or voice commands through Google API speech recognition tools.

•Piyare et al. present a Bluetooth-based home automation system where an Android cell phone running a Python script communicates with an Arduino BT board with digital and analog input/output ports to which sensors and appliances are connected. The smartphone application has a toggle on and off feature for each device. However, Bluetooth connectivity between the smartphone and the Arduino BT board required a range of 50 m or less within a concrete building and mobile platforms other than Symbian do not support the Pyhton application.

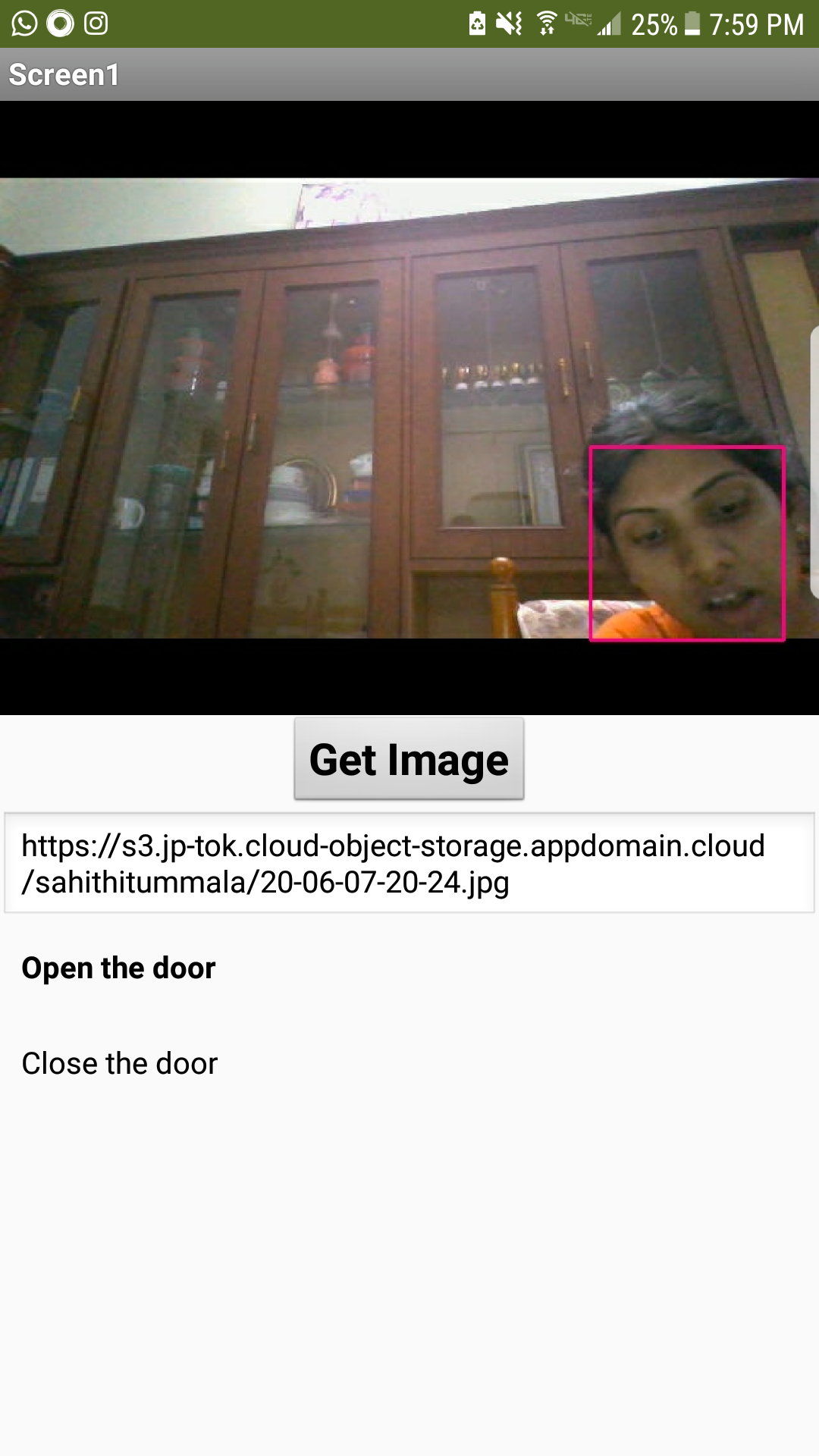
•Behera et al. designed and implemented a real-time smart home automation system using an Arduino Uno board along with an Arduino Wi-Fi Shield and a PC home server. A PIR or motion sensor, an light dependent resistor and an LM35 temperature sensor were used to collect data which was made available on the PC server that also implemented a MATLAB-GUI platform to control the temperature, lights, and fans. The PIR sensor also acted as a security component by detecting possible intrusions and setting off a buzzer to alert the residents.Howediet al. [proposed a low-cost smart home system built upon a similar architecture using the Arduino Uno board, PIR sensors, DHT11 temperature sensors, INA219 high side DC current sensor and servo motors that control doors and windows. The Arduino IDE is used to implement the control and monitoring module of the system while the MIT App Inventor is used to develop a simple Android application. •This paper presents an architecture that can be used as framework to build a low-cost smart home security system. Using affordable components such as microcontrollers from Elegoo and Raspberry Pi and RF signals as a communication channel between these devices, it was possible to develop an IoT system that allows users of a household to view when a particular door has been opened. Schematics for connecting the different components have been provided along with figures to demonstrate them. The data flow between each of these devices have been explained and potential issues that may arise have been discussed. Finally, future work in this area along with potential use cases for this architecture have also been discussed.

