SENIOR DESIGN PROJECT

SMART HOME – YOUR EAGLE'S NEST

STUDENTS:

Giorgi Solomnishvili

Lasha Butkhuzi

Reziko Tsirgvava

Mikheil Makharadze

Aleksi Aleksandria

San Diego State University

15/05/2021

Contents

Abstract

The problem our team chose to work with is the safety and the security of residential areas. We utilized an IoT server that connects the mobile application to devices in the residential area. At home, we installed several sensors that can detect an intruder, fire, and gas leak. We made it possible for the owner to monitor his/her home's safety and security from any place in the world. We implemented a module that can identify humans intelligently. The module determines if a person approaching the residential area is a tenant, a friend of tenants, or a stranger. Furthermore, we installed hardware that tries to extinguish hazards and notifies the owner about their existence.

Introduction

In the 21st century, Safety and Security have become one of the most critical and precise commodities, which are very hard and expensive to come by. Nowadays, we live in a world where nobody is wholly protected from becoming victims of crimes and accidents such as intrusion, robbery, arson, fire, malfunctioning utilities, etc. We have been hearing stories about occurrences of random fires in different parts of Georgia, of accidents caused by having not correctly installed and managed utility systems, and of violent crimes resulting in loss of life.

The reasons mentioned above motivated our team at SDSU_Georgia to dedicate our time and funding to researching sophisticated and relatively inexpensive solutions to the Residential Area's Safety and Security.

We address safety by implementing modules that can detect hazards and generate appropriate responses to limit the damage caused by them. One of the most important accomplishments of our project is making safety monitoring an automated process by utilizing the Internet of Things (IoT). We have sensors that generate information about the safety of the residential area and send that information to the IoT cloud where appropriate response is automatically generated. As a result, our project can turn an ordinary house into a safe home.

We attempted to solve security by implementing a module that intelligently identifies and distinguishes the tenants from ill-intended strangers and potential intruders. We accomplish this by utilizing a trained Artificial Intelligence (AI) system to recognize faces using deep learning. To prevent intrusion and crimes associated with it, our design makes the residential area accessible to only authorized people.

Overall, the solution to the lack of safety and security is implementing an IoT system to generate information about home safety and an AI system to identify potential intruders.

Body

The main goal of we aimed to achieve is the safety and security of a home. Figure 1 represents a block diagram of our project. Our design revolves around the IoT cloud. Our design monitors the amount of Gas, carbon dioxide (CO2), and volatile organic compounds (VOC) in the air for monitoring and maintaining safety. It can detect smoke which is a precursor to fire. Suppose smoke is detected, or the concentration of either compound mentioned above is increased. In that case, the IoT cloud notifies the existence of the hazard to authorities and the homeowner.

Furthermore, if smoke is detected, the water pump is activated to contain and, if possible, eliminate the fire. To ensure the home's security, we use surveillance cameras that can recognize the tenants' faces, and we use motion detectors that send information about the existence of an intruder to the IoT cloud, which in turn activates the GSM module notifying owner and authorities about the intrusion. The specification and functioning of

each sensor, module, and IoT cloud will be discussed thoroughly in the following paragraphs.

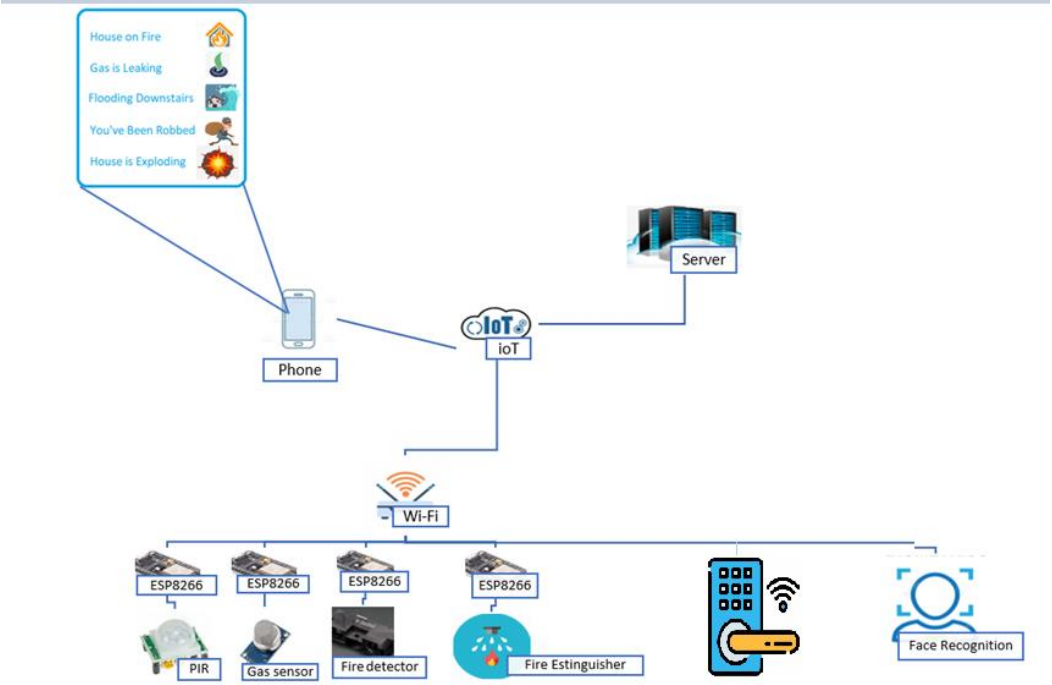


Figure 1

IoT Server and Connectivity

IoT systems are one of the significant tendencies that are changing the world. IoT systems can be understood as a network of a physical object such as embedded sensors, action devices, controller software, etc. IoT technologies are widely used in real-time analytics, machine learning, embedded systems, designing intelligent home devices and appliances, etc. Our project uses IoT to connect fire, gas motion sensors to action devices such as GSM module, fire extinguisher sprinkler, and web application.

The sensors and IoT system exchange information via the Message Queuing Telemetry Transport (MQTT) protocol. This protocol operates at the TCP/IP layer of the internet, supporting bi-directional communication between devices. The devices connected with the MQTT protocol can publish and subscribe to information. MQTT protocol distinguishes between two kinds of network elements: a broker and a client. MQTT broker is a server or cloud that receives messages from publisher devices and routes them to the subscriber clients. Clients can be any devices that are connected to MQTT broker with MQTT protocol. The information exchanged by devices is labeled with topics. Devices can publish data to a particular topic, sending a message to the broker a control

message (Topic). The broker distributes that information to devices that have subscribed to that topic. At first, clients should establish a connection with the broker. After which, they can subscribe to particular topics and publish to them. Figure 2 portrays the client-broker communication process. Client A subscribes to the topic temperature/roof and publishes to that same topic[4]. The same can be said about Client B.

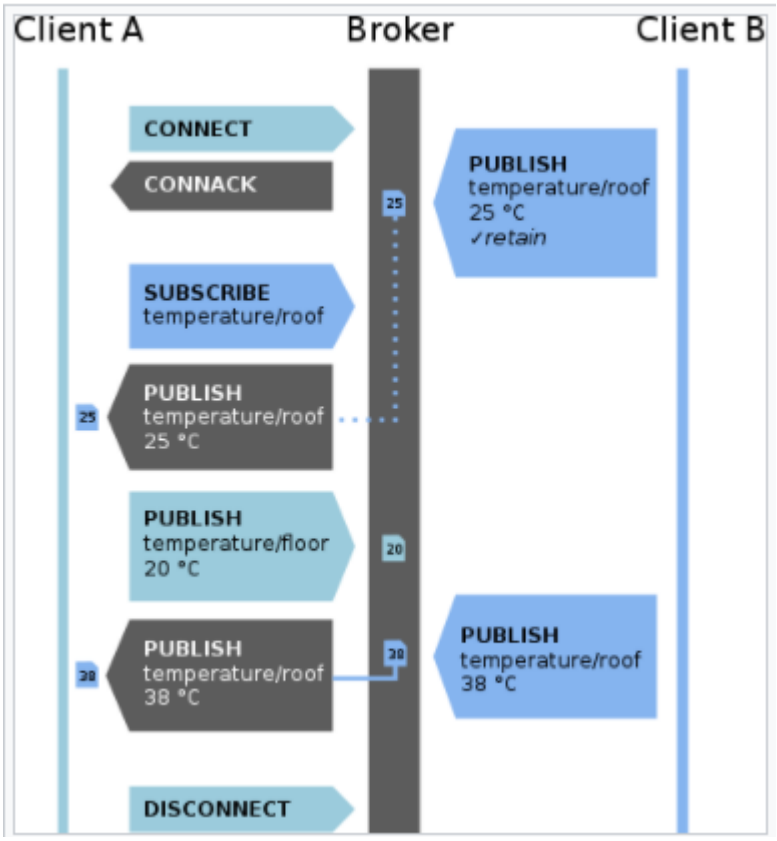


Figure 2

MAXIOT

The MQTT broker utilized in our project is called MAXIOT. The broker requires to be installed on Linux operating system, and it should be running all the time. Therefore, our team rented a remote computer from the digital ocean. The computer has an 80 GB hard disk and 2 GB RAM. We installed Ubuntu on that remote computer and created a user called the IoT with the following commands:

Create a user called iot - Sudo useradd -m iot

Set password iot - Sudo passwd iot

Give the user root privileges - usermod -aG sudo iot

The run file for MAXIOT_SERVER is stored in etc folder and is automatically run as the computer reboots. This automation was achieved by providing a path to run the file in rc.local. The MQTT broker is running on port 4004. If devices want to connect to the broker, they should send the connection request to the public IP of our remote computer

at 4004 port. To efficiently control MAXIOT, we obtained a studio that enables us to establish and destroy connections of the devices with the broker and each other. Figure 3 portrays the studio of our project. On display, two kinds of objects can be seen: Devices and Mediators. Each object has its unique ID.

