Implementation of Smart Home Sensor Networks with Amazon SNS

Dhairya Ajitkumar Patel
MEng ECE
University of Waterloo
Waterloo, Canada
da32pate@uwaterloo.ca

Krishna Kanth Mutta
MEng ECE
University of Waterloo
Waterloo, Canada
kkmpn@uwaterloo.ca

Yash Tiwari
MEng ECE
University of Waterloo
Waterloo, Canada
y2tiwari@uwaterloo.ca

***Abstract*—Smart home monitoring and automation are used to maintain a comfortable living condition. This can be achieved due to the recent advancement in network technologies. There are numerous types of comfort requirements for humans in their houses. The most important of these categories is thermal comfort, which is related to temperature and humidity, followed by visual comfort, which is related to light, hygienic comfort, which is related to air quality, and finally, safety comfort which is related to smoke detection. Protection from trespassers and intruders is also one of the important aspects related to home security. To keep these characteristics within an acceptable range, a system can be built up to monitor them. Making the house smart also entails allowing for the intelligent automatic execution of many commands following the analysis of the collected data. The Internet of Things can be used to automate processes (IoT). This offers the occupant access to specific data in the house as well as remote control of some parameters. In this paper, we are demonstrating the implementation of Smart Home sensor network that encompasses sensors like Light-dependent sensors, Carbon-dioxide (Co2) sensors, humidity sensors, Temperature sensors, and smoke detectors in the form of simulation. In addition to this, we will also be incorporating a face recognition system to prevent illegal trespassing of intruders. The sensor data can be used to monitor the conditions which can be used to detect abnormalities in the system. If any abnormalities are detected by the system, we will be notifying the resident through Amazon Simple Notification Service (SNS). The simulation of our sensor networks would be carried out in Omnett++ environment. Face recognition system and processing of data, notifying the user in case of any abnormality will be done using python.**

***Index Terms*—Smart Home, Omnett++,**

I. Introduction

Wireless sensor networks (WSNs) improve modern life in many different places. In recent years, its use has spread around the globe. The wireless communication network gives the internet of things a new platform. A wide range of items, including Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc., are included in the Internet of Things (IoT). In the network, where data and communication are invisible to the outside world, these things will be encircled [1].

Since the incorporation of IoT technology into smart homes, there have been significant advances in the field of smart home research. Security, home comfort, medical care, data processing, entertainment, and home business - most of these are all included in a smart house [2].

When a homeowner is inside or outside of their home, a smart home is designed to increase their sense of security and tranquilly. So that the advantages of these digital devices to connect and communicate in previous decades grow more attractive with more features as varied controllable apparatuses in the home ascend, which is not a new concept [3]. The most recent IoT application innovation is one of the most readily available setups and a requirement for the modern way of life to monitor and surveillance connected to security. We can access information about security threats, warning notifications, and extra control over home equipment for appropriateness and building surveillance through the use of IoT. Apart from the home security in terms of intruders, there are other aspects that are desirable with the human comfort in their houses like temperature or thermal comfort, humidity, air quality, smoke detection and so on. To cover these aspects in our smart sensor network, we would consider various sensors like Carbon dioxide (Co2) sensors, Humidity sensors, Temperature sensors, and Smoke detectors. Such smart home sensor network systems continuously monitor homes using various sensory systems, such as motion, smoke, gas, temperature, glass break or door break detectors and fire alarm systems, to provide security from natural, incidental, planned, undesired, inadvertent, and human-made hazards.

People and organisations are taking preventive efforts around the world to lessen the harm that various incidents and criminal activities create. Organizations don't just choose for a defensive posture; individuals also value their own and their families' security. [4] Technology is being used by more and more people to protect their personal and family safety. Manufacturers and suppliers of these devices have taken advantage of the increased interest in home and commercial security to boost their own sales for profit. Sales of security systems are soaring as more and more people start to think that these systems may give an extra layer of security and shield their workplace, house, and property from damage in the event of an accident. In many big and medium-sized establishments, security tools including surveillance cameras, monitored and non-monitored intrusion alarm systems, motion detectors, and automatic fire fighting systems are standard. [4] These gadgets have been widely used and put in homes, businesses, and public spaces. Trespassing on private property

Fig. 1. Smart Home Sensor Network

is one of the greatest security worries. The property can now be monitored more easily with CCTV and alarms, but a security officer must still stand in front of monitors to keep an eye on each camera. This is a laborious task that leaves room for error. An unlawful trespasser can go unnoticed by the guard. There is also the issue of compromising the privacy of those inside the home, who might not feel comfortable with security officers always keeping an eye on them. Additionally, it would be expensive to construct such a system and pay the guards' salaries. Incidents that might affect the security can be very dangerous and lead to expensive losses to people's lives. Installing a security system could seem like a way to cope with potential security concerns, but just like most other systems, modern security systems have frequent drawbacks. These flaws include those with these systems' cost, installation, system maintenance, and performance. To assist with such an arduous task and to find any unauthorized employees, a smart system with a sensor network is required. To solve this problem, we will be including Intrusion Detection using Camera along with the other sensors in our Smart home sensor network. We will also enable this Smart home network to trigger messages to the authorized person using Amazon Simple Notification Service (SNS).

This paper primarily focuses on giving users security whether they are at home or away from it. Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication. Amazon SNS Ensure accuracy with message ordering and provides increased security with message encryption and privacy. With the use of this service, messages can be transmitted swiftly, precisely, and affordably. When used with integrated security systems, where information is sent by the security system and received by the user's mobile phone through SMS, mobile phones with SMS capabilities will be highly beneficial and the users can receive continuous updates regarding the well-being of their house or property.

The rest of the paper is structured as follows: Section 2 highlights the related work and key contributions in the area of smart home networks. Section 3 discusses the methodology, design and implementation, and Section 4 presents the experimental results of our project. The paper is concluded in Section 5.

