**Московский авиационный институт**

**(национальный исследовательский университет) «МАИ»**

**Институт №3** **—** «Системы управления, информатика и электроэнергетика»

**Кафедра 307 —** «Цифровые технологии и информационные системы»

Курсовая работа

по дисциплине: **«Визуализация процессов в IoT»**

**Выполнил:**

студент группы М3О-118М-20

Федоров Илья

**Принял:**

Максимов Алексей Николаевич

Москва, 2021

**Оглавление**

Оглавление

[Постановка задачи: 3](#_Toc73319944)

[Используемые инструменты: 3](#_Toc73319945)

[Установка необходимых компонентов: 4](#_Toc73319946)

[Конфигурационные файлы 5](#_Toc73319947)

[Конечный конфигурационный файл (*docker-compose.yml*): 5](#_Toc73319948)

[Конфигурационный файл *telegraf.conf:* 6](#_Toc73319949)

[Создание топиков, запись данных с использованием API для Python (Paho Mqtt). 8](#_Toc73319950)

[Просмотр значений в базе данных: 12](#_Toc73319951)

[Визуализация: 12](#_Toc73319952)

# **Постановка задачи:**

7 Вариант:

написать программу, моделирующую измерения на датчиках температуры в Москве в течении дня и визуализировать их в виде графиков. Задание выполнить с использованием Docker контейнеров.

# **Используемые инструменты:**

Eclipse Mosquitto. mqtt брокер. Версия 1.5

Telegraf, подписывающийся на mqtt топики и передающий данные в InfluxDB формате. Версия latest на май 2021.

InfluxDB – TSDB база данных версия 1.8

Grafana – для визуализации данных. Версия latest на май 2021.

Работа выполнялась на ОС Ubuntu 20.04

# **Установка необходимых компонентов:**

Docker:

* sudo apt update

Затем установите несколько необходимых пакетов, которые позволяют apt использовать пакеты через HTTPS:

* sudo apt install apt-transport-https ca-certificates curl software-properties-common

Добавьте ключ GPG для официального репозитория Docker в вашу систему:

* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Добавьте репозиторий Docker в источники APT:

* sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

Потом обновите базу данных пакетов и добавьте в нее пакеты Docker из недавно добавленного репозитория:

* sudo apt update
* sudo apt install docker-ce

Docker должен быть установлен, демон-процесс запущен, а для процесса активирован запуск при загрузке:

* sudo systemctl status docker

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Установка с использованием Docker-compose. Данный функционал позволяет одним конфигурационным файлом подготовить несколько docker контейнеров для использования их в связке.

# **Конфигурационные файлы**

## **Конечный конфигурационный файл (*docker-compose.yml*):**

version: '3.1'

services:

influxdb:

image: influxdb:1.8

container\_name: course\_influxdb

ports:

- "8083:8083"

- "8086:8086"

- "8090:8090"

volumes:

- influxdb-storage:/var/lib/influxdb

telegraf:

image: telegraf:latest

container\_name: course\_telegraf

links:

- influxdb

volumes:

- ./telegraf.conf:/etc/telegraf/telegraf.conf:ro

grafana:

image: grafana/grafana:latest

container\_name: course\_grafana

ports:

- "3000:3000"

links:

- influxdb

depends\_on:

- influxdb

volumes:

- grafana-storage:/var/lib/grafana

mosquitto:

image: eclipse-mosquitto:1.5

hostname: mosquitto

container\_name: course\_mosquitto

ports:

- "1883:1883"

volumes:

influxdb-storage:

grafana-storage:

**Раздел services:**

На примере influxdb

Image – название образа (по названию ищется образ в репозитории docker)

container\_name – название контейнера (на выбор)

ports – проброс портов с контейнера на локальную ОС

volumes – монтирование хранилища данных на локальной ОС. Необходимо для сохранения данных контейнера после его закрытия.