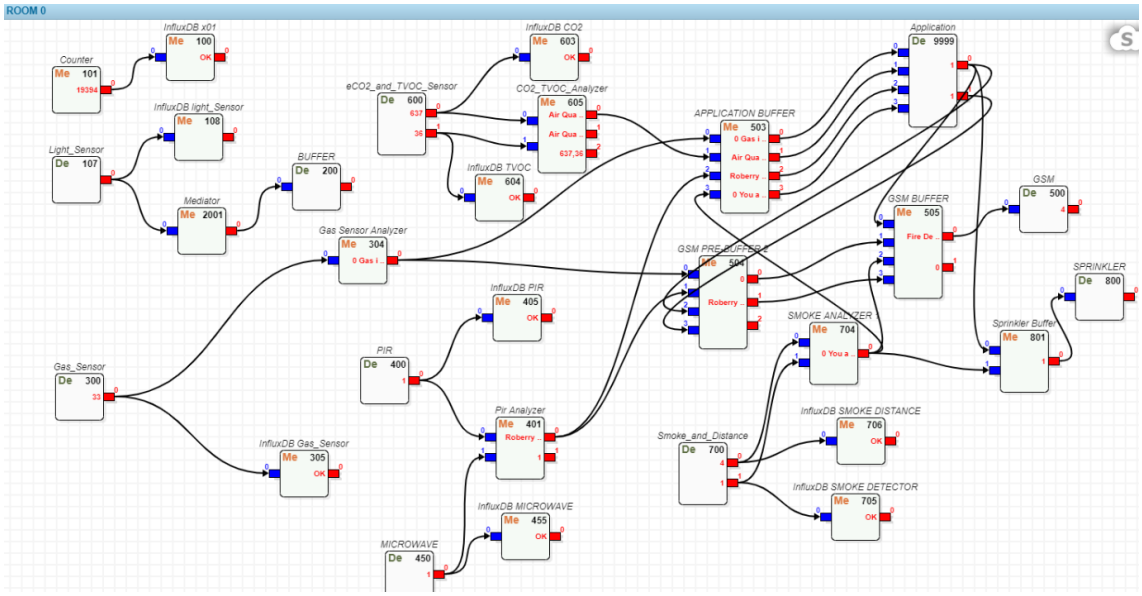


Figure 3

Devices are avatars of actual sensors and action devices. The real, physical sensors connect to their avatars. Sensors send information to their avatars. For example, in figure 3, a device with ID 300 is an avatar for a gas sensor. In every 10 seconds, the gas sensor publishes its reading to a topic 0.

Furthermore, the information that needs to be sent to action devices is delivered to their mediators. For example, a Device with ID 800 is an avatar of a fire extinguisher sprinkler. That Device is subscribed to topic 0, and whenever information is published to topic 0, sprinkler receives it.

Mediators analyze information received from devices and generate an appropriate response that should be delivered to action devices. For example, in figure 3, Mediator with ID 305 stores data published by the gas sensor to databases. Moreover, Mediator with ID 801 analyses information publishes by smoke detectors, and if the fire is detected, the mediator activates the sprinkler.

We used the ESP8266 module (Figure 4) to connect sensors to the IoT server. ESP8266 and IoT servers can communicate using the MQTT protocol.

Figure 4

Eclipse Paho provides us with valuable libraries for MQTT communication. One such library specifically targeted for ESP devices is PubSubClient (Source code for the connection to MQTT Broker can be seen in related materials).

At first, ESP connects to WiFi with the following code:

```
Serial.begin(9600);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

    delay(500);
    Serial.println("Connecting to WiFi..");

}

Serial.println("Connected to the WiFi network");
```

SSID and PASSWORD are the name and passwords of the home owner's WiFi. WiFi.begin starts the connection process. While loop checks connection status and polls until the connection is not established. Next, the esp needs to connect to its avatar on the MQTT broker. The following code establishes this connection:

```
client.setServer(mqttServer, mqttPort);
client.setCallback(callback);

while (!client.connected()) { // client.con
    Serial.println("Connecting to MQTT...")

    if (client.connect("200/BUFFER")) { //

      Serial.println("connected");
      client.publish("0", "4");
      delay(500);
      client.subscribe("0");
    } else {

      Serial.print("failed with state ");
      Serial.print(client.state()); // tell
      delay(2000);

}
```

The first two lines initiate the connection to mqttServer – the public IP of our remote computer, at port mqttPort – 4004. The while loop checks connection status and polls until the connection is established. The if statement executes command client.connect("200/BUFFER"). This command attempts to connect ESP to theDevice with ID 200 and the name BUFFER. If the connection is established, the esp publishes the first message "4" to topic 0 and subscribes to topic 0. Figure 5 displays the establishment of the connection

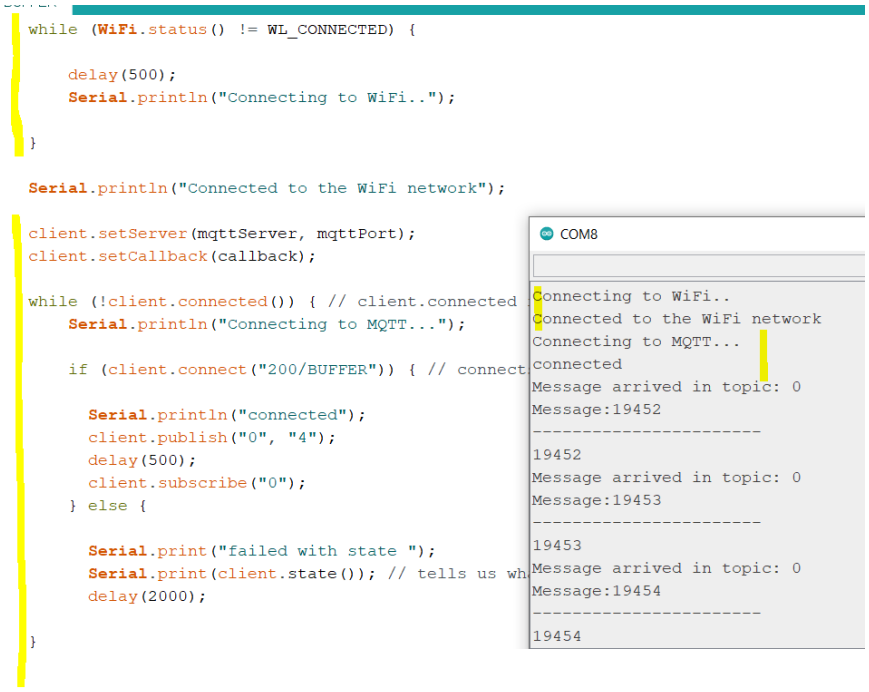


Figure 5

After the connection is made, the esp can publish to topics from 0 to 8 and subscribe topics from 0 to 4. Whenever the information from subscribed topic arrives, a callback function is executed:

```
void callback(char* topic, byte* payload, unsigned int length) {
  char char1[length+1];                                  // This will be
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);

  Serial.print("Message:");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
    char1[i] = (char)payload[i];
  }
  char1[length] = '\0';
  //message = char1;
  Serial.println();
  Serial.println("-----------------------");
```

This function receives messages byte by byte and saves them in a char array called char1.

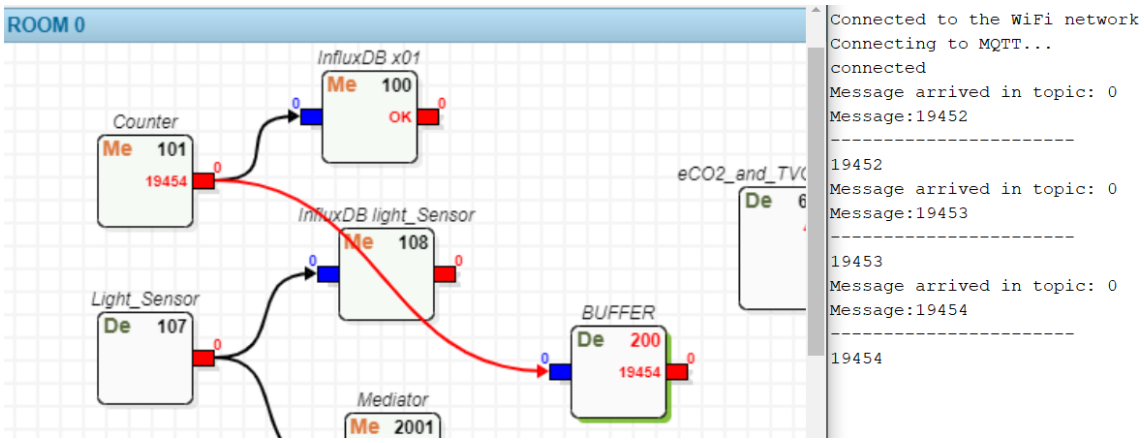


Figure 6

Figure 6 displays an example of publishing/subscribe. An ESP is connected to its avatar with ID 200 and the name BUFFER. The ESP is subscribed to topic 0. Mediator with ID 101 and name counter publishes a number every 10 seconds to topic 0 of BUFFER. This number is delivered to ESP (the right half of the screen represents data received by ESP). Next, the ESP publishes the received data to topic 0. For example, if Counter published 19454, that number will be delivered to ESP (The last line on the right half of the picture) and then published to topic 0 of the Device called BUFFER.

