University of Colima
Faculty of Mechanical and Electrical Engineering
Intelligent Computer Engineering

Analysis of the behavior of the agroclimatic variables related to his broccoli crop.
Data analysis and visualization

Ceballos Contreras Jesús Eduardo 20176724
Larios Bravo Cristian Armando 20188165
López Aguilar Alba Beth-birai 20186885
Olmos Romero Nydia Naomi 20184561

6°D

Place: Mexico, Colima, Coquimatlan.
Date: 19/03/2024.

INTRODUCTION

Precision agriculture (PA) is a farming management strategy based on observing, measuring and responding to temporal and spatial variability to improve agricultural production sustainability. It is used in both crop and livestock production. Precision agriculture often employs technologies to automate agricultural operations, improving their diagnosis, decision-making or performing. Monitoring climate variables and data analysis in agriculture are fundamental components to improve efficiency, productivity and sustainability in the agricultural sector. The phases or tasks of a monitoring system consist of:

- Sensors and monitoring devices
- Data collection
- Analysis of data
- Data visualization
- Data-driven decision making

DEVELOPMENT

Create a Jupyter Notebook and write the Pandas code to:

- Load the five datasets.
- Merge the five files in a only dataset using the **Pandas Merge Function.**

In [ ]:
```
# We import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
# We load the datasets and then join them into one
df1 = pd.read_csv("01.csv")
df2 = pd.read_csv("02.csv")
df3 = pd.read_csv("03.csv")
df4 = pd.read_csv("04.csv")
df5 = pd.read_csv("05.csv")

# Concatenate the DataFrames along axis 0
df = pd.concat([df1, df2, df3, df4, df5], axis=0)
df
```

Out[ ]:

	'Temperature'	'Humidity'	'Humidity of the terrain'	'Luminosity'	'Ph'	'Speed of the wind'
0	17	78	35.0	51	5	4
1	27	96	37.0	60	4	20
2	32	75	15.0	85	5	8
3	16	31	47.0	16	5	2
4	38	41	89.0	17	5	5
...	...	...	...	...	...	...
995	35	43	78.0	42	5	0
996	16	53	90.0	94	7	15
997	29	64	67.0	74	7	17
998	24	56	98.0	96	6	19
999	15	61	62.0	80	7	28

5000 rows × 6 columns

EXPLORATORY ANALYSIS

Perform exploratory analysis of the dataset using basic Pandas commands.

In [ ]:
```
# Review the first rows of the DataFrame to understand the structure of the data.
df.head()
```

Out[ ]:

	'Temperature'	'Humidity'	'Humidity of the terrain'	'Luminosity'	'Ph'	'Speed of the wind'
0	17	78	35.0	51	5	4
1	27	96	37.0	60	4	20
2	32	75	15.0	85	5	8
3	16	31	47.0	16	5	2
4	38	41	89.0	17	5	5

In [ ]:
```
# Review the last rows of the DataFrame to understand the structure of the data.
df.tail()
```

Out[ ]:

	'Temperature'	'Humidity'	'Humidity of the terrain'	'Luminosity'	'Ph'	'Speed of the wind'
995	35	43	78.0	42	5	0
996	16	53	90.0	94	7	15
997	29	64	67.0	74	7	17
998	24	56	98.0	96	6	19
999	15	61	62.0	80	7	28

In [ ]:
```
# Check and handle missing data (NaN values) if necessary.
print(f"Data before dropping NaN values: \n{df.isna().sum()}")
```

```
Data before dropping NaN values:
'Temperature'                0
 'Humidity'                  0
 'Humidity of the terrain'   2
 'Luminosity'                0
 'Ph'                        0
 'Speed of the wind'         0
dtype: int64
```

In [ ]:
```
df = df.dropna()
print(f"Data after dropping NaN values: {df.isna().sum()}")
```

```
Data after dropping NaN values: 'Temperature'                0
 'Humidity'                  0
 'Humidity of the terrain'   0
 'Luminosity'                0
 'Ph'                        0
 'Speed of the wind'         0
dtype: int64
```

In [ ]:
```
# Obtain basic statistical information about numerical variables.
df.describe()
```

Out[ ]:

	'Temperature'	'Humidity'	'Humidity of the terrain'	'Luminosity'	'Ph'	'Speed of the wind'
count	4998.000000	4998.000000	4998.000000	4998.000000	4998.000000	4998.000000
mean	29.975590	64.610844	48.810324	48.379752	5.466587	14.585834
std	9.084543	20.328885	28.610564	29.457490	1.168076	8.666718
min	15.000000	30.000000	0.000000	0.000000	4.000000	0.000000
25%	22.000000	47.000000	23.250000	22.000000	4.000000	7.000000
50%	30.000000	65.000000	49.000000	48.000000	5.000000	15.000000
75%	38.000000	83.000000	73.000000	74.000000	6.000000	22.000000
max	144.000000	99.000000	99.000000	99.000000	28.000000	29.000000

MEASURES OF CENTRAL TENDENCY

For each numerical column, in code cells adds the calculation of measures of central tendency (mean, mode, median, min, max).

MEAN

In [ ]:
```
# Calculate the mean of each column
media = df.mean()
print("Mean:\n", media)
```

```
Mean:
 'Temperature'                 29.975590
 'Humidity'                   64.610844
 'Humidity of the terrain'    48.810324
 'Luminosity'                 48.379752
 'Ph'                          5.466587
 'Speed of the wind'          14.585834
dtype: float64
```

MODE

In [ ]:
```
# Calculate the mode of each column
moda = df.mode().iloc[0]
print("Mode:\n", moda)
```

```
Mode:
 'Temperature'                 44.0
 'Humidity'                   87.0
 'Humidity of the terrain'    57.0
 'Luminosity'                  0.0
 'Ph'                          4.0
 'Speed of the wind'           3.0
Name: 0, dtype: float64
```

MEDIAN

In [ ]:
```
# Calculate the median of each column
mediana = df.median()
print("Median:\n", mediana)
```

```
Median:
 'Temperature'                30.0
 'Humidity'                   65.0
 'Humidity of the terrain'    49.0
 'Luminosity'                 48.0
 'Ph'                          5.0
 'Speed of the wind'          15.0
dtype: float64
```

MIN

In [ ]:
```
# Calculate the minimum of each column
minimo = df.min()
print("Minimum:\n", minimo)
```

```
Minimum:
 'Temperature'                15.0
 'Humidity'                   30.0
 'Humidity of the terrain'     0.0
 'Luminosity'                  0.0
 'Ph'                          4.0
 'Speed of the wind'           0.0
dtype: float64
```

MAX

In [ ]:
```
# Calculate the maximum of each column
maximo = df.max()
print("Maximum:\n", maximo)
```

```
Maximum:
 'Temperature'                144.0
 'Humidity'                    99.0
 'Humidity of the terrain'     99.0
 'Luminosity'                  99.0
 'Ph'                          28.0
 'Speed of the wind'           29.0
dtype: float64
```

DISPERSION MEASURES

For each numerical column, in code cells add the calculation of dispersion measures:

- Range. The difference between the maximum value and the minimum value in a set of data.

In [ ]:
```
# Calculate the range for each column
rango = df.max() - df.min()
print("Range:\n", rango)
```

```
Range:
 'Temperature'                 129.0
 'Humidity'                     69.0
 'Humidity of the terrain'      99.0
 'Luminosity'                   99.0
 'Ph'                           24.0
 'Speed of the wind'            29.0
dtype: float64
```

- Standard Deviation: Standard deviation is a measure of the average dispersion of values with respect to the mean.

In [ ]:
```
# Calculate the standard deviation for each column
desvStd = df.std()
print("\nStandard Deviation:\n", desvStd)
```

```
Standard Deviation:
 'Temperature'                  9.084543
 'Humidity'                    20.328885
 'Humidity of the terrain'     28.610564
 'Luminosity'                  29.457490
 'Ph'                           1.168076
 'Speed of the wind'            8.666718
dtype: float64
```

- Variance: The variance is the square of the standard deviation.

In [ ]:
```
# Calculate the variance for each column
varianza = df.var()
print("\nVariance:\n", varianza)
```

```
Variance:
 'Temperature'                  82.528922
 'Humidity'                    413.263577
 'Humidity of the terrain'     818.564376
 'Luminosity'                  867.743692
 'Ph'                            1.364403
 'Speed of the wind'            75.112003
dtype: float64
```

- Interquartile Range (IQR). The interquartile range is the difference between the third quartile (Q3) and the first quartile (Q1) in an ordered data set.

