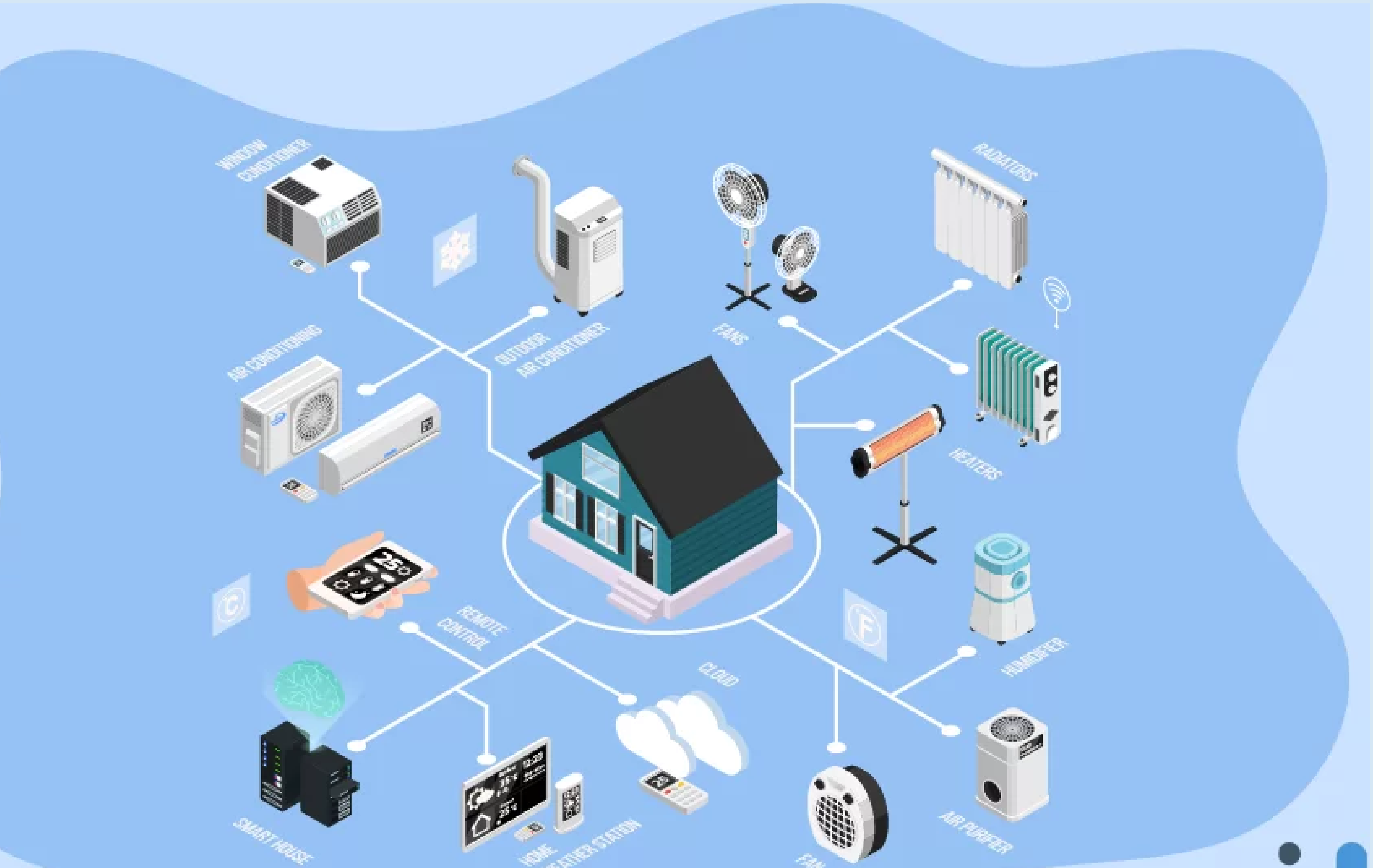
HOME AUTOMATION SYSTEM USING FUZZY LOGIC CONTROLLER

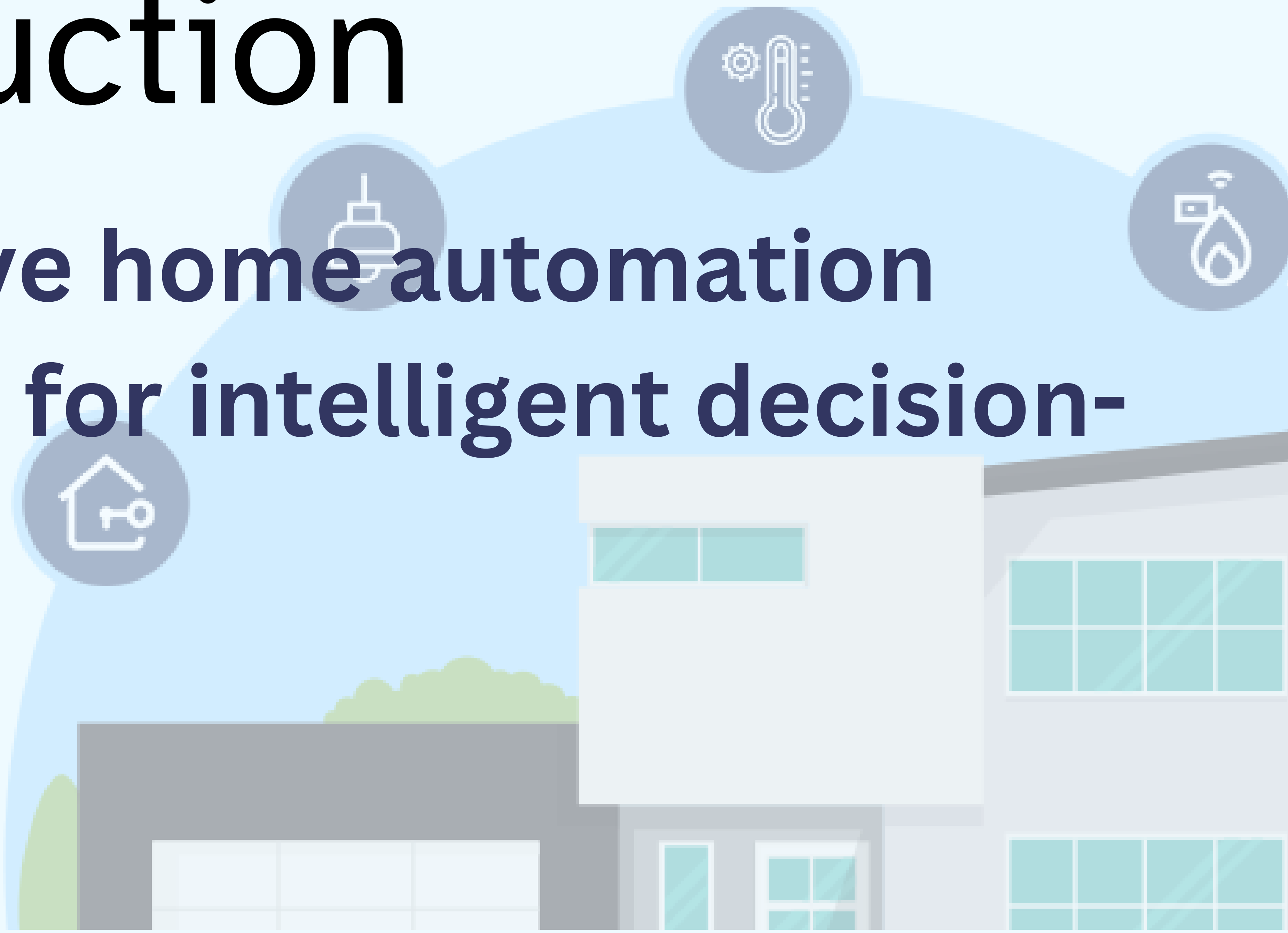
SY BTech DIV A1
HARSH NEVSE 31
MADHURA PATWARDHAN 44
REVATI JAGDALE 51

Guided by Prof. Jyoti Lele

Table Of Contents



Introduction



Creation of a comprehensive home automation system that uses fuzzy logic for intelligent decision-making.

