TEAM NAME - ELECTRICIANS

PROJECT TITLE - Smart Home

Mid Evals
13 November, 2021

TEAM MEMBERS -
Lavisha Bhambri (2020101088)
Mayank Bhardwaj(2020101068)
Shreyash Jain (2020101006)
Gautam Ghai (2020101020)

Overall Objective

Our team, **Electricians**, came up with a solution to automate the home & tackle the issue of safety when we are not present at our homes. We have decided to implement various features to build a Smart Home IOT project under the guidance of our IoT course professor, **Prof. Deepak Gangadharan**, and the assigned IOT teaching assistant **Ms. Pravalika Mukkiri.**

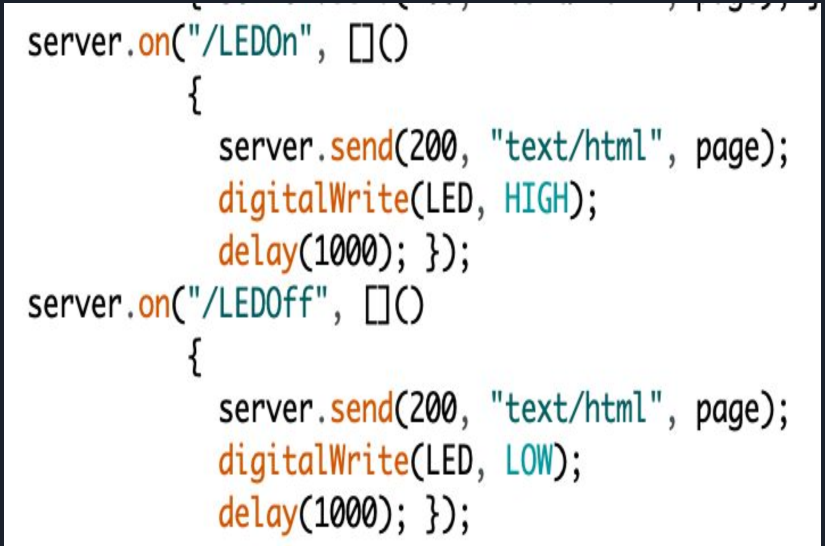
Project Overview

We have divided our project into 4 sub parts where each sub part is a solution to a problem faced by many of us in our day to day home practices.We have made a single page Web Application which is a Web server for the user to have a look at the details anytime whenever he/she is connected to the same wifi as that of ESP32 and incorporate all the functionalities in a single place.

Parts

- ❖ PART-1 : Turning Leds ON and OFF
- ❖ PART-2 : Smart Lights
- ❖ PART-3 : Home Security
- ❖ PART-4 : Fire Alarm

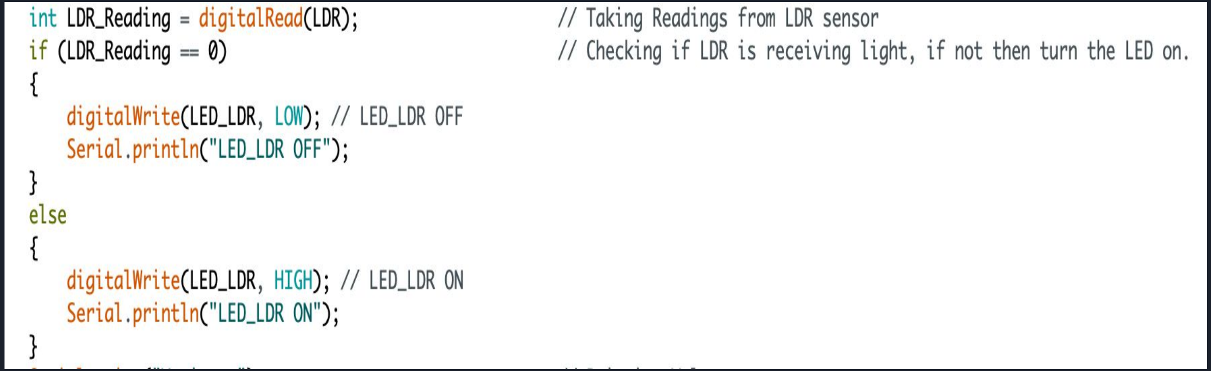
Part - 1 : Turning LEDs ON and OFF



```
server.on("/LEDOn", []()
         {
           server.send(200, "text/html", page);
           digitalWrite(LED, HIGH);
           delay(1000); });
server.on("/LEDOff", []()
         {
           server.send(200, "text/html", page);
           digitalWrite(LED, LOW);
           delay(1000); });
```

For easing out the work of the user we have made a web server and given the option to him using which he/she can switch OFF/ON an electrical appliance for instance, LEDs while he is present in any part of the home.

