

EGRE 510 Final Project: IoT enabled Smart Home

Bruno Da Silva, Mahel Al Islam

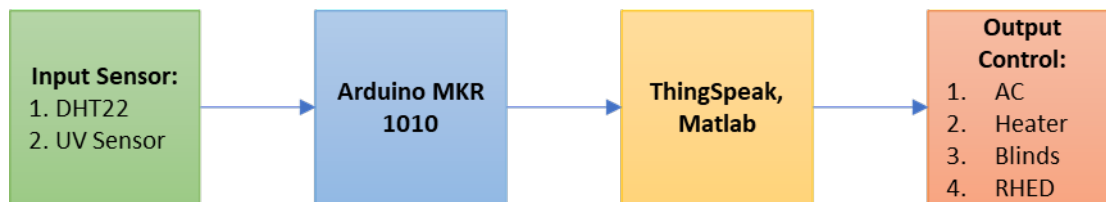
In this project, we have designed and implemented an IoT-enabled smart home using an IoT analytics platform service ThingSpeak and Matlab as our main tools. We have aggregated, visualized, and analyzed live data streams of data from sensors using Arduino, ThingSpeak, and Matlab. We have used two sensors to collect temperature, humidity, and UV index from the surrounding. We have used Arduino MKR 1010 board to read data from sensors and to send this data to the ThingSpeak channel. We have used ThingSpeak application React, Matlab analysis, and time control to execute the actions in different conditions.

Equipment:

- Arduino MKR 1010
- Temperature/Humidity Sensor, DHT 11
- UV Sensor

Working Diagram:

We have collected temperature, humidity, and UV index data from two sensors through the Arduino MKR1010 board and sent this data to The ThingSpeak channel. In Matlab, we did some analysis and sent control signals to AC, heater, blinds, and RHED.



Action Description:

In our study, we have used a time control Matlab analysis application to control the outputs according to the input data. We have used the first three fields of the ThingSpeak channel as the inputs (temperature, humidity, and UV index). We have used the Matlab analysis app to execute the actions. We used a time control app to execute the analysis code every five minutes. We first checked if it's daytime from the timestamp retrieved from the channel data and if it's not bright outside according to the UV index from the UV sensor (field 3) to set the signal to keep the blinds open, otherwise off.

For the UV index calculation, we used the following formula:

$$UV\ index = \frac{V_{sig} \times 307}{200} \dots \dots (1)$$

Then, we put conditions according to the temperature. We divided the temperature region into three parts: High ($T > 78$), Normal ($62 < T < 78$), and Low ($T < 62$). For the high region, we set the signal to turn the AC on (output = 1), the heater and dehumidifier remained off (output = 0). For normal region temperature, we have sent the AC control signal off. We have checked for humidity measurement and if it's higher than the acceptable range ($H > 65$), we sent the signal to turn on the RHED (output = 1). For the low region ($T < 62$), we have sent a signal to turn the heater on. We have checked if it's daytime and bright outside (UV index high) to send the signal to keep the blinds open. After executing all the actions, we sent the output signals as 0 (off) or 1 (on) for AC control, heater control, blind control, and RHED control in fields 4 to 7 accordingly.

