Noise Pollution Monitoring Using Internet of Things

Phase 5 submission document

Project title: Noise Pollution Monitoring

Phase 5: Project Documentation & Submission

Topic: In this section we will document the complete project and prepare it for submission.

Introduction:

Noise pollution, also known as sound pollution, is a significant environmental issue that can have adverse effects on human health, wildlife, and overall quality of life. It is caused by excessive and unwanted sound, often originating from various sources such as industrial activities, transportation, construction, and urban development. To address this problem, monitoring and managing noise pollution have become essential, and the Internet of Things (IoT) offers a powerful and innovative solution.

Noise pollution refers to the presence of high levels of noise that disrupt the normal acoustic environment. This can lead to a range of problems, including hearing damage, sleep disturbances, and stress-related health issues.

The IoT is a network of interconnected devices that can collect and exchange data over the internet. In the context of noise pollution monitoring, IoT devices play a crucial role in gathering, transmitting, and analyzing data related to sound levels in various locations.

The use of IoT technology for noise pollution monitoring is a promising approach to address this critical environmental issue. By providing accurate and real-time data, IoT systems empower governments, organizations, and individuals to take informed actions to reduce noise pollution and create quieter, healthier living environments.

Here's a list of tools and software commonly used in the process:

1.Noise Sensors and Microphones:

Various noise sensors and microphones are used for data collection. These hardware components are often integrated with IoT devices.

2. Data Loggers:

Data loggers are essential for collecting and recording noise data from sensors. They can be standalone devices or integrated into sensor systems.

3. IoT Device Management Platforms:

Platforms like AWS IoT Core or Azure IoT Hub are employed to manage and monitor IoT devices remotely.

4. Dashboard and Visualization Tools:

Tools like Grafana, Tableau, or custom-built dashboards enable real-time and historical visualization of noise data.

5. Geographical Information Systems (GIS):

GIS software, like ArcGIS or QGIS, is used to overlay noise data onto maps, allowing for spatial analysis and visualization.

6. Connectivity Protocols:

Wireless communication protocols such as Wi-Fi, LoRaWAN, Sigfox, and cellular (e.g., 4G/5G) enable data transmission from sensors to central databases or cloud platforms.

7.Cloud Platforms:

Cloud computing platforms like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, and IBM Cloud provide storage, data processing, and analysis capabilities for noise data.

8.IoT Development Platforms:

IoT development platforms like Arduino, Raspberry Pi, or specialized platforms designed for noise monitoring are used to create sensor nodes and IoT devices.

9.Database Management Systems:

Systems like MySQL, PostgreSQL, or NoSQL databases (e.g., MongoDB) store and manage collected noise data.

10.Mobile Applications:

Custom mobile apps can be developed to provide real-time noise monitoring and reporting capabilities to users.

11.Compliance and Reporting Software:

Software for regulatory compliance and reporting, tailored to specific noise regulations, may be used by environmental agencies and organizations.

12.Security Tools:

Security measures and software are crucial to protect the integrity and privacy of noise data, including encryption, firewalls, and intrusion detection systems.

13.Remote Monitoring and Control Software:

Software that enables remote monitoring and control of IoT devices for maintenance and updates.

14.Data Analysis Software:

Data analysis tools and software, such as Python with libraries like Pandas and Matplotlib, are commonly used for processing and visualizing noise data.

15.Notification and Alerting Systems:

Systems that send alerts or notifications to relevant parties when noise levels exceed predefined thresholds.

Required packages and installation:

- **IoT Development Libraries**
- **Noise Sensor Libraries:**
- **Database Libraries:**
- **Web Framework (Optional):**
- **Data Visualization Libraries:**
- **Cloud Service SDKs (e.g., AWS SDK, Azure SDK):**
- **IoT Platform-Specific Packages:**
- **Security Packages (e.g., cryptography, SSL libraries):**

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1.Empathize:

- Start by understanding the needs and pain points of the community affected by noise pollution. Conduct surveys, interviews, and observations to gain insights into their experiences.