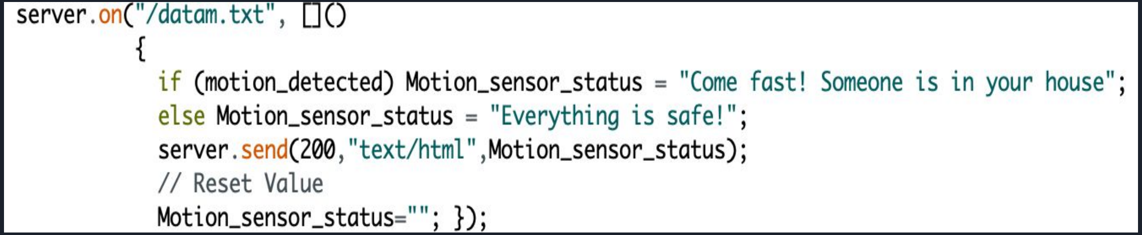
Part - 2 : Smart Light



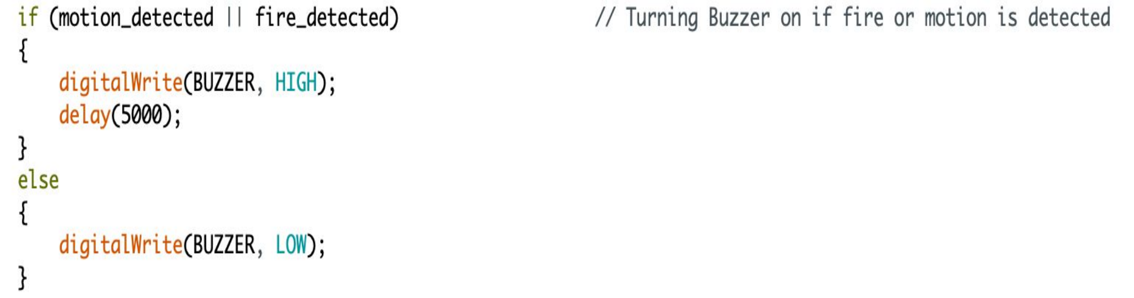
```
int LDR_Reading = digitalRead(LDR);                         // Taking Readings from LDR sensor
if (LDR_Reading == 0)                                       // Checking if LDR is receiving light, if not then turn the LED on.
{
    digitalWrite(LED_LDR, LOW); // LED_LDR OFF
    Serial.println("LED_LDR OFF");
}
else
{
    digitalWrite(LED_LDR, HIGH); // LED_LDR ON
    Serial.println("LED_LDR ON");
}
```

For the smart home lights we will be using the LDR module to detect if it’s day or night. On the basis of the results, if it’s day then the LEDs are switched OFF, else during the night, they are automatically switched ON.

Part - 3 : Home Security



```
server.on("/datam.txt", []()
          {
            if (motion_detected) Motion_sensor_status = "Come fast! Someone is in your house";
            else Motion_sensor_status = "Everything is safe!";
            server.send(200,"text/html",Motion_sensor_status);
            // Reset Value
            Motion_sensor_status=""; });
```

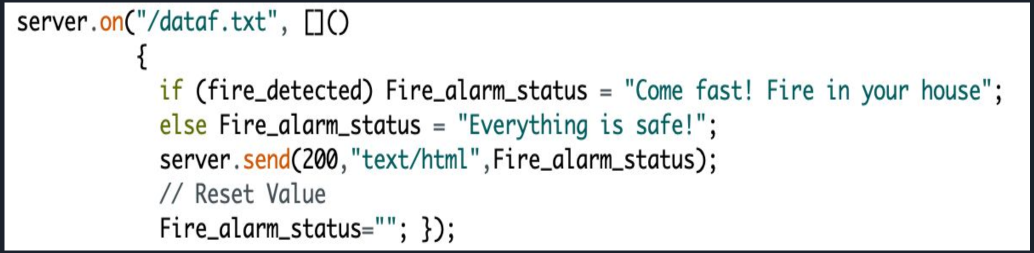


```
if (motion_detected || fire_detected)                  // Turning Buzzer on if fire or motion is detected
{
    digitalWrite(BUZZER, HIGH);
    delay(5000);
}
else
{
    digitalWrite(BUZZER, LOW);
}
```