In [ ]:
```
# Calculate the interquartile range (IQR) for each column
ric = df.quantile(0.75) - df.quantile(0.25)
print("\nInterquartile Range (IQR):\n", ric)
```

```
Interquartile Range (IQR):
 'Temperature'                 16.00
 'Humidity'                   36.00
 'Humidity of the terrain'    49.75
 'Luminosity'                 52.00
 'Ph'                          2.00
 'Speed of the wind'          15.00
dtype: float64
```

- Coefficient of variation (CV). The coefficient of variation is a measure of the relative dispersion of the data. You obtain it by dividing the standard deviation by the mean and multiplying the result by 100.

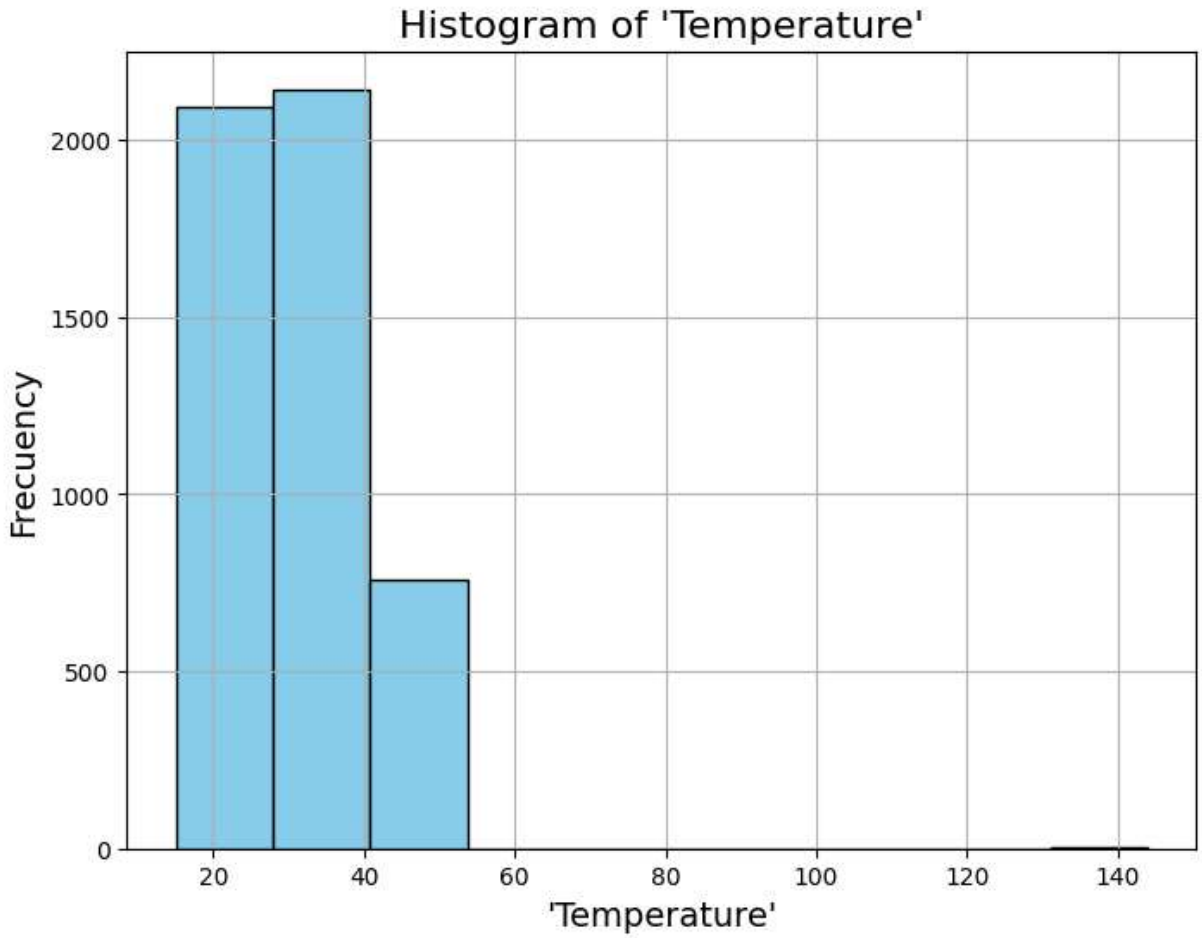
In [ ]:
```
# Calculate the coefficient of variation (CV) for each column
cDv = (df.std() / df.mean()) * 100
print("\nCoefficient of Variation (CV):\n", cDv)
```

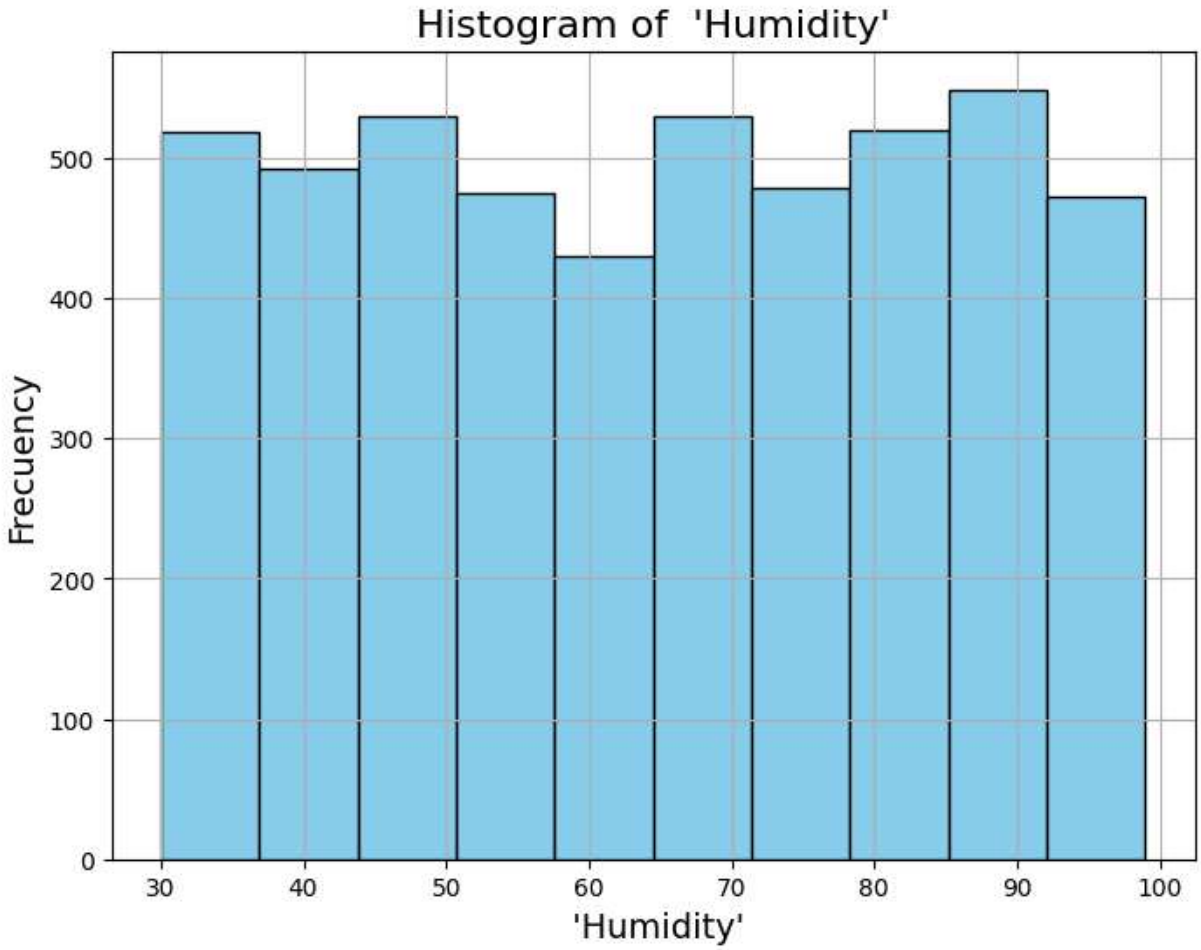
```
Coefficient of Variation (CV):
 'Temperature'                 30.306469
 'Humidity'                   31.463581
 'Humidity of the terrain'    58.615804
 'Luminosity'                 60.888054
 'Ph'                         21.367565
 'Speed of the wind'          59.418734
dtype: float64
```

