|  |  |
| --- | --- |
| **Y** | **Министерство науки и высшего образования Российской Федерации**  **Мытищинский филиал федерального государственного бюджетного**  **образовательного учреждения высшего образования**  **«Московский государственный технический университет имени**  **Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Космический

КАФЕДРА К3 «Прикладная математика, информатика и вычислительная техника»

**ОТЧЕТ ПО ПРОЕКТО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ**

Студент: Некиров Матвей Владимирович

Группа: К3-22Б

Название практики: Ознакомительная практика

Выполнил: студент группы К3-22Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Некиров М.В.

(Подпись, дата) (И.О.Фамилия)

Руководитель: кандидат тех. н-к **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Афанасьев А.В.

(Подпись, дата) (И.О.Фамилия)

*Мытищи 2023*

|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Мытищинский филиал федерального государственного бюджетного**  **образовательного учреждения высшего образования**  **«Московский государственный технический университет имени**  **Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Космический

КАФЕДРА К3 «Прикладная математика, информатика и вычислительная техника»

**ЗАДАНИЕ НА ПРАКТИКУ**

Студент группы К3-22Б: Некиров Матвей Владимирович

Направление подготовки: 01.03.02 «Прикладная математика и информатика»

Направленность подготовки: «Прикладная математика»

Название практики (по учебному плану): Учебная практика

Сроки прохождения практики: с «10» июля 2023 г. по «27» июля 2023 г.

Место прохождения практики: Кафедра ПМИВТ МФ МГТУ им. Н.Э. Баумана

Индивидуальное задание: получение и обработка данных от приборов от 02.02.22 по 30.07.22

Дата выдачи: «10» июля 2023 г.

Дата сдачи отчета: «27» июля 2023 г.

Выполнил: студент группы К3-22Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Некиров М.В.

(Подпись, дата) (И.О.Фамилия)

Руководитель: кандидат тех. н-к **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Афанасьев А.В.

(Подпись, дата) (И.О.Фамилия)

Содержание

[Основная часть 6](#_Toc141733485)

[Выбранные средства для разработки программы 6](#_Toc141733486)

[Алгоритм работы программы 6](#_Toc141733487)

[Используемые библиотеки 6](#_Toc141733488)

[Основные функции 6](#_Toc141733489)

[Виджеты 6](#_Toc141733490)

[Результат работы программы 8](#_Toc141733491)

[Интерфейс 10](#_Toc141733492)

[Заключение 11](#_Toc141733493)

[Приложение 12](#_Toc141733494)

Введение

Современный мир невозможно представить без использования различных приборов, с помощью которых происходит сбор и обработка данных. Однако, для того чтобы эти данные были полезными и применимыми в реальной жизни, необходимо проводить их анализ, обработку и наглядную визуализацию. В рамках проектно-технологической практики на кафедре ПМИВТ, целью является разработка программы с графическим интерфейсом, способной получать, обрабатывать и визуализировать исходные данные, полученные с приборов в формате json. Это позволит получить новые знания и опыт в области научных исследований и работы с данными. В данном проекте мы будем использовать современные среды разработки и методы анализа данных, что позволит выполнить установленное техническое задание. Результаты этой практики могут быть использованы для дальнейших научных исследований.

Данные получены с приборов кафедры К3 ПМИВТ:

РОСА К-2 (01) 448 аудитория

Тест Студии (schHome) г. Пушкино

Сервер СЕВ (01) информационный центр Сервер СЕВ (02) информационный центр Сервер СЕВ (03) информационный центр Тест воздуха (01)

Hydra-L (01) 447 аудитория

Hydra-L (02) 449 аудитория

Hydra-L (03) 446 аудитория

Hydra-L (04) 448 аудитория

Hydra-L (05) 451 аудитория

Hydra-L (06) 453 аудитория

Hydra-L (07) 445 аудитория

Hydra-L (08) 451 аудитория

Опорный барометр (01) 448 аудитория

Сервер dokuwiki (01) информационный центр Сервер dbrobo (01) информационный центр

Сервер webrobo (01) информационный центр Студия поля:

system RSSI – уровень сигнала Wi-Fi

TCS34725 – это датчик RGB (датчик цветности освещения). его поля – это результат предобработки:

redC , greenC , blueC – собственно значения RGB, но обработке не подлежат, поскольку необходимо знать внутренние настройки режима работы датчика, а их в данных нет;

lux – вычисленная освещённость в люксах;

colorTempCT – цветовая температура света в Кельвинах BH1750 – датчик-люксометр.

lux (в люксах)

blink (в %)

blinkmin и blinkmax – результаты промежуточных измерений, из которых вычисляется blink

AM2321 – датчик температуры и относительной влажности.