Aim:

Incorporate fuzzy rules to optimize energy consumption, provide ease of life, and improve overall efficiency in managing various household devices.

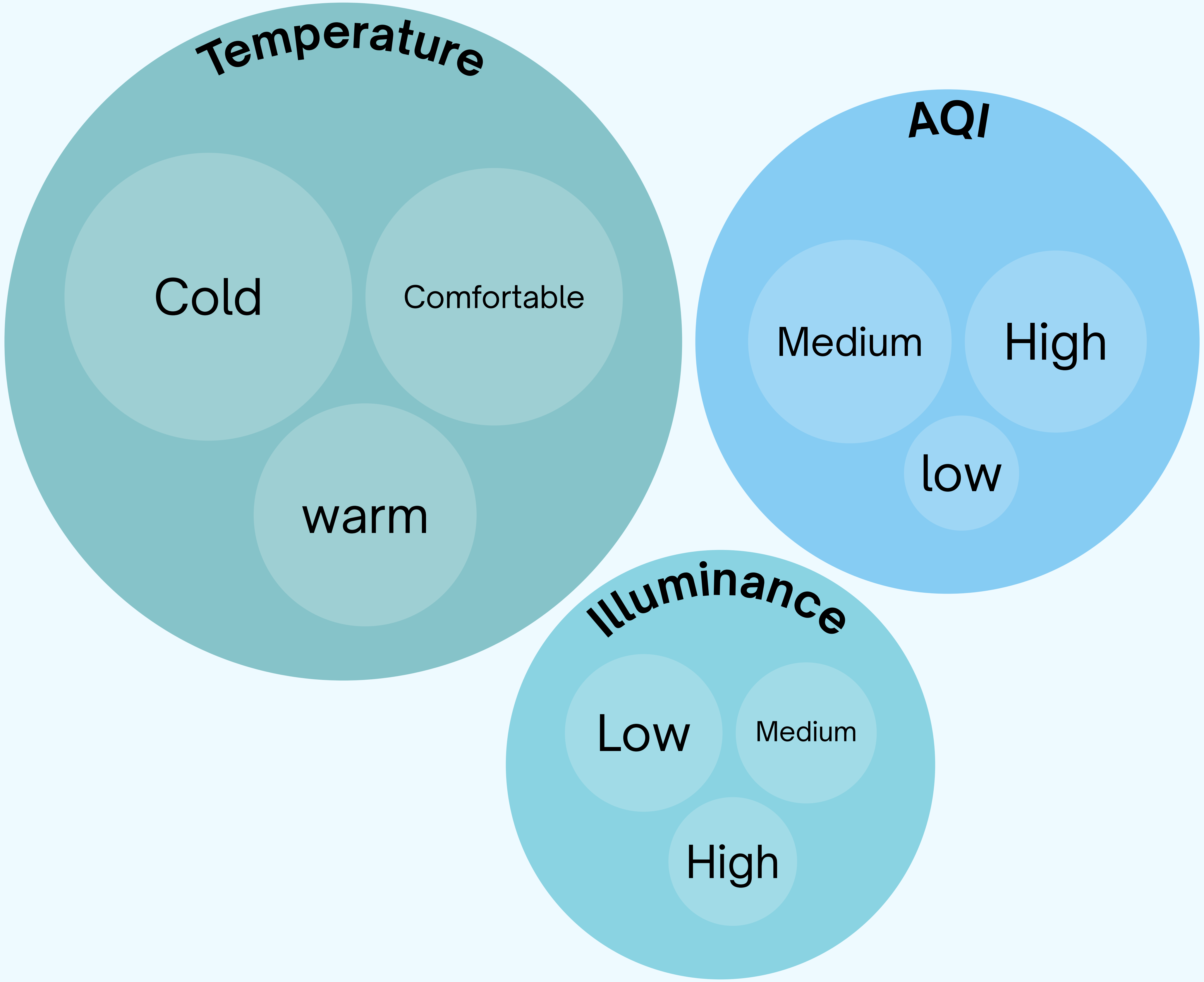
Focus:

Lighting Control: Develop a fuzzy logic system to adjust lighting levels based on factors like time of day, occupancy, and ambient light.

Temperature Regulation: Implement fuzzy logic for controlling heating, ventilation and air conditioning (HVAC) systems, ensuring comfortable and energy-efficient conditions.

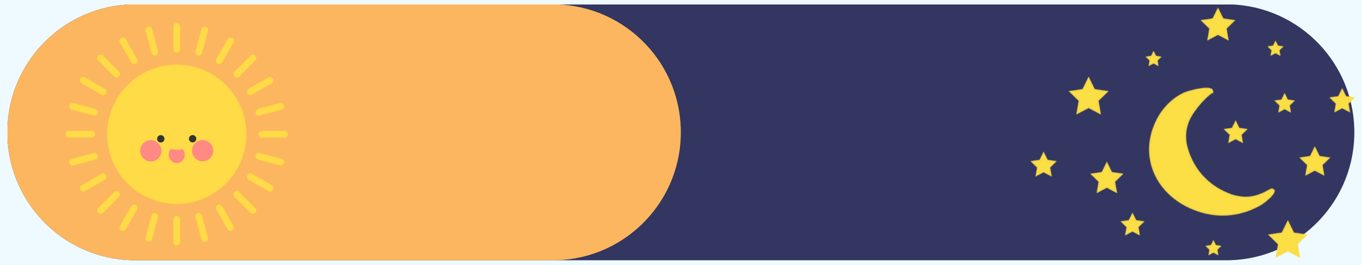
Air Quality index : Integrate fuzzy logic to manage air purifying systems and make them efficient as need requires.

Lingustic Variables Involved:



TIME OF DAY

50%=12pm



Working

Environmental variables measured by INPUT devices

Light
Temp
Intruder action
Pollution
Humidity
Noise
Wind

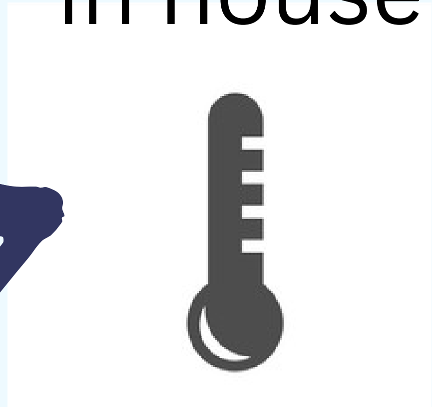
MEDIATOR

Fuzzy controller

Defuzzified output

Recommended action

OUTPUT devices in house



Packages involved

Numpy library

Pandas library

Matplotlib plotting library

Scikit-fuzzy-> fuzzy logic toolbox

MEDIUM OF PROGRAMMING:

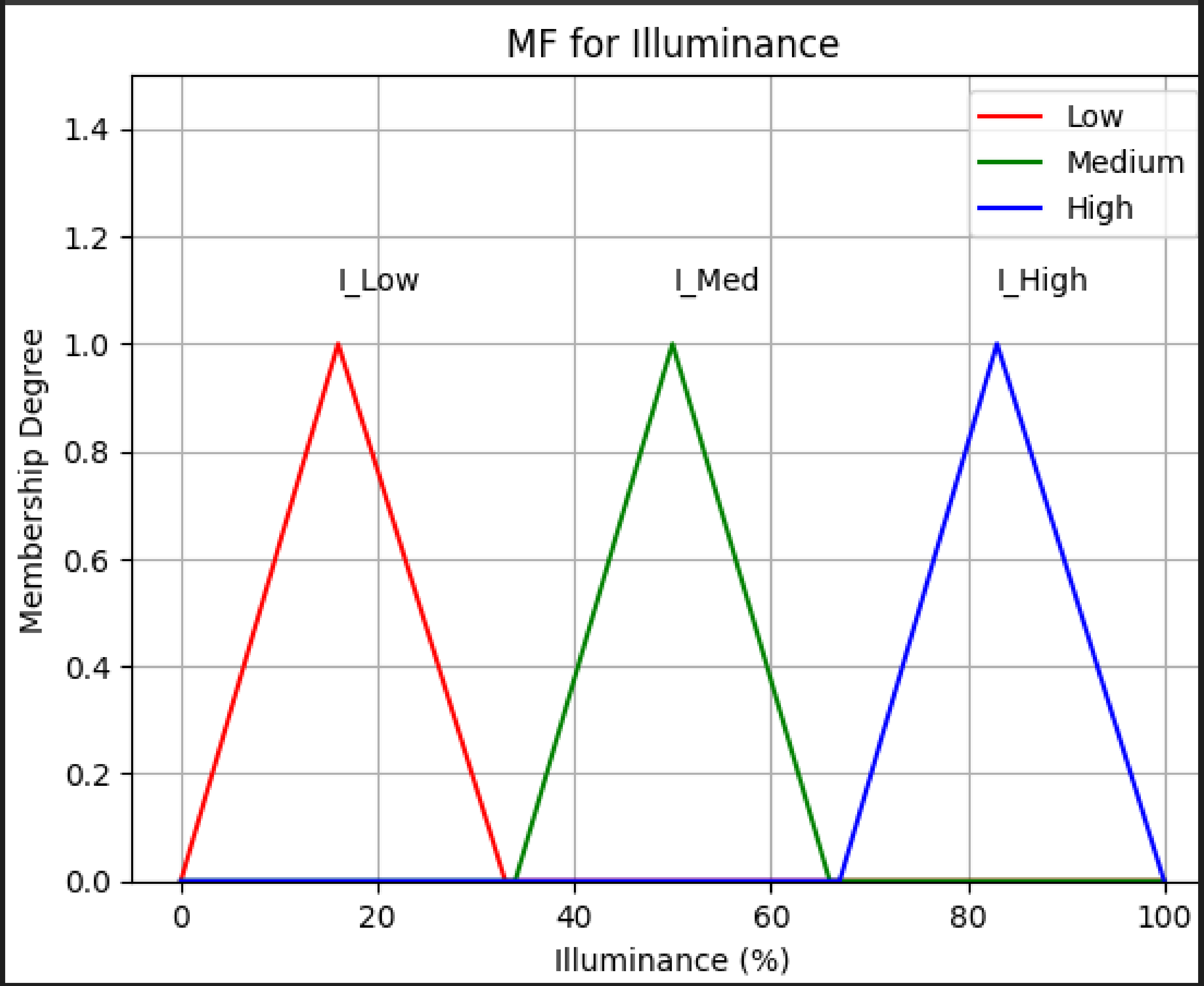
Jupyter Notebook (Python)

Fuzzy Rules:

```
# Illuminance

IF Illuminance IS High AND Time of Day IS Daytime THEN i_low
IF Illuminance IS Low AND Time of Day IS Night or Evening THEN i_high
IF Illuminance IS Low AND time of day is daytime THEN i_med


# Temperature

IF Temperature IS Cold AND Time of Day IS Night THEN t_warm
IF Temperature IS Warm AND Time of Day IS Daytime THEN t_cold
IF Temperature IS Comfortable AND Time of Day IS Evening THEN t_Comfort


# Air quality index(AQI)

IF AQI IS High THEN APS_power_high
IF AQI IS Low THEN APS_power_low
IF AQI IS Medium THEN APS_power_normal
```