InfluxDB

As mentioned above, we have mediators that save information to databases. In the beginning, we were going to use MySql databases. However, while researching that topic, we found out that MQTT Brokers work better and faster with the influx DB databases. The research revealed that influxDB is optimized for real-time analysis and fast storage and retrieval of data. Therefore, we decided to switch to influxDB. It was installed on our remote computer with the following command:

```
sudo su
echo "deb https://repos.influxdata.com/ubuntu bionic stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
sudo apt-get update
sudo apt-get install influxdb
sudo systemctl enable --now influxdb
systemctl status influxdb
```

The influxDB is listening on port 8086. We created an influx database user called "hopeuser" and a database called "hopedb" with the following commands:

```
create user <username> with password <'password'>
create database <databaseName>
```

On the MAXIOT, we have mediators that send received data to databases. MAXIOT allows us to program mediators with python code. We use the influxDB python library to make it possible for the mediator to connect to the influxDB database and create and update measurements (See figure 7).

```
7  from influxdb import InfluxDBClient
8  import datetime
9
10 client = InfluxDBClient(host='localhost', port=8086, username='hop
11 client.get_list_database()
12 client.switch_database('hopedb')
13
14
15 current_time = datetime.datetime.utcnow().isoformat()
16 current_Velue = float(IN_0)
17 json_body = [
18     {
19         "measurement": "Gas_Sensor",
20         "tags": {},
21         "time": current_time,
22         "fields": {
23             "value": current_Velue
24         }
25     }
26 ]
27
28
```

Figure 7

Each measurement consists of two columns. The first one is used as an ID and is filled with the current time. The second one is used for data values. Line 16 on Figure 7 takes data subscribed from topic 0 and sends that information to the measurement. Line 19 specifies the name of the measurement.

Grafana

Up to this point, we have a server that receives/sends data from/to clients and stores data into databases. We could see the values stored in the database from the command line. However, we wanted to have a method to visualize data more clearly. While researching for such methods, we discovered the existence of grafana.

Grafana is an open-source web application used to create graphs based on connected data sources. One of the data sources supported by Grafana is influxDB databases. Therefore, we decided to use Grafana as a visualization tool. We installed the OSS version of Grafana because it was a stable and well-tested one. However, while we were researching Grafana, we found out that the Enterprise version was being developed, which would provide the users with more safety. In the future, when the Enterprise version becomes stable, we would like to switch to the Enterprise version[2].

We installed Grafana on our remote computer. It runs on port 3000. Grafana allowed us to create a dashboard – The Eagle's Nest, and panels within the dashboard for displaying the graphs. We have panels for Counter, VOC, Smoke Distance, Smoke Detector, PIR and MICROWAVE Motion Sensor, Gas Sensor, and CO2. Figure 8 displays panels for the Gas sensor and CO2.

Figure 8

Initially, Grafana was configured in a way that it only allowed access to the dashboard through authorization. Since we wanted to embed Grafana panels in our web application, we changed the configuration and enabled anonymous authorization. See Figure 9. This was achieved by modifying the grafan.ini file, which is stored in our remote computer.

Figure 9

Web Application

Our project allows the homeowner to monitor and control his/her home with the web application. We used a relatively new and effective script language – MS script, to create a web application. The script language is built on JavaScript libraries – jQuery and paho.

The application creates a WebSocket, and this WebSocket connects to the MQTT broker, MAXIOT, as a device. Just like sensors and action devices, the web application has its avatar with ID 9999. If a person wants to sign into his monitoring and control system, he should enter an ID. Each customer will have a unique interface because they will have a different home. Therefore, each customer will need a different avatar for their application. Moreover, part of the application that enables monitoring and controlling should be other for each client. Fortunately, MS script can be

Devce Config:

```
<1:99>
    ..[page],3
..{
    ..header_text,Monitoring and Control Panel,
    ..header_button,Configuration, ui-btn ui-btn-icon-notext ui-co
    ..header_button,name, ui-btn ui-btn-icon-notext ui-corner-all
        Page_change('2');

    ..js,
        function Refresh_isConfig2(){
            put_input_text('it_pop2_Host',Read_Memory('mem_Host'))
            put_input_text('it_pop2_Port',Read_Memory('mem_Port'))
            put_input_text('it_pop2_Device_ID',Read_Memory('mem_De
            put_input_text('it_pop2_Device_Name',Read_Memory('mem_
            if(Connect_Status === 1){
                put_input_text('it_pop2_Status',"Connected");
            }
            else{
                put_input_text('it_pop2_Status',"Disconnected!!!")
            }
        }
```

written in the avatars, just like in the case of mediators. This way, we will be able to accommodate the application to the customer's needs. See Figure 10.

The application allows users to see sensor values, Grafana panels, and surveillance camera streams (See Figure 11a,b,c).

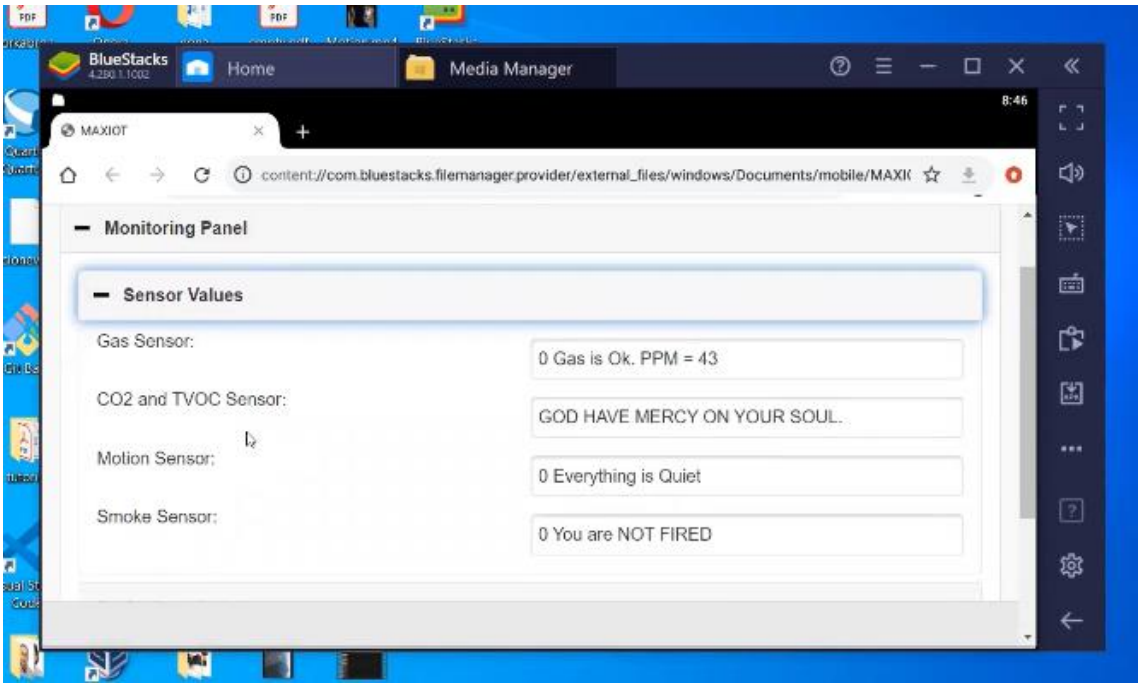


Figure 11a

Figure 11b

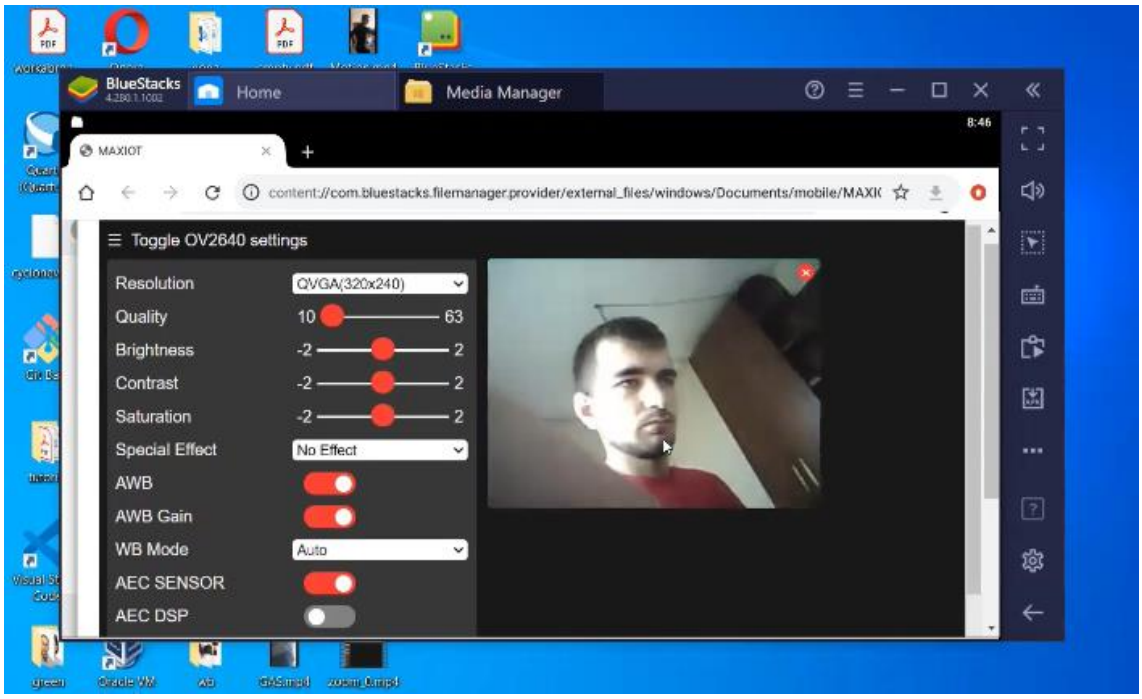
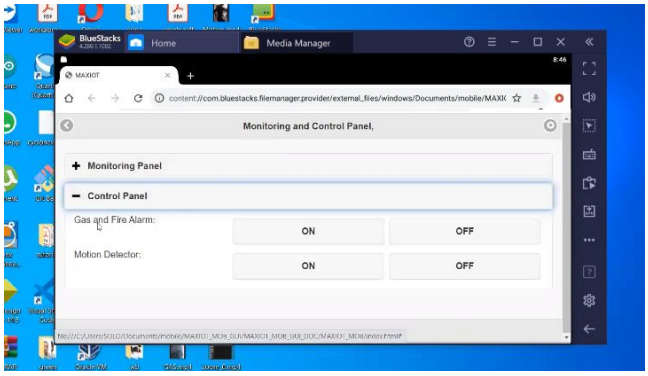


Figure 11 c



The Web Application allows users to control action devices in his/her home. At this point, we offer two action devices – A motion detector and a fire gas alarm system. Motion, Gas, and fire sensors are reading information all the time. However, the alarm and the detector will act based on the sensor values if they are turned on from the Web Application.