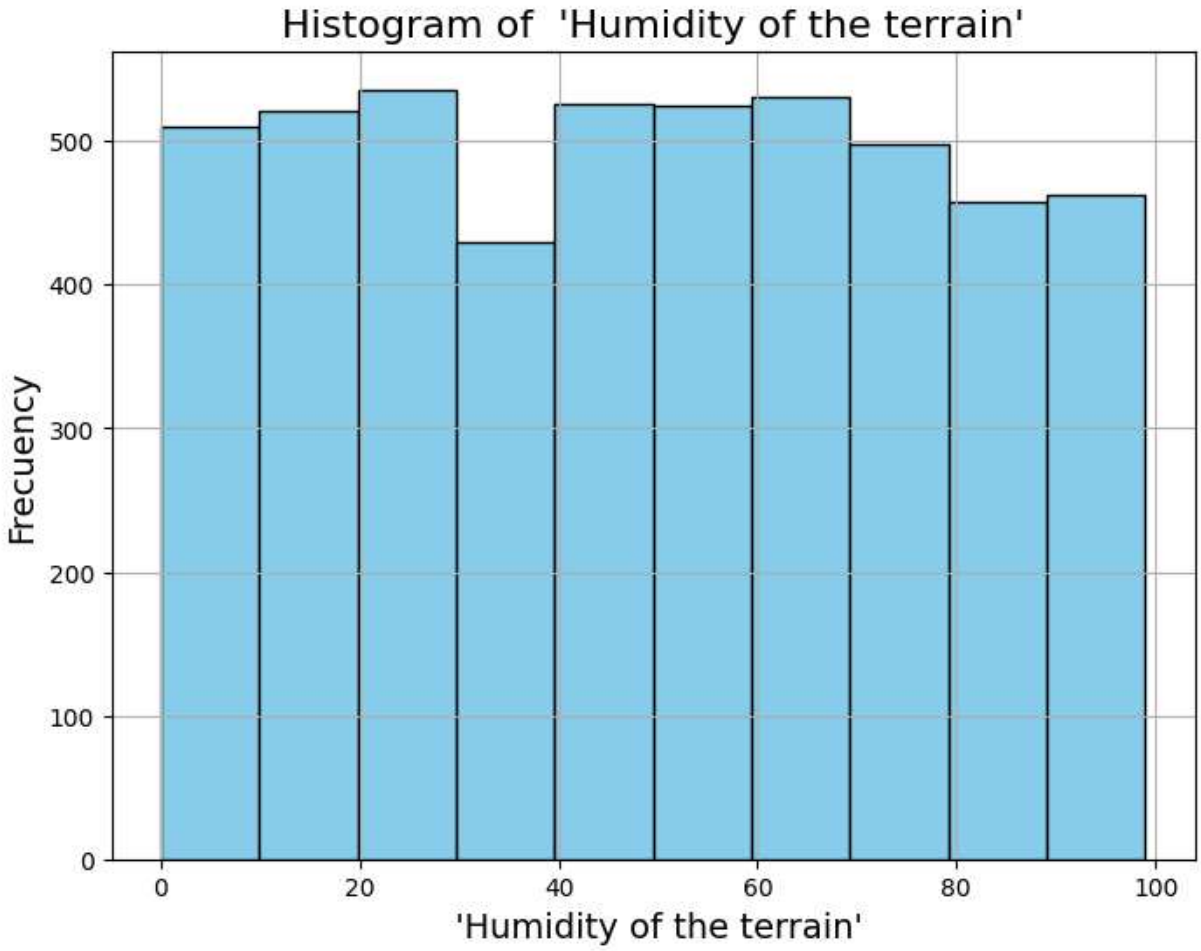
HISTOGRAMS

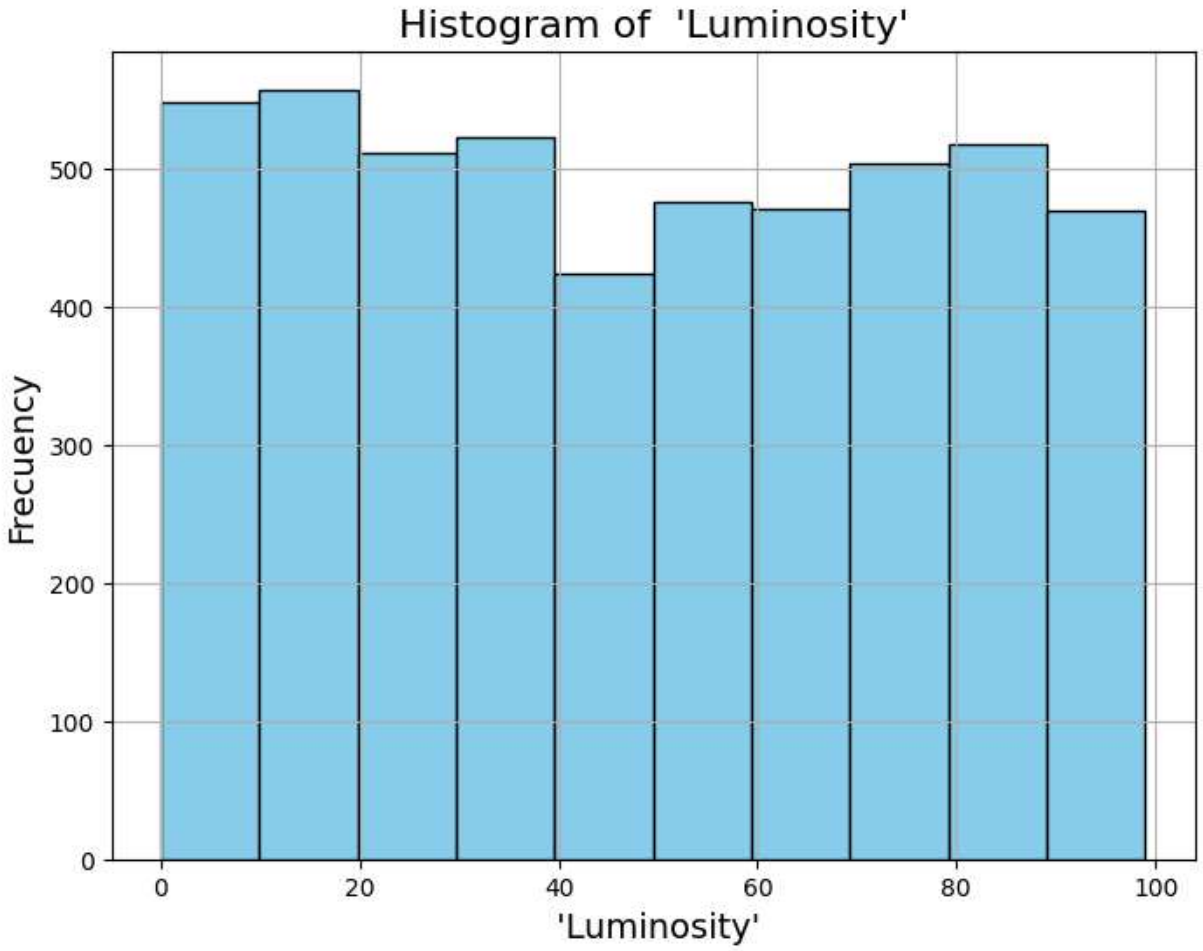
For each numerical column **graph the histogram** of the data using matplotlib library.