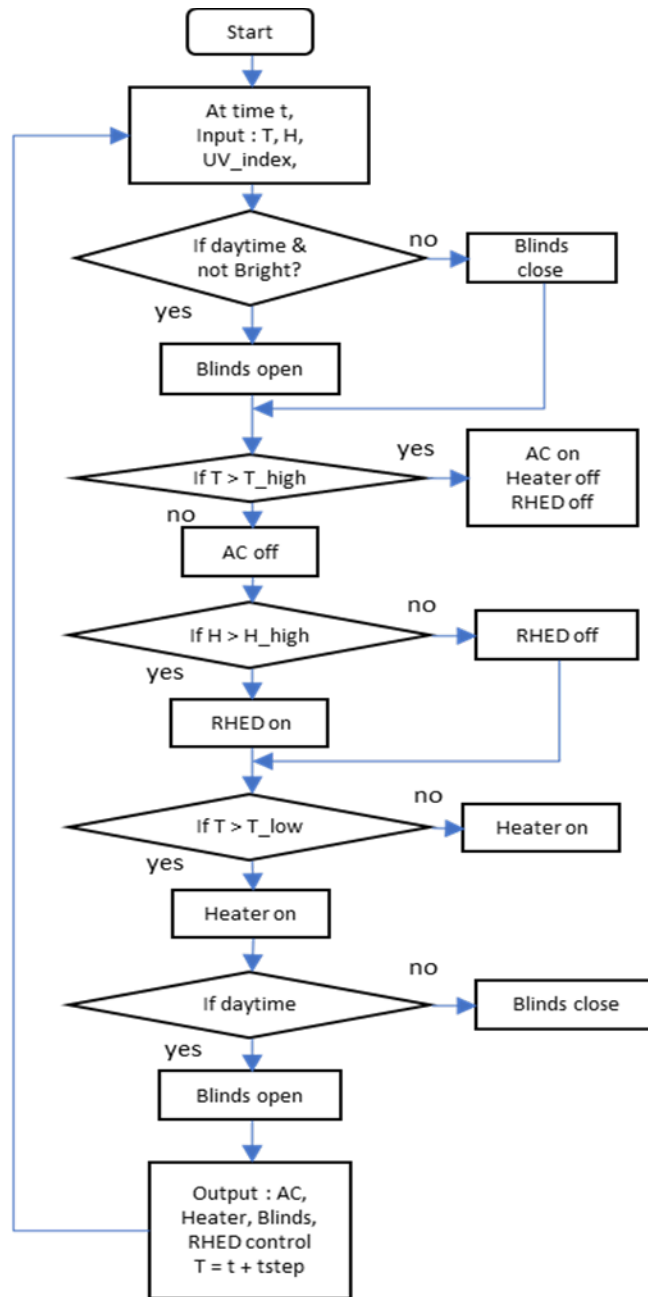


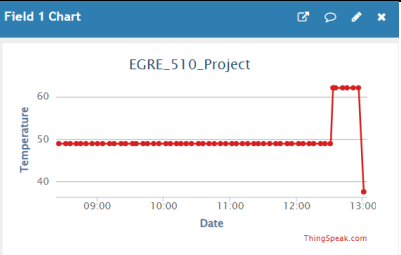
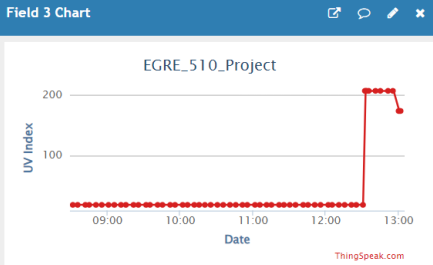
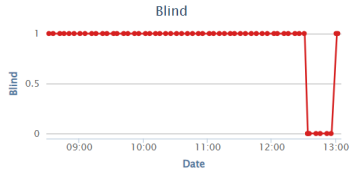

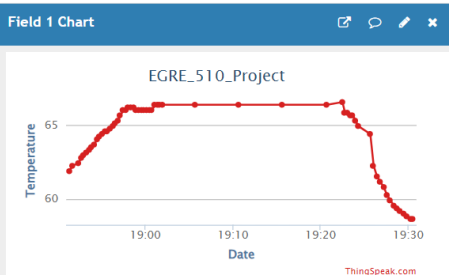
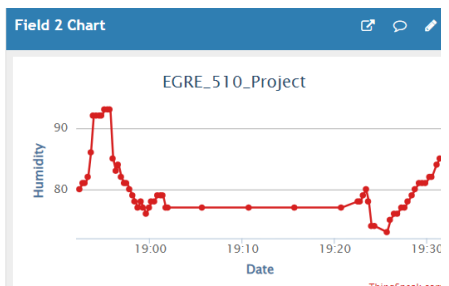
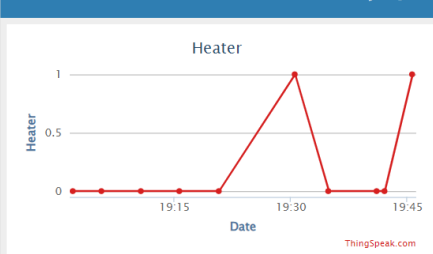
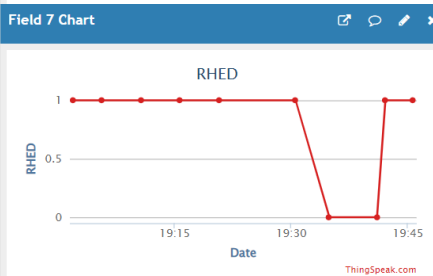