II. Review of Related Work

Numerous experts have addressed the security challenges in IoT devices during the last few years. The critical goals of a few recent studies will be addressed in this section of the paper. Rahman et al. [5] created a very affordable smart home system in 2018 that runs on an Android application and can be accessed wirelessly over GSM from anywhere in the world. The project has a sophisticated electrical monitoring system that enables users to remotely control lights, fans, TVs, and other devices both inside and outside the home. While receiving any form of security alert while residing anywhere in the world, the user can maintain the security system utilizing the Smart Security System application. Smart gardening systems using Android applications and GSM systems were primarily supported by IoT. This system increases human satisfaction and is operated automatically without awaiting instructions from the owner. A smart home system guarantees us a secure living environment, enhanced security, reduced power consumption, and environmental friendliness—things that were unthinkable in previous decades. A security system for opening doors using face recognition and detection algorithms was proposed by Nag et al. [6]. The authorised user receives the incoming data transmitted from the Raspberry Pi via the Telegram application. This approach employs the old dated HaarCascades face detection algorithm, which uses a lot of memory.

In 2017, Joshi et al.'s [7] goal was to create a basic home automation system for controlling a variety of devices that could be viewed and accessed for a very low cost from anywhere in the world. The Raspberry Pi and Arduino were used with the Nrf modules integrated with them to screen the home condition apparatuses and submit the results to the developed server. The web server is dynamically alerted via an alarm or message if any hazardous or threats are discovered via mobile connectivity after periodically checking the parameters or orders that are passed via web page. No matter where they are, users can easily access this program. As a result, the system has the appearance of being advantageously cheap. The proposed hierarchical engineering system is expected to translate a high-tech smart home system into a result that is simple to use.

Using the Node MCU ESP8266 and Arduino-nano as a controller, Adriano and Budi [8] planned and constructed an integrated home security and Internet of Things (IoT) system in 2018. Digital code, RFID to reveal the entrance, and email notifications to customers were all included in the home security framework. The framework that was shown used a DHT-22 sensor to measure the temperature and humidity in the room, a PIR to detect intruders, a sensor for rain to distinguish

rain, LDR sensors to monitor the light level, and a fire sensor to detect an oven fire. In addition, install the light bulbs and solenoid valves that serve as the actuators. The findings of this investigation showed that the system can monitor the condition of the home and remotely control the output of solenoid valves and lights by using an application on a cell phone through web connection.

Suresh S. et al. [9] stated in his paper that "the system intended for home observation and Security system includes of sensors that are supposed to collect the information that may be utilized by the owner to make wise decisions. The temperature sensing component is used to determine the space's temperature since the passive infrared sensor (PIR) is used to detect movements. To coordinate and improve the system's safety, several modules, notably the PIR, temperature, and hence the GSM modules, communicate with one another. The Arduino board is connected to the PIR sensing element, and as a result, the temperature sensor element. The board receives the digital signal. The Arduino board uses the GSM module to transmit and receive signals. Through the GSM module's route, the received signal is sent to the home's owner via text message. The owner sends a signal to the GSM module to turn off the alert if necessary. The Arduino board can receive the signal from the GSM module. The Arduino board transforms this signal into a form that the sensors can understand before sending it to them. Real-time sensor switching is used. The Arduino board is the key component. The Arduino chip contains the code for the motion detector, temperature sensor, and GSM. The SMS is delivered to the homeowner right away after the system is activated. The GSM module contains the necessary signaling contained within it.

According to Jayashri Bangali and Arvind Shaligram [10], an essential component or feature of smart home applications is the automated or intelligent house, which denotes the automation of regular tasks with electrical appliances used in homes and security. The newly developed idea of "smart homes" provides residents with a cosy, practical, and secure environment. Traditional security systems provide an indicator in the form of an alarm to keep residents and their property secure from attackers. A smart home security system, however, offers a lot more advantages. In his project, he put forth two systems, one of which relies on GSM technology and the other of which detects intruders using a web camera. The first security system makes use of a web camera that is mounted inside the home and is controlled by software that is placed on a PC. This system communicates over the Internet. Any movement of an intruder in front of the camera's field of view is detected. The second security mechanism is SMS-based and sends an SMS to the owner using GSM technology. The planned device aims to protect homes against burglars and fires. If any of the aforementioned scenarios occur while the owners are away from their house, the gadget will SMS the emergency number that has been given to the system.

The advantages of Wireless Sensor Networks and GSM technology were combined to create Aayush Aggarwal and R.C. Joshi's [11] WSN and GSM based Remote Home Security System. With no consideration for distance, it may detect intrusion, fire, etc. and notify the user remotely about the incident. In those security systems, an intrusion can be detected if it falls within the WSN dimension.

III. Methodology, Design and Implementation

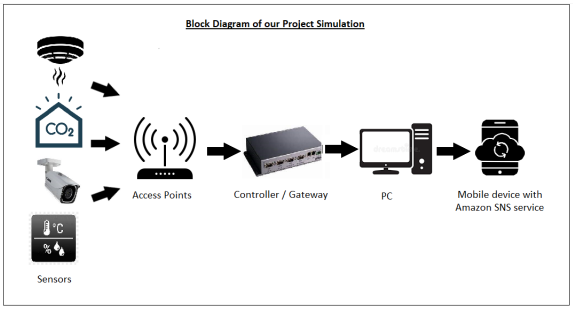

Fig. 2. Block Diagram of the implementation of the project

In this section we are going to discuss the methodology used and discuss our design and implementation of the sensor networks in Omnet++, logic in python and Amazon SNS system. Figure 2 shows the block diagram of our methodology. It is seen that we have used five type of sensors which are Smoke, Carbon-dioxide (Co2), Camera, Humidity and Temperature sensor. These sensors communicate to their respective access-points and send them packets at regular interval. On receiving the packets from the five access-points, the packets are then forwarded using AoDV protocol to the central hub or gateway (controller). This packets / data is then accessed by a local PC where the data are analysed. Finally, a notification is send via cloud server to the end user if the sensor data values cross the predefined threshold.

In this project, the implementation of the sensor network, mode of communication etc is done in Omnet++ simulation environment. The configurations is discussed in further sections. The implementation of logic to set the trigger values and notify the end user is implemented using the combination of Python and Amazon-SNS service.