●Conduct interviews and surveys to gather insights on what users

value in property valuation and what information is most critical for

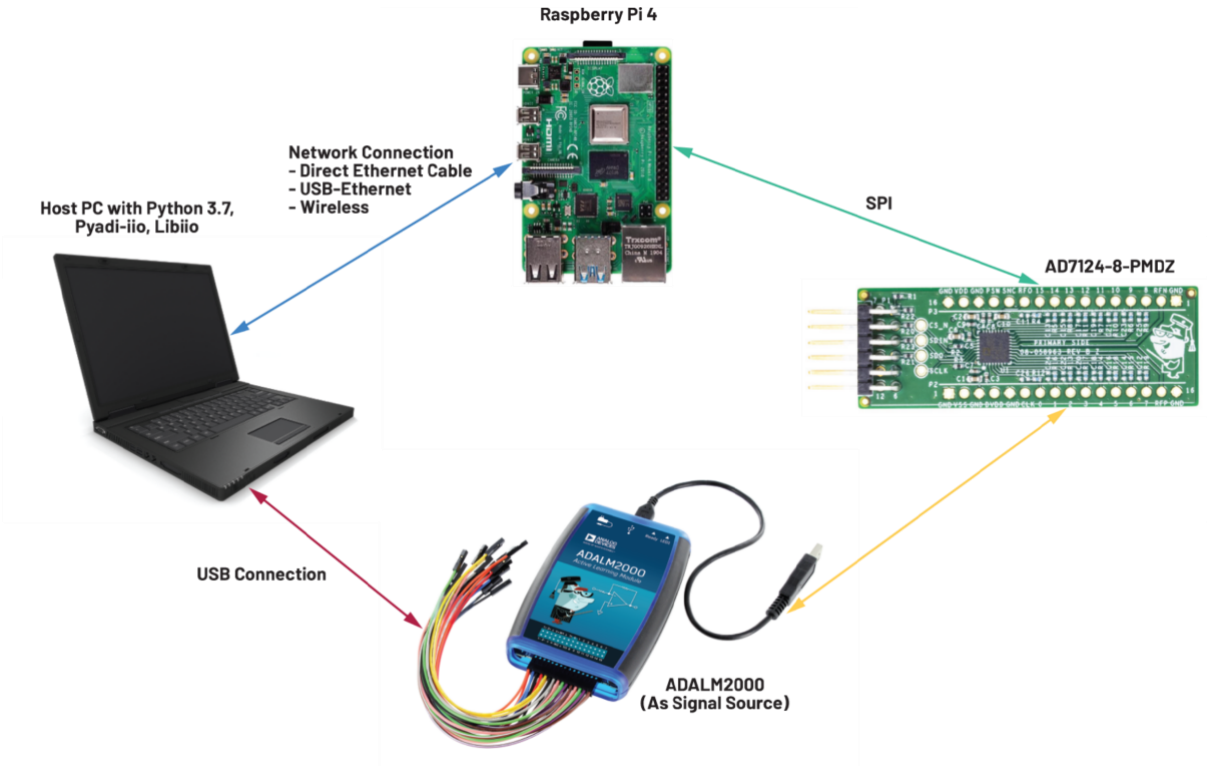
their decision-making.

2. Define:

- Define the problem by creating a clear problem statement. For example, "How might we reduce noise pollution in urban areas to improve residents' quality of life?"

4. Prototype:

- Create prototypes and models of the IoT noise monitoring system. This can include physical sensor setups and software interfaces
- Based on the feedback and testing, proceed with the full implementation of the IoT noise pollution monitoring system.



When presenting your design thinking process and IoT noise pollution monitoring system, use visuals and clear communication to engage your audience.

2.DESIGN INTO INNOVATION

Creative Technology Integration:

- Explore cutting-edge IoT technologies and sensors that can provide more accurate and real-time noise data. For example, consider advanced noise sensors, AI-driven analysis, and 5G connectivity.