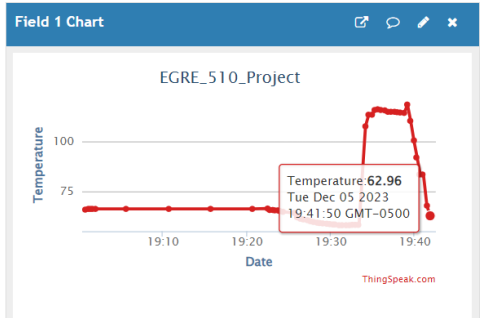
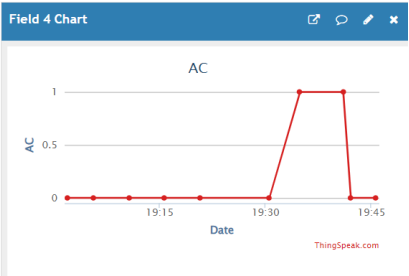
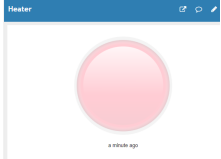
Figure 1: Overall working diagram

Table 1: Case Description

Case	Daytime	Temperature (Field 1)	Humidity (Field 2)	UV index (Field 3)	AC (Field 4)	Heater (Field 5)	Blinds (Field 6)	RHED (Field 7)
1	Yes	-	-	low	-	-	open	-
2	Yes	Low	-	high	off	on	open	-
3	-	High	-	-	on	off	-	off
4	-	Normal	High	-	off	off	-	on
5	-	Low	Low	-	off	on	-	off
6	-	Low	High	-	off	on	-	on
7	No	Normal	Normal	low	off	off	close	off

In table 1, we have shown some example cases and the output fields. Case 1 & 2 covers all the time the blinds should be open and other than that the blinds should be closed. We have shown some input, output channel figures in table 2 with the control actions in table 2.

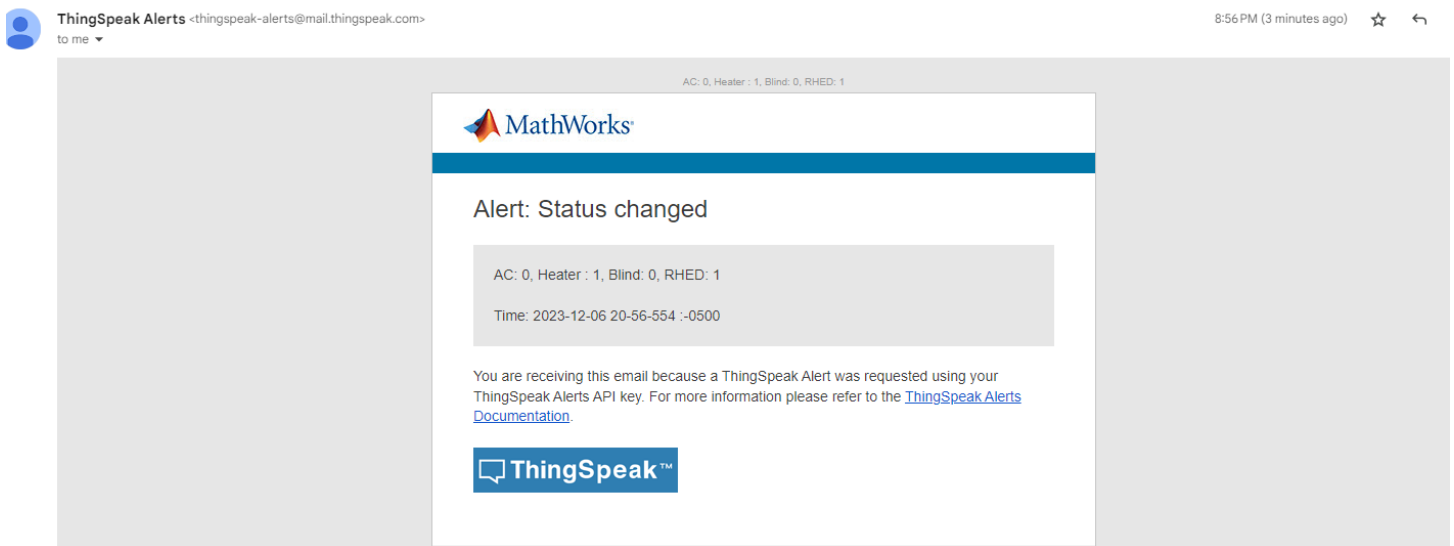
Table 2: Input, Output Fields, and Control action