A. Simulation in Omnet++

Figure 3 shows the field area of 695 X 454 meters in Omnet++ environment which can be assumed to resemble to that of a Smart Home. In this environment, we have used 3 Smoke sensors to prevent false trigger of alarm when there is no fire. The other 4 types of sensors are Humidity, Co2, Temperature and Camera sensor. The configuration details of each of these sensors are as follows.

B. Humidity Sensor

The uses for humidity sensors are incredibly varied. Humidity sensors are used in homes for monitoring and preventive measures for people with illnesses that are impacted by humidity. Additionally, house Heating, Ventilation, and Air Conditioning systems (HVAC systems) include a humidity

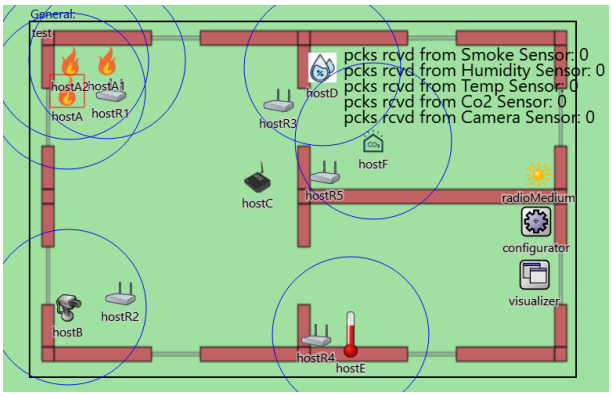

Fig. 3. Setup of Sensor Networks in Omnet++

sensor. The value of the capacitor is measured by the humidity sensor to determine the change in moisture.

In this project we are performing the simulation in a virtual environment like Omnet++ but physical sensors that can be used in real world are of various types. One of the type of physical sensor is the DHT11 and DHT22 sensor. The DHT22 type of sensor is more expensive as compared to the other but it has better sensitivity. While the DHT11 has a temperature range of 0 to 50 degrees Celsius with +-2 degrees precision, DHT22 has temperature measuring range of -40 to +125 degrees Celsius with +-0.5 degrees accuracy. Additionally, the DHT11 sensor's humidity range is 20 to 80 percent with a 5 percent accuracy, compared to the DHT22 sensor's 0 to 100 percent with a 2 to 5 percent accuracy [12].

In Omnet++, the sensor simulation is carried out. Since the humidity sensor's transmission range is under 100 units, it must send data via access-point rather than directly to the controller. The local station is connected to the controller.Python is used for the entire logic implementation process. When the event is triggered, the python broadcasts the messages to the owner's device using SNS cloud services. This message is generated using the Amazon SNS package. In our simulation the humidity sensor has a packet length of 64 bytes and each packet is sent at an interval of 15 seconds.

C. Smoke Sensor

In real world there are various types of smoke sensors available that can support our simulation type, one of them is smoke sensor by Fibaro FGSD-002-EN-A-v1.1. It is battery powered with a battery type of CR123A 3V DC, works on radio frequency of 908.4MHz or 916 MHz. the range of this FGSD-002-EN-A-v1.1 sensor is 30meters indoor which is suitable for our application. The maximum measured temperature by this sensor is 100 degree Celsius. This sensor is highly accurate sensor with an accuracy of 0.5 degrees. Whenever a fire event is triggered, this sensor sends the sound signal as well as a control command to the user. It has an additional feature of performing automatic test of fire every 10 seconds. It also has 3 levels of sensor sensitivity and has a radio power of -5dBm[13]. In our simulation we have used three smoke sensors which are grouped at a location as shown in the field network. Smoke sensors are the sensors which have been considered as the most critical type among all the five sensors. A notification is sent to the end user through Amazon SNS only when at least two of the sensors data values is greater than the trigger set point value. This logic is implemented in python which helps to solve the problem of false triggering of the notification system. As smoke sensor is critical in this project, we have set the packet sent rate at an interval of 5 seconds while the length of packet is kept at 64 bytes for each of the three sensors. The communication range is set at 100 units for all the three sensors.

D. Temperature Sensor

The physical sensor relevant to our application is the Temperature sensor TEMP-STICK-TH-W-FBA by Ideal sciences. It is AA battery powered and the communication is done wirelessly. Its has high accuracy of 0.5 degree Celsius and its temperature range is between -40 Fahrenheit to +140 Fahrenheit[14].

In our simulation, the communication range for temperature sensor is kept at 100 units so that packets are sent to controller via its nearest access-point. The packet size is 64 bytes while the data rate at which each packet is sent is 15 seconds. For logic implementation, a text message is sent to the end user if the values of temperature reach the set threshold. The information on set values are mentioned in the experimental results section.

E. Co2 Sensor

Carbon dioxide sensors are used to detect the level of Co2 in the house and can be used to monitor the air quality of a confined space. For our simulation we have used a co2 sensor which has a communication range of 100 units with packet size of 64 bytes. Each packet is sent at an interval of 5 seconds. If the sensor value is greater than 800 a notification message "Air quality is poor: Action Required" is sent to the end user using amazon SNS service.

F. Intrusion Detection

Many visual image detection sensors are used for intrusion detection of which one of them is the Kinetix sCMOS camera sensor. The sensor type used is Teledyne Photometrics Kinetix Sensor. It comes with 10 megapixel camera with 6.5 micrometer pixel size. It can record the frames with a higher rate of 498 frames per second. It has a wide aperture with 29.4 mm field of view.

In Omnet++ simulation, the video that the camera records is delivered to the controller via the router because the camera has a range of 100 units and the local station is positioned farther from the range. The controller transfers the video data to the local station because it is connected to it.

SNS configuration is used by AWS for message notification. Access key, security password, and location are established through the AWS command line interface for that.

Normally the video data is greater in size as compared to the data transmitted by other sensors. Due to this reason, in

our project, we have set the packet length to 1 Megabytes and each packet is sent at an interval of 20 seconds.