Figure 12

MAXIOT Configuration

We have 8 sensors and devices connected to MAXIOT (See table 1).

Name of Avatar	**ID of Avatar**	**Publish/Subscribe**
Gas Sensor	300	**Publishes**: Topic 0 – the amount of CH4 in the air.
PIR Motion Sensor	400	**Publishes**: Topic 0 – reading of PIR sensor.
MICROWAVE Motion Sensor	450	**Publishes**: Topic 0 – reading of MICROWAVE sensor.
CO2 and TVOC Sensor	600	**Publishes**: Topic 0 – the amount of CO2. Topic 1 – the amount of TVOC.
Smoke and Distance detector	700	Publishes: Topic 1 – 0 if smoke is detected. Topic 0 – distance from the nearest object.
GSM	500	Subscribes: Topic 0 – information about existing hazard or intrusion
SPRINKLER	800	Subscribes: Topic 0 – 1 is published on topic 0 if the fire is detected
APPLICATION	9999	**Subscribes**: Topic 0 – Reading from the gas sensor. Topic 1 – Reading from CO2 sensor. Topic 2 – Reading from Motion sensors. Topic 3 – Reading from fire sensor. **Publishes**:

		Topic 0 – Activation signal for fire and gas alarms. Topic 1 – Activation signal for fire detector.

Table 1

Sensors and Action Devices

Our design can detect a gas leak, fire, and intruder. Furthermore, we have a water pump to extinguish the fire and a GSM module to notify the owner and authorities about the hazard's existence.

Gas Leak Detection

The other feature of the smart, secure, and safe house is detecting an excessive amount of Methane. In that particular case, we use a gas sensor MQ7, also known as "Chemiresistor." Chemiresistive materials change their electric resistance in response to the change in the chemical composition of the environment. The research indicates that MQ can detect a change in Methane concentration, Carbon dioxide, and hydrocarbon gas-liquid. The concentration of the elements mentioned above increases when there is a gas leak. Thus, the MQ sensor enables us to monitor the concentration of chemical components of natural Gas.

MQ7 works on a 5V DC power supply. It is powered from an ESP 3V pin through a step-up converter that elevates 3.3V to 5V. The code downloaded in that ESP enables us to detect a critical concentration of CH4 and then send a warning signal to the IoT server. Other devices like GSM will be activated and send the message to the house owner.

Figure 13

Simultaneously, we have LCD screen NOKIA 5110 (Figure 13) connected to the ESP and MQ7 and displays CH4. ESP and NOKIA screens use SPI communication and libraries Adafruit_GFX and Adafruit_PCD8544 to exchange information. The screen operates within 2.7 – 3.3 voltages. Nokia screen has 8 pins – RST (reset), CE (chip enable), DIN (data in), CLK (clock), VCC (3v), BL (screen LEDs), and GND[1]. Nokia pins are connected to D0 – D4 digital pins of ESP. By having the screen, a homeowner can directly monitor the amount of Methane in the air.

WATER PUMP:

The smart, secure, and safe house should have sufficient facilities to put out the fire and prevent considerable damage caused by burning. For this purpose, we have a pump that is driven by a special pump that works on a 12V DC. The water pump that we purchased in the DAC store is powered by 12V. That amount of voltage is obtained via a 12V adapter connected directly to the socket (220V). The task of the pump is not to allow the water pump to flow water permanently but only in case of fire. Therefore, we used a relay that acts as a switch.

The relay itself is powered and controlled by ESP 8266. In our case, the relay requires 5V DC for adequate performance. To get 5V, we applied a step-up converter and connected its output to the input terminals of the relay. ESP is used not only for power supplying purposes but it receives signals from IoT servers. A unique code is written that receives a signal from the IoT server, either high (1) or low (0), and delivers it to digital pin D1. In the case of the high value of Signal, we have 3.3 V at pin D1, hence the 5V supply for the relay and operating. We have high value only in case of fire. Fire is detected by a smoke detector that sends a signal to the IoT server, and we use the same Signal as an indicator to turn on or off the relay. When the relay is turned on, the water pump is also turned on, and the water splashes. Now we have just a water pump and thus a direct water current. However, we can make progress in that field. We can also apply a water sprinkler that divides a typical water stream into tiny droplets and throw them in a 360-degree direction. This method can give us two advantages. First of all, it can be more efficient in terms of fire extinguishing. On the other hand, it can prevent damage caused by water itself since the pressure of droplets is considerably lower than those of direct water current.

Intruder (Motion) Detection:

Our design can detect a motion in the home. We used a Passive infrared (PIR) and Microwave sensor to detect the motion.

Figure 14

PIR sensor (Figure 14) detects motion by detecting the heat and rapid heat change in a residential area. Once the sensor warms up, it can detect heat and movement in the surrounding areas, creating a protective "grid." If a moving object blocks too many grid zones and the infrared energy levels change rapidly, the sensors trigger an alarm This rapid change of heat indicates that someone is moving into the home. PIR outputs a digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor. The sensitivity range of PIR is up to 6 meters

Microwave sensors (Figure 14) detects motion by utilizing electromagnetic radiation. The sensor can emit electromagnetic waves and capture reflected waves. The sensor can measure the time wave took to reach an object and come back to the source. Microwave sensor uses the doppler effect and projected microwaves to detect the motion.

If nobody is supposed to be at home and the motion is detected, this means that we have an intruder. There is an excellent chance that this intruder is a thief. The owner and the police will be notified about the intruder.

GSM Module:

One of the most vital components of the project is linked to the GSM module. The house can't be smart, safe, and secure if the owner does not know the current situation in his/her home. Just detecting hazards or intruders is not enough. We need to tell the owner about it. This issue is solved through GSM sim800C. It's a device that can send a message at any distance. We tested its range and found out that we can send SMS from Tbilisi to San Diego.

Our GSM is powered from Arduino Uno 5V pin. However, it is programmed through ESP8266. Arduino is only used for power supply purposes. The core techniques to send a message include so-called AT commands. Roughly speaking, it's a language we ask our GSM to do things (sending a message to a certain number). AT commands are included in the special code written in Arduino IDE.

At first, we check the signal quality of Signal with the command: AT+CSQ. Next, we check if the sim card is inserted: AT+CCID. The following step is to configure GSM so that it would send text SMS: AT+CMGF=1. After that, we should tell GSM cell phone number where SMS should be sent AT+CMGS=+995xxxxxxxxx. The codes are uploaded in related materials.

In our IoT server, GSM is linked to several devices. Thus it sends messages according to the Device that emits the Signal. For example, if the gas sensor is sending 1s, there is a critical amount of CH4 in the house, so GSM sends a message to the house owner and notifies of the potential danger. While testing the GSM module, we came across a problem. While it was sending SMS, it was losing connection with MAXIOT. Our code is configured so that if the connection to the IoT server is lost, ESP tries to reconnect it. The reconnection process interfered with sending the SMS. As a result, in the beginning, GSM was able to send SMS 60% at a time. We solved this problem by configuring the code so that after the connection is re-established, GSM sends the last received SMS. That way, our module was able to send SMS 100% at a time.

INDOOR AIR QUALITY MONITORING SYSTEM

To monitor indoor air quality, we use ultra-low-power digital gas sensor CJMCU-811. This breakout board utilizes a digital air quality sensor module CCS811, which provides a convenient way to monitor a wide range of volatile organic compounds (TVOC) and particularly equivalent carbon dioxide (eCO2). The module integrates a metal oxide (MOX) gas sensor, a microcontroller unit (MCU), which includes an Analog-To-Digital

CO_2 [ppm]	Air Quality
2100	BAD Heavily contaminated indoor air Ventilation required
2000	
1900	
1800	
1700	
1600	
1500	MEDIOCRE Contaminated indoor air Ventilation recommended
1400	
1300	
1200	
1100	
1000	FAIR
900	
800	GOOD
700	
600	EXCELLENT
500	
400	

Table 2

Converter (ADC), and an I2C interface. By connecting to the ESP8266 (NodeMCU 1.0) microcontroller over I2C, it will return Total Volatile Organic Compound (TVOC) reading and an equivalent Carbon Dioxide reading (eCO2). The eCO2 output range for CCS811 is from 400ppm up to 32768ppm, and the TVOC content is from 0ppb up to 29206ppb. **Table 2** display the concentration of CO2 and VOCs in the air and how they affect its quality.

