Safe Nest Smart Lock Using OMNET++ Simulation, Wireless Sensors and AWS

1st Snigdha Kakkar
MEng ECE
University of Waterloo
Waterloo, Canada
s2kakkar@uwaterloo.ca

2nd Shraddha Digambar Dhole
MEng ECE
University of Waterloo
Waterloo, Canada
sddhole@uwaterloo.ca

3rd Akash Kadiri
MEng ECE
University of Waterloo
Waterloo, Canada
akadiri@uwaterloo.ca

***Abstract*—The Internet of Things(IoT) has revolutionalized the world and quality of life around us and it is bringing a revolution in each field where we never thought it is possible. One such field where automation and IoT have brought about considerable changes is home security. Across the globe, personal and home security has been a major concern in every household. Home security systems are a useful addition to today's homes where most of the family members go out of the house for their daily activities at the same time. Vision-based security systems have the advantage of being easy to set up, inexpensive and unobtrusive. The traditional home security systems, i.e., closed circuit television (CCTV) can only capture the video without being able to give any warning feedback if there is any suspicious object.**

Our project Safe Locking System (Safe Nest) aims to design and implement a home security system with the capability for human detection with additional functionalities such as Smart weather information and suggestions, garage door open notification. It is developed using sensors to detect any security violation and sends out the alert signal to your mobile device. Our project has following features a) Burglary detection, b) Automatic Main-Door and Garage-Door opening mechanism, c) Auto Light Turn On/Off d) Weather sensor e) Garage door open alert. Our project utilizes Omnet++ as a simulation environment to simulate sensor nodes, data transmission to local controller and to the local station. We have used Python3 for logic implementation and would be manually feeding certain input data to depict the functioning of the Safe Nest App. If any abnormality is detected, then alerts are sent via SMS, or events are triggered using AWS cloud instance.

***Index Terms*—Smart lock, Facial recognition, Internet Of Things, IOT, Wireless communication, Omnet++, Smart home, Open door alert, weather sensor, accuracy, sensor nodes, Python, AWS SNS**

I. Introduction

Internet of Things (IoT) is changing the ways people work and carry on their daily activities. The IoT has brought about a revolution of automation not only in enterprises but also in our day-to-day activities. [1] So, the basic idea behind IoT is connecting devices and collecting data from them and making the data useful in taking actions. These devices are heterogeneous sensing devices in a distributed system from which information is gathered. With Wi-Fi technology coming to our homes and being used in everyday activities, this IoT has become even more useful. It has become easier to complete our daily tasks using IoT-based automation. [2] Here, our main focus is the IoT-based applications in our home security.

The aim of this project is to devise a Safe Nest Security System for homes that features a combination of user-friendly attributes, such as known face detection and recognition, garage door open alert, weather recommendations and alert, and light auto turn ON/Off based on motion, that lead to a creation of a Safe Nest for the home residents.

A. Introduction to problem and Motivation

Traditionally, home automation and security systems primarily worked on mechanized processes where human intervention and support are required to accomplish certain tasks. There were various approaches using which the home security systems were connected and automation was executed, such as Power line based, Wired based, and Wireless. [3] One such example of a home security system is closed circuit television (CCTV).

It is not possible to secure our home with CCTVs, because someone has to constantly monitor the feed of the CCTV and watch for suspicious activity. The CCTV system is also static in nature and cannot react in real-time. In the event of a burglary, this real-time reaction is very critical not only in saving the goods from being stolen but also the life. The CCTV systems are also not interconnected to each other and cannot communicate to different nodes which is disadvantageous.

In the modern era of hyper-connectivity, the ways in which people's privacy can be invaded have changed drastically in the past few years, thus requiring us to change the ways we protect our privacy and ensure our home security. This is the foundation for us to come with our Safe Nest system which is enabled by fast and reliable internet connections and easy access to IoT devices.

This system aims to secure our homes as well as provides additional features that make our lives better such as garage door open alert, motion sensors, and weather alerts which were previously not possible. The proposed solution of the Safe Nest is mainly wireless because the power line-based and wired-based solutions induce a huge amount of complexity

in connection of all the sensors nodes to the centralized controller and also require some of additional converter circuits. With the advent of technologies such as Bluetooth, Wi-Fi, and Zigbee, the problems posed by power line-based and wired solutions have been resolved. Additionally, there is an advantage of operating wireless devices remotely which was earlier not possible.

B. Goal and objectives of proposed solution

The goal of our project is to achieve the above-mentioned features using simulation techniques rendered by Omnet++ and using Python Machine Learning libraries, such as face_recognition, scikit learn to carry out image recognition tasks such as face detection and recognition.

The behavior of the overall network which consists of different types of sensors such as motion sensors, camera sensors, magnetic sensors and weather sensors, etc, has been simulated using Omnet++. The following attributes like the range of each sensor, packet size for each sensor, communication medium, routing protocols, etc. have been clearly outlined and demonstrated in Omnet++ simulation. Four types of sensors have been simulated in omnet++ and python.
These sensors are :

1) Motion sensors- PIR sensors
2) Camera sensors
3) Magnetic sensors- PIR sensors
4) Weather Sensors - measuring precipitation, humidity, temperature

Another important objective for the system is to have the capability to detect the face of the owner of the home and send notifications in case the unknown face is detected. This is achieved using the cloud services of AWS in particular we are using SNS - Simple Notification Service. For example, a message like ”Unauthorized access detected at the main door” is prompted when an unauthorized person tries to access the door. This use of a plethora of IoT-enabled features results in enhanced home security and easy accessibility at a reduced cost and complexity.