## **Конфигурационный файл *telegraf.conf:***

[[inputs.mqtt\_consumer]]

## MQTT broker URLs to be used. The format should be scheme://host:port,

## schema can be tcp, ssl, or ws.

servers = ["tcp://0.0.0.0:1883"]

## Topics that will be subscribed to.

topics = [

"sensors/temp",

"sensors/real\_feel"

"sensors/wing",

]

data\_format = "influx"

[[outputs.influxdb]]

## Multiple URLs can be specified for a single cluster, only ONE of the

## urls will be written to each interval.

urls = ["http://localhost:8086"]

## The target database for metrics; will be created as needed.

## For UDP url endpoint database needs to be configured on server side.

database = "telegraf"

## If true, no CREATE DATABASE queries will be sent. Set to true when using

## Telegraf with a user without permissions to create databases or when the

## database already exists.

skip\_database\_creation = true

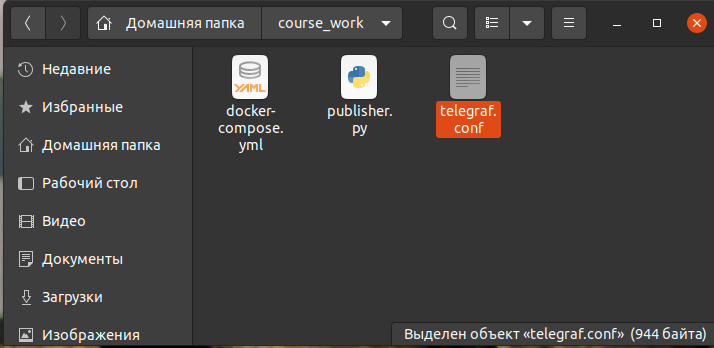
## HTTP Basic Auth

username = "admin"

password = "admin"

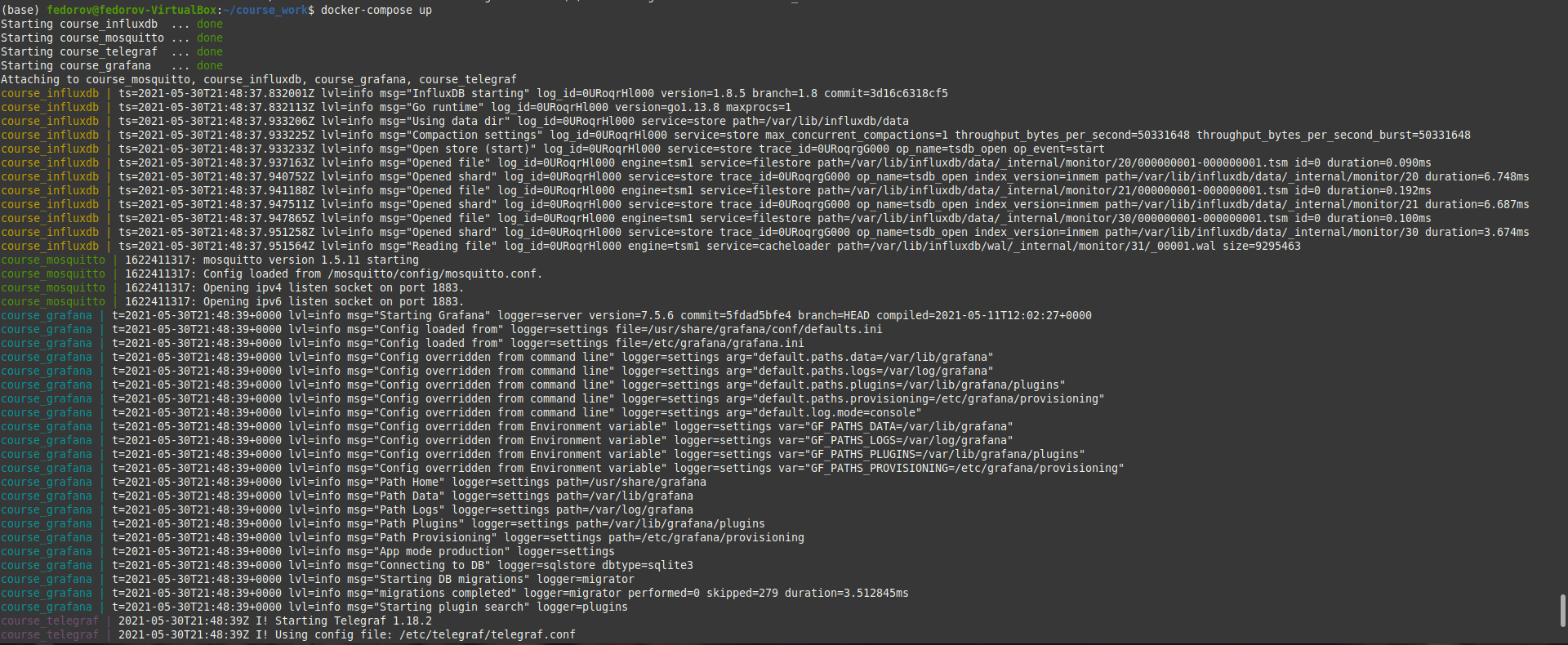
[[inputs.mqtt\_consumer]] – подключаемый модуль telegraf. Собирает данные с указанных в topics = [ ] топиках. Подключается к порту 1883. Данные должны записываться в формате "influx" в [[outputs.influxdb]] модуль в БД "telegraf" по порту 8086.

