UDC 004.62 https://doi.org/10.35546/kntu2078-4481.2023.2.26

L. M. OLESHCHENKO
Candidate of Technical Sciences,
Associate Professor at the Department of Computer Systems Software
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0001-9908-7422

M. O. ILIN
Postgraduate Student at the Department of Computer Systems Software
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0009-0001-0803-3726

SOFTWARE ANALYSIS OF RADIATION AIR POLLUTION STREAMING DATA

In today's world, attention to environmental protection and sustainable development is increasing. Natural resources are limited, so it is important to take measures to control and preserve the ecological state of the environment. One of the effective tools for this is environmental monitoring systems that are able to collect and process streaming data. Streams come in real time from various sensors measuring environmental parameters such as air quality, noise level, water level, pollutant emissions and many others. Analysis of data from systems for monitoring environmental indicators is an important tool for monitoring and improving the state of the environment, allows for more effective management of natural resources, reduces the negative impact of human activity on the environment and promotes sustainable development. One of the main advantages of streaming data processing is the ability to analyze and respond to changes in real time. This allows to promptly detect dangers, deviations from standards, emergency situations or other problems arising in the environment and immediately take measures to eliminate them. The processing of streaming data allows analyzing trends and predicting possible risks and problems in the environment. This allows to take precautionary measures and develop strategies to preserve the ecological state. Data obtained from monitoring systems can serve as a basis for the development of policies and regulatory standards in the field of environmental protection.

The article describes software methods for analyzing streaming data, in particular, for analyzing radiation air pollution. Open data for indicators of radiation air pollution in the Kyiv city were analyzed and data visualization was carried out using analytical tools of the Python programming language. A correlation was found between radiation indicators at different times of the day, the time periods of emergency shutdowns of the sensor were analyzed. The Google Colab cloud environment was used for data analysis. The technical characteristics of the monitoring system of radiation air pollution indicators in the Kyiv city from open data sources of the ThingSpeak platform are described.

Key words: *software methods, streaming data, Python, air pollution monitoring, radioactive particle sensor, ThingSpeak, Internet of Things, ESP12.OLED.*

Л. М. ОЛЕЩЕНКО
кандидат технічних наук,
доцент кафедри програмного забезпечення комп'ютерних систем
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0000-0001-9908-7422

М. О. ІЛЬЇН
аспірант кафедри програмного забезпечення комп'ютерних систем
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0009-0001-0803-3726

ПРОГРАМНИЙ АНАЛІЗ ПОТОКОВИХ ДАНИХ РАДІАЦІЙНОГО ЗАБРУДНЕННЯ ПОВІТРЯ

У сучасному світі збільшується увага до питань охорони навколишнього середовища та сталого розвитку. Природні ресурси обмежені, тому важливо вживати заходів для контролю і збереження стану довкілля. Одним із ефективних інструментів для цього є програмні системи моніторингу показників навколишнього середовища, які здатні збирати та обробляти потокові дані. Потокові дані надходять у режимі реального часу з різних датчиків, що вимірюють параметри середовища, такі як якість повітря, рівень шуму, рівень *води, викиди забруднюючих речовин і багато інших. Аналіз даних систем моніторингу показників повітря є важливим інструментом для контролю та покращення стану навколишнього середовища, дозволяє забезпечити більш ефективне управління природними ресурсами, зменшити негативний вплив людської діяльності на довкілля та сприяти сталому розвитку. Однією з основних переваг обробки потокових даних є здатність до аналізу і реагування на зміни в реальному часі. Це дозволяє оперативно виявляти небезпеки, відхилення від стандартів, аварійні ситуації або інші проблеми, що виникають у навколишньому середовищі та негайно приймати заходи для їх усунення. Обробка потокових даних дозволяє аналізувати тенденції та прогнозувати можливі ризики та проблеми в навколишньому середовищі. Це дозволяє приймати запобіжні заходи та розробляти стратегії для збереження екологічного стану. Дані, отримані з моніторингових систем, можуть слугувати підґрунтям для розробки політик та регулюючих стандартів у сфері охорони навколишнього середовища.*

У статті описано програмні методи аналізу потокових даних, зокрема, для аналізу радіаційного забруднення повітря. Проаналізовано відкриті дані показників радіаційного забруднення повітря у м. Києві та здійснено візуалізацію даних за допомогою аналітичних інструментів мови програмування Python. Знайдено кореляцію між показниками радіаційного випромінювання в різний час доби, проаналізовано періоди часу аварійних відключень датчика. Для аналізу даних використано хмарне середовище Google Colab. Описано технічні характеристики моніторингової системи показників радіаційного забрудненння повітря у м. Київ з відкритих джерел даних платформи ThingSpeak.

Ключові слова: *програмні методи, потокові дані, Python, моніторинг стану радіаційного забруднення повітря, датчик радіоактивних частинок, ThingSpeak, Інтернет речей, ESP12.OLED.*

Problem statement

The importance of monitoring air quality from sensors is that air, as one of the main components of the environment, has a direct impact on human health, the ecosystem and the overall quality of life. Because air is a mixture of gases, aerosols, and other harmful substances, its quality can vary greatly depending on various factors, including industry, transportation, energy, and other sources of pollution. The air quality monitoring is necessary for protection of public health. Poor air quality can lead to the development of various diseases of the respiratory system, cardiovascular diseases and other health problems. Monitoring allows to detect high levels of harmful substances, such as soot, nitrogen oxides, hydrogen sulfide, and others, and take the necessary measures to reduce their concentration and protect public health. Air quality monitoring helps assess the environmental impact of various sources of pollution such as industrial plants, transportation systems, energy, etc. This provides a basis for making decisions to improve efficiency, eradicate hazardous sources of pollution and promote sustainable development.