Membership functions

```python
#ILLUMINANCE

range = np.arange(0, 101, 1)

i_low = fuzz.trimf(range, [0, 16, 33])
i_med = fuzz.trimf(range, [34, 50, 66])
i_high = fuzz.trimf(range, [67, 83, 100])

fig,a = plt.subplots()
a.plot(range, i_low, 'r', linewidth=1.5, label='Low')
a.plot(range, i_med, 'g', linewidth=1.5, label='Medium')
a.plot(range, i_high, 'b', linewidth=1.5, label='High')

plt.title('MF for Illuminance')
plt.xlabel('Illuminance (%)')
a.set_ylim(0,1.5)
plt.ylabel('Membership Degree')

plt.text(16,1.1,'I_Low')
plt.text(50,1.1,'I_Med')
plt.text(83,1.1,'I_High')

plt.legend()
plt.grid()
plt.show()
```



Application of Mamdani inference system

Methodology

1. Interpolation is a technique for adding new data points within a range of a set of known data points.
2. The max-min (or Mamdani) inference:
   Minimum as logical AND
   Maximum as logical OR

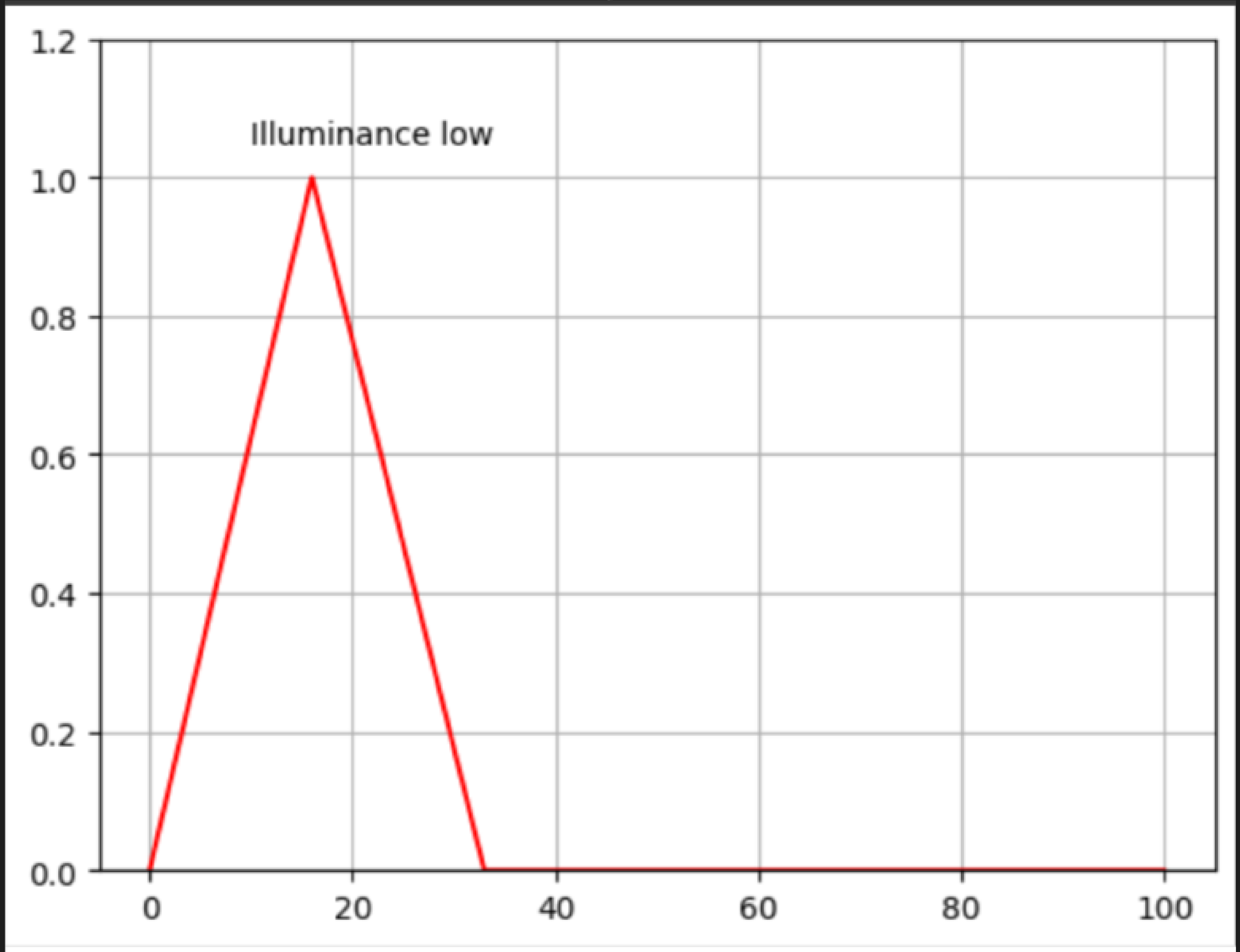
Implementation of rules in mamdani

```
i_low_degree = fuzz.interp_membership(range, i_low, i_value)
i_med_degree = fuzz.interp_membership(range, i_med, i_value)
i_high_degree = fuzz.interp_membership(range, i_high, i_value)

t_cold_degree = fuzz.interp_membership(range, t_cold, t_value)
t_comfortable_degree = fuzz.interp_membership(range, t_comfortable, t_value)
t_warm_degree = fuzz.interp_membership(range, t_warm, t_value)

aq_low_degree = fuzz.interp_membership(range,aq_low,aqi_value)
aq_med_degree = fuzz.interp_membership(range,aq_med,aqi_value)
aq_high_degree = fuzz.interp_membership(range,aq_high,aqi_value)

tod_daytime_degree = fuzz.interp_membership(range, tod_daytime, tod_value)
tod_evening_degree = fuzz.interp_membership(range, tod_evening, tod_value)
tod_night_degree = fuzz.interp_membership(range, tod_night, tod_value)


rule1_1 = min(i_high_degree, tod_daytime_degree)  # IF Illuminance is high AND TOD is daytime THEN i_low
rule1_2 = min(i_low_degree,tod_night_degree)      # IF Illumninance is low and TOD is night THEN i_high
rule1_3 = min(i_low_degree,tod_evening_degree)    # IF Illuminance IS Low AND time of evening THEN i_med
```