Case	Input Fields	Output Fields	Output Action
Case1	 		<p>Daytime & UV Index low: Blinds Open; Nighttime: Blinds close</p> 
Case 6	 	 	<p>Temperature low, Humidity high, RHED On Heater On</p>  
Case 4			 <p>Temperature Normal, Humidity High, AC off,</p>



We have used 4 different widgets (Red – Heater, Green- AC, Yellow – RHED and Blue - Blind) which would light up when the corresponding system is on.

Email Notifications:



To achieve all the test cases, we took several steps. We used ice water and dipped our sensor within a ziplock bag to cool down the temperature. It also increased the humidity. We used a blow dryer to increase the temperature of the sensor.



Figure 2: Design equipment

Channel ID: 2349977

Read API Key: 'GOOTZORSIYFDUPP9'

HVAC Control Code (Matlab Analysis with Time control):

% Enter your MATLAB Code below

```
readChId = 2349977;
writeChId = 2349977;
readKey = 'GOOTZORSIYFDUPP9';
writeKey = 'KTPU0ISXY6Q7E4I';
[data,time] = thingSpeakRead(readChId,'Fields',[1, 2, 3, 4, 5, 7],'ReadKey',readKey)
t=data(1)
h=data(2)
u=data(3)
t_h = 78;
t_l = 62;
h_h = 65;
u_h =100;
dt=timeofday(time)

if dt>duration(7,0,0) && dt<duration(17,0,0)
    d = 1
else
    d = 0
end

if (d ==1 && u<u_h)
    b=1;
else
    b=0;
end

if t >t_h
    AC = 1;
    Heater = 0;
    RHED = 0;
else
    AC = 0;
    if h>h_h
```

```

        RHED=1;
    else
        RHED=0;
    end
    if t>t_1
        Heater = 0;
    else
        Heater = 1;
        if (d ==1 && u>u_h)
            b = 1;
        else b = 0;
        end
    end
end

AC
Heater
b
RHED
thingSpeakWrite(writeChId,[t,h,u,AC,Heater,b,RHED],'Fields',[1,2,3,4,5,6,7],'Writekey',writeKey);

```

Thingspeak Email Alert (Matlab Analysis with Time control):

```

alert_body = 'MATLAB Thingspeak';
alert_subject = 'Status changed';
alert_api_key = 'TAK4PNJF11EPYRYXUHPE8';
alert_url = "https://api.thingspeak.com/alerts/send";

```

```

channelID = 2349977; % channel ID
readAPIKey = 'GOOTZORSIYFDUPP9'; % Read API Key

```

```

% Read the latest values from Fields 4, 5, 6, and 7
fieldValues = thingSpeakRead(channelID, 'Fields', [4, 5, 6, 7], 'ReadKey', readAPIKey);

```

```

% New: Include the field value
alert_body = sprintf('AC: %d, Heater : %d, Blind: %d, RHED: %d', fieldValues);

```

```

% Form the JSON message
jsonmessage = sprintf(['{"subject": "%s", "body": "%s"}'], alert_subject, alert_body);

```

```

% Set options for web request
options = weboptions("HeaderFields", {'Thingspeak-Alerts-API-Key', alert_api_key; 'Content-Type','application/json'});

```

```

% Send the request
result = webwrite(alert_url, jsonmessage, options);

```