BME280 – датчик температуры, атмосферного давления и относительной влажности

* temp
* pressure
* humidity

BMP280 – датчик температуры и атмосферного давления

* temp
* pressure

DS18B20 – датчик температуры фирмы Dallas.

# Основная часть

## Выбранные средства для разработки программы

Программа разработана в среде Jupyter Notebook из-за возможности создавать блоки, быстро изменять и запускать их, что позволяет ускорить разработку и сделать процесс более наглядным.

## Алгоритм работы программы

1. Выгрузка данных с сервера через load.py
2. Преобразование во внутренний формат через fixup.py
3. Загрузка данных в notebook
4. Преобразование дат
5. Инициализация функций фильтрации, усреднения и т.п.
6. Вывод интерактивных клеток, в которых пользователей выбирает тип графика и данные

## Используемые библиотеки

Стандартные:

sys, collections, datetime, json, os, pathlib, bisect, typing

Нестандартные:

pandas, scipy, numpy, matplotlib, ipywidgets, requests, bs4

## Основные функции

date\_df\_correction

draw\_mean

min\_max\_candle

effective\_temp

convert\_labels

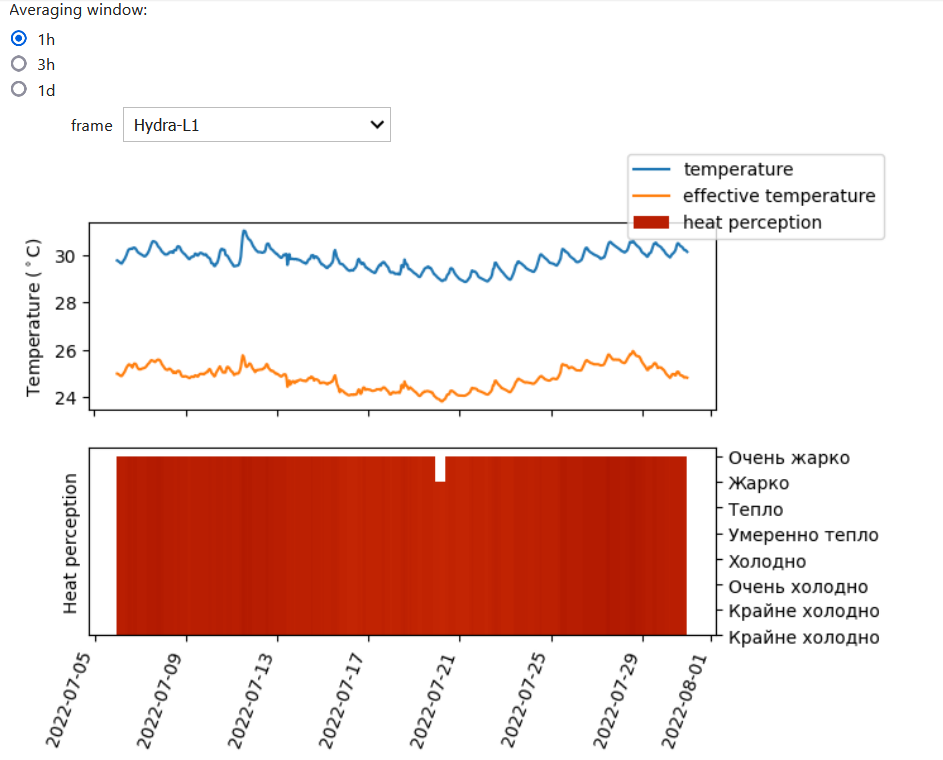
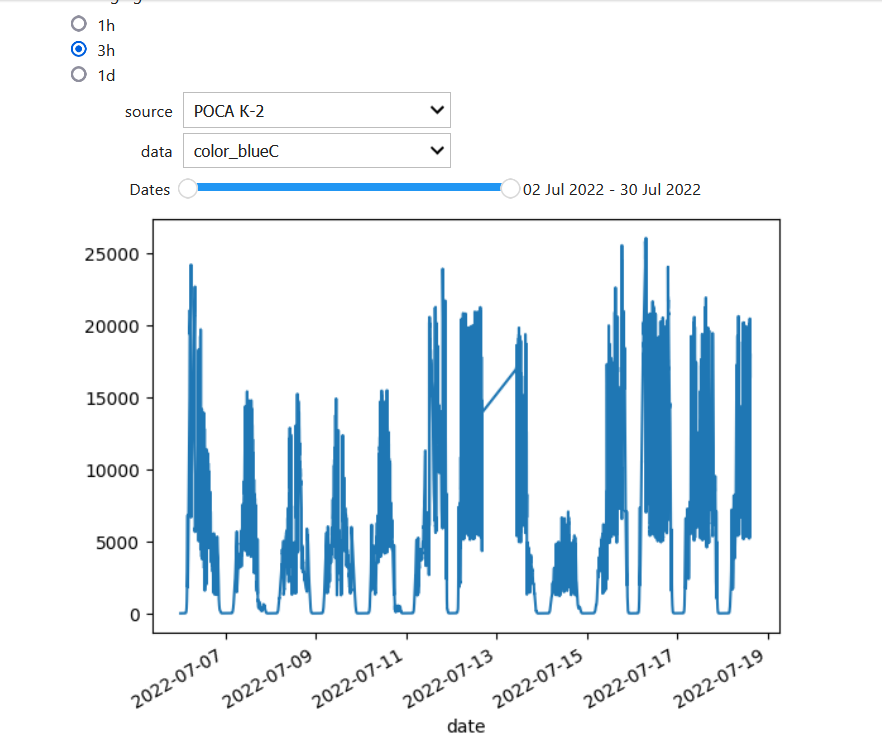
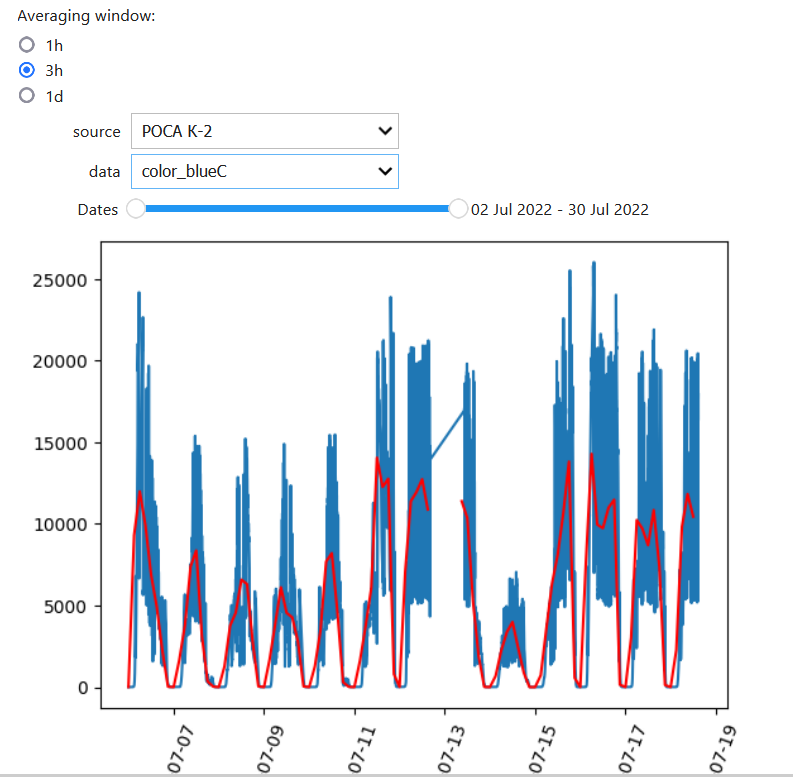
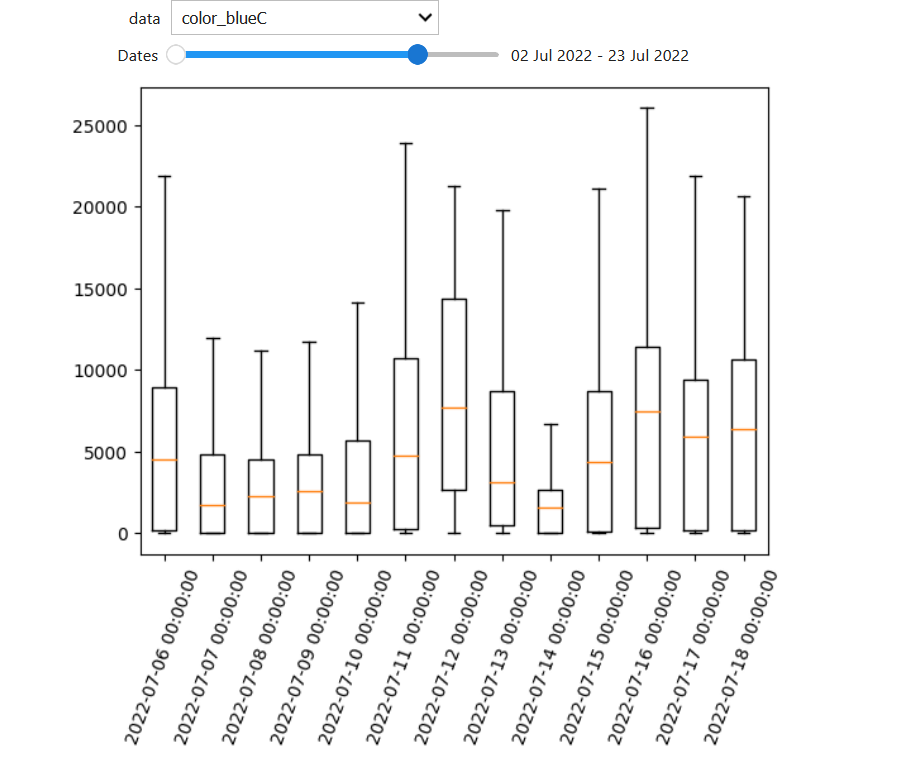
plot\_effective