Interpolation

```python
rule1_1 = min(i_high_degree, tod_daytime_degree)  # IF Illuminance is high AND TOD is daytime THEN i_low
rule1_2 = min(i_low_degree,tod_night_degree)      # IF Illumninance is low and TOD is night THEN i_high
rule1_3 = min(i_low_degree,tod_evening_degree)    # IF Illuminance IS Low AND time of evening THEN i_med

if rule1_1>0:
  fig,a = plt.subplots()
  a.plot(i_low,'r')
  plt.text(10,1.05,'Illuminance low')
  a.set_ylim(0,1.2)
elif rule1_2>0:
  fig,a = plt.subplots()
  a.plot(i_high,'b')
  plt.text(75,1.05,'Illuminance high')
  a.set_ylim(0,1.2)
elif rule1_3>0:
  fig,a=plt.subplots()
  a.plot(i_med,'b')
  plt.text(45,1.05,'Illuminance Medium')
  a.set_ylim(0,1.2)
else:
  fig,a=plt.subplots()
  a.set_ylim(0,1.3)
  plt.text(0.2,0.5,"No specific action suggested")

plt.grid()
```

OUTPUT: ideally illuminance=low



Conclusion

Hence, we have employed Mamdani FIS to give us a crisp output, which will be communicated to the output devices and ultimately result in an independent, power-efficient system.