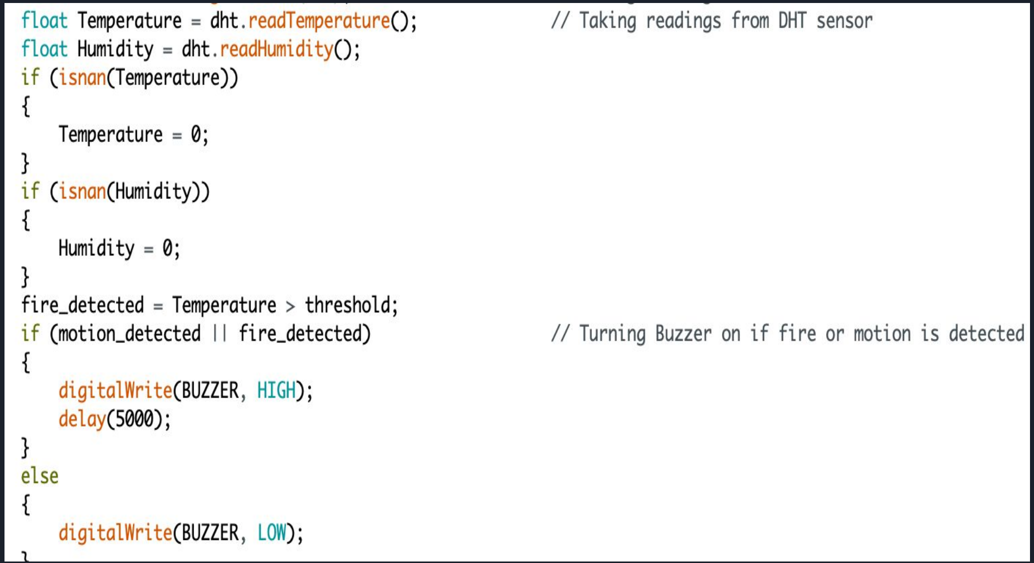
For the safety purpose and for protecting the house from an unknown person we have incorporated the PIR sensor to detect if there is some unwanted motion and we have notified the user by raising an appropriate alarm with the help of a BUZZER and by notifying it on the web server where the message status changes from “EVERYTHING IS SAFE “ to “ COME FAST! SOMEONE IS IN YOUR HOUSE.”

Part - 4 : Fire Alarm

We have also incorporated the DHT11 sensor to incorporate the fire alarm feature for our smart home and as the temperature rises above a certain threshold temperature, the buzzer gets activated and the message status on the web server changes from “EVERYTHING IS SAFE “ to “Come fast! Fire in your house”.

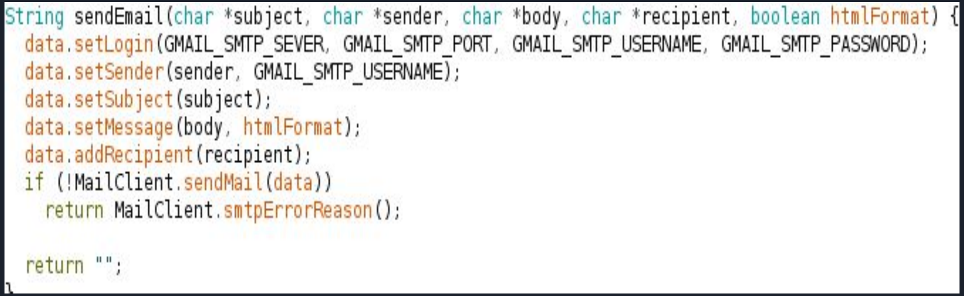


```
server.on("/dataf.txt", []()
          {
            if (fire_detected) Fire_alarm_status = "Come fast! Fire in your house";
            else Fire_alarm_status = "Everything is safe!";
            server.send(200,"text/html",Fire_alarm_status);
            // Reset Value
            Fire_alarm_status=""; });
```