5.Flowchart



6.RESULT

•This paper presents an architecture that can be used as framework to build a low-cost smart home security system. Using affordable components such as microcontrollers from Elegoo and Raspberry Pi and RF signals as a communication channel between these devices, it was possible to develop an IoT system that allows users of a household to view when a particular door has been opened. Schematics for connecting the different components have been provided along with figures to demonstrate them. The data flow between each of these devices have been explained and potential issues that may arise have been discussed. Finally, future work in this area along with potential use cases for this architecture have also been discussed.

FINAL OUTPUT WILL BE

7.Advantages and Disadvantages

•Potential issues may arise through interference on the 433 Hz RF frequency. Many home devices use RF signals to communicate and at a given time there may be more than one RF receiver trying to send signals to the Raspberry Pi or it could be picking up signals that it was not intended to receive. An interference testing with the RF units can be done as a part of future work. In the case of multiple transmitters attempting to communicate with the Raspberry Pi, there would need to be a registration system in place on the Raspberry Pi that kept track of incoming signals and their sources. However, the architecture proposed here does not provide that support

8.Applications

Security in smart applications has evolved from the convergence of wireless technologies, micro-electromechanical systems (MEMS), microservices and the Internet. This concept has the potential to impact how we live and how we work. The Internet of Things is now the focus point for all the technological advancements and research. Every product or entity that measures something or performs a specific function locally has a scope of being connected to cloud. Companies are trying to enhance their efficiency through the IoT by logging maximum data, managing the execution and analysing the data. The IoT enabling this data collection or collation plays a pivotal role. With more and more applications being considered for refinement, the amount of data gets

9.Future Scope

•This design can be used as a reference for further applications to be developed with the current sensor architecture, and it provides a framework using the Raspberry Pi through which other sensors can be added to the smart home network.

•**Reed Switch Applications**

•The door sensor provides a way of seeing whether or not a door has been opened. The most obvious way to extend this application is to record the amount of time a door is open. While this may be useful to individual end users, there lies a possible business interest in knowing how often the doors to a store are opened and closed. This information can then be used by businesses to lower their energy costs.

