PROTYPING A METHOD TO CROWD SOURCE LOCAL NOISE LEVELS

Martin Leff, Keding Wang, Mariam Vashakidze, Zixuan

Fall 2023

CARNEGIE MELLON UNIVERSITY

Table of Contents

Problem Statement and Motivation

Consistently high or dangerous noise levels have been known to cause serious health impacts for those who live near their sources. High noise levels are linked to hearing impairments, sleep disturbance, and heart disease.[1] Additionally, the presence of persistent noise can adversely affect children's learning and development, and even impact local property values. In the cases of roadways as the source of noise, mitigation methods may include modifying speed limits, prohibiting specific types of vehicles from traveling on certain roads, or using traffic control devices to slow vehicles.[2] The US government sets guidelines for what is considered safe noise levels when on work sites, with engineering controls required when exposures exceeding 90 dBa.[3] Understanding the locations of high noise levels is in the interest of local and national governments since they may wish to prevent negative health impacts on citizens by taking these actions. In our project we utilized an analog noise sensor alongside a digital GPS sensor to prototype one method that local governments or citizens could use to map and share the locations of the sources of high noise levels in their neighborhoods.

Solution and Use Cases

Fundamentally, our proposed solution utilizes two types of sensors to map ambient noise levels in an area. Our design was inspired by Speck, a small and low-cost air quality monitor developed and implemented at Carnegie Mellon University.[4] In our design a digital GPS sensor is used to find the position in latitude and longitude on a map where readings are taken. At the same time, a sound level sensor takes readings of the ambient noise level, in decibels (dB). Readings are grouped by location based on their latitude and longitude coordinates, corresponding to a grid where each cell of the grid is a square with dimensions 10 meters by 10 meters in size. This level of grouping avoids over- or under-smoothing the noise level data, allowing localized noise levels to become visible. Once readings are grouped by location, the median noise value is selected and displayed to the user on a map.

Our prototype used a laptop connected to both the noise and the GPS location sensors to collect and display data. The laptop was mounted on a wheeled cart and moved from location to location while readings were taken. However, we envision that a future version could combine both sensors into a single device small enough to be a handheld. This device could be retrievable from a local library or other community hub, where citizens could take them out and measure noise levels in their area. Storing data to a larger database would require an internet connection,

---

[1] Passchier-Vermeer, W., & Passchier, W. F. (2000). Noise exposure and public health. *Environmental Health Perspectives*, *108*(suppl 1), 123–131. https://doi.org/10.1289/ehp.00108s1123

[2] US Department of Transportation. (2017, August 24). *Highway Traffic Noise Analysis and Abatement Policy and Guidance*. FHWA. https://www.fhwa.dot.gov/environMent/noise/regulations_and_guidance/polguide/polguide01.cfm

[3] Centers for Disease Control and Prevention. (2018b, December 11). *Loud noise can cause hearing loss*. Centers for Disease Control and Prevention. https://www.cdc.gov/nceh/hearing_loss/default.html

[4] *Meet Speck*. Speck by Airviz. (n.d.). https://www.specksensor.com/

so the device would connect to Wi-Fi to do so. If no connection was available, the device could store data temporarily until a connection was established and then send the data at that time. In a more advanced version of this device, we would also allow users to track their own data specifically by querying the database for their own user ID.

In our design most of the data inputs are crowd-sourced by users. However, their inputs are stored in a central database that would be accessible to government officials. These officials may choose to inspect the maps that are complete with readings from many users to identify locations with high ambient noise levels. Once identified, policymakers can implement programs to mitigate the noise in those locations. The current version of our design does not group noise readings temporally; all readings in the same location, regardless of the time of day they were taken, are grouped together and displayed on the map. However, in a future version of the design we would also group the readings by time, so that areas with high noise levels at specific times can be identified. This would allow policymakers to fine tune their programs intended to mitigate high noise levels, since noise sources may not be constant within a location.