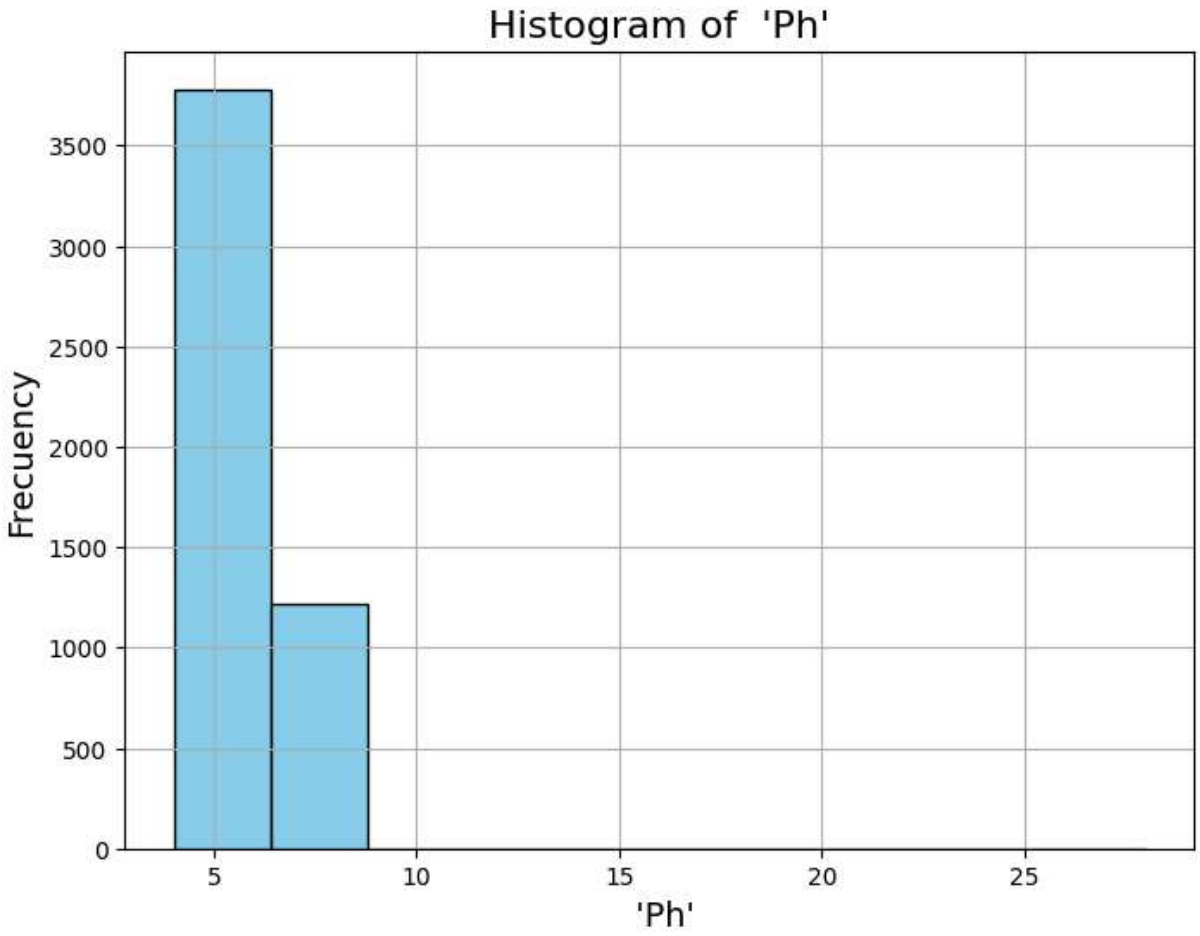
In [ ]:
```
# Get the numeric columns
columnas_numericas = df.select_dtypes(include=['int64', 'float64']).columns

# Plot histogram for each numeric column
for columna in columnas_numericas:
    plt.figure(figsize=(8, 6))
    plt.hist(df[columna], bins=10, color='skyblue', edgecolor='black')  # Personali
    plt.title(f'Histogram of {columna}', fontsize=16)
    plt.xlabel(columna, fontsize=14)
    plt.ylabel('Frecuency', fontsize=14)
    plt.grid(True)
    plt.show()
```

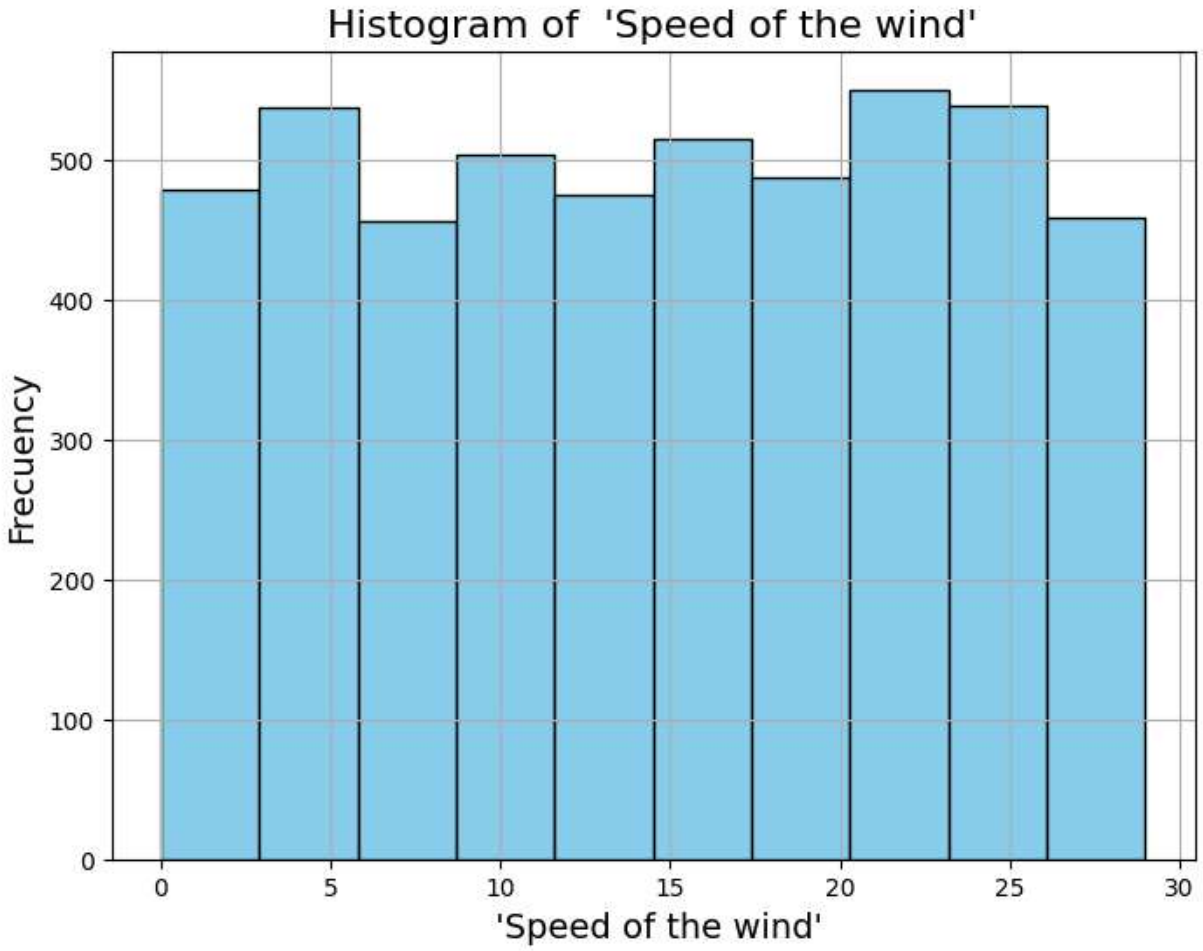

Histogram of 'Temperature'
Frecuency
2000
1500
1000
500
0
20
40
60
80
100
120
140
'Temperature'


Histogram of 'Humidity'
Frecuency
0
100
200
300
400
500
30
40
50
60
70
80
90
100
'Humidity'


Histogram of 'Humidity of the terrain'
Frecuency
0
100
200
300
400
500
0
20
40
60
80
100
'Humidity of the terrain'


Histogram of 'Luminosity'
Frecuency
0
100
200
300
400
500
0
20
40
60
80
100
'Luminosity'


Histogram of 'Ph'
Frecuency
3500
3000
2500
2000
1500
1000
500
0
5
10
15
20
25
'Ph'



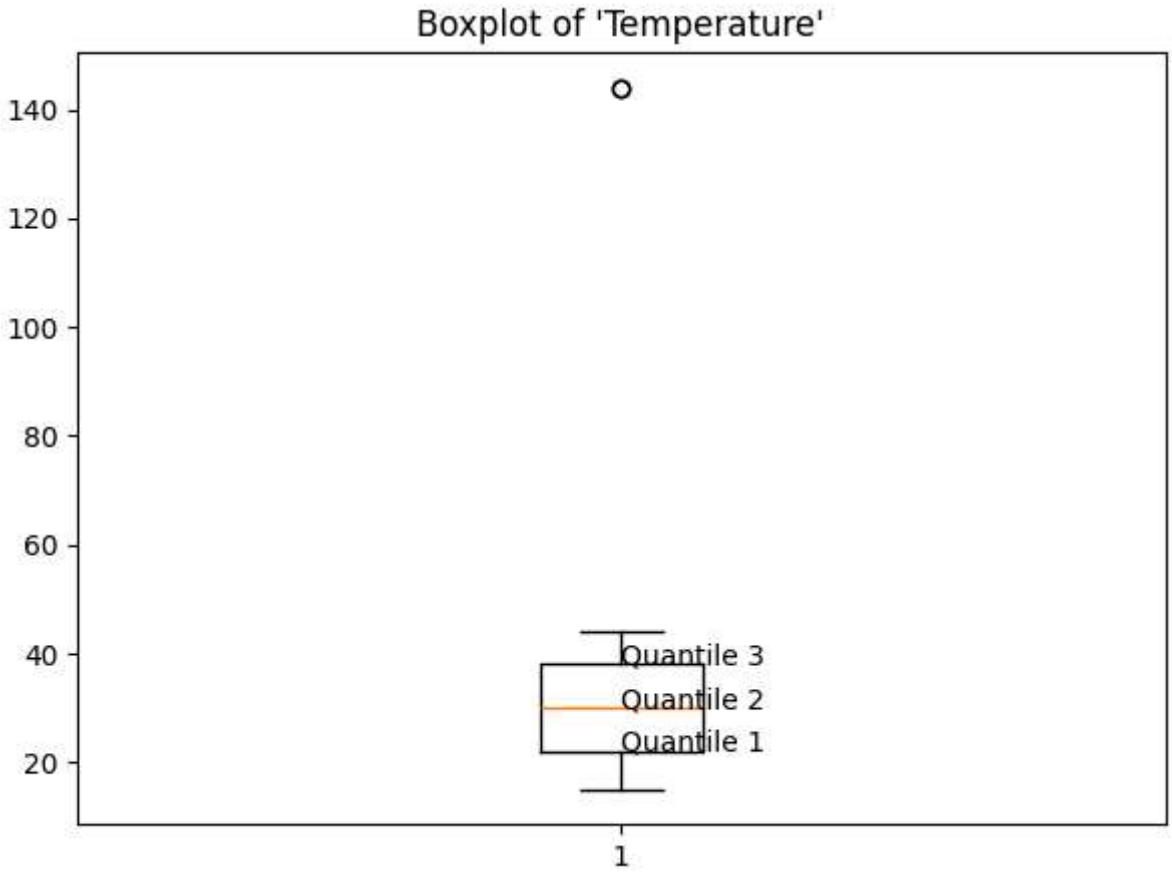
WISKER GRAPHICS

For each numerical column create a box or whisker graphics using matplotlib library.