Data Integration:

●Consider how data from multiple IoT sensors can be integrated to create a holistic view of noise pollution in an area. Explore machine learning and data analytics for deeper insights

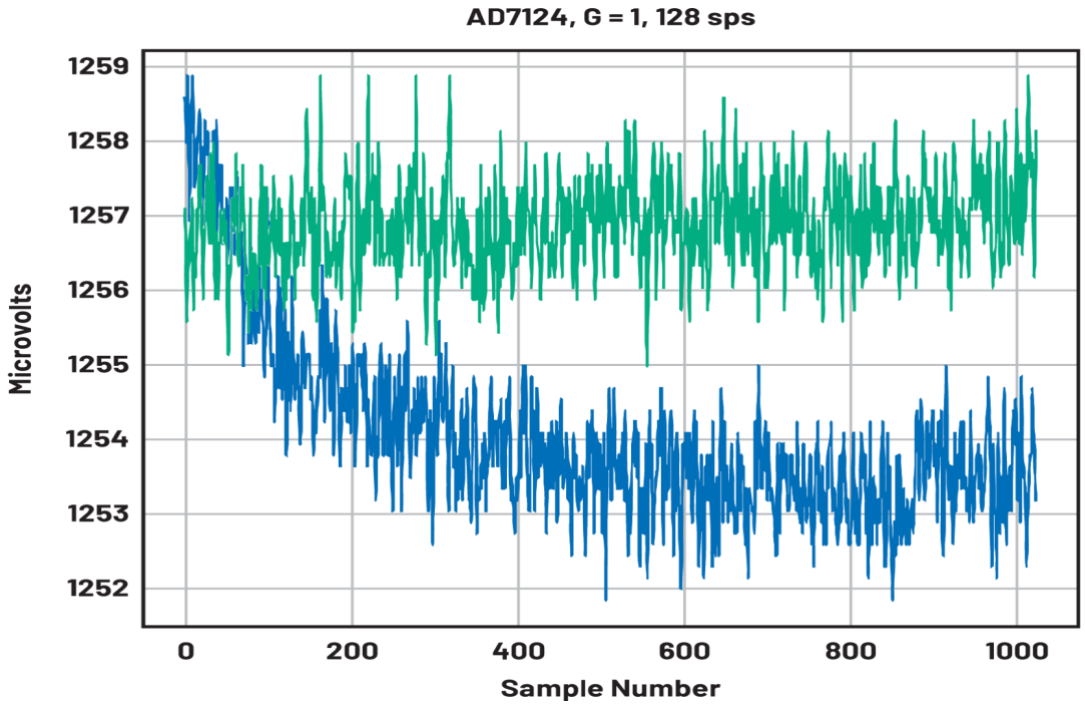
PYHON PROGRAM:

```
# AD7124-8 Basic Data Capture

import adi  # pyadi-iio library
# Connect to AD7124-8 via Raspberry Pi
my_ad7124 = adi.ad7124(uri="ip:analog.local")
ad_channel = 0  # Set channel
# Set PGA gain
my_ad7124.channel[ad_channel].scale = 0.0002983
my_ad7124.sample_rate = 128  # Set sample rate
# Read a single "raw" value
v0 = my_ad7124.channel[ad_channel].raw
# Buffered data capture
my_ad7124.rx_output_type = "SI"  # Report in volts
# Only one buffered channel supported for now
my_ad7124.rx_enabled_channels = [ad_channel]
my_ad7124.rx_buffer_size = 1024
my_ad7124._ctx.set_timeout(100000)  # Slow
data = my_ad7124.rx()  # Fetch buffer of samples

print("A single raw reading: ", v0)
print("A few buffered readings: ", data[:16])
del my_ad7124  # Clean up
```

Output:



Identify the Problem:

- Start by clearly defining the problem of noise pollution in urban areas and understanding its implications on public health and well-being.

User-Centered Design:

- Embrace a user-centered design approach. Understand the needs and pain points of the community affected by noise pollution through surveys, interviews, and observations.

Design Thinking:

- Apply design thinking principles, including empathy, problem definition, ideation, prototyping, and testing, to refine your innovative concept.