Description of the Physical Phenomena

Sound Level

Sound is a wave in air pressure created by the vibration of an object and propagated through a medium such as air or water. The sound is received by a sensor such as a human ear or the noise sensors we used in our prototype. Sound has many characteristics, including frequency, amplitude, wavelength, and speed. The amplitude of the sound wave is related to the intensity of sound (loudness). The unit of sound intensity level is the decibel (dB). Decibels are measured on a logarithmic scale against a reference sound intensity, as showing in the following equation:[5]

$$L_I = 10\, log_{10} \left| \frac{I}{I_0} \right| dB$$

Where L_I is the sound intensity in decibels, I is the sound sound intensity in watts per square meter (W/m^2), and I_0 is a reference sound intensity. Typically, I_0 is set to 1 picoWatt per square meter (pW/m^2), which is approximately the lowest sound intensity hearable by a human in room conditions. Since this is measured on a logarithmic scale, an increase in 10 dB represents a doubling of the perceived loudness of a sound source. Figure 1 displays a descriptive range of various sound levels, in decibels:[6]

[5] *Sound Intensity*. HyperPhysics Concepts. (n.d.). http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/intens.html

[6] Sygrove, C. (2023, January 2). *Decibel Chart: All You Need to Know*. MDHearing. https://www.mdhearingaid.com/blog/decibel-chart/

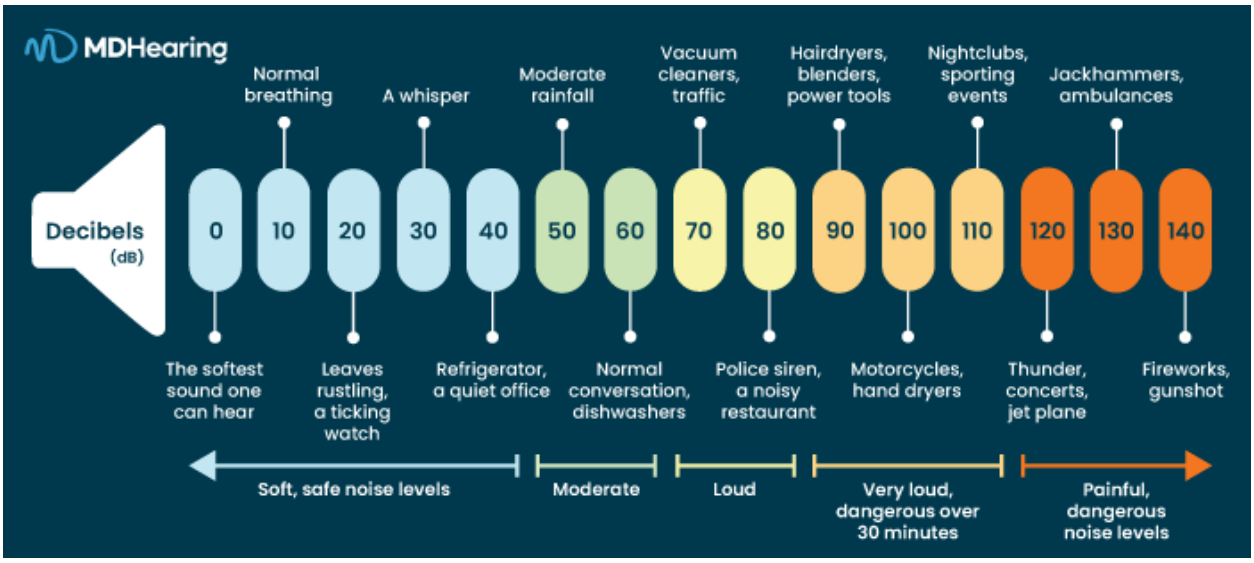

Figure 1. Decibel hearing chart qualitatively describing a range of noise levels from 0 dB to 140 dB.

Based on Figure 1 we would anticipate decibel readings of between 60 and 90 while taking measurements outside, since this represents the range from the sound of household appliances to the sounds of more intense appliances and power tools.

Position in Latitude and Longitude

For tracking objects and places located on the Earth's surface, we typically use a coordinate system in units of Latitude and Longitude. Each degree Latitude or Longitude is roughly equivalent to 111 km in distance. The maximum Latitude value is $90°$ N while the minimum is $90°$ S. Longitude ranges from $-180°$ to $180°$, measured West to East across the globe. The combination of a latitude and longitude coordinate provides the exact position of an object on the Earth's surface.