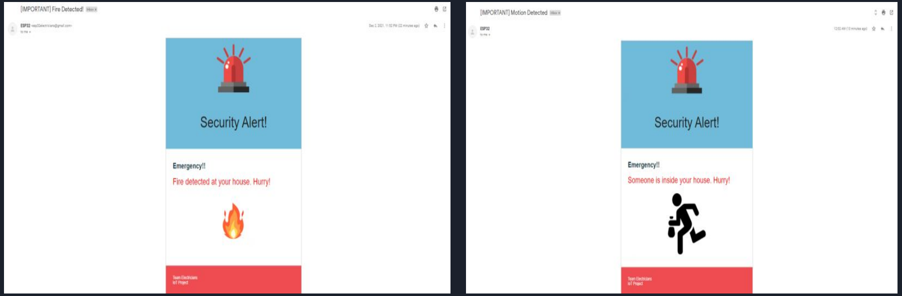
```
float Temperature = dht.readTemperature();                    // Taking readings from DHT sensor
float Humidity = dht.readHumidity();
if (isnan(Temperature))
{
    Temperature = 0;
}
if (isnan(Humidity))
{
    Humidity = 0;
}
fire_detected = Temperature > threshold;
if (motion_detected || fire_detected)                         // Turning Buzzer on if fire or motion is detected
{
    digitalWrite(BUZZER, HIGH);
    delay(5000);
}
else
{
    digitalWrite(BUZZER, LOW);
```

Email Notification Alert



```
String sendEmail(char *subject, char *sender, char *body, char *recipient, boolean htmlFormat) {
  data.setLogin(GMAIL_SMTP_SEVER, GMAIL_SMTP_PORT, GMAIL_SMTP_USERNAME, GMAIL_SMTP_PASSWORD);
  data.setSender(sender, GMAIL_SMTP_USERNAME);
  data.setSubject(subject);
  data.setMessage(body, htmlFormat);
  data.addRecipient(recipient);
  if (!MailClient.sendMail(data))
    return MailClient.smtpErrorReason();

  return "";
}
```

Code Snippet



[IMPORTANT] Fire Detected!

ESP32 <esp32electricians@gmail.com>
to me

Dec 2, 2021, 11:52 PM (22 minutes ago)

Security Alert!

Emergency!!

Fire detected at your house. Hurry!

Team Electricians
IoT Project

[IMPORTANT] Motion Detected

ESP32
to me

12:02 AM (13 minutes ago)

Security Alert!

Emergency!!

Someone is inside your house. Hurry!

Team Electricians
IoT Project

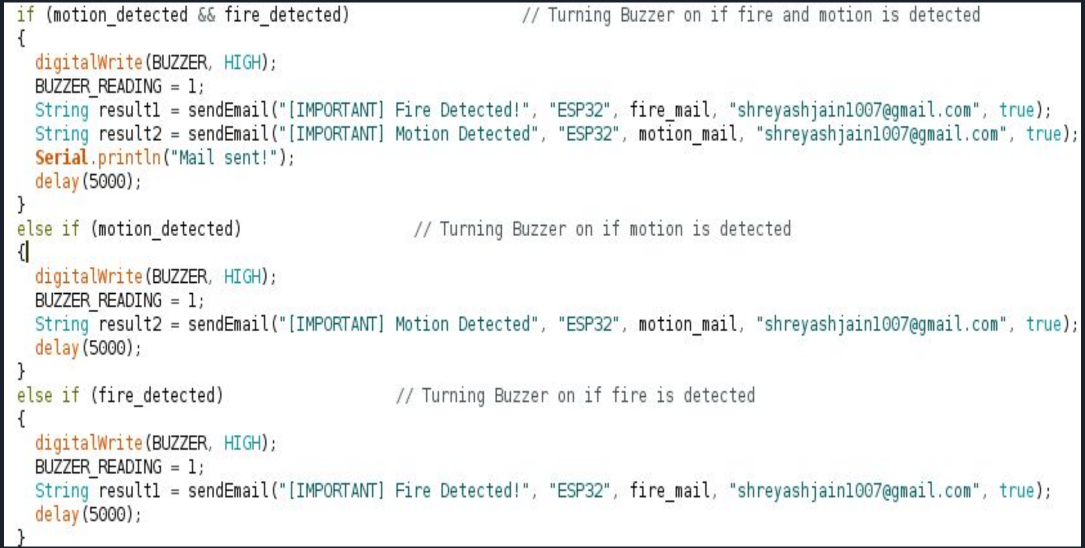
Email Notifications in case of fire alert & thief

We have also provided the feature of emergency alert via email notifications in case of two emergent situations:-

1. When some unknown person enters the home in the absence of the house owner,
2. When there is fire in the house.

Using the above function, the sender (team Electricians) sends the email to the recipient (house owner) with a subject.