G. General Configurations

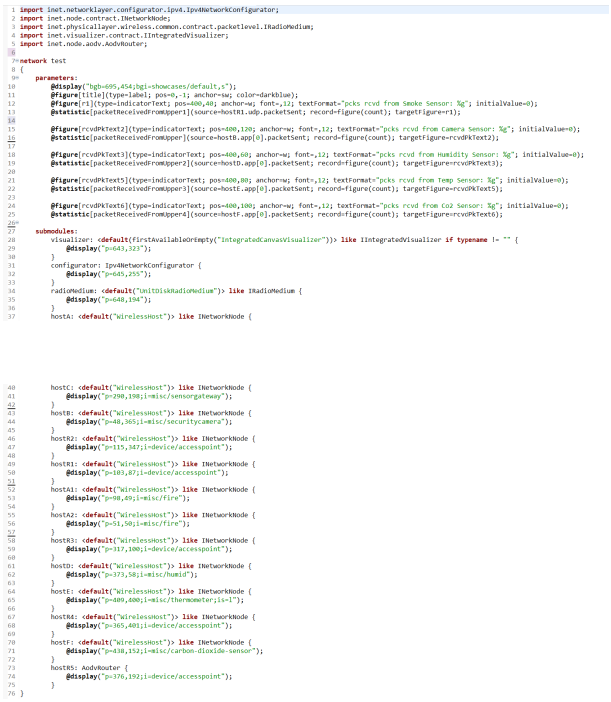

Fig. 4. Ned file in Omnet++

The Figure 4 shows the image of code (*.ned*) used to set up the field of sensor networks in Omnet++. Figure 5 shows the code of the *.ini* file. It is seen that we have used 5000 as our port address, packet name is set to 'UDP Data'. We have also kept the Duplex communication as false as we don't want sink to send the data to source. We have set MAC header length as 23 Bytes and the bit-rate for hosts is kept as 11 Mbps. The communication range of all the access points are kept different as they are not at same distance from the central gateway - 'host C'. The communication ranges for access-points R1, R2, R3, R4, R5 are set to 250, 250, 125, 250, 100 units. Access-point R1 is configured with Smoke sensors, R2 with Camera sensor, R3 with Humidity sensor, R4 with Temperature sensor and host R5 is configured with Co2 sensor. The central hub or the host C has large communication range of 250 units and is placed in the field in such a way that it communicates with all the access-points. AoDV routing protocol is used in this environment which helps to forward the packet from the sensors to the central hub / gateway.

H. MAC layer - IEEE802.11

IEEE802.11 has been selected as the MAC layer, allowing the nodes and Controller to communicate with each other over WiFi. It defines the collection of media access control (MAC) and physical layer (PHY) protocols for implementing wireless local area network (WLAN) computer communication. IEEE 802.11 is a subset of the IEEE 802 set of local area network (LAN) technical standards. The standard and amendments are the most commonly used wireless computer networking standards in the world and serve as the foundation for wireless network products bearing the Wi-Fi brand. The majority of home and workplace networks employ IEEE 802.11 to enable wireless communication and Internet access between computers, printers, smartphones, and other devices.

The LAN/MAN Standards Committee of the Institute of Electrical and Electronics Engineers (IEEE) is responsible for developing and maintaining the standards (IEEE 802). The standard's foundational version was released in 1997, and changes have since been made. The current edition of the standard effectively nullifies each change, yet the business world frequently markets to the revisions since they succinctly describe the capabilities of their products. As a result, each iteration tends to establish itself as a standard in the industry.

IEEE 802.11 operates at a variety of frequencies, including the 2.4 GHz, 5 GHz, 6 GHz, and 60 GHz bands. Despite the fact that the IEEE 802.11 specifications describe possible channels, the permitted radio frequency spectrum availability varies greatly per regulatory region.

The protocols are frequently used to transmit Internet Protocol traffic and are typically used in conjunction with IEEE 802.2. They are built to smoothly inter-operate with Ethernet.

I. Logic Implementation using Python and Amazon SNS

Python is the programming language that is used to collect all the sensor inputs and perform the necessary validations. It serves as the user interface for the Amazon SNS service as well. With the help of this SNS service, we may send messages containing data to an Amazon SNS topic, and the topic's subscribing applications will receive the messages asynchronously.

Establishing a secure connection between our Python code and the Amazon SNS service is therefore the first step. This is done using the "Boto3" package, which offers Python API for AWS infrastructure services, and the unique security access key generated in the AWS services site, as shown in figure 6.

Once a secure connection is set up, the main method is executed which gives the user a series of options for simulation as shown in figure 7. On selecting the desired option, function calls are made to test the sensor functionalities. The functionalities are as follows:

I The inputs from the smoke detector is in terms of LEL values. To avoid false alarms 3 sensors are used to detect the smoke levels and when at least two of them are showing readings greater than 20, the SNS message is triggered to the user stating the increased level of smoke as shown in figure 8. Additionally, this message can also be triggered to the fire department.