Methodology

We used a multifaceted approach that integrates technology and data analysis to measure noise levels across a wide area. We employed the Taidacent industrial grade noise decibel detection module (TTL-12V) noise level sensor[7] to accurately gauge the decibel readings across various locations. This type of sensor works by converting changes in air pressure caused by sound into electrical signals. Specifically, the changes in air pressure change the shape of a flexible diaphragm within the sensor, resulting in a change in capacitance. In turn, this changes the electrical signal of the output.[8] Furthermore, the sensor gives different weight to different received sounds based on their frequency, like the response of human ears. The signal is then

[7] *Taidacent Industrial Noise Decibel Detection Module Sound Sensor to Measure Sound Levels in Decibels RS485 5V Uart Dust Noise Meter*. Amazon.com. (n.d.). https://www.amazon.com/Taidacent-Industrial-Decibel-Detection-Measurement/dp/B07QMLYV5B

[8] *Electret Condenser Microphones Explained*. Alan Butcher Components. (n.d.). https://www.abcomponents.co.uk/electret-condenser-microphones/

filtered and amplified to show a single sound level reading at each point in time. The transfer function to convert the readings from the sensor into a dB reading is as follows:

$$Sound\ Level = 30\ dB + \left(\frac{3.3\ V}{3\ V} * \frac{reading}{65535} * 90\ dB\right)$$

In the transfer function, 30 dB represents the minimum value that can be measured by the sensor, corresponding to a 0 V output. 90 dB represents the difference between the maximum decibel reading of the sensor (120 dB) and the minimum. The 3.3 V and 3 V factors are representative of the input and output voltages, respectively. This sensor is connected to the Raspberry Pi Pico W, which typically measures voltage values between 0 and 3.3 V using 12 bit resolution. However, Micropython transmits the value as 2 bytes (16 bits), casting the read value to be an integer from 0 to 65535. As a result, we defined our transfer function to divide the read value by 65535. A sample circuit diagram is shown in Figure 2:

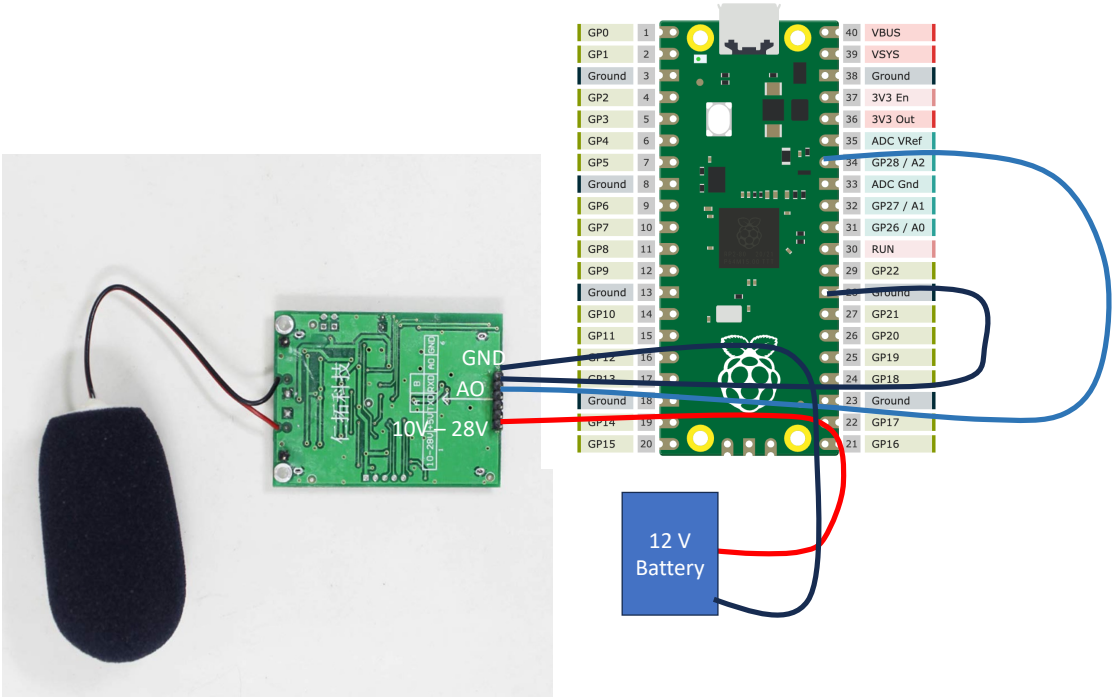

Figure 2. Circuit diagram connecting the Taidacent noise level sensor to the Raspberry Pi Pico W and a 12 V battery