Related research

Researchers are constantly working on developing and improving sensor technologies used in air quality monitoring [1]. Studies focus on enhancing the accuracy, precision, and reliability of sensors to ensure the collection of high-quality data. This involves advancements in sensor calibration, data validation techniques, and the evaluation of sensor performance in different environmental conditions [2]. Researchers use monitoring data to create detailed air pollution maps and models. These maps provide spatial representations of air quality, identifying hotspots of pollution and understanding pollutant dispersion patterns [3]. The research involves the integration of monitoring data with geographical information systems (GIS) and the development of predictive models to estimate pollution levels in areas without monitoring stations [4].

Numerous studies investigate the health effects of air pollution using data from monitoring systems [5]. Researchers analyze the relationship between pollutant concentrations and adverse health outcomes such as respiratory diseases, cardiovascular issues, and even mortality rates. These studies provide valuable insights into the impacts of air pollution on public health and support the development of policies and regulations to mitigate these effects [6].

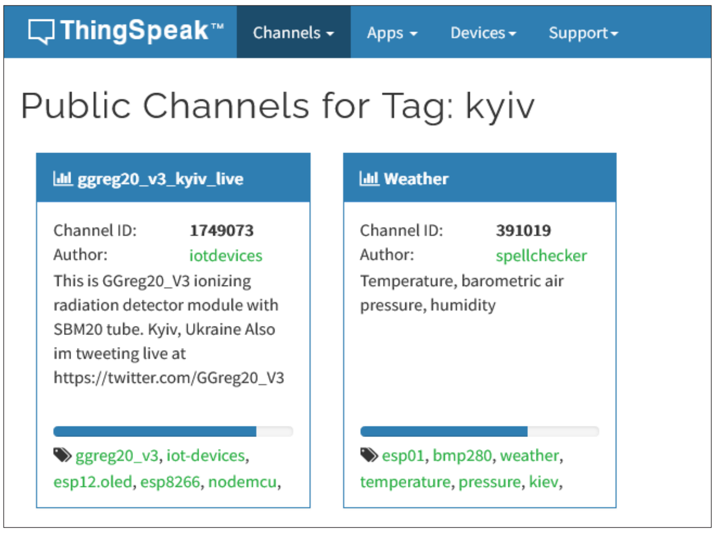
Monitoring systems help in identifying pollution sources and assessing their contributions to air pollution. Researchers employ advanced data analysis techniques, such as source apportionment models, to identify pollution sources such as industries, transportation, and biomass burning. This knowledge aids in designing targeted emission control strategies and implementing effective pollution reduction measures [7].

Citizen science initiatives involve the active participation of communities in monitoring air quality. Researchers explore the role of citizen scientists in data collection, validation, and interpretation. They study the effectiveness of community-based monitoring networks and assess the impact of community engagement on raising awareness, influencing policy decisions, and promoting behavioral changes to improve air quality [8].

The main goal of the article is streaming data software analysis of the air radiation indicators from open-source monitoring system.

Existing software solutions for sensor data analysis

ThingSpeak is an open-source Internet of Things (IoT) platform and cloud service that allows users to collect, analyze, and visualize sensor data from connected devices. It provides a simple and easy-to-use interface for managing IoT data and creating applications based on that data (Fig. 1).


ThingSpeak™ Channels Apps Devices Support

Public Channels for Tag: kyiv

ggreg20_v3_kyiv_live

Channel ID: 1749073
Author: iotdevices
This is GGreg20_V3 ionizing radiation detector module with SBM20 tube. Kyiv, Ukraine Also im tweeting live at https://twitter.com/GGreg20_V3

ggreg20_v3, iot-devices, esp12.oled, esp8266, nodemcu,

Weather

Channel ID: 391019
Author: spellchecker
Temperature, barometric air pressure, humidity

esp01, bmp280, weather, temperature, pressure, kiev,

Fig. 1. ThingSpeak IoT platform and open data channels for research

The IoT device in Fig. 1 is a GGreg20_V3 radioactive particle detector with an SBM-20 Geiger tube. Description of the data source: the GGreg20_V3 radioactive particle detector is an electronic sensor module for building a personal Geiger counter and determining the level of ionizing radiation. This IoT data source on thingspeak.com provides the following metrics measured by its sensor in real time: ggreg20_v3_cpm, ggreg20_v3_usvperh, and ggreg20_v3_dose_usv (Fig. 2).