To wire the module to the microcontroller, we connected VCC to 3.3V, GND to GND, SCL to D1, SDA to D2, WAKE to GND. Other pins can be left unconnected. **Figure 15** shows the scheme of the hardware. For that scheme, we constructed PCB using Altium designer. For the software, we installed Adafruit CCS811 Library on Arduino IDE. We sent the information on our IoT server, where we wrote the logic for maintaining an acceptable level of air quality. If the bad quality of the air is detected, the user will be informed immediately via the phone. The message will contain the current air quality level and advice on what action to take. The user will also see the trends of eCO2 and TVOC on the **Grafana** website and monitor the current air quality level using the mobile application.

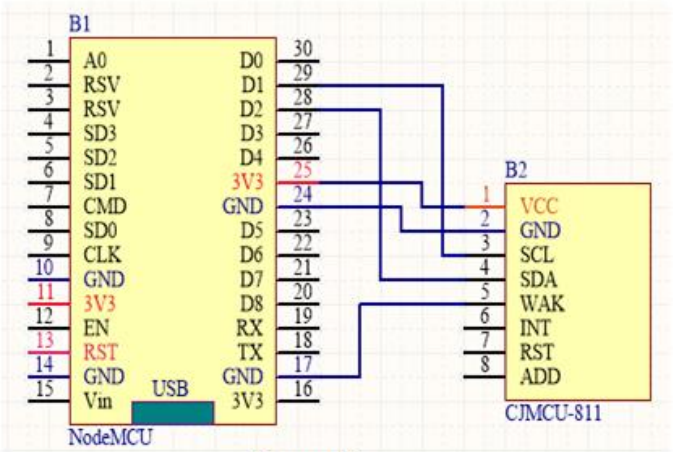


Figure 15

FIRE DETECTION AND ALARM SYSTEM

To detect smoke and notify the user about fire in the house, we designed a light scattering or photoelectric smoke detector, which operates on the Tyndall effect. Tyndall effect is a phenomenon in which a beam of light scatters on, striking the particles present in its path. This scattering of light is responsible for the blue color of the sky. In a light scattering smoke detector, a photocell and light source are separated from each other by a darkened chamber, such that the light source does not fall on the photocell (Figure 16). After the smoke enters the chamber, the light from the source is scattered and falls on the photocell (figure 17), which then triggers the alarm. To trigger the alarm, we designed a scheme, which is displayed in figure 18.

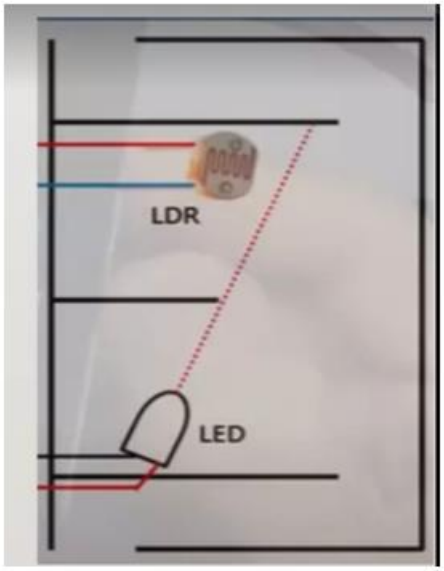


Figure 16

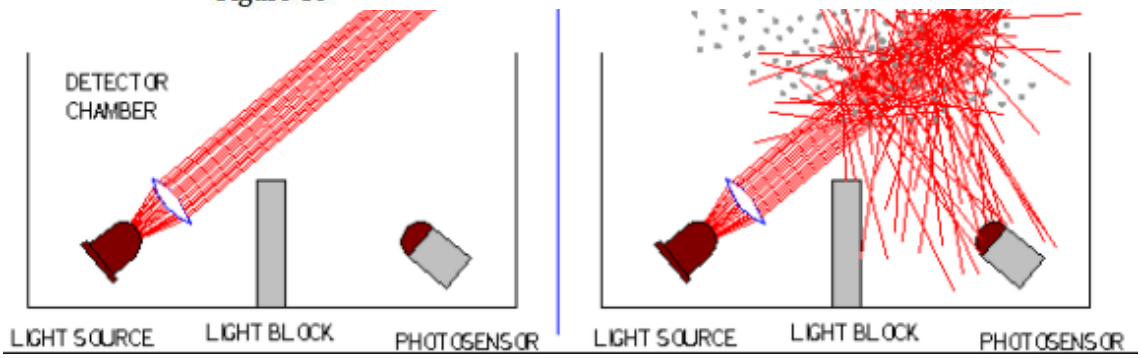


Figure 17

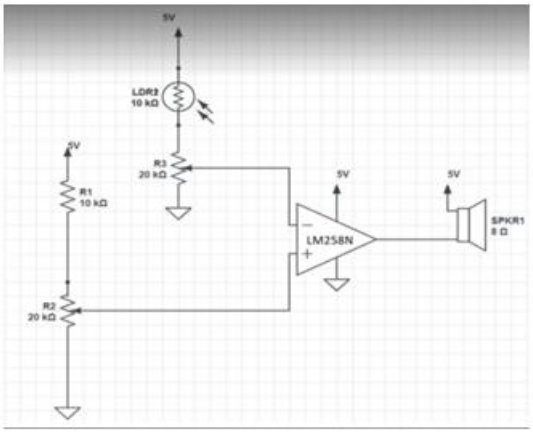


Figure 18

As the scattered light falls on the photosensor, we get the change in resistance in our case LDR. This change can be used to detect the change in voltage by the LM258N module easily. In the positive input terminal of LM258N, we have fixed voltage, which is greater than the voltage in the negative input terminal. So nominally, we have 5 volts on the output terminal, which doesn't trigger the buzzer. However, if the light is detected, the voltage in the negative terminal will rise, and the output will change to 0 volts, which

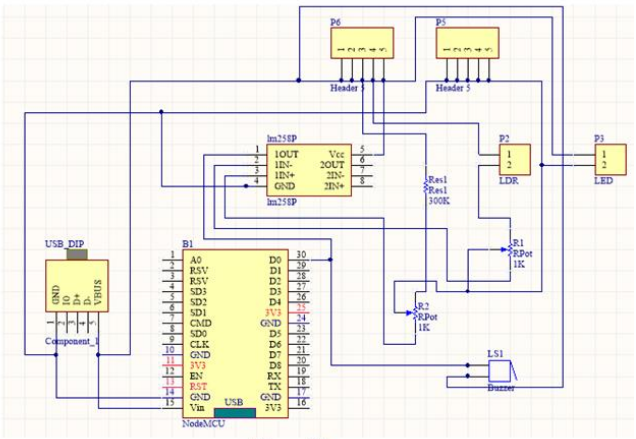


Figure 19

then triggers the buzzer. We also connected the output terminal of the LM258N module to the digital input pin of the ESP module to send the smoke detector status on our IoT cloud. For a final scheme in ALTIUM designer is displayed in figure 19. In the scheme, you will find two potentiometers to adjust the sensor's sensitivity in different environments. Figure 20 shows the complete smoke detector module with the sensor, the PCB part, and the mechanical part.

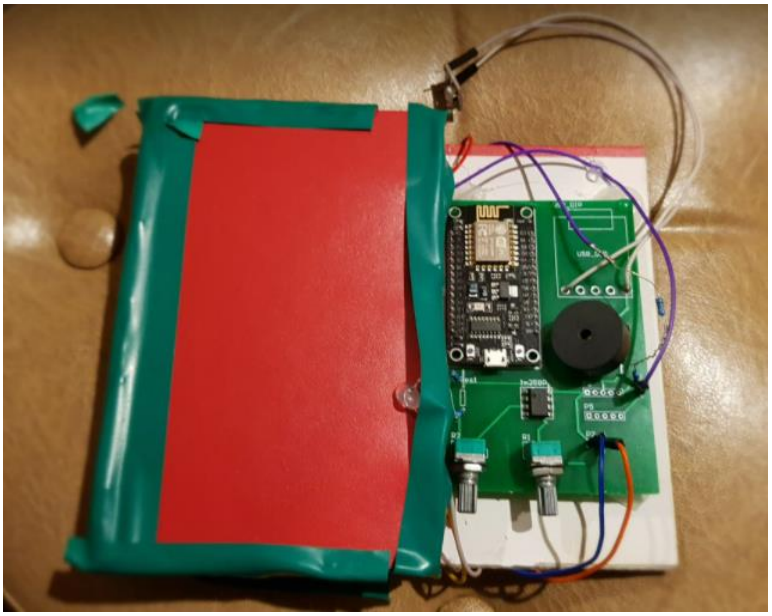


Figure 20

SMOKE DETECTION USING IR DISTANCE SENSOR

The second method we used to detect fire is monitoring the change in distance using an infrared distance sensor.

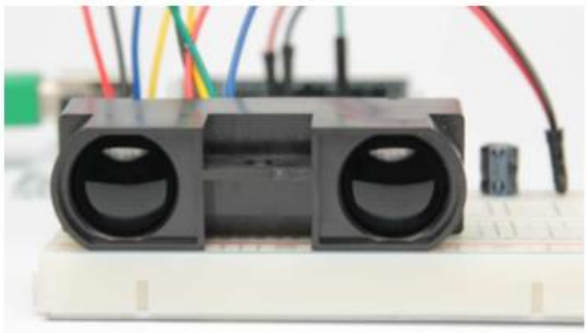


Figure 21

This sensor (figure 21) has two eyes, one sends the Signal, and the other catches the reflected Signal from the object. The time needed for the Signal to be sent until catching reflected is used to calculate the distance. In our case, because the sensor uses infrared light, we can detect heat or fire, which has high spectra of infrared light.

The Sensor Module has the range of detection from 1 to 4 meters, so it should be installed on the ceiling. When the fire occurs, smoke will go up, and the sensor will detect the rapid change in distance. If there is no fire, the difference in distance is around 1 to 2 cm. The scheme for the sensor is displayed in figure 22.