II The temperature sensor sends the temperature in degree Celsius. On temperature greater than 25 degrees Celsius a message is sent to the user stating "Warning! House is too hot, please turn on the Air Conditioner switch". However, on temperature less than 16 degree Celsius, a warning message to turn on the heater is triggered as shown in figure 9.

```
[General]
network = test
sim-time-limit = 900s #Simulation Period
*.host*.ipv4.arp.typename = "GlobalArp"

# Configurations for all the three Smoke Sensors
*.hostA*.numApps = 1
*.hostA*.app[0].typename = "UdpBasicApp"
*.hostA*.app[0].destAddresses = "hostC" #hostC is the controller / destination
*.hostA*.app[0].destPort = 5000
*.hostA*.app[0].messageLength = 64B #Length of message 64 Bytes
*.hostA*.app[0].sendInterval = 5s #1 packet is sent after 5s
*.hostA*.app[0].packetName = "UDPData"

# Configurations for Camera Sensor
*.hostB.numApps = 1
*.hostB.app[0].typename = "UdpBasicApp"
*.hostB.app[0].destAddresses = "hostC" #hostC is the controller / destination
*.hostB.app[0].destPort = 5000
*.hostB.app[0].messageLength = 1024B #Length of message 1 Megabytes
*.hostB.app[0].sendInterval = 20s #1 packet is sent after 20s
*.hostB.app[0].packetName = "UDPData"
# Configurations for Humidity Sensor
*.hostD.numApps = 1
*.hostD.app[0].typename = "UdpBasicApp"
*.hostD.app[0].destAddresses = "hostC" #hostC is the controller / destination
*.hostD.app[0].destPort = 5000
*.hostD.app[0].messageLength = 64B #Length of message 64bytes
*.hostD.app[0].sendInterval = 15s #1 packet is sent after 15s
*.hostD.app[0].packetName = "UDPData"

# Configurations for Temperature Sensor
*.hostE.numApps = 1
*.hostE.app[0].typename = "UdpBasicApp"
*.hostE.app[0].destAddresses = "hostC" #hostC is the controller / destination
*.hostE.app[0].destPort = 5000
*.hostE.app[0].messageLength = 64B #Length of message 64bytes
*.hostE.app[0].sendInterval = 15s #1 packet is sent after 15s
*.hostE.app[0].packetName = "UDPData"

# Configurations for Co2 Sensor
*.hostF.numApps = 1
*.hostF.app[0].typename = "UdpBasicApp"
*.hostF.app[0].destAddresses = "hostC"
*.hostF.app[0].destPort = 5000
*.hostF.app[0].messageLength = 64B #Length of message 64bytes
*.hostF.app[0].sendInterval = 15s #1 packet is sent after 15s
*.hostF.app[0].packetName = "UDPData"
# Configuration for host C / Destination / Controller
*.hostC.numApps = 1
*.hostC.app[0].typename = "UdpSink"
*.hostC.app[0].localPort = 5000

#Common configurations for all sensors and hosts
*.host*.wlan[0].mac.fullDuplex = false
*.host*.wlan[0].radio.receiver.ignoreInterference = true
*.host*.wlan[0].mac.headerLength = 23B
*.host*.**.bitrate = 11Mbps

#setting Visualizations
*.visualizer.sceneVisualizer.descriptionFigure = "title"
*.visualizer.mediumVisualizer.displaySignals = true
*.visualizer.physicalLinkVisualizer.packetFilter = "UDPData*"

#Setting up the communication ranges of all the Access_points
*.hostR3.wlan[0].radio.transmitter.communicationRange = 125m
*.hostR1.wlan[0].radio.transmitter.communicationRange = 250m
*.hostR2.wlan[0].radio.transmitter.communicationRange = 250m
*.hostR4.wlan[0].radio.transmitter.communicationRange = 250m
*.hostR5.wlan[0].radio.transmitter.communicationRange = 100m

*.host*.forwarding = true
```

```
*.configurator.optimizeRoutes = false
*.host*.ipv4.routingTable.netmaskRoutes = ""

#Setting more Visualzations
*.visualizer.physicalLinkVisualizer.displayLinks = true
*.visualizer.dataLinkVisualizer.displayLinks = true
*.visualizer.networkRouteVisualizer.displayRoutes = true
*.visualizer.*LinkVisualizer.lineShift = 0
*.visualizer.networkRouteVisualizer.lineShift = 0
*.visualizer.networkRouteVisualizer.packetFilter = "UDPData*"
*.host*.wlan[0].mac.useAck = true

#Communication Ranges of sensors
*.hostA*.wlan[0].radio.transmitter.communicationRange = 100m
*.hostB.wlan[0].radio.transmitter.communicationRange = 100m
*.hostD.wlan[0].radio.transmitter.communicationRange = 100m
*.hostE.wlan[0].radio.transmitter.communicationRange = 100m
*.hostF.wlan[0].radio.transmitter.communicationRange = 100m
*.hostC.wlan[0].radio.transmitter.communicationRange = 275m
*.hostA*.wlan[0].radio.displayCommunicationRange = true
*.hostB.wlan[0].radio.displayCommunicationRange = true
*.hostD.wlan[0].radio.displayCommunicationRange = true
*.hostE.wlan[0].radio.displayCommunicationRange = true
*.hostF.wlan[0].radio.displayCommunicationRange = true
*.host*.typename = "AodvRouter"
*.visualizer.dataLinkVisualizer.packetFilter = "AODV*"

#Mac as IEEE802.11/
*.host*.wlan[*].mgmt.typename = "Ieee80211MgmtAdhoc"
*.host*.wlan[*].agent.typename = ""
*.host*.wlan[*].radio.typename = "Ieee80211UnitDiskRadio"
*.host*.wlan[*].bitrate = 11Mbps #Bit rate = 11 Mbps
```

Fig. 5. Ini file in Omnet++

```
import boto3
import time
client = boto3.client(
        "sns",
        aws_access_key_id="xxxx",
        aws_secret_access_key="xxxx",
        region_name="ca-central-1"
    )
```

Fig. 6. Establishing secure connection with amazon SNS

```
def main ():
    flag = 0
    while flag !=6:
        print("\nPlease select one of the following features:")
        print("1. Smoke Detector \n2.Temperature Sensor \n3. Humidity Sensor \n4. Co2 Sensor \n5. Face Recognition \n6. Exit")
        flag = int(input())
        match flag:
            case 1: smoke_sensor()
            case 2: temp_sensor()
            case 3:humidity_sensor()
            case 4:co2_sensor()
            case 5: face_recog()
            case 6: print("Thank you")
main()
```

Fig. 7. Main method to test individual functionalities

This functionality can further be extended by automating the process of turning on the heaters and air conditioners.

III The humidity sensor transmits digital output which ranges from 1-100. Since the accuracy of these sensors are high only one sensor is being used for simulation as well. As shown in the figure 10, a message is triggered to the user if the humidity value is more than 30 or else the system prints that the humidity is in normal range.

IV The Co2 sensor sends the level of Co2 in the air in ppm. The safe limit is 800ppm. Thus, if the readings are more than 800, a message is triggered to the user as shown in figure 11.

V The facial recognition functionality is implemented by using the face-recognition and cv2 library. The camera continuously sends the video data to python and image is captured from these by using the cv2 library which detects face by using the facial features. Once the image is captured, it is stored and compared with the registered user's image by using the face-recognition library.