II. Literature Review

The area of automation that has seen the quickest growth is home automation and security, which has had a significant impact on people’s lifestyles both directly and indirectly. Various technologies have been used to accomplish the home automation idea and approach. The monitoring and control of lights, fans, overhead entertainment systems like TVs, and other appliances are no longer the only focus of modern home automation systems. Today’s home automation and security systems operate increasingly complex systems including unauthorized entry detection, gas, and smoke detection, autonomous irrigation systems, and much more in addition to lighting and fan control. Numerous relevant papers and works were considered in the context of our implementation, a few of them have been briefly discussed below along with a comparison to what we want to implement and its benefits over the current system.

The paper:“Facial Recognition Based Smart Door Lock System” [4] focuses primarily on the implementation of access control facial recognition systems and can be extended to a broad circumference spectrum. Using facial recognition, motion sensor-based control systems, and others that can allow efficient access control can be utilized, so that security breaches can be solved. The main idea of the paper is to design a facial recognition security system that can be effortlessly connected with a smart home system using a Raspberry Pi. The relay circuit that manages a door’s magnetic lock was coupled to the output of the facial recognition algorithm which helps these intelligent devices scan and recognize faces, preventing access from unknown or unauthorized people.

Recent major research initiatives have primarily focused on video-based face modeling/tracking, identification, and system integration. Based on these databases, new datasets have been created, and analyses of recognition tactics have been conducted. It is not an exaggeration to say that one of the most popular applications of pattern recognition, picture analysis, and comprehension is face recognition. But, the main drawback of a typical door lock is that anyone can open one by making a copy of the key or stealing one, making it impossible for friends and relatives to visit us without physically being there. This referred paper tries to address these problems. In order to effortlessly convert this standard door lock into a smart lock that can open the door whenever we arrive in front of the gate or want it to open for anyone else without being physically present, we must update the door.

The hardware components employed in the existing design of the research paper we referred to, include Raspberry Pi-3Model B+, Raspberry camera, electric solenoid lock, door, storage card, relay, voltage regulator, and screen for display while the software components include Raspbian OS, OpenCV/Facial Recognition Libraries, Python, and WIFI. Raspberry pi serves as the main device controller in the solution proposed in the paper. Raspberry pi configures the camera to capture and store the image. Sensors are also directly linked to the raspberry pi which also has a plug-and-play USB interface that can be equipped with door motion. The Raspberry Pi camera is automatically opened by code, which then interacts with a live video feed to provide facial recognition-based real-time conversation. The system’s functioning is as follows: A face’s live video feed is captured by the video capture equipment. Python programs working together to enable recognition, detection, and lock activation can recognize the face from the video feed, compare it with

what is in the database, and decide whether to trigger the lock to open or remain locked.

Another research paper, “Smart Door with Facial Recognition” [5]focuses primarily on the implementation of a door locking system using facial recognition. The major concept is to make life easier and to convert normal doors into smart doors which help to open immediately when the authorized person is in front of the door. It does not only make the system easier but makes it more secure than usual, as we see a lot of cases related to crime happening to the elderly and young children this can a problem solver, one of the main attractions of the product is that the face detection is enabled in it and the fingerprints which can help to not enter the unknown person into the household unless and until the permission is allowed.

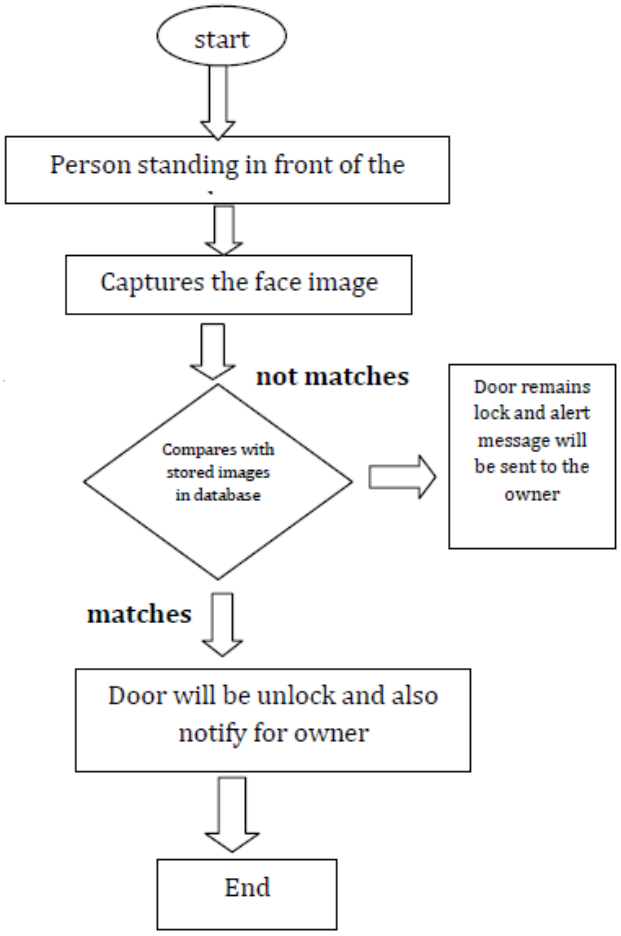

Fig. 1. Flow Diagram of existing Design from the referred Research Papers

The hardware components employed in this referred paper’s design include a set of electromagnetics like ESP32 CAM (for capturing face), UART TTL programmer, Relay module, Solenoid lock, LED (red and green), Breadboard (for connections), power supply and the software components include C Sharp and WIFI. As per this design, say that a person has been authorized to control the lock system, the person is standing in front of the door, the camera recognizes the person using the HD camera installed and the data will be saved in the cloud-computing, and the program has been created using the set of electromagnets utilized here, which will be controlled by a Raspberry Pi and microcontroller so the app now uses the data to compare and allows the person, there might be few affecting scenarios to this case regarding the paper, one being the disadvantage of the theft or lost case of the phone so the phone should be secured if in case of these scenarios the other case regarding the paper is to allow the elderly and younger people to be authorized by the remote control with high security.