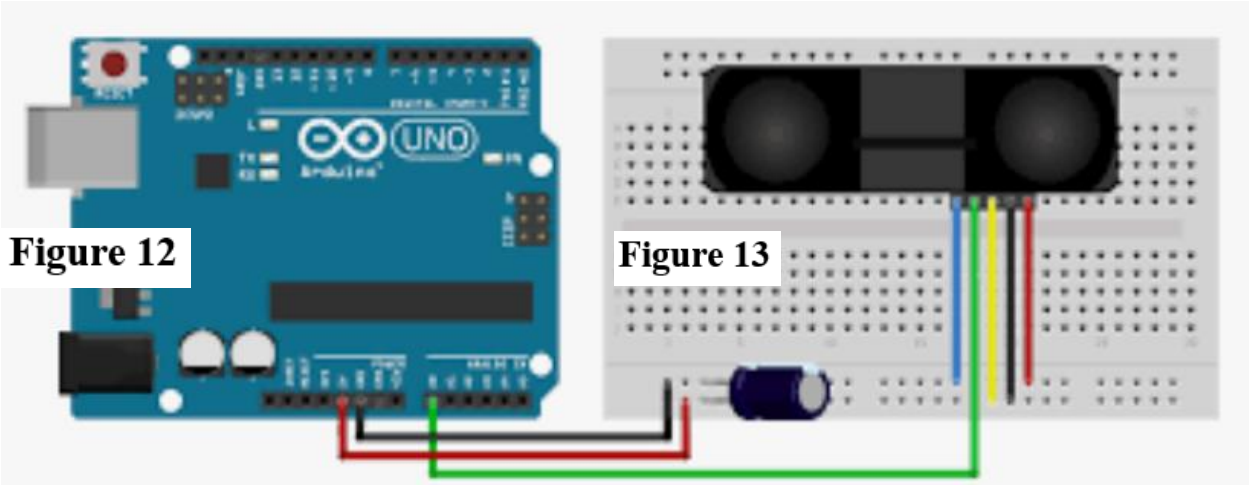


Figure 22

We cannot send information using the ARDUINO board directly to the cloud. So, we used TX/RX communication to send data from Arduino to the ESP module and from ESP to the cloud. The connection is simple. We just used two digital pins and configured serial communication in the code.

Face Recognition

We have a unit that can recognize the faces of people at the front door. The owner will give individuals authorization so that the front door will open for them as they show up at the gate. For the owner to provide a person an authorization, we need to have a reference picture. The reference picture should go through the Facial Recognition Process, consisting of three phases: Face Detection, Pre-processing of the image, and Training[3].

Phase 1 - Face Detection:

The method we will use to detect the face is called Histogram of Oriented Gradients (HOG).

First of all, the picture should be converted into greyscale since we do not need the color to detect or recognize faces. Now we have only black and white pixels. Our program divides the image into several boxes. Each of those boxes contains nine neighboring pixels.

Figure 23

The program compares each pixel's darkness to that of its neighboring pixels. Our goal is to determine the direction towards which the image is getting darker. The program determines gradients in each box. The arrows are placed along the gradients. Thus, the program creates the shape of objects. Figure 23 shows a reference picture and the shape of the face. Once the face is detected, it is cropped from the reference image and is ready for the second phase.[3]

Phase 2 – Pre-Processing:

During the second phase, the face is being pre-processed. At the end of phase 1, we cropped the face, but the face might be turned towards a different direction. For computers, a person looking to the right and the face of the same person looking to the left looks completely different. To remedy this complication, the program should rotate the picture so that the eyes, tip of the nose, and lips are always in the same place.[3]

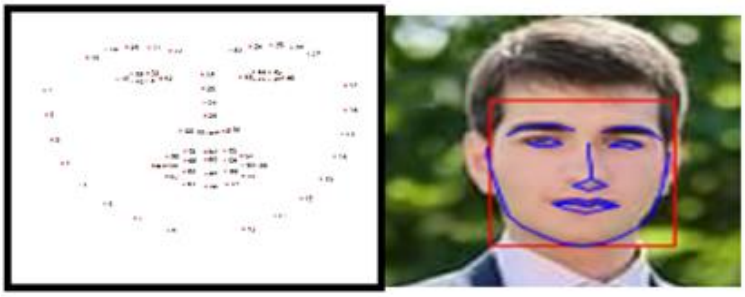


Figure 24

The program looks for facial landmarks that are unique to every human being. Figure 124shows the position of those landmarks and landmarks of the reference face.

Phase 3 – Training:

During the third phase, facial landmarks are stored and assigned with the name of the person[3]. See Figure 25.

In the future, when this person – SOLO, looks into the camera, he will be identified as an authorized person, and the owner of the house will be notified that SOLO has arrived. See Figure 26

Figure 25

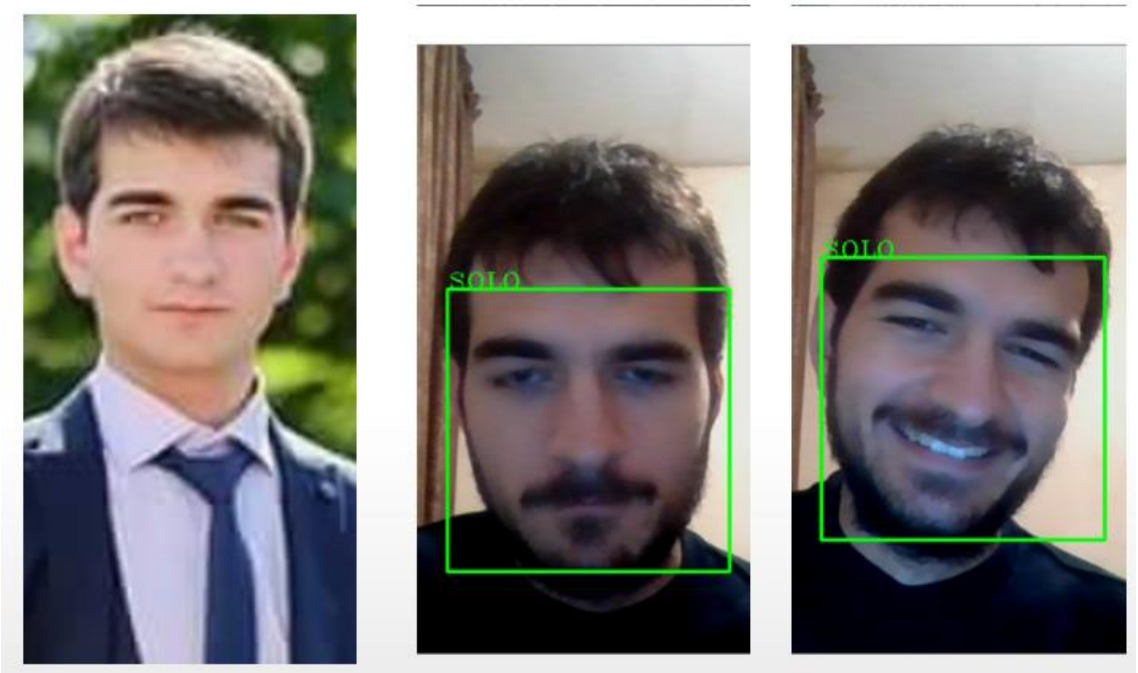


Figure 26

We used ESP 32 CAM to design a face recognition unit. The ESP has a camera attached to it that supplies us with a video stream. Furthermore, we run a local webserver where the video stream is displayed. The video stream and face recognition control panel are incorporated in the Web Application. From the server, the homeowner can add faces of authorized personals or tenants. See Figure 27.

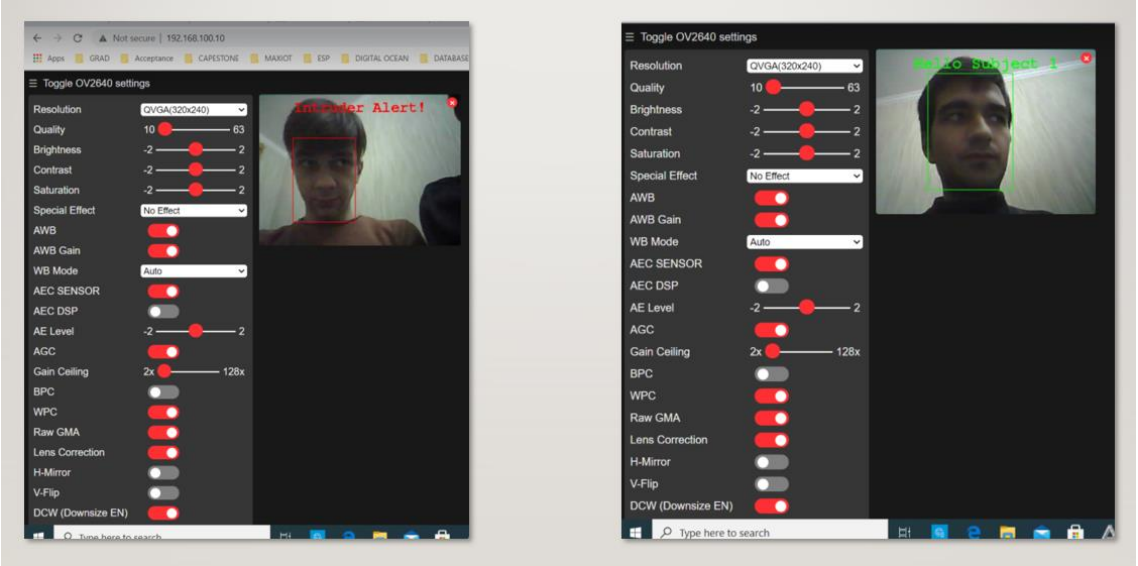


Figure 27

Conclusion

Our project addressed the 21st century's one essential issue, Safety and Security. Our primary purpose was to make those two commodities affordable and readily available to citizens who live in a world where nobody is completely protected from becoming victims of crimes and accidents such as intrusion, robbery, arson, fire, malfunctioning utilities, etc.