update\_columns

printer

effective\_printer

# Результат работы программы



## Интерфейс

В первом интерактивном окне пользователю представлены 2 группы radio-button. Первые позволяют выбрать один из трёх видов графиков: как есть, усредненные данные, свечи. Под кнопками расположены выпадающие списки, позволяющие выбрать датчик и данные. Ниже расположен двойной слайдер, через который выбираются даты, участвующие в построении графика

Во втором окне сверху отображаются 3 кнопка – выбор окна усреднения: за час, за 3 и за день. Ниже расположен выпадающий список для выбора датчика. В самом низу отображается график температуры, влажности и тепловосприятия.

# Заключение

В ходе проектно-технологической практики были обработаны и визуализированы данные в среде Jupyter Notebook с использованием matplotlib и ipywidgets. Разработанная программа полностью соответствует поставленному техническому заданию. Практика позволила нам получить новые знания и опыт в области обработки и визуализации данных, а также применить полученные навыки на практике.

# Приложение

%matplotlib inline

from datetime import datetime

import numpy as np

import scipy.stats as stats

import matplotlib.pyplot as plt

import pandas as pd

import ipywidgets as widgets

from bisect import bisect

from matplotlib.colors import Normalize

from pathlib import Path

import os  
def load\_dataset(path: Path) -> pd.DataFrame:

df = pd.read\_json(path)

df.date = pd.to\_datetime(df.date, format="%Y-%m-%d %H:%M:%S")

df.set\_index('date', inplace=True)

return df

ROOT = Path(os.getcwd()).resolve()

dataframes = {}

for file in os.listdir('datasets'):

path = ROOT / 'datasets' / file

print(file, path)

dataframes[path.stem] = load\_dataset(path)

print('OK')

def zscore\_cleaning(data) -> pd.DataFrame:

z = np.abs(stats.zscore(data))

data = data[(z < 3).all(axis=1)]

return data

def quantile\_cleaning(data) -> pd.DataFrame:

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = data.apply(stats.iqr)

data = data[~((data < (Q1 - 1.5 \* IQR)) | (data > (Q3 + 1.5 \* IQR))).any(axis=1)]

return data

def draw\_mean(data: pd.DataFrame, window: int | str = '1h') -> None:

data = data.dropna()

res = data.resample(window).mean()

plt.plot(data)

plt.plot(res, color='red')

plt.xticks(rotation=70)

plt.plot()

plt.show()

def min\_max\_candle(data: pd.DataFrame) -> None:

gb = data.dropna().groupby(pd.Grouper(freq='d'))