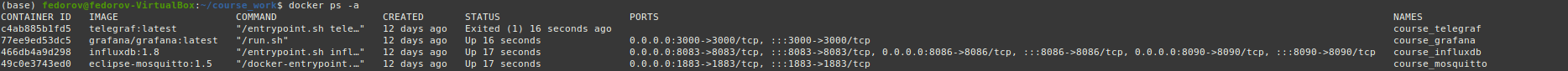
Итоговая структура проекта:



Запуск docker-compose проекта:

* Открыть в папке проекта консоль
* Запустить сборку контейнеров командой *docker-compose up*



Команда docker ps -a проверяет список запущенных и созданных контейнеров:

# **Создание топиков, запись данных с использованием API для Python (Paho Mqtt).**

pip install paho-mqtt

import paho.mqtt.client as mqttClient

import time

import random

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Connected to broker")

global Connected # Use global variable

Connected = True # Signal connectio

else:

print("Connection failed")

Connected = False # global variable for the state of the connection

broker\_address = "0.0.0.0"

port = 1883

# password = "1"

client = mqttClient.Client() # create new instance

client.on\_connect = on\_connect # attach function to callback

client.connect(broker\_address, port=port) # connect to broker

client.loop\_start() # start the loop

while Connected != True: # Wait for connection

time.sleep(0.1)

# температура на датчиках по москве.

# подразумевается 24 измерения на 1 день. программа будет моделировать до нажатия ctrl+c

# in\_sun

# in\_shadow

# real\_feel

try:

time\_now = 0

day = 1

month = 1

year = 2021

delta = 1

value = -5

real\_feel = value

wing = False

flag = False

while True:

if (3< month < 6) or (9< month < 12):

delta = 2

else: delta = 1

if time\_now == 24:

time\_now = 0

day += 1

if day == 30:

day = 1

month += 1

if 3 <= month < 8:

value += random.uniform(3, delta\*5)

flag = True

else:

value -= random.uniform(3, delta\*5)

if month == 12:

month = 1

year += 1

wing = random.choice([0, 1])

if 7 < time\_now < 20: # температура обычно колеблется днем, ночью чаще всего она снижается

value = value + random.uniform(0, delta/2)

if value >= 50:

value = 50

else:

value = value - random.uniform(0, delta/2)