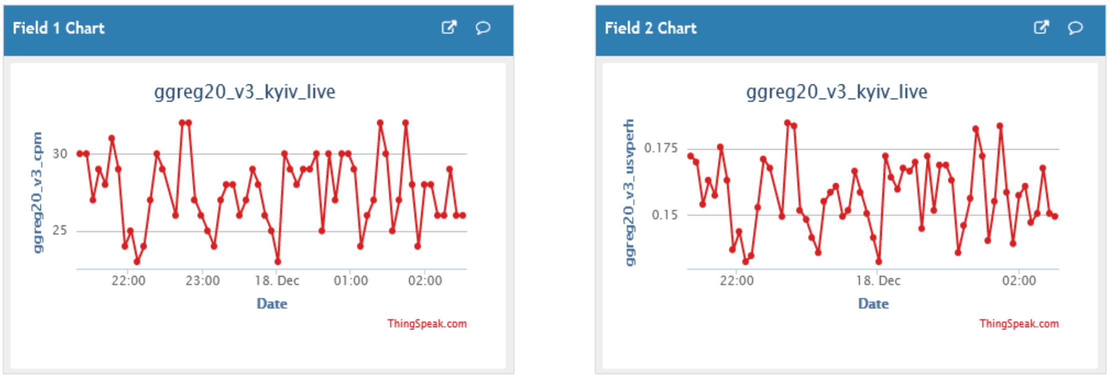


Fig. 2. Graphs of GGreg20_V sensor indicator values

ThingSpeak enables users to collect data from various sources, such as sensors, devices, and IoT platforms. It supports a wide range of communication protocols, including HTTP, MQTT, and TCP/IP, making it compatible with a variety of devices and technologies. The core concept in ThingSpeak is a «channel.» A channel represents a stream of data from a specific source or device. Users can create multiple channels to organize and manage their data effectively.

ThingSpeak provides cloud-based storage for collected sensor data. It offers a time-series database that can handle large volumes of data and supports efficient retrieval and analysis of historical data. ThingSpeak offers built-in analytics capabilities, allowing users to perform basic mathematical operations, filtering, and visualization on the collected data. It provides functions to calculate averages, maximums, minimums, and other statistical measures.

The platform also includes built-in charting tools for creating real-time graphs and visualizing data trends. ThingSpeak allows users to define triggers and reactions based on specified conditions. Triggers can be set to activate when certain data thresholds are reached, and reactions can be defined to perform actions such as sending notifications or triggering external processes. ThingSpeak integrates with various IoT platforms, services, and hardware devices. It supports integration with MATLAB for advanced data analysis and modeling. Additionally, it provides APIs and webhooks for easy integration with other applications and services. ThingSpeak is open-source, which means that its source code is freely available for modification and customization. It has an active user community that shares code, examples, and offers support to users.

There are several existing software solutions for sensor data analysis that can be considered as analogues to Thingspeak. InfluxDB is a time-series database specifically designed for handling and analyzing sensor data. It provides high-performance storage and retrieval of time-stamped data, along with built-in querying capabilities. InfluxDB also offers integration with visualization tools like Grafana for data visualization and monitoring. Grafana is an open-source analytics and visualization platform that can be used with various data sources, including sensor data stored in databases like InfluxDB. It provides flexible and interactive dashboards to visualize sensor data in real-time, perform data exploration, and create custom analytics and alerts. AWS IoT Analytics is a managed service provided by Amazon Web Services (AWS) that enables the collection, processing, and analysis of sensor data at scale. It offers features such as data ingestion, data transformation, data storage, and advanced analytics capabilities to derive insights from sensor data. AWS IoT Analytics integrates with other AWS services for seamless data integration and further analysis. Microsoft Azure IoT Hub is a cloud-based platform that allows for the ingestion and management of sensor data from connected devices. It provides capabilities for data processing, storage, and analysis, as well as integration with other Azure services such as Azure Stream Analytics and Azure Machine Learning for advanced analytics and predictive modeling.

Research results

We use of Python programming language technologies to analyze streaming data from online platforms to which data from sensors are sent. First, it was necessary to choose the nearest source of IoT data, in order to subsequently analyze the data from the selected source using the Python language. For this, the site https://thingspeak.com/channels/public was used, where we can sort public IoT data sources by tag. Since the place of residence is Kyiv city, the IoT data sources were filtered according to the "kyiv" tag. Only two IoT data sources were found for the "kyiv" tag, namely: a data source named "ggreg20_v3_kyiv_live" with a unique ID of 1749073, and a data source named "Weather" with a unique ID of 391019 (Fig. 1). Since the data source named "ggreg20_v3_kyiv_live" with the unique identifier 1749073 contained detailed information about this IoT device and its structure in the description, it was decided to choose it for further research. Because another data source named "Weather" with a unique identifier of 391019 did not provide a description of IoT device or a description of the structure of the sensor on the ThingSpeak platform.

So, we describe the selected IoT data source with the name "ggreg20_v3_kyiv_live". The IoT data source named "ggreg20_v3_kyiv_live" with unique ID 1749073 uses the GGreg20_V3 radioactive particle detector with SBM-20 Geiger tube as an IoT device. Description of the data source: the GGreg20_V3 radioactive particle detector is an electronic sensor module for building a personal Geiger counter and determining the level of ionizing radiation. This IoT data source on thingspeak.com provides the following metrics measured by its sensor in real time: ggreg20_v3_cpm, ggreg20_v3_usvperh, and ggreg20_v3_dose_usv:

• ggreg20_v3_cpm is the number of pulses per minute. The name indicates ggreg20_v3, this is the actual name of the module, and cpm is deciphered as counts per minute, that is, it is the number of pulses per minute. This indicator characterizes the current power of ionizing radiation per minute cycle. The greater the number of pulses, the greater the level of radiation. From this value, we can get the dose of radioactive radiation in μSv/h by using the following formula: cpm * 0.0092.