Prototyping:

- Create prototypes of the innovative IoT noise monitoring system. This could involve physical sensor setups, software interfaces, or even augmented reality (AR) visualization.

Scalability and Sustainability:

- Plan for scalability and sustainability, ensuring that the innovative system can expand to cover larger areas and be maintained over time.

Visualization and Public Engagement: -

- Develop engaging and informative data visualizations that not only provide noise data but also educate and engage the public on the issue of noise pollution.

Real-World Testing:

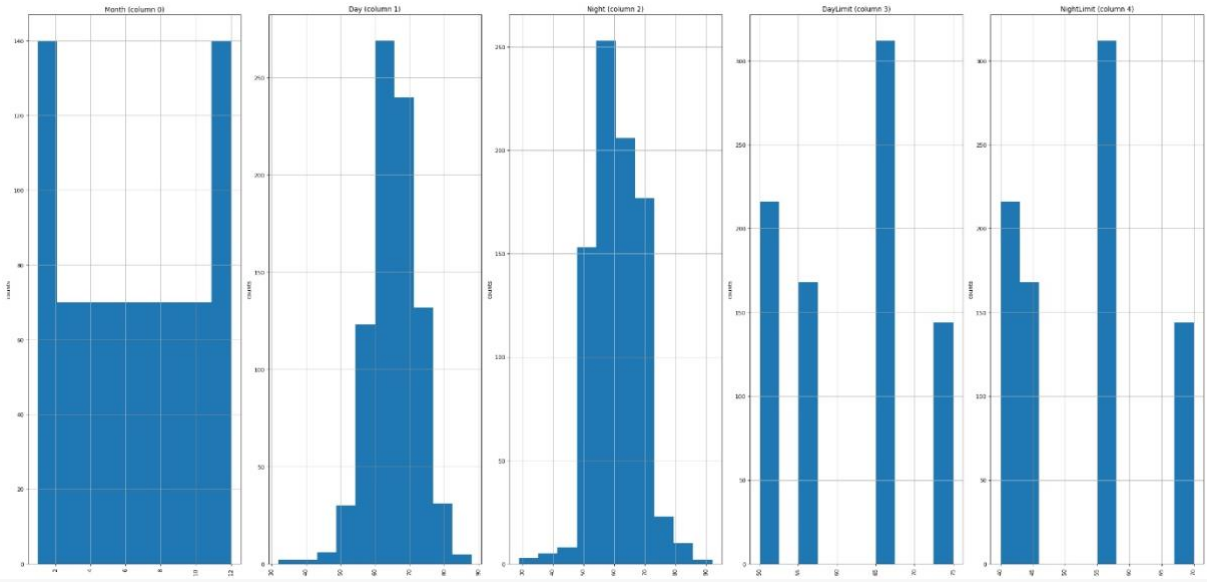
- Conduct real-world testing and gather feedback from the community to refine the innovation.

3.Development Part-1

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

for dirname, _, filenames in os.walk:
    for filename in filenames:
        print(os.path.join(dirname, filename))
# Distribution graphs (histogram/bar graph) of column data
def. plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 5
0]] # For displaying purposes, pick columns that have between 1 and 50 un
ique values
    nRow, nCol = df.shape
columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow),
 dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
```

```
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```



4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.,

Feature Engineering:

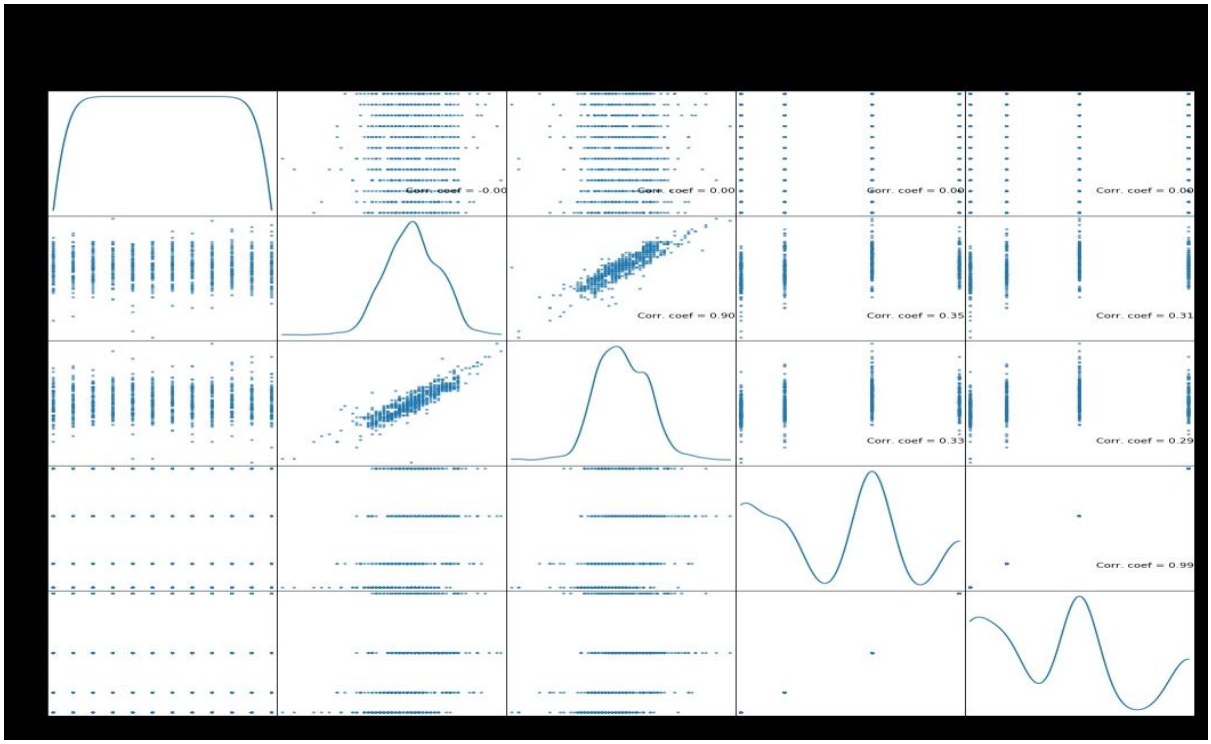
- Converting the noise level data to a logarithmic scale. This is because human perception of noise is logarithmic, meaning that a small increase in noise level at low levels is perceived as a larger increase than the same increase at high levels.
- Creating features that represent the time of day, day of the week, and season. This is because noise pollution levels can vary depending on these factors.
- Creating features that represent the location of the IoT devices. This is because noise pollution levels can vary depending on the environment, such as whether the device is located in a noisy urban area or a quiet rural area.

Model training:

- Support vector machines (SVMs)
- Logistic regression
- Random forests
- Neural networks

CODE:

```
                import tensorflow as tf
import paho.mqtt.client as mqtt
import time
# Load the machine learning model
model = tf.keras.models.load_model("noise_classifier.h5")
# Create an MQTT client
client = mqtt.Client()
# Connect to the MQTT broker
client.connect("localhost")
# Start a loop to read the microphone array data, classify the noise sources, and publish the results to the IoT platform
while True:
  # Read the microphone array data
    audio_data = get_microphone_array_data()
   # Classify the noise sources
    noise_classes = model.predict(audio_data)
 # Publish the noise classification results to the IoT platform
    client.publish("noise_classification", json.dumps(noise_classes))
  # Wait for 1 second
    time.sleep(1)
# Disconnect from the MQTT broker
client.disconnect()
```