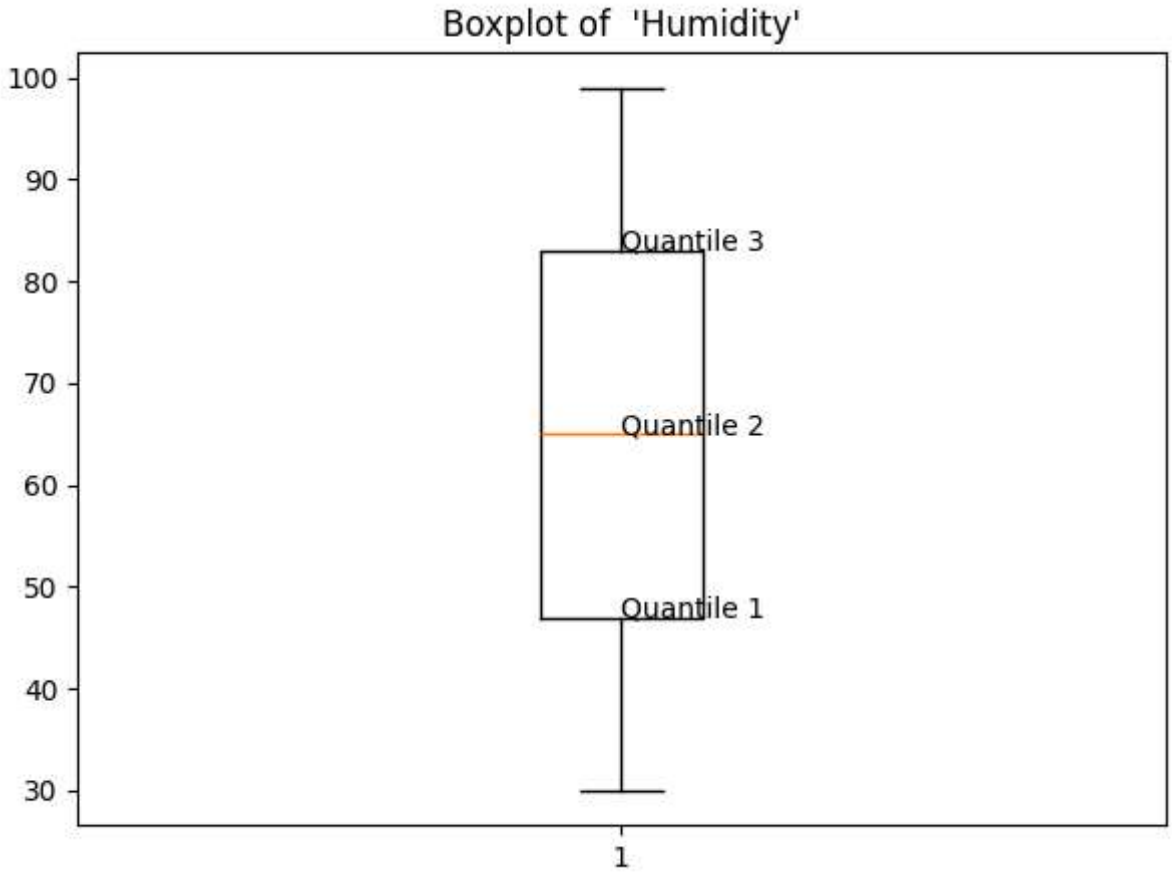
In [ ]:
```
# Get the numeric columns
columnas_numericas = df.select_dtypes(include=['int64', 'float64']).columns


for columna in columnas_numericas:
    # Calculate quartiles
    median = df[columna].median()
    q1 = df[columna].quantile(0.25)
    q2 = df[columna].quantile(0.50)
    q3 = df[columna].quantile(0.75)

    print(f"----{columna}---")
    print("Median:", median)
    print("Quantile 1:", q1)
    print("Quantile 2:", q2)
    print("Quantile 3:", q3)


    # Create the boxplot
    plt.figure(figsize=(7, 5))
    plt.boxplot(df[columna], vert=True)
```

```
    # Add labels to each quartile
    plt.text(1, q1, 'Quantile 1')
    plt.text(1, q2, 'Quantile 2')
    plt.text(1, q3, 'Quantile 3')
    plt.title(f'Boxplot of {columna}')

    plt.show()
```

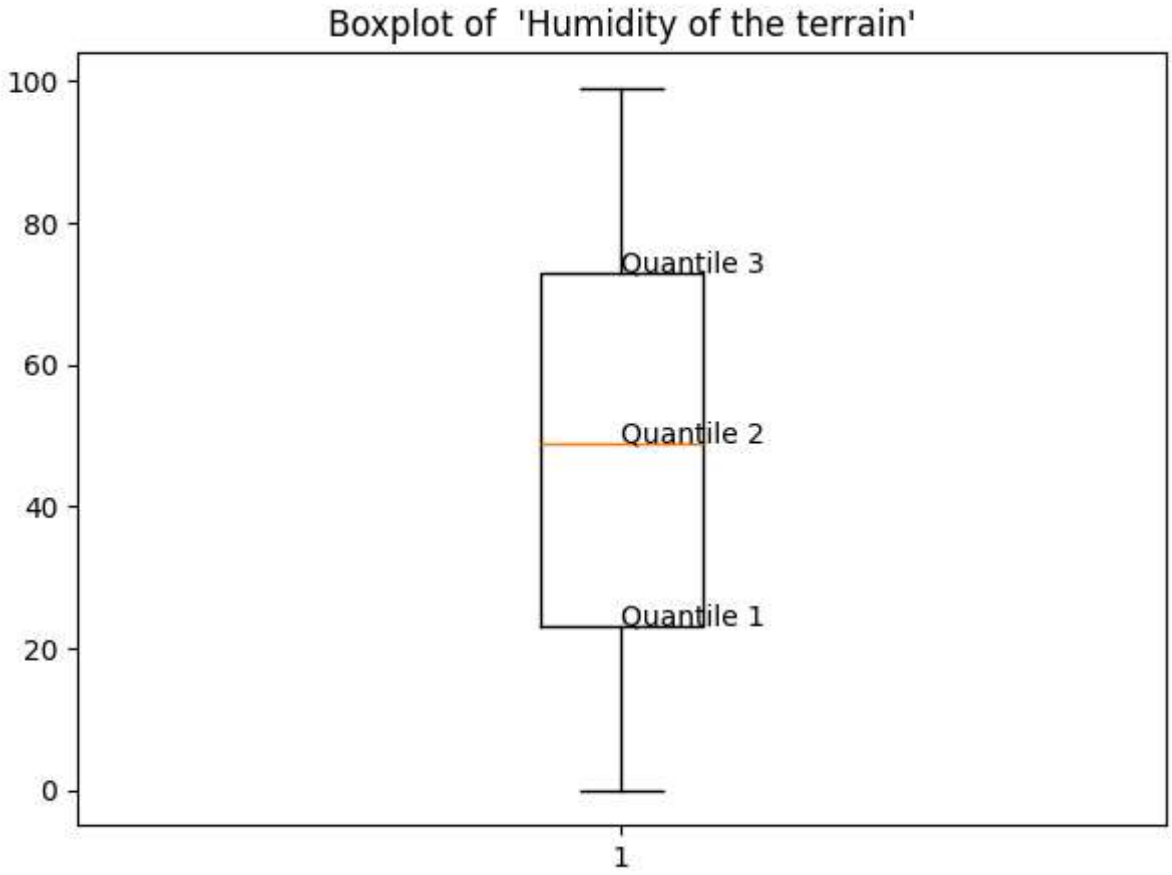
```
----'Temperature'---
Median: 30.0
Quantile 1: 22.0
Quantile 2: 30.0
Quantile 3: 38.0
```