Email Notification Alert



```
if (motion_detected && fire_detected)                    // Turning Buzzer on if fire and motion is detected
{
  digitalWrite(BUZZER, HIGH);
  BUZZER_READING = 1;
  String result1 = sendEmail("[IMPORTANT] Fire Detected!", "ESP32", fire_mail, "shreyashjain1007@gmail.com", true);
  String result2 = sendEmail("[IMPORTANT] Motion Detected", "ESP32", motion_mail, "shreyashjain1007@gmail.com", true);
  Serial.println("Mail sent!");
  delay(5000);
}
else if (motion_detected)                    // Turning Buzzer on if motion is detected
{
  digitalWrite(BUZZER, HIGH);
  BUZZER_READING = 1;
  String result2 = sendEmail("[IMPORTANT] Motion Detected", "ESP32", motion_mail, "shreyashjain1007@gmail.com", true);
  delay(5000);
}
else if (fire_detected)                    // Turning Buzzer on if fire is detected
{
  digitalWrite(BUZZER, HIGH);
  BUZZER_READING = 1;
  String result1 = sendEmail("[IMPORTANT] Fire Detected!", "ESP32", fire_mail, "shreyashjain1007@gmail.com", true);
  delay(5000);
}
```

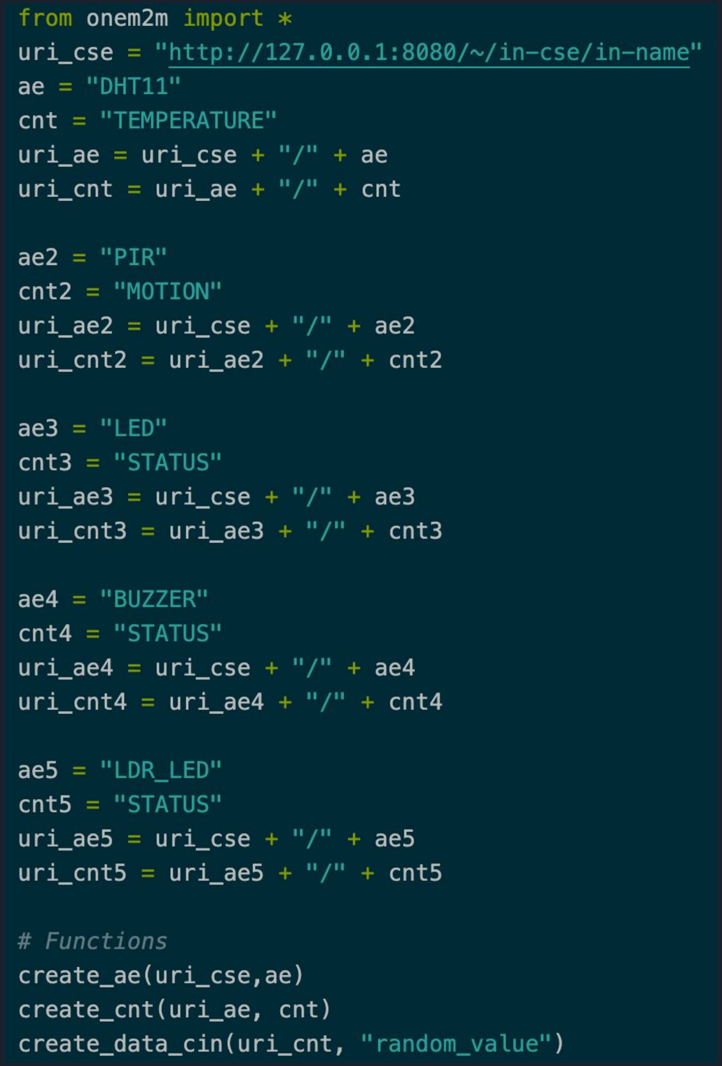
This code block shows the situations in which we are sending the email notifications to the email of the house owner in case of :-