if value <= -50:

value = -50

time\_now += 1

if wing == 1:

real\_feel = value - delta

temp = "temp\_measurement,type=temperature value={0}".format(round(value, 2))

print(temp)

client.publish("sensors/temp", temp)

temp = "realfeel\_measurement,type=real\_feel value={0}".format(round(real\_feel, 2))

print(temp)

client.publish("sensors/real\_feel", temp)

temp = "wing\_measurement,type=was\_wing value={0}".format(wing)

print(temp)

client.publish("sensors/wing", temp)

print(time\_now," часа, ",day," день",month," месяц")

time.sleep(0.05)

except KeyboardInterrupt:

client.disconnect()

client.loop\_stop()

Программа моделирует погоду в Москве, с учетом времени года (месяца), времени суток и наличия ветра.

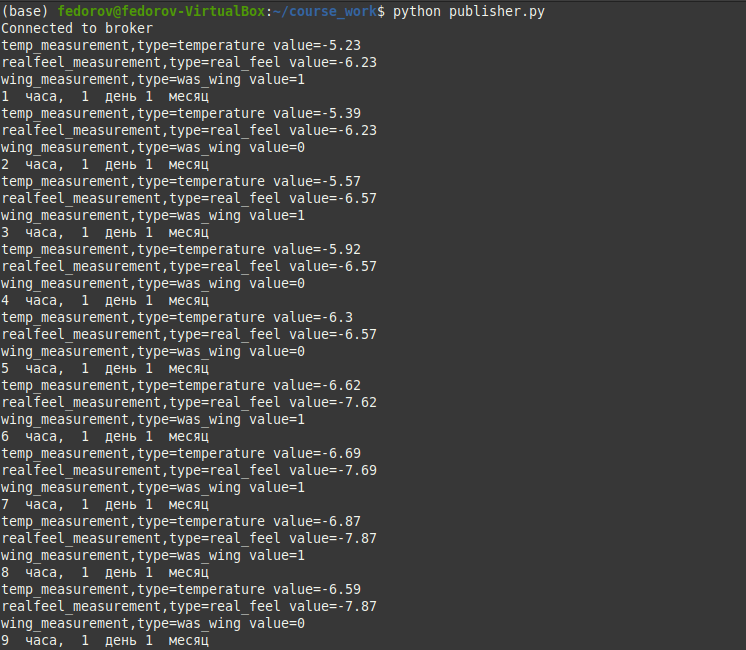
Данные записываются в следующие топики (порт 1883 локальной машины):

* "sensors/wing" – наличие ветра
* "sensors/real\_feel" – как «реально ощущается» (зависит от наличия ветра)
* "sensors/temp" – реальное значение температуры

Программа запускается в консоли (в директории проекта) командой:

*python ./publisher.py*

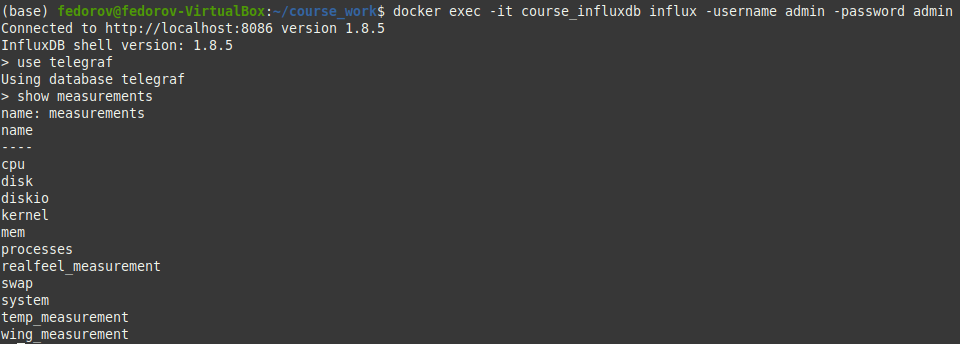
Контейнеры должны быть запущены, необходимые порты открыты.



