Expe-smart house Project

GUO Bin
EL Kouch Yasmine

GUO Bin
EL Kouch Yasmine

Abstract

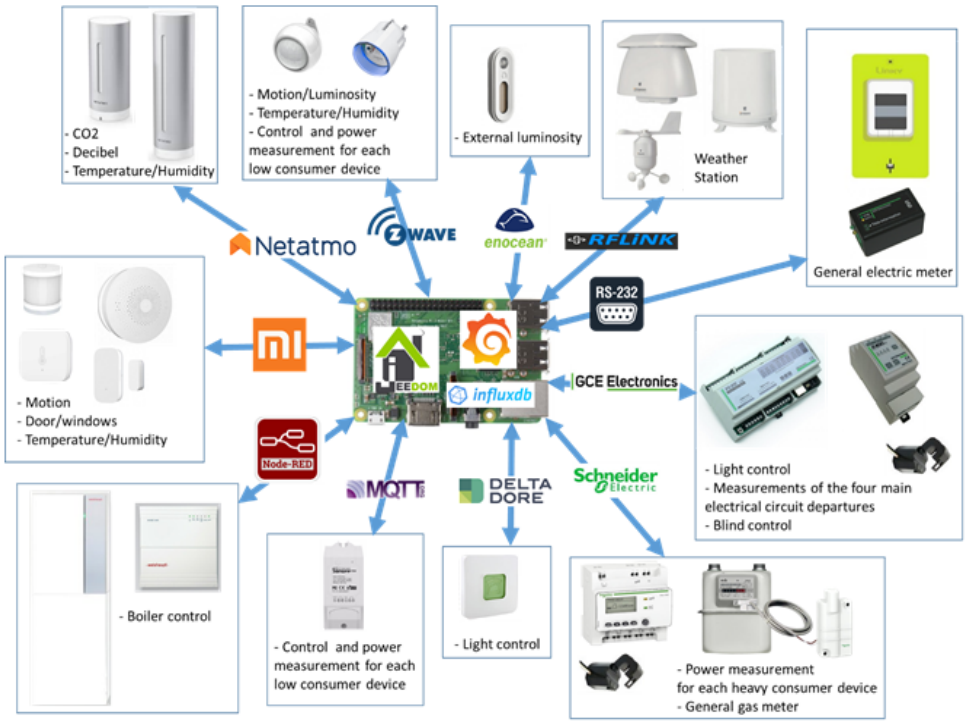
The smart home project helps to learn more about home automation technology. It is developed on the basis of open source hardware and software. And it is also flexible in adding new technologies or sensors. Such a system allows control of every kilowatts consumed and saves energy and money spent on electricity, heating and water supply.

This project provides access to around 340 measurement points, accessible in real time via a Grafana portal fed with an Influxdb database. The data we use for analysis in this project comes from a household of 120 m² in which a family of 5 lives. Here we mainly consider the energy usage in the living room. By collecting data on the different sensors, we can analyze the energy consumption in the living room and give some reasonable suggestions to residents.

Keywords : smart home, Grafana, Influxdb, sensors, collect data, energy consumption.

Performance of a smart house [1]

The smart home analysed in this project counts several sensors, here we give an example of the architecture used.



The reported measurements are :

- Consumption of electricity, gas and water
- Temperature, humidity and brightness of each room
- Opening position of each door and window
- Motion detectors
- State of lighting
- Analysis of the air in each room
- Outdoor weather conditions

It is worth noting that the sensor itself cannot store data. It can be connected to a computer via Wi-Fi, and the data measured will be sent to the web page on Grafana. Each ip address corresponds to a different sensor, by changing the address, we can reach different data.

About software

Grafana [2] is an open source platform designed for the monitoring, analysis and visualization of IT metrics. It comes with a web server allowing access to it via an HTTP API.Grafana generates its charts and dashboards from time series databases such as Graphite, InfluxDB or OpenTSDB.

For the project , we worked with InfluxDB [3] which is an open-source time series database (TSDB) developed by InfluxData. It is optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, IoT sensor data, and real-time analytics.

Object

We can collect the data from all sensors. The data is obtained by using grafana and saved in csv files. We will later treat the data with python. We are interested in controlling the energy, the heater and the lighting in the living room depending upon the data. We will then design programs that will allow us to better understand the data collected.