For our project, we had proposed a new design based on the existing system designs, considering the security and dependability of the sensors and their performance. In addition, to the existing work we tried implementing weather sensors to measure the precipitation, humidity, and temperature of the external weather and send alerts. Also, we added more functionalities such as electrical devices control such as lights/fans using motion sensors, Door open/close detection using magnetic sensors helpful for garage doors. Also, our suggested method removes the use of GSM module and suggests using the AWS SNS system to establish SMS cloud triggering in place of the GSM module.

Furthermore, in order to extend this project in real-time we would advocate to enhance the functionality of open garage door alert to open window alert across the house perimeter. This would be especially useful during the time of a snow storm or a wind storm. Additional functionality of automated audio welcome to home and coffee brewing would be a nice touch when an authorized visitor gets in the main door. The weather alerts could be paired with the internal thermostat as well to regulate the temperature inside the house. The next parts cover the code, simulation results, and design in detail.

III. Methodology Design and Implementation

We have used 1 PIR motion sensor for detecting motion and turning the light ON/OFF, 1 video camera for unauthorized person detection through the video, 1 magnetic PIR sensor for detecting if the garage door is open or closed, 3 weather sensors for measuring the precipitation, humidity and temperature of the external weather and sending alerts accordingly. The locations of the sensors are quite far away and thus they cannot directly communicate with the local controller. Hence, we have placed 4 routers (access points) that transmit the data from each of the sensors to the controller.

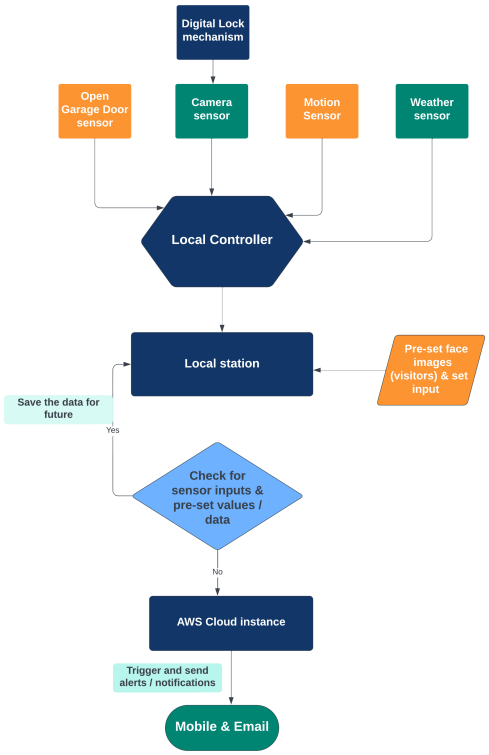

Fig. 2. Safe Nest Flow Diagram

For this project, all the sensors' data was wirelessly transmitted to a controller that is in turn connected to a local station (our desktop). The logic implementation on the basis of the sensors' data has been done in the local station i.e. our desktop in Python3. If there is any kind of abnormality or alerting situation, the core logic triggers a notification from the AWS SNS service in the cloud to our registered phone number as an alert SMS.

A. Omnet++ Simulation

Omnet++ simulation environment [6] was chosen to simulate the sensor network for the entire Safe Nest system. 1 Motion (PIR) Sensors, 1 Camera Sensor, 1 Magnetic (PIR) Sensor and 3 Weather sensors (Humidity, Precipitation and Temperature) were distributed in a simulation area of 500x500 units with a simulation time of 900 seconds. These sensors pooled the data collectively to a wireless controller, that primarily served the purpose of a gateway for sensor data. This controller then passed on the data to a local PC. This further triggered relevant triggering events to the AWS SNS cloud. Each sensor has a limited range of 100 units thus 4 access points were placed in the simulation plane to enable data forwarding to the wireless controller/gateway. The sensor nodes and access points functioned as AODV routers, therefore data hopping could take place between the sensors nodes themselves as well as the access points. IEEE802.11 has been selected as the MAC layer i.e., the nodes communicate over WiFi with the Wireless Controller. The controller shared connection with the local PC over a physical ethernet cable that enabled them to share data.

Every sensor has varied transmit times and different packet sizes. The transmit time for the motion sensor and weather sensors are fixed at 10s each while the packet size is 64 bytes each. This has been taken this way because these particular sensors would transmit small amounts of data at small regular intervals. The transmit time for camera and PI sensor is 50s and 100s each while their packet size is 1024 bytes and 512 bytes respectively. This is due to the fact that these two sensors would send large amount of data not as frequently as the other 2 sensors.

The figure here, displays the simulation environment and set up of Safe Nest system in Omnet++. The sensors have been simulated using Omnet++ and there is a circle around each of them depicting their ranges. From this figure, it is also evident that the sensors cannot communicate directly and thus 4 access points (router nodes) have been positioned in between the sensors and the controller.