Подразумевается, что данные с датчика температуры берется каждый час и пишутся в топик. Функцией *time.sleep()* можно регулировать скорость модуляции (в секундах). Так, при значении 0.05 за одну секунду будет моделироваться 20 часов измерений. За 10 минут – 500 дней, что будет отображено в визуализации.

# **Просмотр значений в базе данных:**

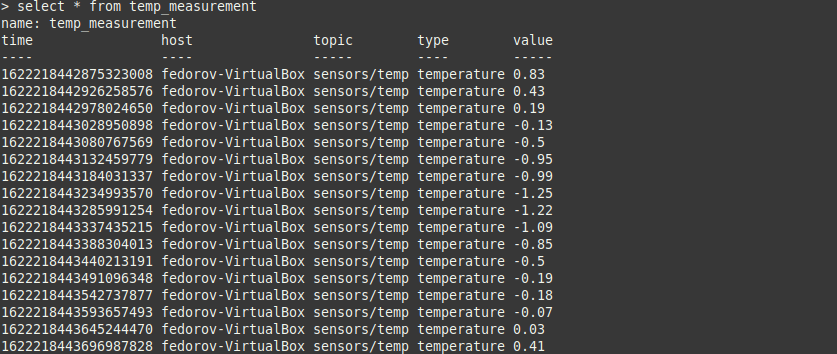
*docker exec -it course\_influxdb influx -username admin -password admin*



После выбора данных и вывода списка всех измерений видно, что создались те измерения, которые мы указали в переменной *temp* (имеющей формат “influx”).

Выведем несколько записанных значений :