The images are first encoded and then compared against each other as shown in figure 13. If these images are matched, access is granted, and the simulation prints "Access Granted" or else a message to Amazon SNS is triggered stating unauthorized access. Thus, making the user aware about the intrusion

```
def smoke_sensor():
    # Smoke Sensor -> Ideal LEL value is less than 20
    smoke_s1 = int(input('LEL value from Controller 1: '))
    smoke_s2 = int(input('LEL value from Controller 2: '))
    smoke_s3 = int(input('LEL value from Controller 3: '))

    if (smoke_s1>= 20 & smoke_s2 >=20) | (smoke_s2>= 20 & smoke_s3 >=20) | (smoke_s3>= 20 & smoke_s1 >=20) :
        print('Warning: Smoke Detected')
        ### need to write AWS code
        msg = dt_string +'\n'+ "Warning! Smoke Limit Exceeded in your house "
        client.publish(
        PhoneNumber="+12269788073",
        Message= msg
        )
        time.sleep(1)
    else:
        print('No Smoke detected in the area!')
        time.sleep(1)
```

Fig. 8. Smoke Detection

```python
def temp_sensor():
    temp_s1 = int(input('Temperature value from the sensor: '))
    if(temp_s1 > 25):
        print('Warning: Too hot, turning on the AC')
        ### need to write AWS code
        msg = dt_string +'\n'+ "Warning! House is too hot, Please turn on the Air Conditioner switch"
        client.publish(
        PhoneNumber="+12269788073",
        Message= msg
        )
        time.sleep(1)
    elif(temp_s1 < 16):
        print('Warning: Too cold, turning on the Heater')
        ### need to write AWS code
        msg = dt_string +'\n'+ "Warning! House is too cold, Please turn on the Heater switch"
        client.publish(
        PhoneNumber="+12269788073",
        Message= msg
        )
        time.sleep(1)
    else:
        print('No action required, ideal temperature at house!')
        time.sleep(1)
```

Fig. 9. Temperature Detection, Monitoring and Control

```python
def humidity_sensor():
    # Humidity Sensor -> Ideal range is 30 to 50%
    humidity_s1 = int(input('Humidity value from the sensor: '))
    if(humidity_s1 < 30):
        print('Warning: Humidity range not appropriate')
        ### need to write AWS code
        msg = dt_string +'\n'+ "Warning! Humidity is low, Please turn on the humidifier"
        client.publish(
        PhoneNumber="+12269788073",
        Message= msg
        )
        time.sleep(1)
    else:
        print('No action required, humidity levels are proper!')
        time.sleep(1)
```

Fig. 10. Humidity Detection and Monitoring

```python
def co2_sensor():
    # Carbondioxide (CO2) Sensor -> Ideal value is below 800ppm
    co2_s1 = int(input('PPM value from the CO2 sensor: '))

    if(co2_s1 > 800):
        print('Warning: CO2 levels are high')
        ### need to write AWS code
        msg = dt_string +'\n'+ "Warning! CO2 levels exceeded in the house. Please evacuate or open the windows"
        client.publish(
        PhoneNumber="+12269788073",
        Message= msg
        )
        time.sleep(1)
    else:
        print('No action required, ideal CO2 at house!')
        time.sleep(1)
```

Fig. 11. Co2 detection and Monitoring

```python
def face_recog():

    videoCaptureObject = cv2.VideoCapture(0)
    result = True
    while(result):
        ret,frame = videoCaptureObject.read()
        cv2.imwrite("unknown_person.jpg",frame)
        result = False
    videoCaptureObject.release()
    cv2.destroyAllWindows()
```

Fig. 12. Face detection Code

```python
#comparing the images
img1 = face_recognition.load_image_file("Picture_new.jpg")
img2 = face_recognition.load_image_file("unknown_person.jpg")
img1_enc = face_recognition.face_encodings(img1)[0]
img2_enc = face_recognition.face_encodings(img2)[0]
result = face_recognition.compare_faces([img1_enc],img2_enc)
if result[0] == True:
    print("Access Granted")

    time.sleep(1)
else:
    print("Security Breach")
    msg = dt_string +'\n'+ "Warning! Unidentified person is trying to enter your house"
    client.publish(
    PhoneNumber="xxxxxxxxxx",
    Message= msg
    )
    time.sleep(1)
```

Fig. 13. Face Comparison and Secure Entry Code

IV. Experimental Results

This section explains in depth of the simulation of our sensors communicating to the control station wirelessly. We have also demonstrated the simulation of python and amazon SNS system which takes sensor values as the input and sends notifications to end user using the SNS service. This section is divided into two parts. First section shows the network simulation in Omnet++ and the second part is simulation of implementation of our logic in Python and Amazon SNS environment.

A. Simulation Results of Sensor Networks in Omnet++.

As mentioned the simulation of the network sensors is done for the period of 900 seconds. The Figure 14 shows the snapshot of the simulation of the network sensors during a particular time stamp. In the figure it is seen that all sensors first send the packets to their access-points. After that each of the five access-points send the packet to the central hub 'host C' or the controller. During a particular time stamp it is seen that total packets sent by three smoke sensors is 273, packets received by 'host C' or controller from humidity, temperature and Co2 sensors are 30 each while the packets received by 'host C' from camera sensor is 22.

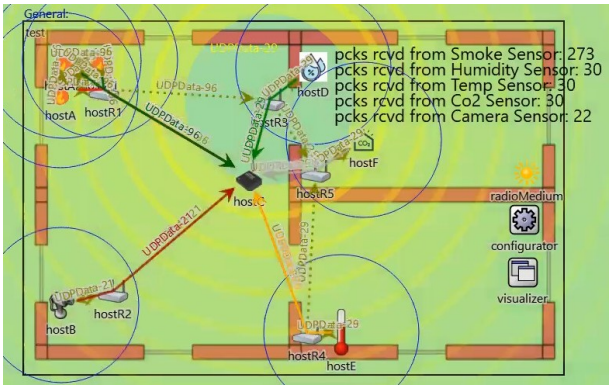

Fig. 14. Omnet++ Simulation in Progress

Figure 15 shows the the final values of packets received after the period of 900s of simulation in Omnet++. It is seen that the total amount of packets received from the three smoke sensors are 539. The packets received from the humidity, temperature and Co2 sensors are 59 each and the total amount of packets received from camera sensors by 'host C' is 44.