GUO Bin
EL Kouch Yasmine

Data collection

This step is very necessary for data analysis later. Our purpose is to gather all the data in the same table and store it in csv format. In order to achieve this goal, we need to use the influxdb package. Influxdb allows us to process and analyze time series data. We use 30-minute intervals to extract the measurement data and add it to the table, then use function **ffill** to fill in the missing values in the table.

Here is a part of our program :

```
data7 = pd.read_csv('C:\pythonProjectsmarthouse\output_Lumière.csv', sep=',')
data7["datetime"] = data7["time"].apply(stringdate_to_datetime)

data8 = pd.read_csv('C:\pythonProjectsmarthouse\output_Température.csv', sep=',')
data8["datetime"] = data8["time"].apply(stringdate_to_datetime)

data9 = pd.read_csv('C:\pythonProjectsmarthouse\output_TV_Power.csv', sep=',')
data9["datetime"] = data9["time"].apply(stringdate_to_datetime)


data1.set_index('datetime', inplace=True)
print("isnull", data1.isnull().any())
data1.ffill()
newdata1 = data1.resample('30T').sum()
print(newdata1)
```

- sep=‘,’ : It is a parameter of the print function, used to define the interval between output data.

- c:\pythonProjectsmarthouse\output Température.csv : The path road of temperature. csv file. By using the function pd.read_csv, we can read the file.

- ffill : Fill the missing values in the table.

- data1.resample(‘30T’).sum : Reprocess the original sample, re-sampling at 30-minute intervals

After collecting all data in files (each file for the data of one sensor), we then try to understand the values on them and give an analysis to process data efficiently.

The final table is shown below.

GUO Bin
EL Kouch Yasmine

datetime	CO2	window	TVpower	...	humidity	light	luminosity
2021-05-30 07:20:00	1034.0	0.0	0.0	...	55.0	0.0	519.0
2021-05-30 07:40:00	2783.0	0.0	0.0	...	164.0	0.0	1027.0
2021-05-30 08:00:00	2788.0	0.0	0.0	...	55.0	0.0	1011.0
2021-05-30 08:20:00	1800.0	0.0	0.0	...	110.0	0.0	961.0
2021-05-30 08:40:00	2626.0	0.0	0.0	...	110.0	0.0	902.0
...	...	...	...	...	...	...	...
2021-06-02 18:00:00	2398.0	0.0	0.0	...	118.0	0.0	198.0
2021-06-02 18:20:00	1219.0	0.0	0.0	...	60.0	0.0	157.0
2021-06-02 18:40:00	1773.0	0.0	0.0	...	120.0	0.0	96.0
2021-06-02 19:00:00	1670.0	0.0	0.0	...	60.0	0.0	90.0
2021-06-02 19:20:00	1818.0	0.0	0.0	...	180.0	0.0	30.0

Control energy

Our main object is to control energy, to manage it and to know when we are wasting energy and when it's all good. In order to have better consumption, we should test the CO2 concentration and check whether the windows are open at the same time to determine if there is energy waste.

It is worth noting that under normal situations, we should also consider the heating, but due to technical reasons we can not get the data about the heating, so we look here to consider just the two parameters of the window and the carbon dioxide concentration.

We use this program to test energy waste :

```
# Waste energie
window = newdata1['Window']
co2 = newdata2['CO2']
for i in range(len (window)):
    if window[i] == 1 and co2[i] < 1000:
        print("we are wasting energy")
    elif window[i] == 1 and co2[i] >= 1000 or window[i] == 0 and co2[i] <= 1000:
        print("good")
    elif window[i] == 0 and co2[i] > 1000:
        print("we need to open window")
```

```
good
we need to open window
we need to open window
good
we need to open window
we need to open window
we need to open window
we need to open window
good
we need to open window
we need to open window
good
we need to open window
we need to open window
we need to open window
we need to open window
we need to open window
we need to open window
we need to open window
we need to open window
we need to open window
```

Control heating

Sensors in the living room can allow us to monitor the temperature and adjust automatically the energy usage for the room. This is a great solution to that annoying cold room in the house which usually requires several blankets. We choose then to only analyse the data from temperature and humidity to know if it's necessary to open the heater or if it's all good.