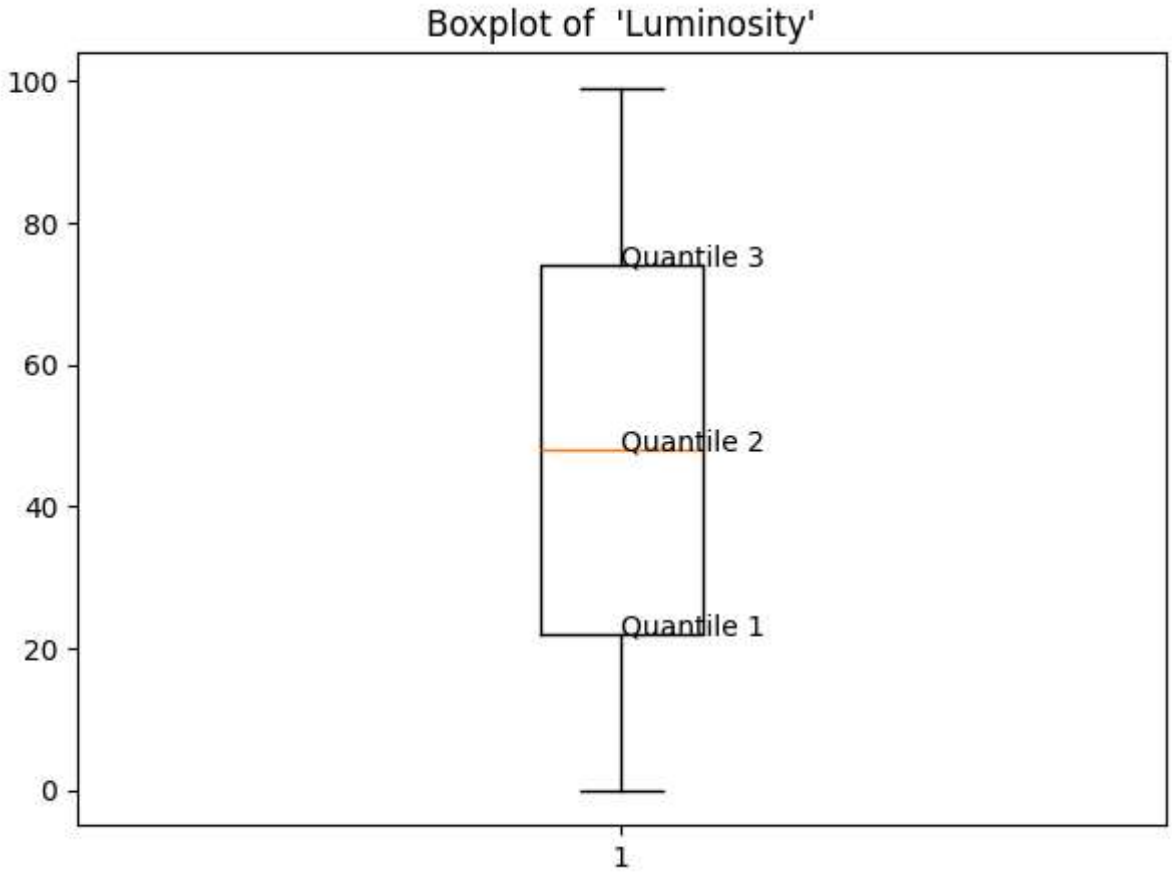

```
---- 'Humidity'---
Median: 65.0
Quantile 1: 47.0
Quantile 2: 65.0
Quantile 3: 83.0
```



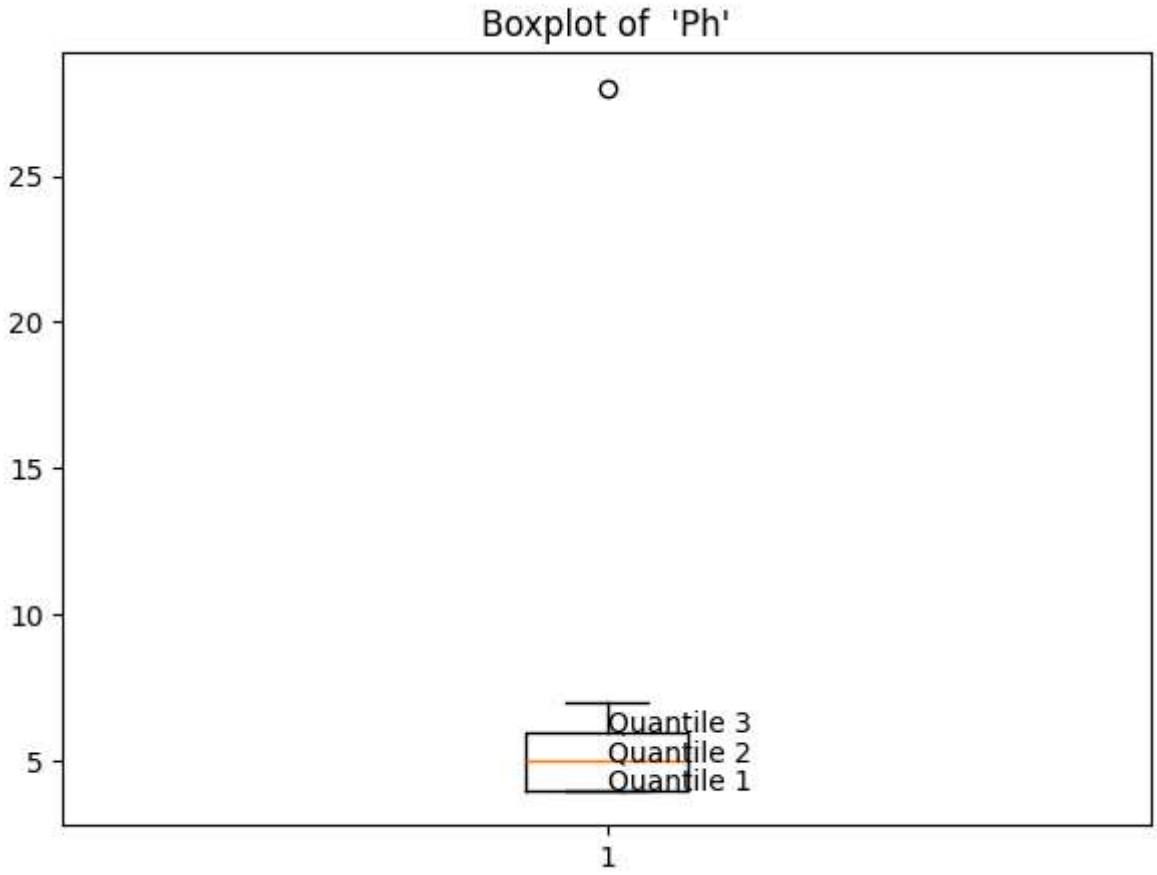

```
---- 'Humidity of the terrain'---
Median: 49.0
Quantile 1: 23.25
Quantile 2: 49.0
Quantile 3: 73.0
```