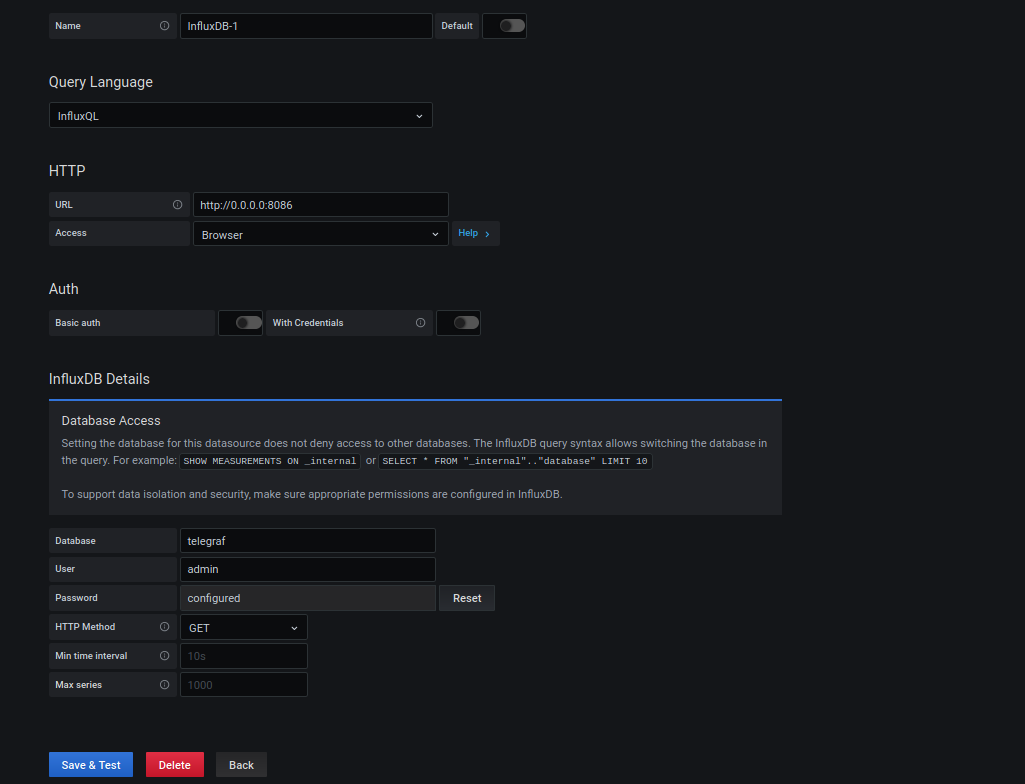
*SELECT \* FROM temp\_measurement*



# **Визуализация:**

Необходимо открыть браузер

* перейти по адресу [*http://localhost:3000/*](http://localhost:3000/)
* Авторизация: логин: *admin* пароль: *admin*
* Настроить источник данных

**

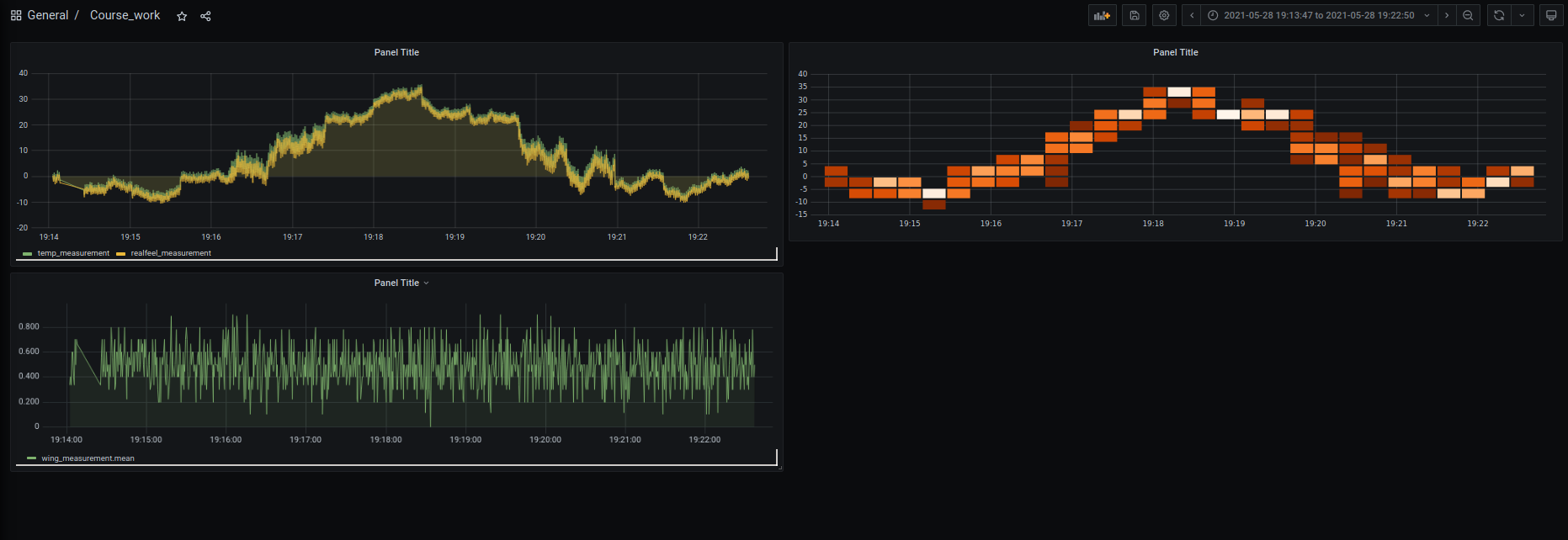
Затем необходимо создать новый dashboard:



Создано было 3 панели, показывающие следующее:

* Температура измеренная и реально ощущаемая на одной плоскости
* График ветрености (был или нет)
* Тепловая карта (самые частые значения температуры белые)

Итоговый dashboard выглядит следующим образом:



Температура смоделирована за 1 год, начиная с 1 января.