Figure 16 shows the comparison of Packet Delivery Ratio of each type of sensor. Packet Delivery Ratio (PDR) is defined as the ratio of total number of packets received to the packet sent. It is seen that the packet delivery ratio for the smoke sensors is the highest at 99.81%, while the PDR of Co2, Temperature and Humidity sensors are all same at 98.33% and the PDR of Camera sensors is recorded as 97.78%. The reason that PDR of the smoke sensors is highest this can be due to the fact that all of the smoke sensors are in vicinity of access point i.e. 'host R1', therefore lesser packet loss.

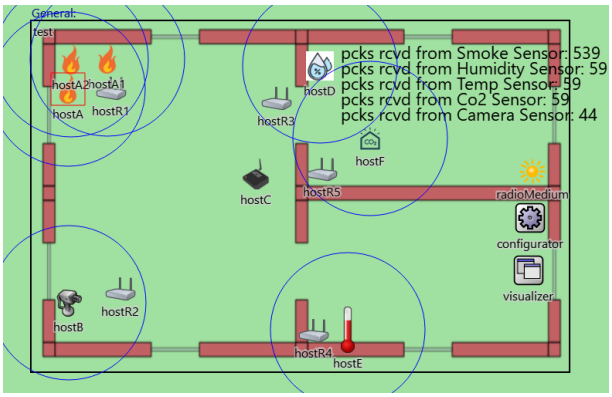

Fig. 15. Completion of Simulation in Omnet++

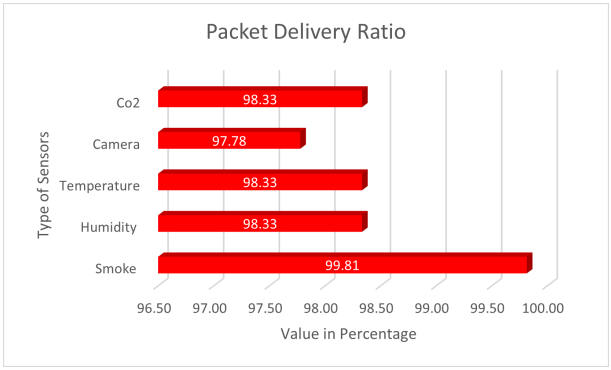

Fig. 16. Packet Delivery Ratio

Figure 17 shows the comparison results of end-to-end delay or latency period in seconds of different types of sensors used during the simulation. Latency period can be defined as the average amount of time taken by packets to reach from source to destination. Latency is dependent on the distance between the source and destination. It can be clearly observed from the figure that Co2 and Humidity sensor being closest to the controller has the least amount of latency period of 0.081s and 0.115s respectively. The latency of Smoke, Camera and Temperature sensor is 0.33s, 0.31s, 0.27s respectively.

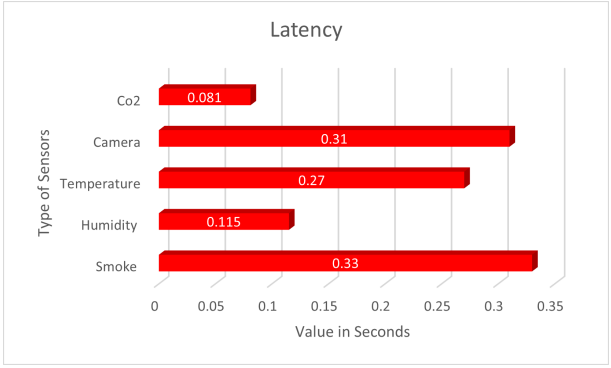

Fig. 17. Latency

B. Simulation of Sensor Data in Python and Amazon SNS

The python code is an infinite while loop as it must continuously read values from the sensors. Thus, the python simulation runs in an equivalent way where the user is prompted to select from a given list of options where each option assesses the functionality of that sensor. The execution terminal looks as in figure 18.

```
Please select one of the following features:
1. Smoke Detector
2.Temperature Sensor
3. Humidity Sensor
4. Co2 Sensor
5. Face Recognition
6. Exit
```

Fig. 18. Functionality selection Menu

As shown in figure 19, the smoke detector functionality is selected. Since 2/3 values are more than 20, a warning is issued, and message is sent to the user on his registered number as seen in figure 25

```
You have selected option: 1
LEL value from Controller 1: 45
LEL value from Controller 2: 67
LEL value from Controller 3: 18
Warning: Smoke Detected
```

Fig. 19. Smoke Detector Output by Python Simulation

Similarly, a warning is triggered when the temperature sensor is selected, and temperature range is more than 25 or less than 16. The python output is shown in figure 20 and message output is shown in figure 25

```
You have selected option: 2
Temperature value from the sensor: 34
Warning: Too hot, turning on the AC

Please select one of the following features:
1. Smoke Detector
2.Temperature Sensor
3. Humidity Sensor
4. Co2 Sensor
5. Face Recognition
6. Exit
2
You have selected option: 2
Temperature value from the sensor: 13
Warning: Too cold, turning on the Heater
```

Fig. 20. Temperature Detector Output by Python Simulation

The humidity sensor message is triggered since its value is less than 30. The output is shown in figure 21 and figure 25.