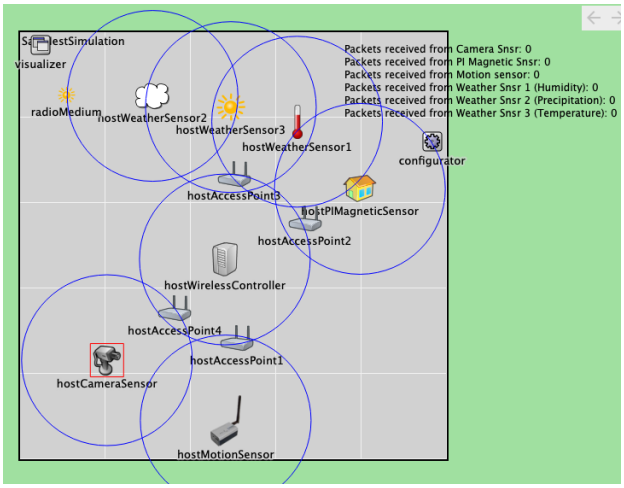

Fig. 3. Omnet++ Simulation Snapshot

B. Burglary detection and automatic door opening for authorized visitors

The camera sensor is located at the door of the house/residence. It is basically used to detect any attempt of unauthorized access at the door. The camera records the video of the person trying to open the door and matches with the stored faces. [7]

If there is an unauthorized person trying to access, it sends a notification to the local station. [8] The local station further notifies the home owner by triggering an alert notification via AWS SNS service to the owner's phone number. The figure above depicts the data transmission from the camera sensor to the controller using the access point as the range

Fig. 4. Unauthorized Person Detection Process Flow

Fig. 5. Garage Door Open Alert Process Flow

of camera sensor is limited to 100 units. Additionally, it also shows the notification alert flow from the local station to the user's phone in case there is an unauthorized access attempt. The core logic is written in Python and the video data transmission is being shown in Omnet++.

AWS utilizes SNS service configuration to send an alert notification. To enable it, AWS command line interface has been downloaded and it's access key, secret access key and location were configured. Following this, BOTO3 python library was installed in Python3. [9] If the face does not match the faces stored in the database, then a push notification - "Alert: Unauthorized access on the door!" is sent to the phone. On the other hand, if the face matches, then the door is unlocked and a message is displayed - "Door unlocked, access granted!" [10]

C. Garage Door Open Alert

1 Magnetic PIR sensor [11] has been used to check whether the garage door is open or not. This sensor has a range of 100 units so it cannot communicate directly with the controller and thus an access point (router) is used in between. This data flow has been simulated and shown using Omnet++.

Locally generated data has been fed into the python code (Local station) for the Raspberry PIR magnetic sensor. The sensor would basically take in GPIO input value. If this pin value is 12, it would print a message "Garage door is closed!", else it would send a notification to the user that the garage door has been left open.

The figure here depicts the data transmission of the PIR sensor to the controller via router and to the local station. Additionally, it also depicts the message request being sent to AWS SNS service and an alert message of the door being left open has been sent to the user.

D. Auto Light Turn On/Off

1 motion PIR sensor [12] has been used to detect the motion and turn the LED ON/OFF accordingly. Sensor implementation has been done using Omnet++. The sensor transmits the data to controller and further to the local station. The local station triggers the LED ON/OFF as per the core logic and sensor inputs.

The figure here depicts the data transmission of the motion PIR sensor to the controller via router and to the local station. The local system takes in the old PIR value (i.e. previously if the motion was detected or not), current PIR value (i.e. at present if the motion has been detected) and LED value (Light is at present ON or OFF) from the user. It prints a message – "Motion detected!" if the old PIR sensor value was False whereas the current PIR sensor value is True,

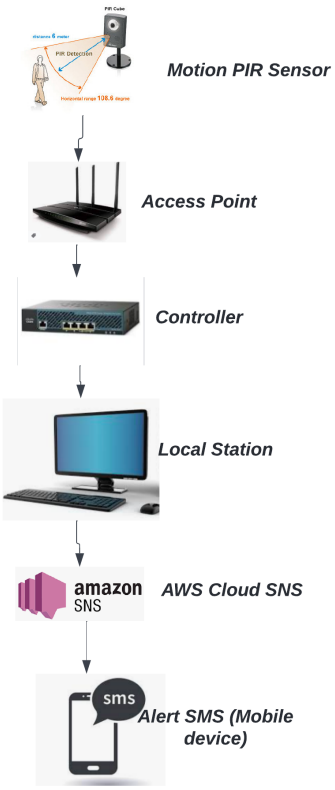

Fig. 6. Auto Light Turn ON/OFF Process Flow

and LED was earlier turned Off. It further turns ON the LED as the motion has been observed. On the contrary, it prints a message "Motion ended!" if the LED was ON earlier and there is no motion detected in the current PIR value (i.e. current PIR value is False) whereas old PIR value showed some movement (i.e. old PIR value is True). It further turns OFF the LED.

E. Weather Alert

There are three kinds of weather sensors being used: 1 Precipitation sensor, 1 humidity sensor and 1 temperature sensor. These sensors take in the inputs from the environment and trigger alert notifications for the user accordingly. [13]

The simulation for these sensors has been done in Omnet++. The range of each of these sensors is 100 units so it cannot directly transmit data to the controller. Instead a router has been used to transmit the data from the sensor to the controller via the router. The controller is further connected to the local station.