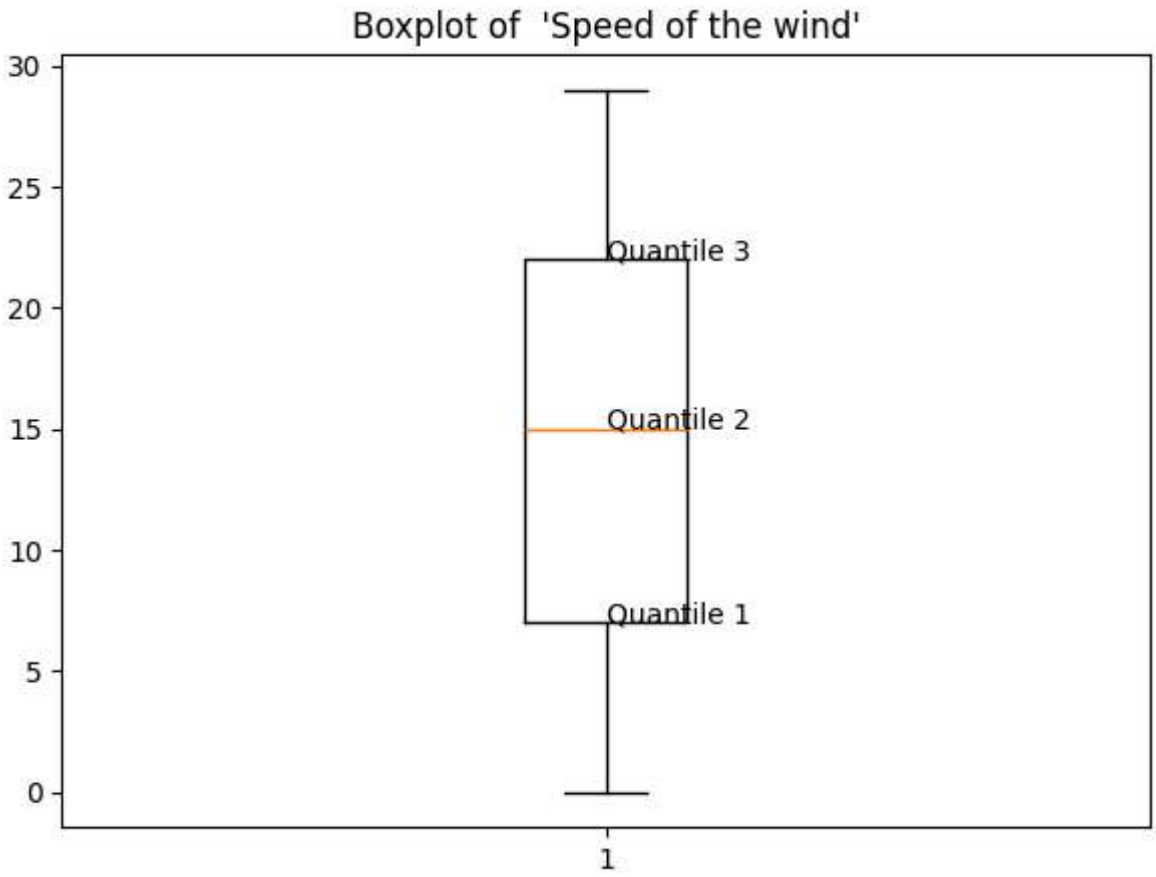

```
---- 'Luminosity'---
Median: 48.0
Quantile 1: 22.0
Quantile 2: 48.0
Quantile 3: 74.0
```



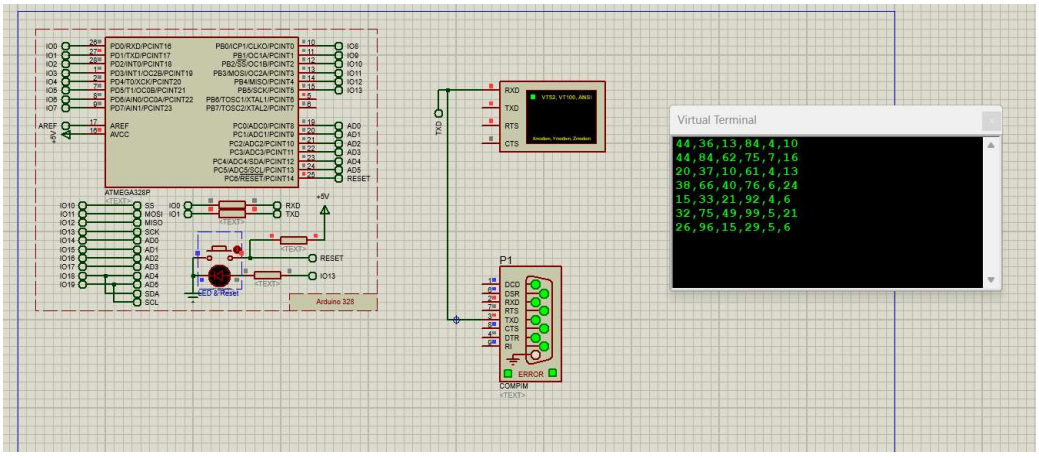

```
---- 'Ph'---
Median: 5.0
Quantile 1: 4.0
Quantile 2: 5.0
Quantile 3: 6.0
```




```
---- 'Speed of the wind'---
Median: 15.0
Quantile 1: 7.0
Quantile 2: 15.0
Quantile 3: 22.0
```



PROTEUS CIRCUIT. ARDUINO



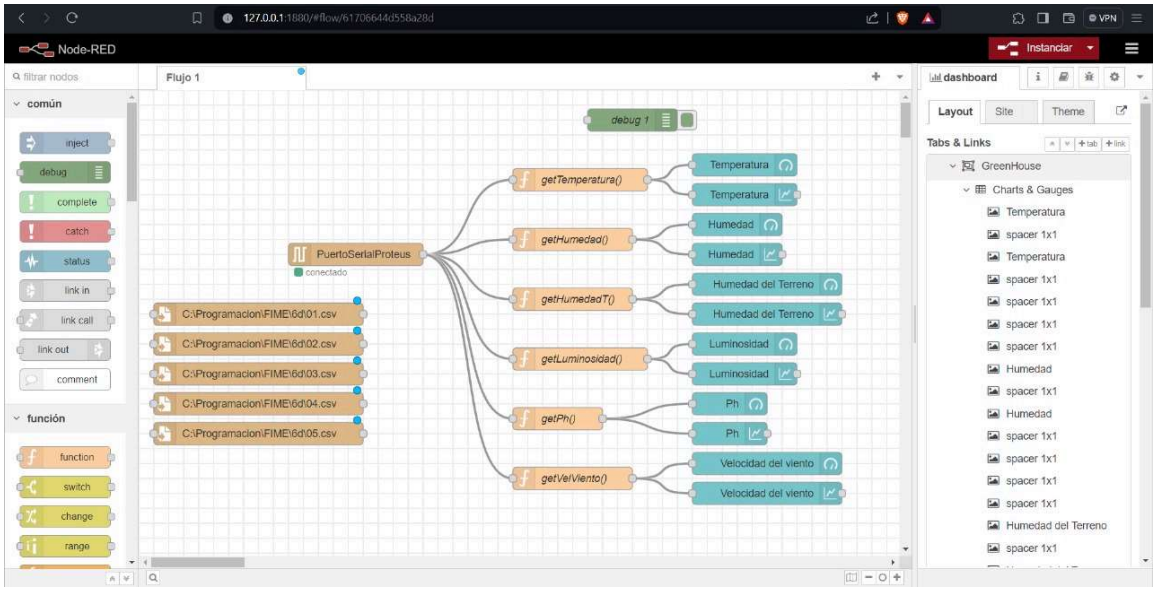
DATASETS GENERATION

NODE RED WEBSITE

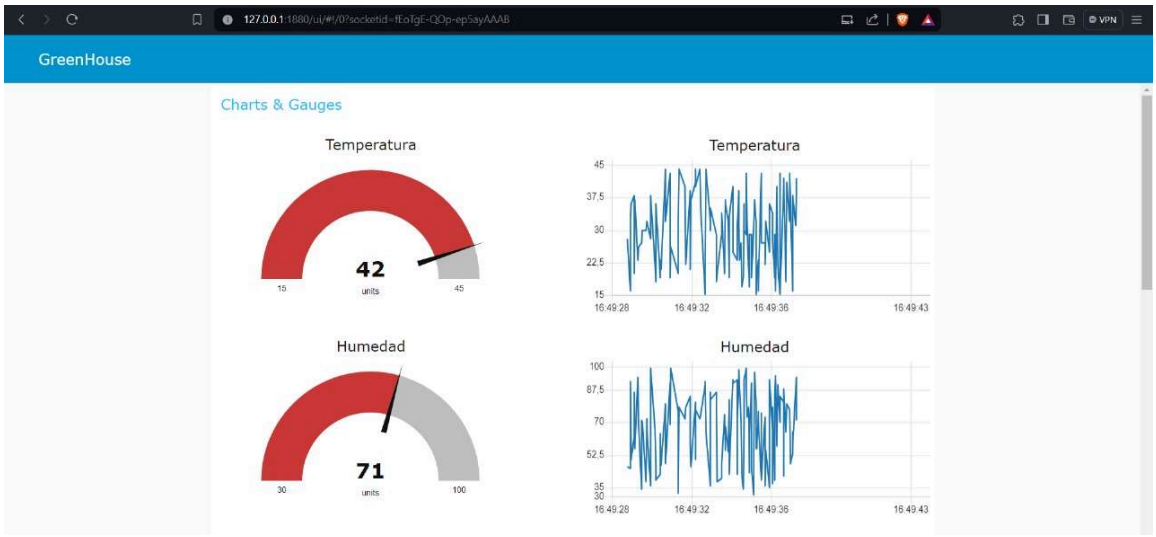
Node-red Green House website with all charts and gauges