We use this program to control the heater :

```
# Open heater
temperature = newdata8['Température']
humidity = newdata5['Humidité']
for i in range(len (temperature)):
    if temperature[i] < 15 and humidity[i] > 50:
        print("we need to open the heater")
    else:
        print("ok")     # tous les résultats sont "ok" car on est en été donc pas besoin d'ouvrir le chauffage
```

We consider that the heater should be turned on if it is cold, which means a temperature below 15 degrees and humidity above 50.

GUO Bin
EL Kouch Yasmine

Lighting control

As a waste of energy, we also consider lighting control. Simply, the same principle as before, we assume that the brightness is greater than 55 and the room lights are turned on as a waste of energy, and the lights in the room should be turned off.

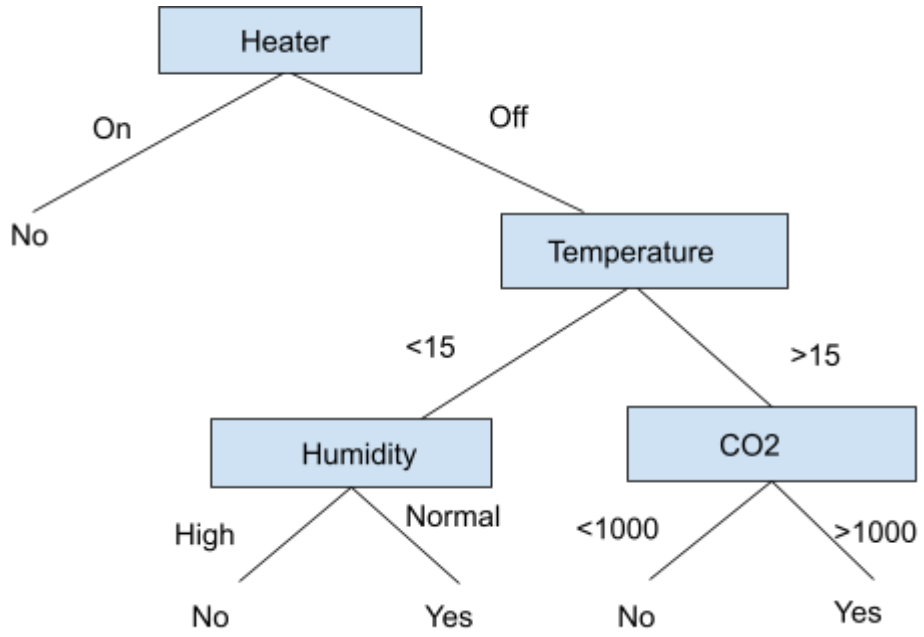
We use this program to control the lighting :

```
# Open lamp
luminosity = newdata6['Luminosité']
lamp = newdata7['Lumière']
for i in range(len (lamp)):
    if lamp[i] == 1 and luminosity[i] > 55:
        print("we are wasting energy")
    elif lamp[i] == 1 and luminosity[i] <= 55 or lamp[i] == 0 and luminosity[i] >= 55:
        print("nice")
    elif window[i] == 0 and luminosity[i] < 55:
        print("we need to open lamp")
```

Decision Tree Classifier

Another needed treatment into the data is using the decision trees. It can really be a very good solution for analysing data. Decision trees as part of the category of supervised algorithms, make it possible to predict a value (prediction) or a category (classification). In fact , we collect only the important variables.

The decision tree below classifies a time according to whether it is suitable to open the window and return the classification associated with the particular leaf. (in this case Yes or No).



For example :

if (Heater = 0 (off) , Temperature > 15 , CO2 < 1000)

Then, it would therefore be classified as a negative instance.

Conclusion

To summarize, these are the conclusions that can be extracted from the present project :

- Sensors are good helpers who can help us to make a more efficient use of energy. By using the program, we can analyze energy waste and give suggestions to the occupants.

- Decision trees are a very useful technique in machine learning. In this project, we are learning about Supervised learning (classification), which means that the training data are accompanied by labels indicating the class of the observations.

References

http://miniprojets.net/index.php/2021/01/29/security-risk-on-a-smart-home/
https://www.skedler.com/what-is-grafana-why-use-it-everything-you-should-know/
https://www.influxdata.com/products/

GUO Bin
EL Kouch Yasmine

INP-Grenoble ENSE3