•A second future application for this architecture could include connecting a Bluetooth module to the Mega 2560 board to identify individuals entering through the door by pairing with an end users phone. It could also be used by parents to check on a child that may try to leave home at an odd hour.

•**Further Raspberry Pi Applications**

•Since the Raspberry Pi uses an RF receiver it can not only receive the transmission from the magnetic reed switch but also be outfitted with the ability to receive transmissions from other RF-enabled devices in the household. The Raspberry Pi could be used as a hub for RF-enabled smart home devices throughout the house. The concept can be used to develop a more advanced home security system that would include PIR motion sensors placed in other areas of the house to detect intrusion while homeowners are away. Using, RF-enabled bulbs the smart home system

•The Raspberry Pi can serve as a fog computing device which can store information locally before sending it to the cloud server. This can be useful when applications might need real time support or low latency that cannot be provided by the cloud service.

•**Android Application Improvements**

•The Android application for this project, written in Java, is a basic application. Future work may involve adding time zone support from the device so as not to depend solely on Unix timestamps. Multiple options to view and display the data located in the MongoDB database, can also be added. A calendar feature can be integrated so users can have a more robust view of the door open events in their household. The system can be enhanced to display alerts to users

10.References

•N Carter, MA Hoque, and MS Ahmed, Simulating vehicle movement and multi-hop connectivity from basic safety messages, in SoutheastCon 2018, IEEE (St. Petersburg, FL, USA, 2018), pp. 1-6. <https://doi.org/10.1109/secon.2018.8479081>

•MS Ahmed, MA Hoque, and AJ Khattak, Demo: real-time vehicle movement tracking on android devices through bluetooth communication with DSRC devices, in 2016 IEEE Vehicular Networking Conference (VNC), IEEE (Columbus, OH, USA, 2016), pp. 1-2. <https://doi.org/10.1109/vnc.2016.7835956>

•MS Ahmed and MA Hoque, Partitioning of urban transportation networks utilizing real-world traffic parameters for distributed simulation in SUMO, in 2016 IEEE Vehicular Networking Conference (VNC), IEEE (Columbus, OH, USA, 2016), pp. 1-4. <https://doi.org/10.1109/vnc.2016.7835942>

•MS Ahmed, MA Hoque, J Rios-Torres, and A Khattak, Demo: freeway merge assistance system using dsrc, in Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (Snowbird, UT, USA, 2017), pp. 83-84. <https://doi.org/10.1145/3131944.3131957>

•D Jordan, N Kyte, S Murray, MA Hoque, MS Ahmed, and A Khattak, Poster: investigating doppler effects on vehicle-to-vehicle communication: an experimental study, in 2nd acm International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (2017), pp. 77-78. <https://doi.org/10.1145/3131944.3131959>

•M Hoque, X Hong, and B Dixon, Innovative Taxi hailing system using DSRC infrastructure, in ITS America 22nd Annual Meeting & Exposition, ITS America (https://www.itsa.org/) (National Harbor, MD, USA, 2012), pp. 4.

11.APPENDIX:

1. import cv2
2. import numpy as np
3. import datetime
4. #ObjectStorage
5. import ibm\_boto3
6. from ibm\_botocore.client import Config, ClientError
7. #CloudantDB
8. from cloudant.client import Cloudant
9. from cloudant.error import CloudantException
10. from cloudant.result import Result, ResultByKey
11. import requests
12. face\_classifier=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")
13. # Constants for IBM COS values
14. # Constants for IBM COS values
15. 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
16. COS\_API\_KEY\_ID = "0zyizLBNVwxjZti0X0hzmlWGSIDdAnNSc9eEPQF0Npz-" # eg "W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"
17. COS\_AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
18. COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/17e1a34a6d5c4d0b85fab4bae65dc411:0fbaf3f0-6258-4dd4-ba62-696148646ea7::" # eg "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"
19. #
20. # Create resource
21. cos = ibm\_boto3.resource("s3",
22. ibm\_api\_key\_id=COS\_API\_KEY\_ID,
23. ibm\_service\_instance\_id=COS\_RESOURCE\_CRN,
24. ibm\_auth\_endpoint=COS\_AUTH\_ENDPOINT,
25. config=Config(signature\_version="oauth"),
26. endpoint\_url=COS\_ENDPOINT
27. )
28. #Provide CloudantDB credentials such as username,password and url
29. client = Cloudant("62b4635a-df31-4ec6-80e3-bb8138800224-bluemix", "bc2de9c4fb270a9eeb74c2d97b6541265972cc745a56e02d01e436fb5f1277a4", url="https://62b4635a-df31-4ec6-80e3-bb8138800224-bluemix:bc2de9c4fb270a9eeb74c2d97b6541265972cc745a56e02d01e436fb5f1277a4@62b4635a-df31-4ec6-80e3-bb8138800224-bluemix.cloudantnosqldb.appdomain.cloud")
30. client.connect()
31. #Provide your database name
32. database\_name = "sample"
33. my\_database = client.create\_database(database\_name)
34. if my\_database.exists():
35. print("'{database\_name}' successfully created.")
36. def multi\_part\_upload(bucket\_name, item\_name, file\_path):
37. try:
38. print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))
39. # set 5 MB chunks
40. part\_size = 1024 \* 1024 \* 5
41. # set threadhold to 15 MB
42. file\_threshold = 1024 \* 1024 \* 15
43. # set the transfer threshold and chunk size
44. transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(
45. multipart\_threshold=file\_threshold,
46. multipart\_chunksize=part\_size
47. )
48. # the upload\_fileobj method will automatically execute a multi-part upload
49. # in 5 MB chunks for all files over 15 MB
50. with open(file\_path, "rb") as file\_data:
51. cos.Object(bucket\_name, item\_name).upload\_fileobj(
52. Fileobj=file\_data,
53. Config=transfer\_config
54. )
55. print("Transfer for {0} Complete!\n".format(item\_name))
56. except ClientError as be:
57. print("CLIENT ERROR: {0}\n".format(be))
58. except Exception as e:
59. print("Unable to complete multi-part upload: {0}".format(e))
60. #It will read the first frame/image of the video
61. video=cv2.VideoCapture(0)
62. while True:
63. #capture the first frame
64. check,frame=video.read()
65. gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)
66. #detect the faces from the video using detectMultiScale function
67. faces=face\_classifier.detectMultiScale(gray,1.3,5)
68. print(faces)
69. #drawing rectangle boundries for the detected face
70. for(x,y,w,h) in faces:
71. cv2.rectangle(frame, (x,y), (x+w,y+h), (127,0,255), 2)
72. cv2.imshow('Face detection', frame)
73. picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
74. cv2.imwrite(picname+".jpg",frame)
75. multi\_part\_upload("sahithitummala", picname+".jpg", picname+".jpg")
76. json\_document={"link":COS\_ENDPOINT+"/"+"sahithitummala"+"/"+picname+".jpg"}
77. new\_document = my\_database.create\_document(json\_document)
78. # Check that the document exists in the database.
79. if new\_document.exists():
80. print(f"Document successfully created.")
81. r = requests.get('https://www.fast2sms.com/dev/bulk?authorization=OMyK5jnSDx9CG40kTNihZ6szEpYRqBPJaQAdr7v1bHg2cmLfoUgiV2jnM75hLRKcC6QAS9ePqOWBJ3dy&sender\_id=FSTSMS&message=Some one at door&language=english&route=p&numbers=9030644234')
82. print(r.status\_code)
83. #drawing rectangle boundries for the detected eyes
84. #waitKey(1)- for every 1 millisecond new frame will be captured
85. Key=cv2.waitKey(1)
86. if Key==ord('q'):
87. #release the camera
88. video.release()
89. #destroy all windows
90. cv2.destroyAllWindows()
91. break

THANKING YOU