All the logic implementation has been done in Python in the local station. Locally generated inputs for four inputs: precipitation, humidity, minTemperature and maxTemperature would be used to trigger notifications to the user. For instance, if the precipitation value is greater than equal to 50 and humidity is greater than equal to 30, then the alert notification

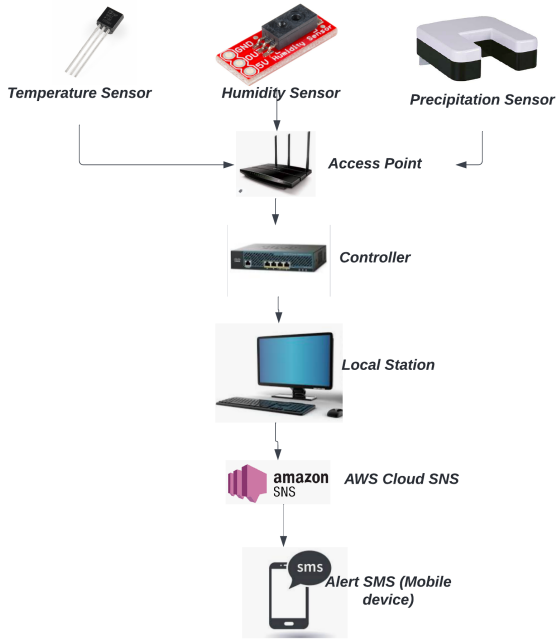

Fig. 7. Weather Alert Process Flow

would be: 'It might rain. Please carry the wind and rain shield for the stroller. Do not forget your umbrella!'. Similarly, if the humidity level is enough, an event is triggered and a notification is sent to the user that there is no need to turn on the sprinklers.

This message is generated using AWS SNS service in the cloud and a push notification is sent via SMS on the user's phone.

IV. Experimental Work to Demonstrate the Working and Performance of Proposed Solution

This section outlines the simulation results of the Safe Nest project using Omnet++, Python core logic and AWS SNS service. This section is primarily segregated into five subsections: Omnet++ simulation, Python code simulation, AWS SNS service simulation for the 4 different kinds of sensors. Snapshots of the simulation have been attached from the used front ends of this project Safe Nest.

A. Omnet++ Simulation Results

A thorough analysis of the data transfer rate, latency and throughput were have been conducted while running the simulation of the project environment for 900 seconds.

At the end of the 900 seconds' simulation, the camera sensor sends around 600 packets, the magnetic PIR sensor sends 550 packets, the motion PIR sensor sends 1450 packets, the three weather sensors sent 900, 1400 and 1450 packets respectively.

The plots of the packet delivery ratio, throughput rate, and latency [14] have been analyzed at the end of the simulation.

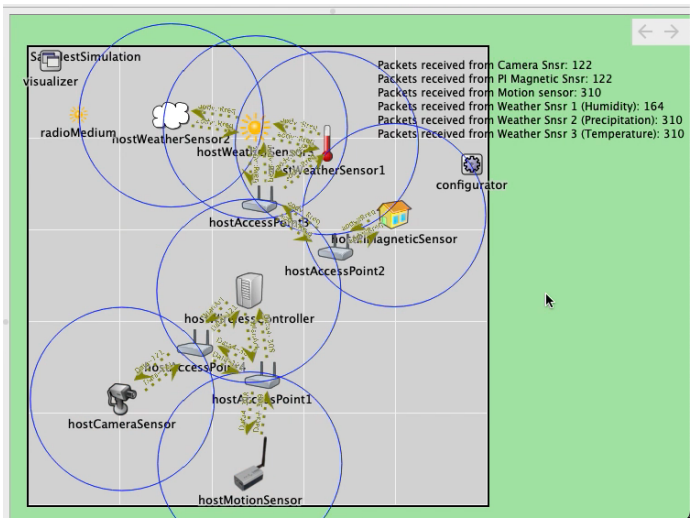

Fig. 8. Omnet++ Simulation snapshot

The observations have been plotted as recorded.

I. Packet Delivery Ratio: Packet delivery ratio for all the sensors is in between 85 and 100 percent. This means that almost all of the data being sent is reaching the controller. This depicts that the four access points have been placed correctly for this network. The reason for the packet delivery ratio of being close to 100 percent but not exactly 100 percent is due to the simulation design and time limit of 900 seconds. Had the simulation been extended to another 5 seconds, the Packet delivery ratio might have reached 100 percent.

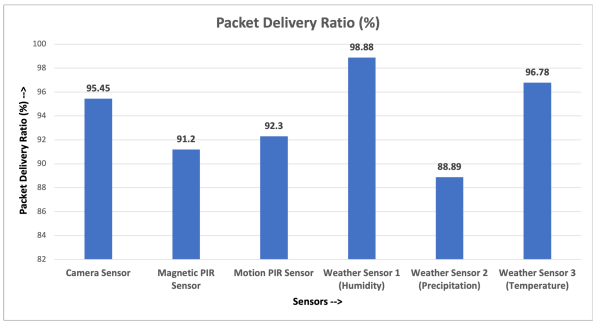

Fig. 9. Packet Delivery Ratio plot for all the sensors

An interesting point to ponder upon is that one of the weather sensors (weatherSensor2) is out of range for the only access point closer to it (hostAccessPoint3) yet the packet delivery ratio is closer to 100 percent. This is due to the fact that it transfers the data to the other weather sensor which in turn sends the data to the access point and further to the controller.

II. Latency rate:

The latency rate for all the sensors is almost between 0.45 and 0.55 but the best end-to-end latency rate is for the motion PIR sensor when compared with the rest of the sensors. Camera and weather sensors have quite similar latency rates.