• ggreg20_v3_usvperh is the dose rate of ionizing radiation, or in other words, the radiation intensity, which is given in the dimension of μSv/h (microsievert per hour). The name indicates ggreg20_v3, this is the actual name of the module, and usvperh stands for uSv per hour. This indicator allows to assess the radiation situation at the current moment. With a normal radiation background, this value should be less than 50 μSv/h.

• ggreg20_v3_dose_usv is the equivalent dose of ionizing radiation accumulated since the IoT device was started. This indicator is given in μSv (microsievert). This indicator always increases, as it shows the received dose of ionizing radiation from the moment the device is started. That is, the longer the device works, the higher the received radiation dose becomes. So, the above-described IoT data source allows to obtain in real time the ionizing radiation dose rate in the Kyiv city, as well as the accumulated radiation dose at the current moment, starting from the start of the device. To find out the structure of the IoT device itself and its sensor, we look at the tags that were added on the site for the selected data source. Among them we have: "ggreg20_v3", "Iot-devices", "esp12.oled", "esp8266", "Kyiv", "Ukraine", "radiation", "sbm20", "geiger". The sensor is based on the MCU microcontroller "ESP12.OLED" because it has "esp12.oled" in the tags. The ESP12.OLED board is a ready-to-program universal ESP8266 MCU controller with a graphic display and interfaces for connecting digital and analog sensors and actuators (Fig. 3). It is used as a central module of IoT devices. This device is based on the NodeMCU ESP8266 board, which is a microcontroller manufactured by Espressif with a built-in Wi-Fi interface. The ESP12.OLED controller includes: GPIO connectors; MCU controller ESP8266-12; built-in monochrome OLED display SSD1306 128×64 0.96″ with I2C interface; Flash button (D3/GPIO0/P18); Rst button; RGB LED; UART interface for testing and programming.

This IoT device also uses "GGreg20_V3 Radioactive Particle Detector with SBM-20 Geiger tube", which is a sensor that measures radioactive radiation (Fig. 4).

The GGreg20_V3 radioactive particle detector is an electronic sensor module for building a personal Geiger counter and determining the level of ionizing radiation. For this, the sensor device provides an output, which can be used to count pulses on the controller. The radiation level can be indicated by light and sound signals. This module can be used to determine the power of ionizing radiation both indoors and outdoors, both in a handheld/pocket design and in stationary mode.

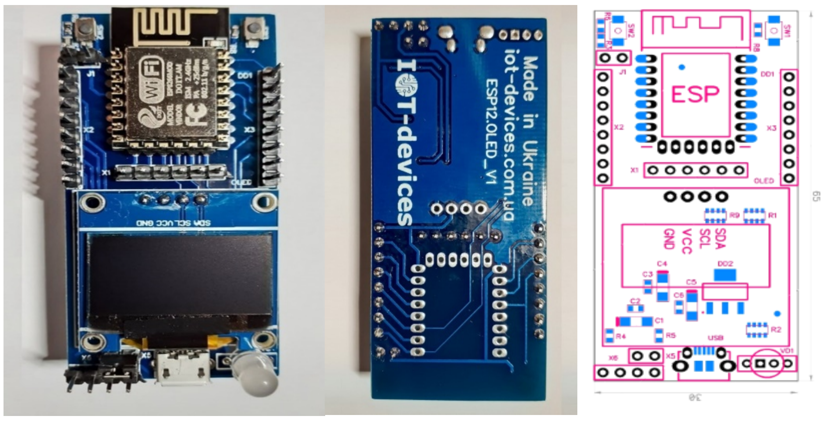


Fig. 3. Top and bottom, scheme of the ESP12.OLED board

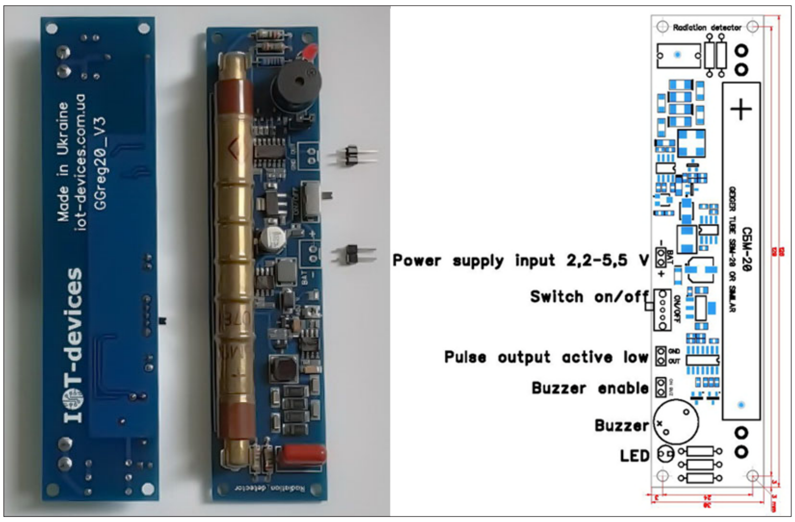


Fig. 4. The view and the scheme of the GGreg20_V3 detector

This sensor emits pulses at the GPIO output, which the microcontroller can count, and thus the power of ionizing radiation can be determined by the number of pulses per unit of time. The Geiger tube SBM-20, which can be easily replaced, fixes the ionizing radiation in this device.