The Co2 sensor message is not triggered as the ppm value is 450. However, when the value entered is 879, which is more

```
You have selected option: 3
Humidity value from the sensor: 12
Warning: Humidity range not appropriate
```

Fig. 21. Humidity Detector Output by Python Simulation

than 800, a message regarding these high levels is sent to the user as shown in figure 22 and 25.

```
You have selected option: 4
PPM value from the CO2 sensor: 450
No action required, ideal CO2 at house!

Please select one of the following features:
1. Smoke Detector
2.Temperature Sensor
3. Humidity Sensor
4. Co2 Sensor
5. Face Recognition
6. Exit
4
You have selected option: 4
PPM value from the CO2 sensor: 879
Warning: CO2 levels are high
```

Fig. 22. Co2 Detector Output by Python Simulation

The face detection functionality is using the local laptop camera for simulation purpose. When any person is present in front of the camera, his/ her face is detected, captured and compared with the image stored in the database. Access is granted in the first test as shown in figure 23 since it has recognized the correct user. However, in the second test a random person's image is stored in the database for comparison. Since the faces were different a message was triggered to the user about the intrusion as shown in figure 24 and 25.

```
Please select one of the following features:
1. Smoke Detector
2.Temperature Sensor
3. Humidity Sensor
4. Co2 Sensor
5. Face Recognition
6. Exit
5
You have selected option: 5
Access Granted
```

Fig. 23. Face Detection pass case by Python Simulation

The figure 25 shows all the messages received by the user for each of the following cases. All these functionalities have been meticulously tested and it yielded 100% accuracy in informing the user.

V. Conclusion

The simulation of sensor networks was successfully implemented using parameters defined in Omnet++ environment.

```
Please select one of the following features:
1. Smoke Detector
2.Temperature Sensor
3. Humidity Sensor
4. Co2 Sensor
5. Face Recognition
6. Exit
5
You have selected option: 5
Security Breach
```

Fig. 24. Face Detection fail case by Python Simulation

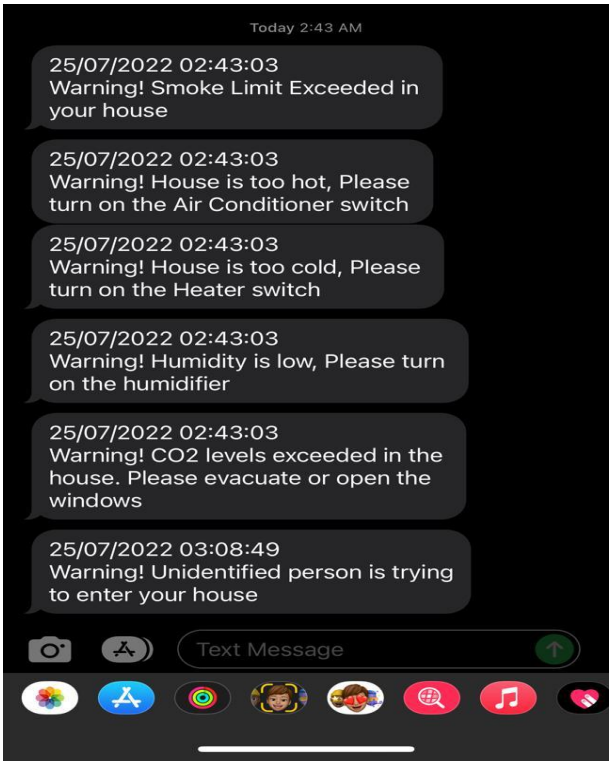

Fig. 25. Messages to the user

The simulation results like packets received from all the five types of sensors, Packet Delivery Ratio, Latency period show results that successfully prove our methodology and setup configurations. The type of sensor data and threshold values of all five type of sensors are taken from reviewing the research papers. Finally, the logic of our implementation is simulated in python environment on local station and a successful notification message is sent via Amazon SNS service to notify the end user in case of abnormality.

References

[1] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision architectural elements and future directions", Future Generation Computer Systems, vol. 29, pp. 1645-1660, 2013

[2] B. Li and J. Yu, "Research and application on the smart home based on component technologies and Internet of Things", Advanced in Control Engineering and Information Science, vol. 15, pp. 2087-2092, 2011

[3] Y. Amri and M. A. Setiawan, “Improving Smart Home Concept with the Internet of Things Concept Using RaspberryPi and Node MCU,” in IOP Conference Series: Materials Science and Engineering, 2018, vol.325, no. 1, p. 012021
[4] J. Y. a. S.-S. Lee, ”Human Movement Detection and Identification Using Pyroelectric Infrared Sensors,” NBCI, 2014
[5] W. Rahman, H. A. Rashid, R. Islam, and M. M. Rahman, “Embodiment of IOT based Smart Home Security System,” ijraset, 2018
[6] A. Nag, J. N. Nikhilendra, and M. Kalmath, “IOT Based Door Access Control Using Face Recognition,” 2018 3rd International Conference for Convergence in Technology (I2CT), 2018
[7] J. Joshi et al., “Performance enhancement and IoT based monitoring for smart home,” in 2017 International Conference on Information Networking (ICOIN), 2017, pp. 468 – 473
[8] D. B. Adriano and W. A. C. Budi, “Iot - based integrated home security and monitoring system,” in Journal of Physics: Conference Series, 2018, vol. 1140, no. 1, p. 012006
[9] Suresh S, Yuthika S and G. Adithya Vardhini, “Home Based Fire Monitoring and Warning System” (978-1-5090-5515-9/16/$31.00 IEEE 2016)
[10] Jayashri Bangali, Arvind Shaligram ”Design and Implementation of Security Systems for Smart Home based on GSM technology ”, International Journal of Smart Home Vol.7, No.6 (2013), pp.201- 208
[11] Aayush Aggarwal, R.C. Joshi, ”WSN and GSM based Remote Home Security System”, International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings published in International Journal of Computer Applications® (IJCA)
[12] Manish, Dejan, Chowdhury, G. R., Nedelkovski, D., Jory, Slater, C.-topher, Marius, Jesper, Davi, Klaatu, Hamid, Kousik; Raan. (2022, February 18). DHT11 amp; DHT22 sensors temperature and humidity tutorial using Arduino. How To Mechatronics. Retrieved August 1, 2022, from https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/
[13] Smoke sensor. FIBARO Manuals. (n.d.). Retrieved August 1, 2022, from https://manuals.fibaro.com/smoke-sensor/
[14] Temp Stick Wireless Temperature Sensor + 24/7 monitoring, Alerts amp; Unlimited Historical Data. connects directly to WIFI. free iphone and Android app, check-in from anywhere! - . Retrieved August 1, 2022, from https://www.amazon.ca/Wireless-Temperature-Monitoring-Unlimited-Historical/dp/B01M1OPOZB?th=1