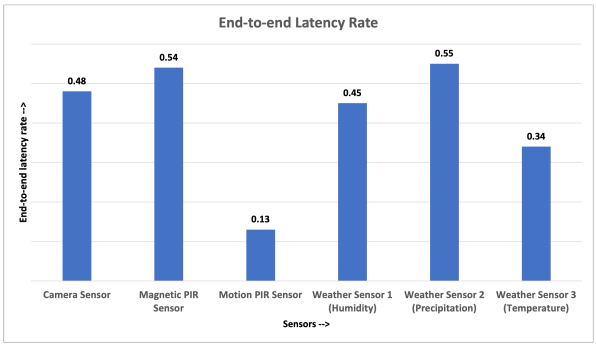

Fig. 10. End-to-end latency rate plot for all the sensors

This might be attributed to the positioning of the sensors in the network.

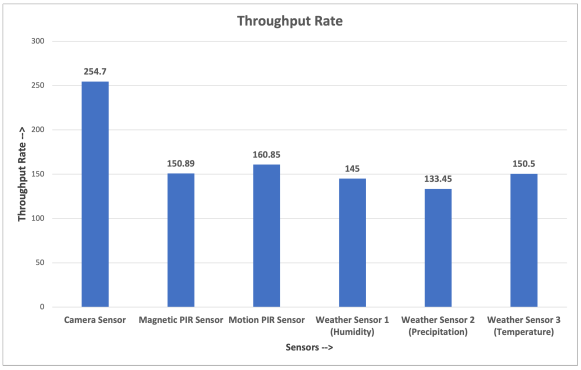

Fig. 11. Throughput rate plot for all the sensors

III. Throughput rate: Throughput rate can be defined as the amount of data transmitted by a particular sensor. It is quite evident from the below figure that the camera sensor has the maximum throughput rate, followed by motion PIR, weather and magnetic sensors.

B. Python and AWS Simulation

The python code results and AWS triggers' simulation have been observed and recorded individually for every sensor. The description of each of the python code logic, execution and simulation have been listed below:

a) Burglary detection and automatic door opening for authorized visitors: : The camera generates frames of the persons standing in front of it. From those frames, faces are detected and compared with the faces stored in the system. "OpenCV" and "face_recognition" libraries have been used in python for this purpose.

The face detection code is as follows:

If the detected face matches with the ones in the database, a red box with Name: Access granted appears around the image. The code also sends a signal to the servomotor to open the door and prints a message – "Door unlocked, welcome

Face Recognition -

1. Burglary detection & Automatic Door opening mechanism

```
camera = cv2.VideoCapture(0)

# Load a second sample picture and learn how to recognize it.
snigdha_image = face_recognition.load_image_file("Snigdha/snigdha.jpeg")
snigdha_face_encoding = face_recognition.face_encodings(snigdha_image)[0]

# Load a second sample picture and learn how to recognize it.
pebbles_image = face_recognition.load_image_file("Pebbles/pebbles.jpeg")
pebbles_face_encoding = face_recognition.face_encodings(pebbles_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    snigdha_face_encoding,
    pebbles_face_encoding
]
known_face_names = [
    "Snigdha",
    "Pebbles"
]
# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

while True:
    success, frame = camera.read()  # read the camera frame
    if not success:
        break
    else:
        # Resize frame of video to 1/4 size for faster face recognition processing
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_small_frame = small_frame[:, :, ::-1]

        # Only process every other frame of video to save time
         # Find all the faces and face encodings in the current frame of video
         face_locations = face_recognition.face_locations(rgb_small_frame)
         face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
         face_names = []
         for face_encoding in face_encodings:
             # See if the face is a match for the known face(s)
             matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
             name = "Unidentified"
             accessmessage = "Unauthorized access"
            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]
                accessmessage = "Access granted"

            face_names.append(name)

        # Display the results
        for (top, right, bottom, left), name in zip(face_locations, face_names):
            # Scale back up face locations since the frame we detected in was scaled to 1/4 size
            top *= 4
            right *= 4
            bottom *= 4
            left *= 4

            # Draw a box around the face
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

            # Draw a label with a name below the face
            cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX

            cv2.putText(frame, name + ": " + accessmessage, (left + 6, bottom - 6),
                        font, 1.0, (255, 255, 255), 1)

            ##Action in case of unauthorized access - Notify the house owner by SNS
            if accessmessage == "Unauthorized access":
                topicArn = 'arn:aws:sns:us-east-2:344935830395:notifyMe'
                snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                aws_secret_access_key = secret_access_key, region_name = 'us-east-2')
                #snsClient.publish(TopicArn = topicArn, Message = 'Alert:
                #Unauthorized access on the door!!')


            ##Action in case of authorized access - Unlock the door / garage door
            if accessmessage == "Access granted":
                print("Door unlocked, welcome home!!!")


            cv2.imshow('SAFE NEST LIVE',frame)
            key = cv2.waitKey(1)
            # stop loop by ESC
            if key == 27: # The Esc key
                break
camera.release()
cv2.destroyAllWindows()
```

Fig. 12. Face Detection and Recognition Code in Python

home!!!". On the other hand, if the face is unknown to the system, it would trigger an event that would further send a notification message to the user that there is an unauthorized access attempt on the door. In that case, a red box with tag - "Unknown: Unauthorized access" gets displayed around the face. The alert message sent to the user is as follows: "Alert: Unauthorized access on the door!!"
For instance, the face detected is one of the two faces stored in the system: "Snigdha" or "Pebbles", then the box around the face would say: "Snigdha: Access granted" and a message would get printed as: "Door unlocked, welcome home!!!". On the other hand, if the face is unknown, then an alert message (SMS) gets triggered via AWS SNS service as: "Alert: Unauthorized access on the door!!"