Before using the data for analysis, we import the necessary Python libraries for working with the data:

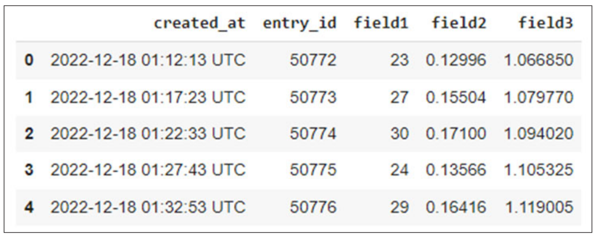
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
%matplotlib inline
```

We download a dataset of indicators that were recorded by the ionizing radiation sensor ggreg20_v3_kyiv_live. Since the file was provided by the site in CSV format, the appropriate function was used to load it into the dataframe:

```
# Import the feed.csv - exported recent data from IoT sensor ggreg20_v3_kyiv_live
# web page of the sensor: https://thingspeak.com/channels/1749073
radiationSensorDf = pd.read_csv(‘/content/drive/MyDrive/data_colab/feed.csv’)
```

We check the correctness of the import by displaying the first five records in the dataframe:

```
radiationSensorDf.head()
```

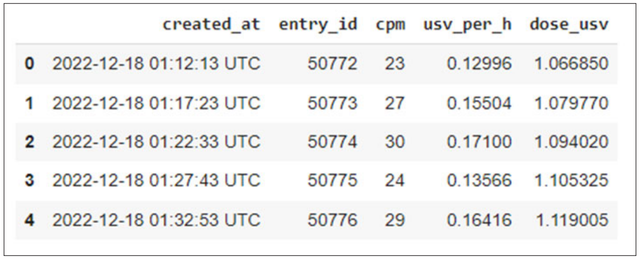


	created_at	entry_id	field1	field2	field3
0	2022-12-18 01:12:13 UTC	50772	23	0.12996	1.066850
1	2022-12-18 01:17:23 UTC	50773	27	0.15504	1.079770
2	2022-12-18 01:22:33 UTC	50774	30	0.17100	1.094020
3	2022-12-18 01:27:43 UTC	50775	24	0.13566	1.105325
4	2022-12-18 01:32:53 UTC	50776	29	0.16416	1.119005

As we can see, the content of the CSV file was correctly loaded into the dataframe. The table has five columns: created_at, entry_id, field1, field2, field3. The column created_at is responsible for the date and time of taking the indicator, entry_id is the unique identifier of the indicator measurement.

We rename the field1 column to cpm, because it is the number of pulses per minute (cpm stands for counts per minute), rename the field2 column to usv_per_h, because it is the dose of ionizing radiation (irradiation intensity), which is given in the dimension of μSv/h (microsievert per hour). Also we rename the column field3 to dose_usv, because it is the equivalent dose of ionizing radiation that has accumulated since the start of the IoT device, which is given in the dimension of microsievert:

```
columnNamesMapper = {
    "field1": "cpm",
    "field2": "usv_per_h",
    "field3": "dose_usv"
}
radiationSensorDf.rename(columns=columnNamesMapper,
inplace=True)radiationSensorDf.head()
```



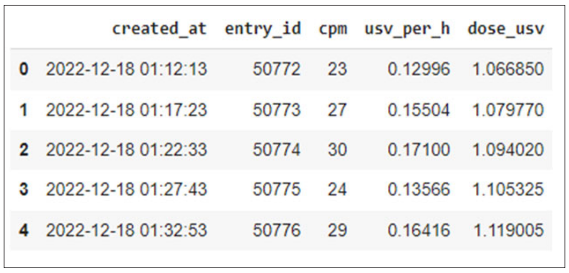
	created_at	entry_id	cpm	usv_per_h	dose_usv
0	2022-12-18 01:12:13 UTC	50772	23	0.12996	1.066850
1	2022-12-18 01:17:23 UTC	50773	27	0.15504	1.079770
2	2022-12-18 01:22:33 UTC	50774	30	0.17100	1.094020
3	2022-12-18 01:27:43 UTC	50775	24	0.13566	1.105325
4	2022-12-18 01:32:53 UTC	50776	29	0.16416	1.119005

We check the data types of each column in the dataset and make sure that there are no missing values in the rows:

```
radiationSensorDf.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   created_at  100 non-null    object
 1   entry_id    100 non-null    int64
 2   cpm         100 non-null    int64
 3   usv_per_h   100 non-null    float64
 4   dose_usv    100 non-null    float64
dtypes: float64(2), int64(2), object(1)
memory usage: 4.0+ KB
```

Therefore, there are no missing values in the table. Therefore, it is not necessary to filter the dataframe to exclude empty lines. As we can see, the data type for the created_at column is object. Since this column shows the date and time of fixing the sensor value, we convert all the values of this column to the datetime type by applying the appropriate function:

```
radiationSensorDf['created_at'] = pd.to_datetime(radiationSensorDf['created_at']).
dt.tz_localize(None)
radiationSensorDf.head()
```

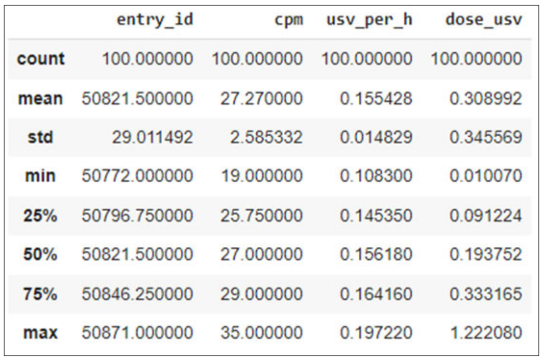


	created_at	entry_id	cpm	usv_per_h	dose_usv
0	2022-12-18 01:12:13	50772	23	0.12996	1.066850
1	2022-12-18 01:17:23	50773	27	0.15504	1.079770
2	2022-12-18 01:22:33	50774	30	0.17100	1.094020
3	2022-12-18 01:27:43	50775	24	0.13566	1.105325
4	2022-12-18 01:32:53	50776	29	0.16416	1.119005

Make sure that the created_at column now has the correct data type:

```
radiationSensorDf.info()
<class ‘pandas.core.frame.DataFrame’>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   created_at  100 non-null    datetime64[ns]
 1   entry_id    100 non-null    int64
 2   cpm         100 non-null    int64
 3   usv_per_h   100 non-null    float64
 4   dose_usv    100 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(2)
memory usage: 4.0 KB
```

We also analyze basic statistical information: minimum, maximum, average value, etc.: `radiationSensorDf.describe()`



	entry_id	cpm	usv_per_h	dose_usv
count	100.000000	100.000000	100.000000	100.000000
mean	50821.500000	27.270000	0.155428	0.308992
std	29.011492	2.585332	0.014829	0.345569
min	50772.000000	19.000000	0.108300	0.010070
25%	50796.750000	25.750000	0.145350	0.091224
50%	50821.500000	27.000000	0.156180	0.193752
75%	50846.250000	29.000000	0.164160	0.333165
max	50871.000000	35.000000	0.197220	1.222080

So, we can see that the maximum value of the power of ionizing radiation is 0.197 μSv/h, and the minimum is 0.108 μSv/h. The average value is 0.155 μSv/h.

We can see the same information about other columns. In addition, in the above statistical information, we can see the value of the standard deviation, percentiles of the level of 25%, 50%, 75%. We visualize sensor data using Python capabilities. To do this, we plot the dependence of all three measured values (cpm, usv_per_h, dose_usv) on the date and time of their fixation. We create a separate function that will create a graph with the appropriate captions:

```
def plot_sensor_data(column_name: str, column_label: str, color: str):
  fig, ax = plt.subplots(figsize=(15, 5))
  ax.plot(radiationSensorDf[“created_at”], radiationSensorDf[column_name],
f’{color}o-’)
  plt.xticks(rotation=90)
  ax.xaxis.set_major_locator(mdates.HourLocator())
  ax.xaxis.set_minor_locator(mdates.MinuteLocator(interval=15))
  ax.xaxis.set_major_formatter(mdates.DateFormatter(‘%d.%m %H:%M’))
  plt.ylabel(column_label, fontsize=15)
  plt.xlabel(‘Timestamp’, fontsize=15)
  plt.show()
```

We have the following three graphs, which visualize the dependence of the number of pulses per minute, the power of the dose of ionizing radiation (μSv/h), the equivalent dose of ionizing radiation (μSv) on time, respectively (Fig. 5, 6, 7):

```
plot_sensor_data(“cpm”, ‘Count of impulses per minute’, ‘g’)
plot_sensor_data(“usv_per_h”, ‘Dose rate, uSv/h’, ‘b’)
plot_sensor_data(“dose_usv”, ‘Equivalent dose, uSv’, ‘r’)
```

As we can see in the graphs above, there were four periods in which there was no electricity, as a result of which the sensor could not record the relevant indicators during this time. Since the equivalent dose of ionizing radiation shows the accumulated radiation from the start of the device’s operation, after the appearance of electricity, the corresponding graph starts from the zero mark and increases approximately uniformly until the next shutdown. We find the correlation between the two selected parameters for the time period from 12/18/22 01:00 to 12/19/22 01:00. To perform this task, the same IPython notebook was also used in the Google Colab environment. To build a heat map, import the Seaborn library:

```
import seaborn as sns
```

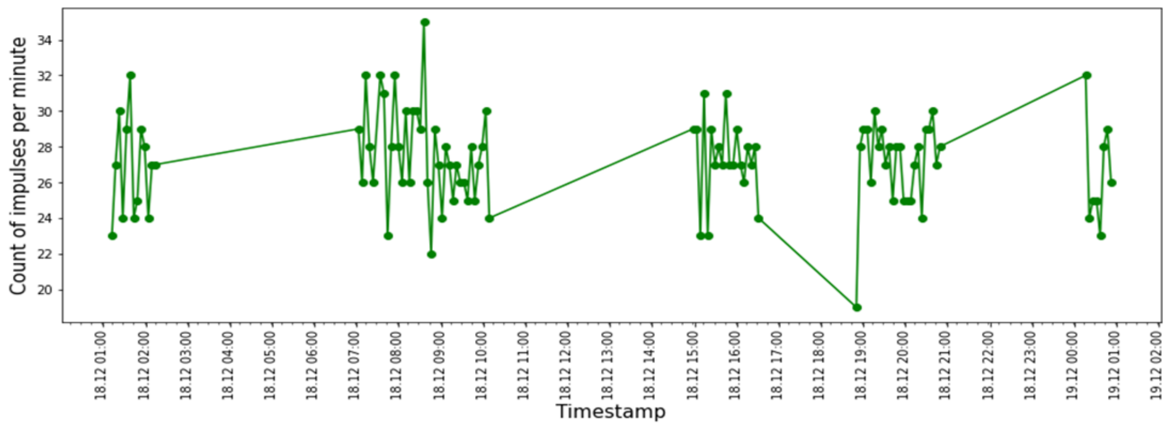


Fig. 5. The dependence of the number of pulses per minute

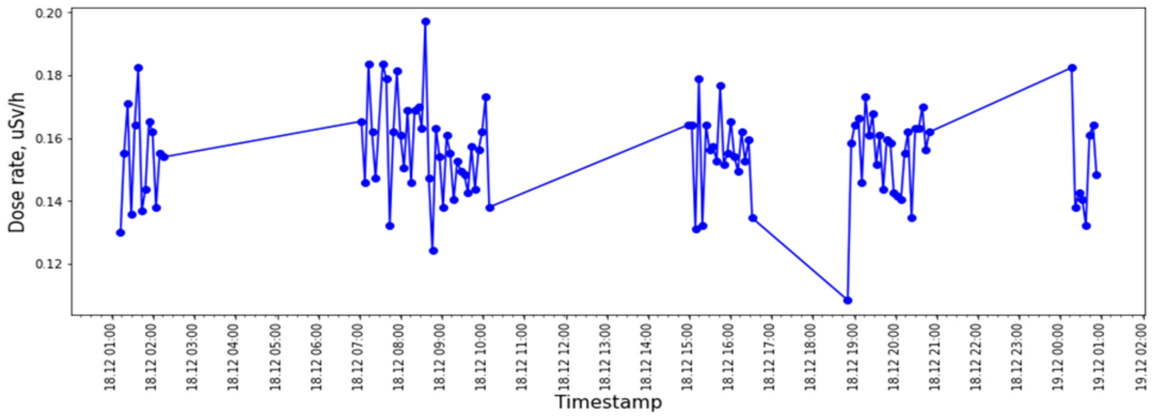


Fig. 6. The dependence of the power of the dose of ionizing radiation

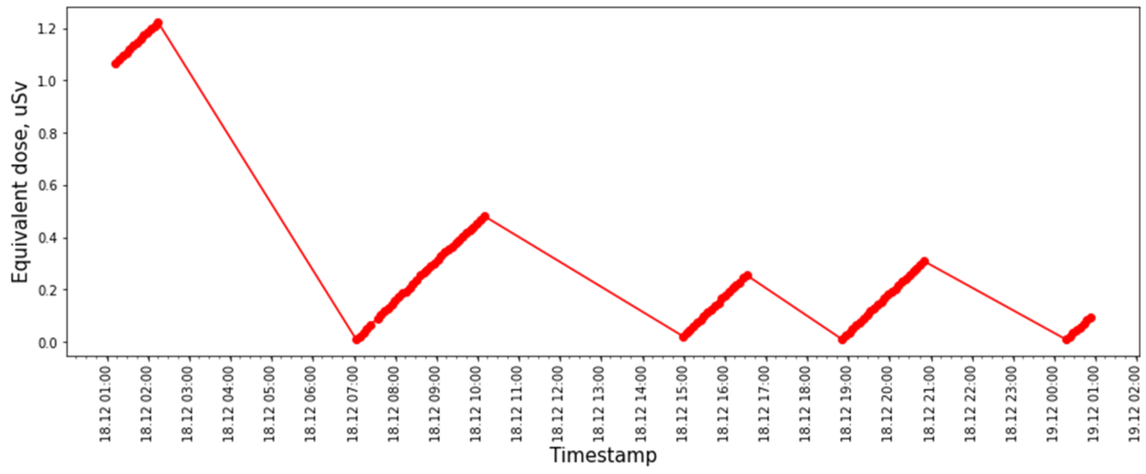


Fig. 7. The dependence of the equivalent dose of ionizing radiation

Before looking for a correlation between the parameters, first we remove the redundant characteristics. Namely, we will delete the entry_id column, since it simply displays the serial number of the measurement and has no value:

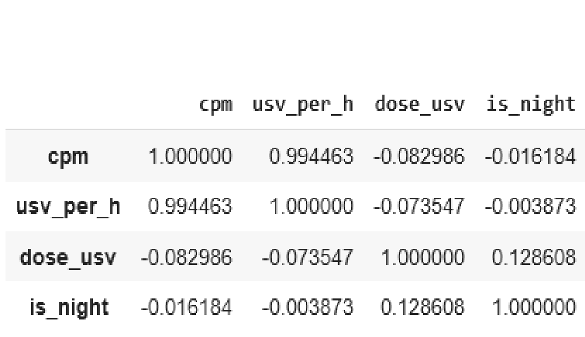
```
# The `entry_id` column has no relevance at this time, so it can be dropped.
radiationSensorDfModified = radiationSensorDf.drop('entry_id', axis=1)
radiationSensorDfModified.head()
```

We also add a new parameter to the dataframe. Since the dose of ionizing radiation at night can be higher or lower than the same value during the day, we will add a new column is_night to the dataframe. It will contain Boolean values. If the hour value from created_at for a measurement is greater than 15 or less than 8, then this measurement will have True in the is_night column and False in other cases:

```
radiationSensorDfModified[“is_night”] = radiationSensorDfModified.apply(lambda row:
row.created_at.time().hour >= 16 or row.created_at.time().hour <= 7, axis=1)
radiationSensorDfModified.head(n=25)
```

We build a correlation matrix for the parameters: cpm, usv_per_h, dose_usv, is_night (Fig. 8). To determine the correlation value, we will use the Pearson method:

```
radiationSensorModifiedCorr = radiationSensorDfModified.corr(method=’pearson’)
radiationSensorModifiedCorr
```



	cpm	usv_per_h	dose_usv	is_night
cpm	1.000000	0.994463	-0.082986	-0.016184
usv_per_h	0.994463	1.000000	-0.073547	-0.003873
dose_usv	-0.082986	-0.073547	1.000000	0.128608
is_night	-0.016184	-0.003873	0.128608	1.000000

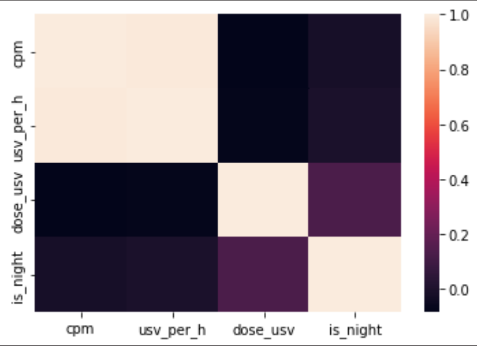


Fig. 8. Correlation of radiation indicators

Conclusions and future work

So, it can be seen that two parameters, namely: cpm and usv_per_h correlate very well, unlike the others. Since the correlation value of these two variables is 0.994, this is a very strong positive correlation. That is, with an increase in the number of pulses per minute, the power of the ionizing radiation dose also increases. This is a logical consequence, since the dose rate of ionizing radiation is calculated based on the number of pulses per minute by multiplying this value by a constant factor. We also conclude that the parameter is_night is not correlated with usv_per_h, because the correlation value is –0.004, that is, there is no dependence between these variables, so we can conclude that the initial assumption was wrong. Other parameters are also not correlated with each other, since other correlation indicators are also close to zero. This research is limited by the number of sensors of the monitoring systems in the Kyiv city. During military attacks on the territory of Ukraine, there are many threats of environmental pollution, especially air pollution, which affects the health and life expectancy of the Ukrainian population. The future development of this research is the use of complex sensors of air quality, namely gas concentrations, particle size to identify relationships between indicators.

References

1. Teh, H.Y., Kempa-Liehr, A.W. & Wang, K.IK. (2020) Sensor data quality: a systematic review. *Big Data* 7, 11. https://doi.org/10.1186/s40537-020-0285-1
2. Peltier, R.E., Buckley, T.J. (2020) Sensor technology: a critical cutting edge of exposure science. *Expo Sci Environ Epidemiol* 30, 901–902. https://doi.org/10.1038/s41370-020-00268-3
3. Rahman, M.H., Agarwal, S., Sharma, S. et al. (2023) High-Resolution Mapping of Air Pollution in Delhi Using Detrended Kriging Model. *Environ Model Assess* 28, 39–54. https://doi.org/10.1007/s10666-022-09842-5
4. Ramos, R.V., Blanco, A.C. (2022) Integrated GIS and air dispersion modeling approach for roadside pollutant mapping: a case study in Baguio City, Philippines. *Spat. Inf. Res*. 30, 371–383. https://doi.org/10.1007/s41324-022-00438-5
5. Keswani A., Akselrod H., and. Anenberg S. C. (2022) Health and Clinical Impacts of Air Pollution and Linkages with Climate Change. *NEJM Evid* 2022; 1(7). DOI: 10.1056/EVIDra2200068
6. Mujtaba, G., Shahzad, S.J.H. (2021) Air pollutants, economic growth and public health: implications for sustainable development in OECD countries. *Environ Sci Pollut Res* 28, 12686–12698. https://doi.org/10.1007/s11356-020-11212-1
7. Morantes, G., González, J.C. & Rincón, G. (2021) Characterisation of particulate matter and identification of emission sources in Greater Caracas, Venezuela. *Air Qual Atmos Health*. https://doi.org/10.1007/s11869-021-01070-2
8. Kaginalkar A., Kumar S., Gargava P., Kharkar N. and Niyogi D. (2022) SmartAirQ: A Big Data Governance Framework for Urban Air Quality Management in Smart Cities. *Front. Environ. Sci*. 10:785129. doi: 10.3389/fenvs.2022.785129