Y = [np.array(gb.get\_group(x)) for x in gb.groups]

X = [x for x in gb.groups]

plt.boxplot(Y, labels=X, showfliers=False)

\_ = plt.xticks(rotation=70)

def effective\_temp(t: np.array, h: np.array) -> np.array:

return t - 0.4 \* (t - 10) \* (1 - h / 100)

HEAT\_PERCEPTION = {-30: 'Крайне холодно', -24: 'Крайне холодно', -12: 'Очень холодно', 0: 'Холодно',

12: 'Умеренно тепло', 18: 'Тепло', 24: 'Жарко', 30: 'Очень жарко'}

\_HP\_VALUES = list(HEAT\_PERCEPTION)

\_HP\_LABELS = list(HEAT\_PERCEPTION.values())

def convert\_labels(x: int) -> int:

return bisect(\_HP\_VALUES, x)

def plot\_effective(t: np.ndarray, h: np.ndarray) -> None:

fig, (ax1, ax2) = plt.subplots(2, 1)

fig.autofmt\_xdate(rotation=70)

eff = effective\_temp(t, h)

ax1.plot(t, label='temperature')

ax1.plot(eff, label='effective temperature')

ax1.set\_ylabel(r'Temperature ($^\circ$C)')

x = [convert\_labels(x) for x in eff]

colormap = plt.get\_cmap("turbo")

rescale = Normalize(np.min(\_HP\_VALUES), np.max(\_HP\_VALUES))

ax2.bar(eff.index, x, width=25 / len(x), label='heat perception', color=colormap(rescale(eff)))

ax2.yaxis.tick\_right()

ax2.set\_ylabel('Heat perception')

ax2.set\_yticks(ticks=np.arange(len(\_HP\_VALUES)), labels=\_HP\_LABELS)

fig.legend()

Интерактивная клетка: выбор приборов, данных, типа графика, временной промежуток наблюдения, усреднение за промежуток  
meanButtons = widgets.RadioButtons(options=['1h', '3h', '1d'], description='Averaging window:')

action = widgets.RadioButtons(options=['Real Data', 'Average', 'Min/Max'], description='Action:')

dfDropdown = widgets.Dropdown(options=list(dataframes), description='source')

columnDropdown = widgets.Dropdown(options=dataframes[list(dataframes)[0]], description='data')

start\_date, end\_date = datetime(2022, 7, 2), datetime(2022, 7, 30)

dates = pd.date\_range(start\_date, end\_date, freq='D')

options = [(date.strftime(' %d %b %Y '), date) for date in dates]

selection\_range\_slider = widgets.SelectionRangeSlider(

options=options,

value=(start\_date, end\_date),

index=(0, len(options) - 1),

description='Dates',

orientation='horizontal',

layout={'width': '500px'}

)

def update\_columns(change):

owner = change['owner'].value

l = dataframes[owner].columns.tolist()

columnDropdown.options = l

columnDropdown.value = l[0]

dfDropdown.observe(update\_columns)

def printer(action, meanButtons, df, clmn, selection\_range\_slider):

s, e = selection\_range\_slider

df = dataframes[df][clmn]

mask = (df.index > s) & (df.index <= e)

match action:

case 'Real Data':

df.loc[mask].plot()

case 'Average':

draw\_mean(df.loc[mask], meanButtons)

case 'Min/Max':

min\_max\_candle(df.loc[mask])

widgets.interact(printer, meanButtons=meanButtons, selection\_range\_slider=selection\_range\_slider, action=action,

df=dfDropdown, clmn=columnDropdown);

График эффективной температуры и теплоощущения

effective\_df = {'Hydra-L': dataframes['Hydra-L'], 'Hydra-L1': dataframes['Hydra-L1']}

effectiveTempDropdown = widgets.Dropdown(options=list(effective\_df))

def effective\_printer(meanButtons, frame):

df = effective\_df[frame]

df = df.resample(meanButtons).mean()

plot\_effective(df.BME280\_temp, df.BME280\_humidity)

widgets.interact(effective\_printer, meanButtons=meanButtons, frame=effectiveTempDropdown);