2. Garage door open alert

```
## Garage door status & open alert
### Assumption: we are using a Raspberry PI sensor for the garage door. Here, we will
## use user-input to simulate the
### reading of Raspberry PI sensor
sensor_pi = input('GPIO input value for the garage door..')

sensor1 = int(sensor_pi)

if sensor1 == 12: ##the GPIO pin 12 would be True if the door is closed else it
    ## would be False
    print("Garage door is closed!!")

else:
    print("Garage door is open!!")
    topicArn = 'arn:aws:sns:us-east-2:344935830395:notifyMe'
    snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                             aws_secret_access_key = secret_access_key,
                             region_name = 'us-east-2')
    #snsClient.publish(TopicArn = topicArn, Message = 'Alert: Garage door is open,
    ## please close it!!')
```

Fig. 13. Garage Door Open Alert Code in Python

b) Garage Door Open Alert:: For the simulation of the input from the magnetic PIR sensor, user-defined input values have been taken as an input (GPIO input value) and a pin value of 12 is set for the Garage door state to be closed. A simple if-then-else loop is executed here. If the GPIO input value is 12, it would print a message "Garage door is closed!". But if the GPIO input value is anything else other than 12, then it would send a notification to the user as: "Alert: Garage door is open, please close it!!"

c) Auto Lights Turn ON/OFF:: Similar to the Garage door open alert simulation, the inputs from the motion sensor have been simulated by taking user-defined input values as Motion PIR sensor values. There are three values being taken in as the user-defined inputs: current PIR value (True/False), old PIR value (True/False) and LED value (ON/OFF). If a PIR value is True, it means that the motion has been detected and vice versa i.e. if it is False, this means there is no movement observed. A LED value of ON means that the light is turned ON and a value of OFF means the light is turned OFF.

For instance, old PIR value is False i.e. there was no movement earlier. Current PIR value is True i.e. motion has been detected at present and LED value was OFF i.e. lights were not turned ON earlier. So, our core logic would turn the lights on by setting LED value as ON and display the message "Motion detected!". On the other hand, if the old PIR value was True i.e. a movement was observed in the past and LED value was ON i.e. lights were turned ON. Now, if the current PIR value is False i.e. no motion has been detected at present, then the core logic would turn off the lights by setting the

3. Auto Light Turn On/Off

```
## Assumption: Triggered by a motion sensor. Here, we assume that the
## particular sensor is a PIR sensor which leads
## to a change in the LED ON/OFF value as per the movement detected

# Setup digital input for PIR sensor:
pir = input('Current PIR Sensor value:  ')
#boolean value taken False initially i.e. no motion initially
pir_value = True if pir.lower()=='true' else False

# Setup digital output for LED:
led = input('LED_value (ON/ OFF): ')
led_value = True if led.lower()=='on' else False
#boolean value taken False initially: lights are turned off initially

# Main loop that will run forever:
oldPIR = input('Old PIR Sensor Value: ')
old_value = True if oldPIR.lower()=='true' else False

while True:
    if pir_value:
        # PIR is detecting movement! Turn on LED.
        led_value = True
        # Check if this is the first time movement was
        # detected and print a message!
        if not old_value:
            print('Motion detected!')
    else:
        # PIR is not detecting movement. Turn off LED.
        led_value = False
        # Again check if this is the first time movement
        # stopped and print a message.
        if old_value:
            print('Motion ended!')
    old_value = pir_value
    break
```

Fig. 14. Auto Light Turn ON/OFF Code in Python

4. Weather sensor

```
## Assuming that this weather sensor would sense and take 4 inputs
## to trigger notifications. Those four inputs would be:
## precipitation, humidity, minTemperature and maxTemperature
## Here, we are assuming that these inputs would be provided by
## the user but in the real-world scenario these
## would take values from a Weather API on a sensor

##getting the inputs
precipitation = int(input('Precipitation value from the sensor: '))
humidity = int(input('Humidity value from the sensor: '))
minTemperature = int(input(
    'Minimum temperature value for the day from the sensor: '))
maxTemperature = int(input(
    'Maximum temperature value for the day from the sensor: '))

if precipitation >= 50 and humidity>= 30:
    print('Alert: It might rain. Please carry the wind and rain shield for the stroller.
          Do not forget your umbrella!')

    snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                             aws_secret_access_key = secret_access_key,
                             region_name = 'us-east-2')
    #snsClient.publish(TopicArn = topicArn, Message = 'Alert: It might rain.
          ##Please carry the wind and rain shield for the stroller.
          ## Do not forget your umbrella!')

else:
    print('Alert: Action needed - triggering to front yard sprinklers!!')
    snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                             aws_secret_access_key = secret_access_key,
                             region_name = 'us-east-2')
    #snsClient.publish(TopicArn = topicArn, Message = 'Alert: Action needed
          ## - triggering to front yard sprinklers!!')

if maxTemperature <= 8 and minTemperature >= 2:
    print('Alert: Carry a jacket along with you')
    #topicArn = 'arn:aws:sns:us-east-2:132898630311:notifyMe'
    snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                             aws_secret_access_key = secret_access_key,
                             region_name = 'us-east-2')
    #snsClient.publish(TopicArn = topicArn, Message = 'Alert:
          ## Carry a jacket along with you')

elif maxTemperature < 2:
    print('Cover the stroller with snow muff. Keep the snow clearing brush in the car.')
    #topicArn = 'arn:aws:sns:us-east-2:132898630311:notifyMe'
    snsClient = boto3.client('sns', aws_access_key_id = access_key_id,
                             aws_secret_access_key = secret_access_key,
                             region_name = 'us-east-2')
    #snsClient.publish(TopicArn = topicArn, Message = 'Cover the stroller with snow muff.
          ## Keep the snow clearing brush in the car.')

else:
    print('It is a nice weather!! Enjoy..')
```

Fig. 15. Weather Alert Code in Python

LED value as OFF. A message would be displayed: ”Motion ended!”

d) Weather Alert:: For the simulation of the inputs of the weather alert, user-defined inputs have been taken for the external weather conditions: precipitation, humidity levels, minimum Temperature and maximum Temperature of the day. These values decide what message or alert will be sent to the user based on the threshold values set initially.