We addressed safety by implementing modules that can detect hazards and generate appropriate responses to limit the damage caused by them. Our project made safety monitoring an automated process by utilizing the Internet of Things (IoT). As a result, our project was able to turn an ordinary house into a safe home.

We solved security by implementing a module that intelligently identifies and distinguishes the tenants from ill-intended strangers and potential intruders. We accomplished this by utilizing a trained Artificial Intelligence (AI) system to recognize faces using deep learning.

This was our first project which means that it is not perfect, and certain things could have been done better. One of those things is a water pump. Now we have just a water pump and thus a direct water current. However, we can make progress in that field. We can also apply a water sprinkler that divides a typical water stream into small droplets and throw them in a 360-degree direction. This method can give us two advantages. First of all, it can be more efficient in terms of fire extinguishing.

On the other hand, it can prevent damage caused by water itself since the pressure of droplets is considerably lower than those of direct water current. Furthermore, we could have improved our project by having an action device that turns off Natural Gas Supply if the gas leak is detected. In addition to this, it is our ambition to make our eagle's nest more intelligent. Currently, it can make decisions on its own and activate action devices. However, we would like it if our home could communicate with words and understand voice commands. The topic of the intelligent home is vast, and, providing enough funding, multiple exciting projects can be done with it.

Overall, we attempted to generate a solution to the issue of the lack of safety and security. We accomplished this by implementing an IoT system to generate information about the state of the home safety and an AI system to identify potential intruders so that tenants. Our project turns an ordinary house into a smart, safe and secure home where tenants are more protected from the perils of the real world.

Specifications

All materials that are required to replicate our project are uploaded in the Project Related Materials section.

In the file, you can see a folder named Gerbers. This folder contains Gerber files for our PCBs. We designed 5 PCBs for co2, Gas, fire, motion monitoring.

In the file, you can see a folder named Invoices. There you will see cheques and bills of components that we purchased.

There is also a folder called components which specifies what components were used by each team member.

You can see PowerPoint presentations, video demonstrations, duties chart, poster, and project proposal, alongside with bloc diagram.

The most important folder is the Source codes folder.

There you will find codes that were used for our project.

The folder named GAS: contains the source code for the gas leak detector module.

The folder named Buffer: contains the source code for receiving information from MAXIOT and publishing that information to MAXIOT. Hence the name Buffer.

The folder named Distance – Send: contains the source code for Arduino. Arduino reads distance values from the distance sensor and serially sends them to ESP.

The folder named Smoke_Detector_AND_Distance_Received: contains the source code for ESP. It receives information about smoke distance from Arduino and sends that information alongside the smoke detector's reading to MAXIOT.

FACE Recognition folder contains two folders. One is filled with python codes used for development. The other is filled with source code for the final face recognition unit that should be downloaded on ESP32 CAM.

GSM_ESP_PART: contains source code for ESP connected to GSM module.

The folder named MICROWAVE: contains source code for esp connected to microwave sensor.

The PIR folder, sprinkler, TVOC_CO2: contain source codes for esps connected to PIR, sprinkler, TVOC_CO2.

The folder named Web App and Web Page contains source codes for the application and website.

The folder called MAXIOT contains python codes for MAXIOT mediators.

Team Member Contributions

Giorgi Solomnishvili:

My responsibilities in the process of assembling an intelligent and secure house comprised of getting MAXIOT server up and running, connecting ESP8266 NODEMCU to IOT cloud, installing Grafana and figuring out a way how to embed Grafana graphs in Web Application, installing influxDB on the web server, assembling device mediator connections on MAXIOT and writing python code for that, and implementing face recognition and surveillance system.

The MQTT broker utilized in our project is called MAXIOT. I rented a remote computer from the digital ocean. The computer has an 80 GB hard disk and 2 GB RAM. I installed Ubuntu on that remote computer and created a user called IoT. The run file for MAXIOT_SERVER is stored in the etc folder and automatically runs as the computer reboots. This automation was achieved by providing a path to run the file in rc.local. The MQTT broker is running on port 4004. If devices want to connect to the broker, they should send the connection request to the public IP of our remote computer at 4004 port. To efficiently control MAXIOT, I obtained a studio that enables us to establish and destroy connections of the devices with the broker and each other.

I decided to connect action devices and MAXIOT with the ESP8266. I discovered a library called PubSubClient that allowed ESPs to connect to their avatars on MAXIOT, subscribe to information from them, and publish data to them. I wrote a code that will enable ESPs to connect to the MQTT server at port 4004. After the initiation, I included a while loop that polls until the connection is established. There might be cases when esp gets disconnected from WiFi and MQTT Broker. I wrote two functions that allow ESP to reconnect to WiFi and MAXIOT.

Since we wanted to store sensor values into the database, I decided to install influxDB on our server. InfluxDB is listening on port 8086. I created an influx database user called "hopeuser" and a database called "hoped."

To visualize values stored in databases, I installed Grafana on our remote computer. It runs on port 3000. Grafana allowed us to create a dashboard – The Eagle's Nest, and panels within the dashboard for displaying the graphs. I made panels for Counter, VOC, Smoke Distance, Smoke Detector, PIR and MICROWAVE Motion Sensor, Gas Sensor, and CO2. Figure 8 displays panels for the Gas sensor and CO2.

Initially, Grafana was configured in a way that it only allowed access to the dashboard through authorization. Since we wanted to embed Grafana panels in our web application, I changed the configuration and enabled anonymous authorization. This was achieved by modifying the grafan.ini file, which is stored in our remote computer. In that way, it became possible to embed Grafana panels in our web application using iframe.

I assembled connections between avatars of our sensors and devices with mediators. I wrote python codes for mediators as well.

We have a unit that can recognize the faces of people at the front door. I used ESP 32 CAM to design a face recognition unit. The ESP has a camera attached to it that supplies us with a video stream. Furthermore, I run a local web server on ESP32, where the video stream is displayed. The video stream and face recognition control panel are incorporated in the Web Application.

I was in charge of managing the project schedule and edit reports. I wrote the abstract and introduction of our final report. I also wrote the part that describes IoT Server and Connectivity, MAXIOT, InfluxDB, Grafana, MAXIOT Configuration and Face Recognition.

Lasha Butkhuzi:

My responsibilities in assembling a smart and secure house comprised monitoring indoor air quality, designing two kinds of smoke detectors, including a fire alarm system, and designing schemes for all the circuits we used and PCBs for that schemes.

To monitor indoor air quality, I used ultra-low-power digital gas sensor CJMCU-811, which provides a convenient way to monitor a wide range of volatile organic compounds (TVOC) and particularly equivalent carbon dioxide (eCO2). By connecting to the ESP8266 (NodeMCU 1.0) microcontroller over I2C, I displayed the concentration of CO2 and VOCs in the air and how they affect its quality. To wire the module to the microcontroller, I connected VCC to 3.3V, GND, GND, SCL to D1, SDA to D2, and WAKE to GND. Other pins can be left unconnected. For the system, I constructed PCB using Altium designer. For the software, I installed Adafruit CCS811 Library on Arduino IDE. I sent the information on our IoT server, where I wrote the logic for maintaining an acceptable level of air quality. If the bad quality of the air is detected, the user will be informed immediately via the phone. The message will contain the current air quality level and advice on what action to take. The user will also monitor eCO2 and TVOC and the current air quality level using the mobile application.

My second personal contribution to the group was to design two kinds of smoke detectors. Firstly, I designed a light scattering smoked detector, which operates on the Tyndall effect. Secondly, I constructed a mechanical part for that sensor, and when the smoke enters the chamber, the light from the source is scattered and falls on the photocell, which I detect by LM258P IC and trigger the alarm. After, I wrote ESP code to send an alarm signal on the IoT server and notify the user. For the second smoke detector, I used a SHARP IR distance sensor. It is installed on the ceiling and detects the rapid change in the distance caused by smoke. The distance value is sent to the Arduino Uno board. I used RX/TX serial communication to connect Arduino to the Esp module after writing the code to send the Distance value on the IoT server and notify the user.

Finally, I designed PCBs for all the circuits sent by my teammates and those I used for my devices. Using Altium Designer, firstly, I designed schemes for the courses. If the devices or components were not in the software library, I created schematic and PCB

libraries for them. Next, I designed PCBs and ordered them from JLCPCB.com after transferring the components and circuits from the breadboards to the PCBs.

I was in charge of managing the parts procurement and poster edition. I wrote the part that describes FIRE DETECTION AND ALARM SYSTEM and INDOOR AIR QUALITY MONITORING SYSTEM.

Reziko Tsirgvava:

My responsibility in assembling a smart and secure house was linked to gas detection, fire extinguishing, and informing the house owner about the hazard. Mainly, I have tested the performance of a gas sensor, MQ7, that can measure an amount of CO in living areas. Apart from testing the sensor's performance, my job was also to send the value ESP Nodemcu microcontroller that is responsible for sending it to the IoT server. I have written a code in Arduino IDE where it is specified that the hazard message will be sent if and only if the amount of CO will exceed a critical mark. In addition, to display the value of CO at the Nokia LSD screen that is powered by ESP pins and an MQ7 Sensor.