1. Thief and fire
2. Only thief
3. Only Fire

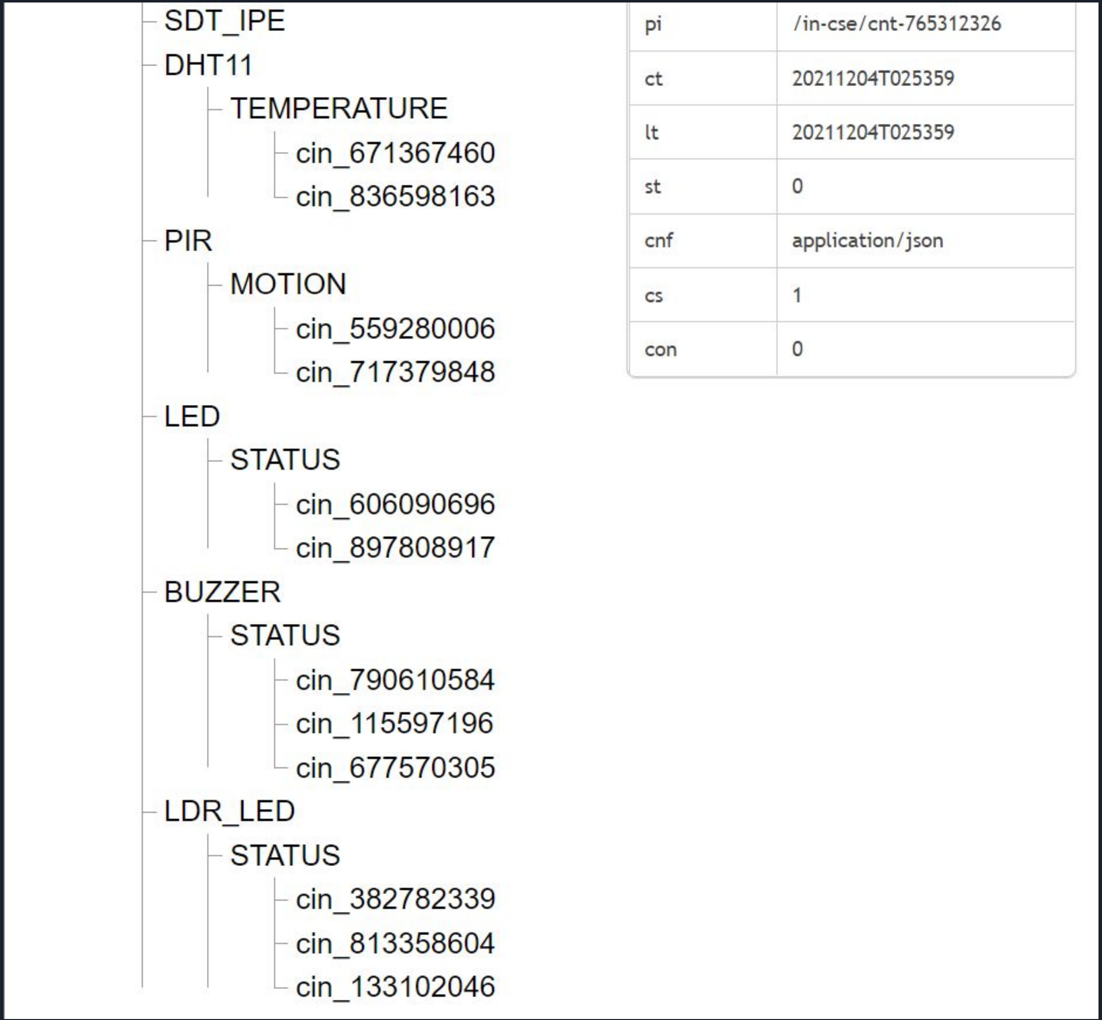
OM2M

We have created containers with the help of a python programme for which the code snippet is as shown:

We have also enabled data sending into the OM2M server with the help of the ‘createCi’ function that is shown in the upcoming slide.



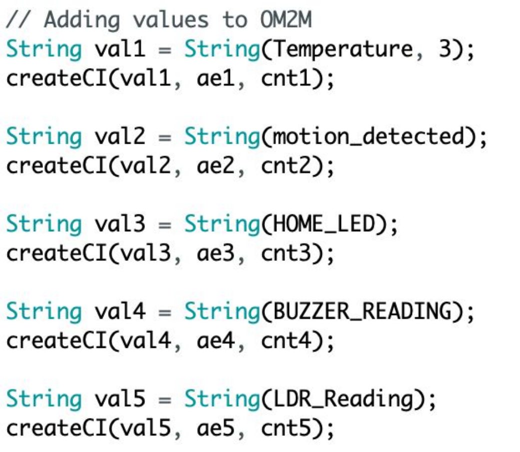
```
from onem2m import *
uri_cse = "http://127.0.0.1:8080/~/in-cse/in-name"
ae = "DHT11"
cnt = "TEMPERATURE"
uri_ae = uri_cse + "/" + ae
uri_cnt = uri_ae + "/" + cnt

ae2 = "PIR"
cnt2 = "MOTION"
uri_ae2 = uri_cse + "/" + ae2
uri_cnt2 = uri_ae2 + "/" + cnt2

ae3 = "LED"
cnt3 = "STATUS"
uri_ae3 = uri_cse + "/" + ae3
uri_cnt3 = uri_ae3 + "/" + cnt3

ae4 = "BUZZER"
cnt4 = "STATUS"
uri_ae4 = uri_cse + "/" + ae4
uri_cnt4 = uri_ae4 + "/" + cnt4

ae5 = "LDR_LED"
cnt5 = "STATUS"
uri_ae5 = uri_cse + "/" + ae5
uri_cnt5 = uri_ae5 + "/" + cnt5

# Functions
create_ae(uri_cse,ae)
create_cnt(uri_ae, cnt)
create_data_cin(uri_cnt, "random_value")
```