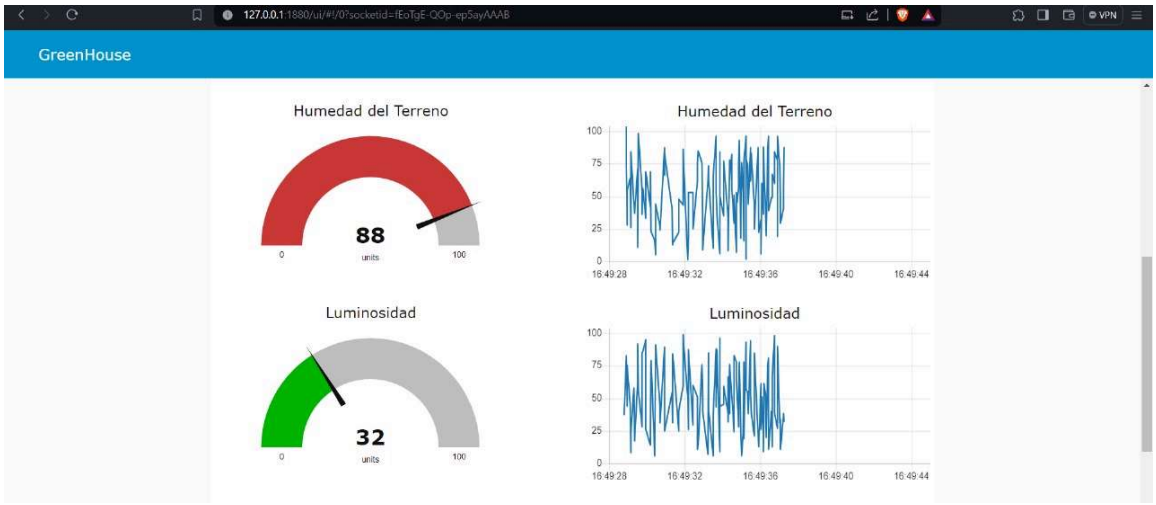
Node-red website config flows

JSON config flows filename = GreenHouse-flows.json



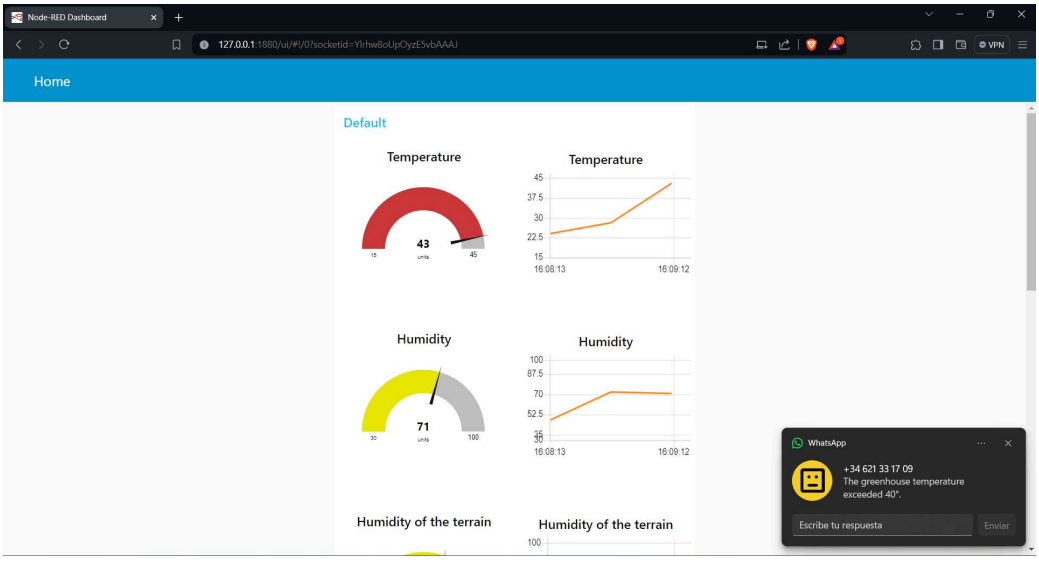
Node-red website views (GreenHouse)



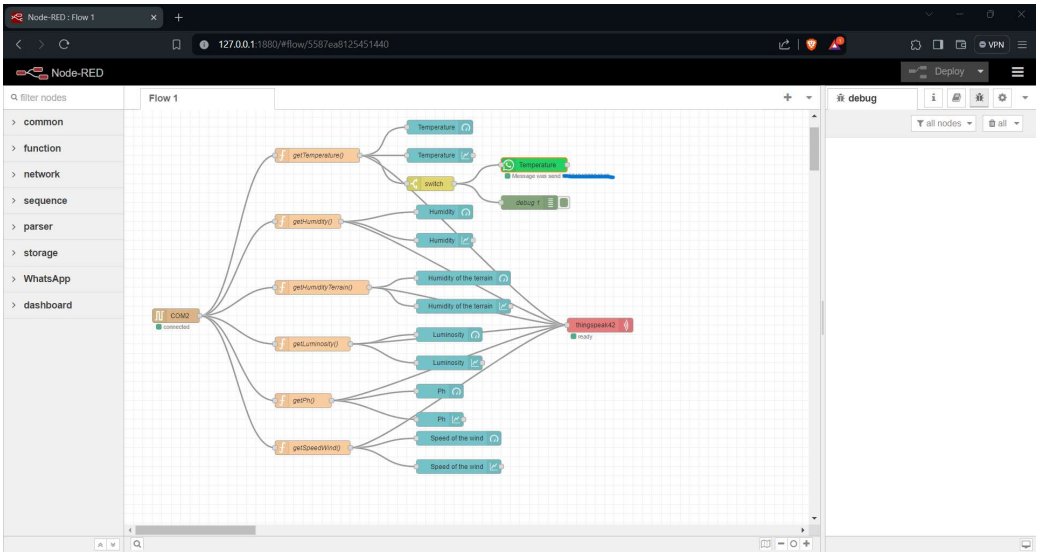


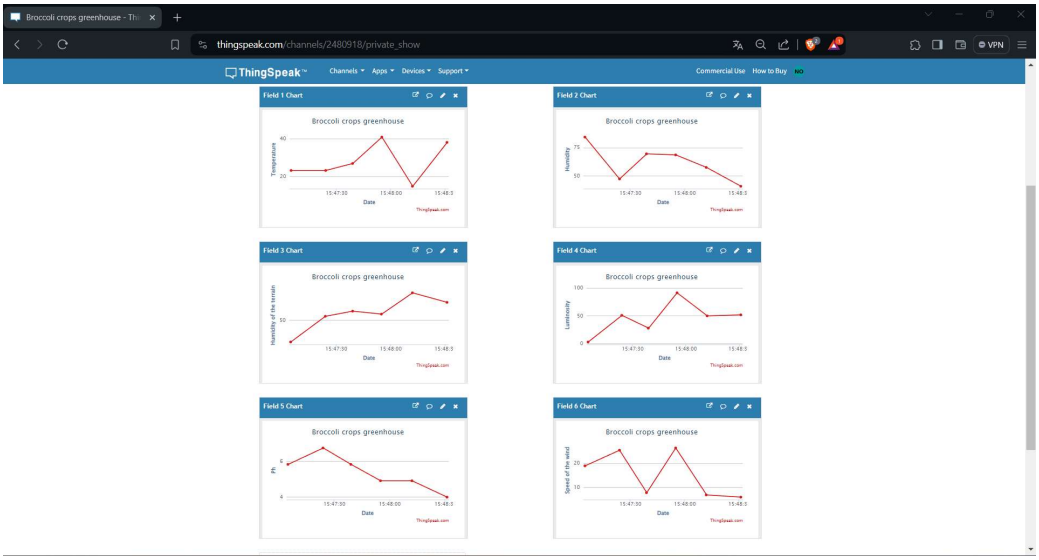
Use of a WhatsApp Chatbot to send high temperatures alerts

We manage to use a WhatsApp chatbot to send an alarm when the temperature exceed the 40°.



Use of the could of ThingSpeak


Node-RED : Flow 1
127.0.0.1:1880/#flow/5587ea8125451440
Node-RED
Deploy
filter nodes
common
function
network
sequence
parser
storage
WhatsApp
dashboard
Flow 1
debug
all nodes
all
COM2
connected
getTemperature()
getHumidity()
getHumidityTerrain()
getLuminosity()
getPh()
getSpeedWind()
Temperature
switch
Humidity
Humidity of the terrain
Luminosity
Ph
Speed of the wind
thingspeak42
ready


Broccoli crops greenhouse - Th
thingspeak.com/channels/2480918/private_show
ThingSpeak
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Field 1 Chart
Broccoli crops greenhouse
Field 2 Chart
Broccoli crops greenhouse
Field 3 Chart
Broccoli crops greenhouse
Field 4 Chart
Broccoli crops greenhouse
Field 5 Chart
Broccoli crops greenhouse
Field 6 Chart
Broccoli crops greenhouse