The other part was my duty as a water pump. It's a device that drives a water current using a motor working properly on 12V. A 12V adapter powers this pump. However, to open and close at desired moment, I used a solid-state relay. The relay, operating at 5V DC, is powered by an ESP 3.3V pin elevated to 5V through the step-up converter. The primary purpose of ESP is to receive a message from the IoT server following to smoke detector that sends signals 1s and 0s. 1s in case of being fire and 0s in case of having no hazard. In case of the presence of fire, the smoke detector turns on, sends the Signal to IoT serve through the ESP microcontroller, then the IoT server sends that Signal to the local ESP, and that controller turns on the relay and lets 12 V drive water pump. This is a relatively small model that can be increased if needed in terms of extinguishing real-life fires.

The third part which I was working on is about GSM. It's a module that sends and receives messages wirelessly. Just like ordinary cell phones, GSM requires a sim card to send messages anywhere in the world. GSM is powered from esp 5V pin, and it is also executed via ESP 8266. Generally, to send messages, GSM needs special AT Commands. The instruction of AT Commands was found in internet resources. Arduino code was required to use AT Commands properly and "ask" GSM to send a desired message in the desired direction. When I assembled the scheme, I was wondering whether I could send the message anywhere. I tested and sent my friend and capstone partner Giorgi Solomnishvili, who lives about 200m away from my house. Then I send the message to my mom, who lives in Kutaisi, and finally, I do the same job for my Iranian friend who lives currently in San Diego. All of the attempts were successful, which ensured me in good performance of GSM. Later this was confirmed again. While using our sensors, our GSM was working property accurately and sent a message on time.

I was in charge of managing the team presentations and overseeing component delivery. I wrote the part that describes Water Pump, Sprinkler, GSM module.

Mikheil Makharadze:

My responsibilities in the process of assembling an intelligent and secure house comprised of developing a web application.

Our project allows the homeowner to monitor and control his/her home with the web application. I used a relatively new and effective script language – MS script, to create a web application. The script language is built on JavaScript libraries – jQuery and paho.

The application creates a WebSocket, and this WebSocket connects to the MQTT broker, MAXIOT, as a device. Just like sensors and action devices, the web application has its avatar with ID 9999. If a person wants to sign into his monitoring and control system, he should enter an ID. Each customer will have a unique interface because they will have a different home. Therefore, each customer will need a different avatar for their application.

Moreover, part of the application that enables monitoring and controlling should be different for each client. Fortunately, MS script can be written in the avatars, just like in the case of mediators. This way, I will be able to accommodate the application to the customer's needs.

The application allows users to see sensor values, Grafana panels, and surveillance camera streams.

The Web Application allows users to control action devices in his/her home. At this point, we offer two action devices – A motion detector and a fire gas alarm system. Motion, Gas, and fire sensors are reading information all the time. However, the alarm and the detector will act based on the sensor values if they are turned on from the Web Application.

I was in charge of managing the budget, alongside Reziko Tsirgvava. I wrote the part that describes Web Application.

Aleksi Aleksandria:

My responsibilities in assembling an intelligent and secure house comprised designing a motion detector unit, monitoring the creation and update of databases, and designing a website to popularize our project.

Our design can detect a motion in the home. I used a Passive infrared (PIR) and Microwave sensor to detect the motion.

PIR sensor detects motion by detecting the heat and rapid heat change in a residential area. Once the sensor warms up, it can detect heat and movement in the surrounding areas, creating a protective "grid." If a moving object blocks too many grid zones and the infrared energy levels change rapidly, the sensors trigger an alarm This rapid change of heat indicates that someone is moving into the home. PIR outputs a digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor. The sensitivity range of PIR is up to 6 meters. I came up with the idea to power up the PIR sensor from the VV pin of ESP8266. In the beginning, we did not know that there exists an ESP pin that is capable of outputting 5v. After reading the ESP datasheet, I discovered that manufacturers consider that many sensors operate at 5v and added a 5v output pin to ESP boards.

Microwave sensors detect motion by utilizing electromagnetic radiation. The sensor can emit electromagnetic waves and capture reflected waves. The sensor can measure the time wave took to reach an object and come back to the source. Microwave sensor uses the doppler effect and projected microwaves to detect the motion.

If nobody is supposed to be at home and the motion is detected, this means that we have an intruder. There is an excellent chance that this intruder is a thief. The owner and the police will be notified about the intruder.

I was in charge of managing the project's exhibition and design its illustrations. I wrote the conclusion of our final report. I also wrote the part that describes the motion detector.

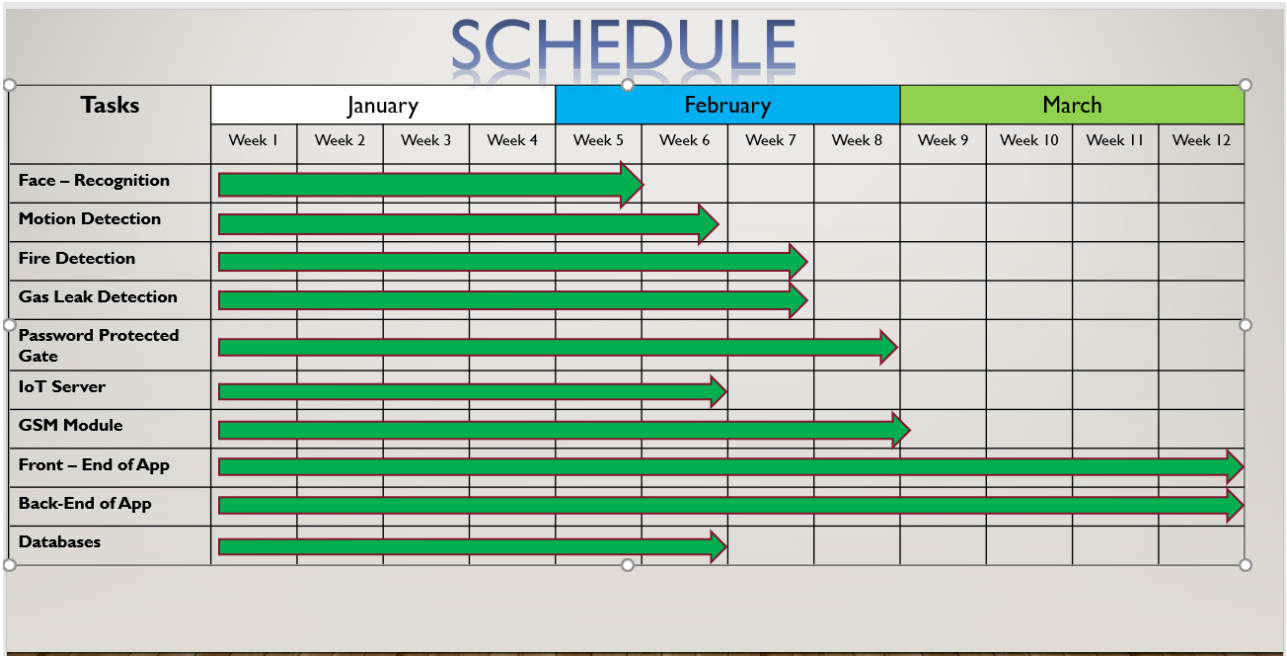
References

1) Interface Nokia 5110 Graphic LCD Display with Arduino (2021). Retrieved from: https://lastminuteengineers.com/nokia-5110-lcd-arduino-tutorial/
2) James Turnbull (1 December 2014). The Art of Monitoring. James Turnbull. pp. 206–. ISBN 978-0-9888202-4-1. Retrieved from: https://books.google.ge/books?id=w5QfDAAAQBAJ&pg=PA206&redir_esc=y#v=onepage&q&f=false
3) Modern Face Recognition with Deep Learning (2016). Retrieved from: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78
4) Rouse, Margaret (2019). "internet of things (IoT)." IoT Agenda. Retrieved 14 August 2019. Retrieved from: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT
5) Send Receive SMS (2021). Retrieved From: https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/.
6) At Commands Reference Guide (2006). Retrieved from: https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf
7) ESP8266WiFi Library (2017). https://arduino-esp8266.readthedocs.io/en/2.7.1/esp8266wifi/readme.html#quick-start

Appendices

Source codes, schematics, and Gerber files are submitted in project related materials section. See specifications.

Gant Chart



Bill Of Materials

Invoices are submitted with project-related files materials.

Components	**Price in $ - Purchased Abroad**	**Price in GEL – Purchased in Georgia**
ESP8266 (x15)	58.65 $	-
Digital Multi Metter	49.19 $	-
Temperature Sensor	13 $	-
Pi Cam	15 $	-
Solenoid Valve	13 $	-
ESP32 CAM (x3)	24 $	-
Soldering Iron	39 $	-
Arducam Camera Adapter	50 $	-
PCBs	20.09 $	
Keypad	-	3 Gel
MQ 135	-	7 Gel
Nokia Screen	-	15 Gel
I2C adapter	-	5 Gel
PIR	-	4 Gel

HDMI Female Adapter	-	29.2 Gel
USB Heads	-	9 Gel
Pi cam	10.9 $	-
Microwave Sensor	5.78 $	-
Relays	12 $	-
Sharp Distance Sensor	13.44 $	-
Delivery from Aliexpress	47.71 $	-
HDMI to VGA adapter	-	15.6 Gel
RPI 4	90 $	-
HDMI-HDMI	-	16.4
Delivery from Georgian Stores	-	32 Gel
RFID	-	14.7 Gel
Breadboards	-	24
Resistors, LM258, MQ4, MQ7, potentiometer	-	22
Soldering Iron, 12V power supply, soldering wire	-	135
Stereo Array Camera	49 $	-
Digital Ocean Computer Renting	70 $	-
Sum	580.76 $	331.9 Gel = 98 $

The total cost is 678.76$