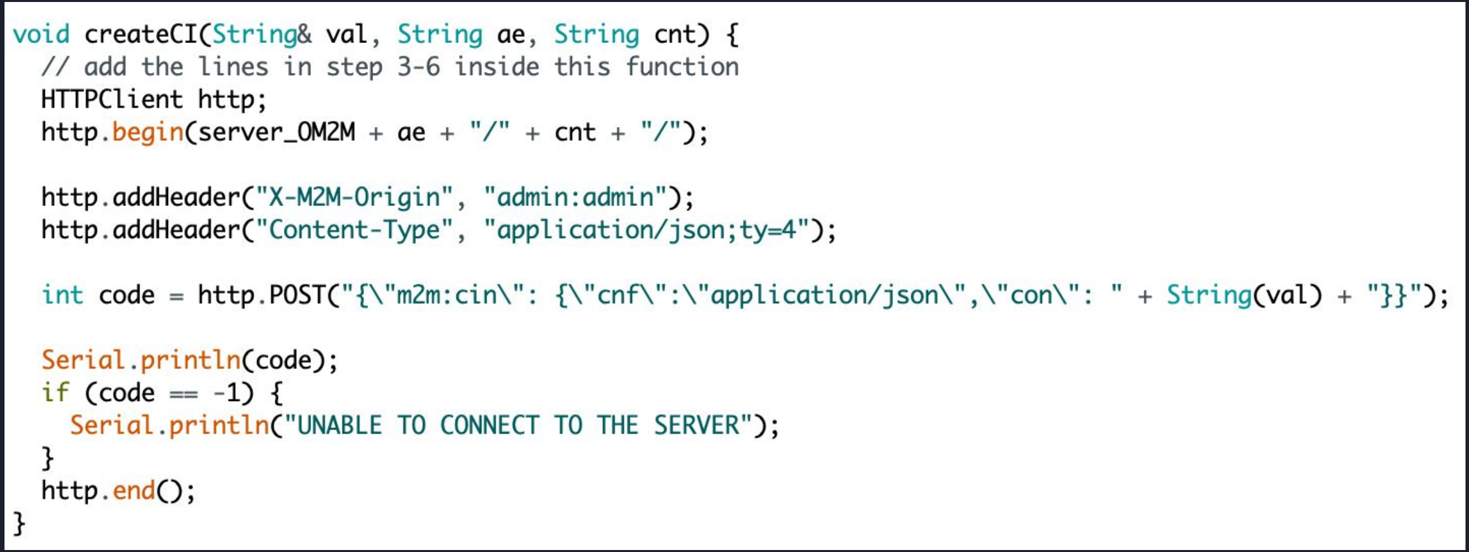
```
├ SDT_IPE
├ DHT11
│   ├ TEMPERATURE
│   │   ├ cin_671367460
│   │   └ cin_836598163
├ PIR
│   ├ MOTION
│   │   ├ cin_559280006
│   │   └ cin_717379848
├ LED
│   ├ STATUS
│   │   ├ cin_606090696
│   │   └ cin_897808917
├ BUZZER
│   ├ STATUS
│   │   ├ cin_790610584
│   │   ├ cin_115597196
│   │   └ cin_677570305
├ LDR_LED
│   ├ STATUS
│   │   ├ cin_382782339
│   │   ├ cin_813358604
│   │   └ cin_133102046
```

pi	/in-cse/cnt-765312326
ct	20211204T025359
lt	20211204T025359
st	0
cnf	application/json
cs	1
con	0