For instance, if the precipitation value is greater than or equal to 50 and humidity value is greater than or equal to 30, then an alert message will get triggered to the user via AWS SNS service: ”Alert: It might rain. Please carry the wind and rain shield for the stroller. Do not forget your umbrella!”. Similarly, if the value is not in this range, then an alert will get sent as follows: ”Alert: Action needed - triggering to front yard sprinklers!!”. This way the core logic will remind the home owner that they need to turn on the sprinklers. In similar way if the temperature is within the range 2 degrees and 8 degrees, then it would notify the home owner via an alert message as: ”Alert: Carry a jacket along with you”. On the same lines, if the maximum temperature is predicted to be less than 2 degrees, then an alert would be sent to the owner as follows: ”Cover the stroller with snow muff. Keep the snow clearing brush in the car.” If none of these condition meet, then a simple message would be printed as follows: ”It is a nice weather!! Enjoy..”

CONCLUSION

A complete Safe Nest home security system was built and simulated using AWS SNS service, four different types of IoT sensors and python simulation of the core logic. The Omnet++ simulation of the network and the four types of sensors was built to resemble the real sensor networks by using diverse parameters in the simulation environment. We had also computed and plotted the following: Packet Delivery Ratio, latency rate and throughput rate for the various sensors being used in the simulation network. The core sensor logic was coded and implemented on our local station. The first functionality was successfully simulated where in the recognized face was granted access whereas an unrecognized one was not. The alert notification of unauthorized access was also received successfully via the SNS service of AWS cloud instance in the event of unauthorized access. The next functionality of the deliverable was that we were able to simulate the motion activated LED lights using a PIR motion sensor. The next important functionality was a weather alert which notified the home owner of the weather conditions for the day, for instance if it was about to rain, a notification was sent that the user must carry an umbrella. Similarly, an alert was sent via SMS through AWS SNS service for carrying the snow shovel in the event a snow storm was about to happen during the day. Another part of the project was checking if the garage

door was left open. The AWS SNS service was able to trigger SMS notifications to be sent to pre-registered users. Thus, a complete loop of Safe Nest features was implemented and simulated successfully.

REFERENCES

[1] Mundle, Kent. "Home Smart IoT Home: Domesticating IoT." Toptal Design Blog, www.toptal.com/designers/interactive/smart-home-domestic-internet-of-things: :text=IoT

[2] Mysa. "How IoT and Smart Home Technology Is Shaping Our Lives." Home Automation — Blog — Mysa, 1 Apr. 2021, getmysa.com/blogs/home-automation/how-iot-and-smart-home-technology-is-shaping-our-lives.

[3] What is home automation system? - Structure, Types. (n.d.-b). ElPro- Cus - Electronic Projects for Engineering Students. Retrieved August 15, 2021, from https://www.elprocus.com/home-automation-systemsapplications/.

[4] Elechi, Promise Ekwueme, Uchechukwu Okowa, Ela. (2022). Facial Recognition Based Smart Door Lock System. Journal of Scientific and Industrial Research. 6. 95-105.

[5] Gomathy, C K. (2022). Smart Door with Facial Recognition. 2395-0056.

[6] Wireless Tutorial — INET 4.4.0 documentation. (n.d.). Inet.omnetpp.org. Retrieved July 21, 2022, from https://inet.omnetpp.org/docs/tutorials/wireless/doc/index.html.

[7] Face Detection using Python and OpenCV with webcam. (2018, November 6). GeeksforGeeks. https://www.geeksforgeeks.org/face-detection-using-python-and-opencv-with-webcam/.

[8] Python, R. (n.d.). Face Recognition with Python, in Under 25 Lines of Code – Real Python. Realpython.com. https://realpython.com/face-recognition-with-python/.

[9] Oleszak, M. (2021, September 21). Working with Amazon SNS with Boto3. Medium. https://towardsdatascience.com/working-with-amazon-sns-with-boto3-7acb1347622d.

[10] Face Detection with Python using OpenCV Tutorial. (n.d.). Www.datacamp.com.https://www.datacamp.com/tutorial/face-detection-python-opencv.

[11] Scholarworks@gvsu,S., Ridge,B.(2016). Remotely Monitor and Manage a Garage with IoT Remotely Monitor and Manage a Garage with IoT. https://scholarworks.gvsu.edu/cgi/viewcontent.cgi?article=1251context=cistechlib.

[12] PIR Motion Sensor. (n.d.). Adafruit Learning System. Retrieved July 21, 2022, from https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/circuitpython-code.

[13] Foong, N. W. (2020, September 25). How to Create a Weather Alert System in Python. Medium. https://towardsdatascience.com/how-to-create-a-weather-alert-system-in-python-5fab4b42e49a.

[14] OMNeT++ WLAN PROJECTS. (n.d.). OMNeT++. Retrieved July 21, 2022, from https://omnet-tutorial.com/omnet-wlan-projects/.