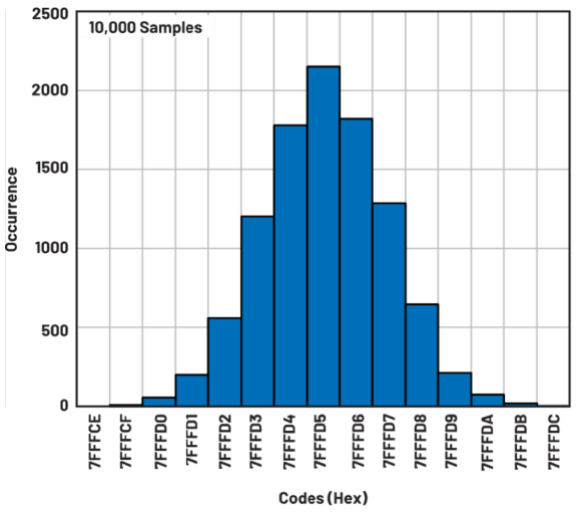


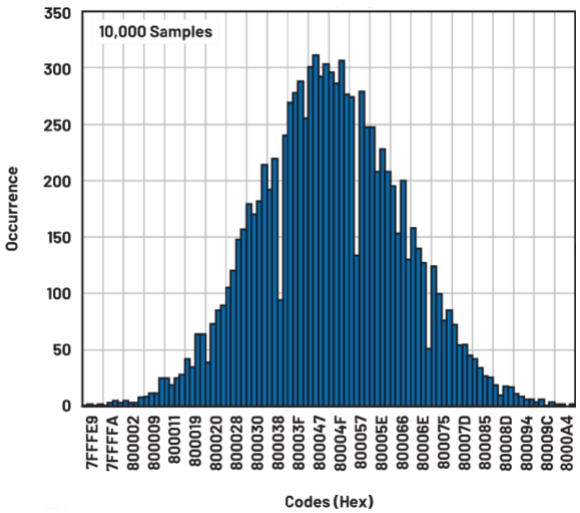
Conclusion:

IoT-based noise pollution monitoring systems can be used to protect public health, improve environmental quality, and increase the quality of life for people in urban and rural areas alike.

1. **Smart cities:** IoT-based noise pollution monitoring systems are being used in smart cities to monitor noise levels from traffic, construction, and other sources. The data collected from these systems is used to develop and implement noise pollution reduction strategies.
2. **Industrial areas:** IoT-based noise pollution monitoring systems are being used in industrial areas to monitor noise levels from factories and other industrial activities. The data collected from these systems is used to ensure that noise levels are within compliance limits and to protect the health of workers and nearby residents.
3. **Public transportation:** IoT-based noise pollution monitoring systems are being used on public transportation vehicles to monitor noise levels and ensure that they do not exceed safe limits.

```
# AD7124-8 Filter Response
import numpy as np
import matplotlib.pyplot as plt
resp = []
freqs = np.linspace(1, 121, 100, endpoint=True)
for freq in freqs:
    print("testing ", freq, " Hz")
    send_sinewave(my_siggen, freq)   # Set frequency
    time.sleep(5.0)                  # Let settle
    data = capture_data(my_ad7124)   # Grab data
    resp.append(np.std(data))  # Take RMS value
    if plt_time_domain:
        plt.plot(data)
        plt.show()
    capture_data(my_ad7124)   # Flush
# Plot log magnitude of response.
response_dB = 20.0 * np.log10(resp/0.5*np.sqrt(2))
print("\n Response [dB] \n")
print(response_dB)
plt.figure(2)
plt.plot(freqs, response_dB)
plt.title('AD7124 filter response')
plt.ylabel('attenuation')
plt.xlabel("frequency")
plt.show()
```





```
# Noise Source Measurement
import numpy as np
navgs = 32  # Avg. 32 runs to smooth out data
ns = 2**16
vsd = np.zeros(ns//2+1)  # /2 for onesided
for i in range(navgs):
  ch1 = np.asarray(data[0])  # Extract ch 1 data
  ch1 -= np.average(ch1)  # Remove DC
  fs, psd = periodogram(ch1, 1000000,
                        window="blackman",
                        return_onesided=True)
  vsd += np.sqrt(psd)
vsd /= navgs
sinc4_50 = np.convolve(sinc2_50, sinc2_50)

# Here's the SINC4-ish filter from datasheet
# Figure 91, with three zeros at 50Hz, one at 60Hz.
filt_50_60_rej = np.convolve(sinc3_50, sinc1_60)
```