Here we are passing value that has to be updated in the function and the application entity type and the container name is also added and depending on that the values will be updated in the OM2M server.



```
// Adding values to OM2M
String val1 = String(Temperature, 3);
createCI(val1, ae1, cnt1);

String val2 = String(motion_detected);
createCI(val2, ae2, cnt2);

String val3 = String(HOME_LED);
createCI(val3, ae3, cnt3);

String val4 = String(BUZZER_READING);
createCI(val4, ae4, cnt4);

String val5 = String(LDR_Reading);
createCI(val5, ae5, cnt5);
```

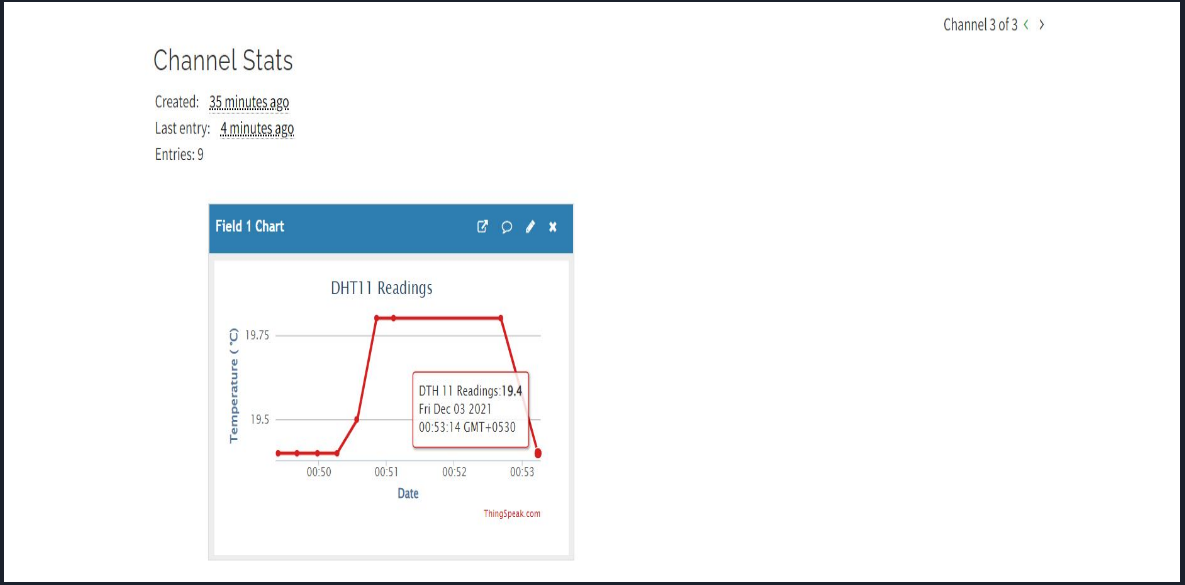


```
void createCI(String& val, String ae, String cnt) {
  // add the lines in step 3-6 inside this function
  HTTPClient http;
  http.begin(server_OM2M + ae + "/" + cnt + "/");

  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");

  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");

  Serial.println(code);
  if (code == -1) {
    Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}
```

THINGSPEAK

We publish data of DHT11 to ThingSpeak channel so that the user can keep track of temperature changes in the house that will give him valuable information later on, incase of a mishap, like when did the fire start, how quickly did it spread and was brought under control. This is useful because while the user will get notifications on his email regarding the status of fire in his house, the user may not be in a position to physically reach the house and do something about it. He may have to call the fire-station and inform neighbours about the fire but because the data will be regularly updated on ThingSpeak, he can keep track of the situation and won’t be in the dark. This data can prove to be valuable for investigations as well.

Also just in case the user fails to receive email notifications due to unforeseeable reasons such as the mail server being down, he/she can still count on ThingSpeak to check if there are any changes in the temperature being detected and whether the temperature being reported has crossed the threshold limit.



Channel 3 of 3

Channel Stats

Created: 35 minutes ago

Last entry: 4 minutes ago

Entries: 9

Field 1 Chart

Components Used

1. Breadboard
2. ESP32
3. PIR motion sensor
4. LDR sensor
5. DHT11 sensor
6. Buzzer
7. LEDs
8. Jumper Wires

Demo Video

Link for video:

https://drive.google.com/file/d/1V_8QHihuPX_YogXIe1BSthLe7ELW6gXn/view?usp=sharing