Figure 2 displays how the noise level sensor could be connected to the Raspberry Pi Pico W, which in turn processes and stores the noise level readings. Since this sensor has additional power requirements, we included a 12 V battery in the circuit to meet the sensor's specifications. In the diagram, the red wire connects the power input of the sensor to the 12 V battery. The black wires connect the sensor and the battery to ground. The blue wire connects the analog output of the sensor to one of the analog input pins on the Raspberry Pi Pico W.

These noise level measurements were geographically pinpointed using the GPS sensor Columbus P-7 Pro Submeter with 0.5-meter accuracy,[9] ensuring precise location tracking for each noise level recording. The collected data was stored in a database, which served as a repository for the noise and location data, allowing for efficient management and retrieval. Subsequently, the aggregated data from this database was visualized on a map, facilitating an intuitive understanding of the noise pollution landscape across the area under study. This systematic mapping not only highlights the current noise levels but would be instrumental in identifying critical hotspots requiring immediate attention for noise mitigation efforts.

Design Stage Uncertainty

The parameters that our sensor records are a timestamp, the noise level in decibels, and the longitude and latitude positions at the time of measurement. Our GPS sensor was the Columbus P-7 Pro Submeter with 0.5-meter accuracy, with USB and Bluetooth GNSS Receiver, with the following specifications:

- 0.5-meter accuracy
- 5 Hz update rate
- Dual-Frequency (L1 + L5)
- 6 constellations (GPS, GLONASS, Galileo, BeiDuo, QZSS and IRNSS)
- Quad output interfaces (Bluetooth SPP, Bluetooth BLE, 2 USB Serial Devices/COM ports)
- Multi-platform compatible (Windows, MacOS, Linux, Android, IOS, iPadOS)

The uncertainty of the GPS device was represented by the 0.5-meter accuracy specification. This implies that the GPS coordinates returned from the sensor are within 0.5 meters of the true position. The readings from the sensor were returned in decimal degrees latitude and longitude, which were converted using a conversion factor into meters. One degree latitude or longitude is equivalent to about 111 km, so the 0.5-meter accuracy corresponds to about 0.000005 degrees of latitude or longitude.

Our noise level sensor was the Taidacent industrial grade noise decibel detection module sound sensor sound level meter sound measurement (TTL-12V), with the following specifications:

- Noise accuracy $\pm$ 0.5dB
- Measuring range 30dB to 120dB
- Response range 20Hz to 12.5 kHz
- Response time fast 500ms, slow 1.5s
- Frequency weighting A
- Output signal TTL, RS485, analog
- Operating temperature -20 degrees Celsius to +60 degrees Celsius, 0%RH to 80%RH

[9] *Columbus P-7 Pro submeter (0.5 meter accuracy) USB and Bluetooth GNSS receiver*. GPSWebShop. (n.d.). https://gpswebshop.com/products/columbus-p-7-pro-submeter-0-5-meter-accuracy-usb-and-bluetooth-gnss-receiver

- Power consumption 18.9mA at 5V; 31.0mA at 12V; 27.8mA at 24V
- Operating voltage 4.5-5 V (default); 10-28V (optional)

The uncertainty of the noise level sensor was represented by the accuracy of ± 0.5dB, indicating that the actual noise level would be within 0.5 dB of the reading. The response time of the sensor was set to "slow," meaning noise level readings were taken every 1.5 seconds. To handle the uncertainty of the noise measurements, we selected the median value of the noise readings collected in the same location. By choosing the median, we hoped to minimize the impact of outliers on our dataset and to identify a true representative sound level for each location. Choosing the median value helped distinguish locations with loud but infrequent noise disturbances from locations with more consistent but higher amplitude average noise levels.

User Interface Design

Our system was essentially a mobile environmental monitoring station combining a noise sensor and a GPS sensor. It was designed to collect sound level data in decibels (dB) and corresponding geographical location data in real-time. While both sensors were connected to one serial port each on the same computer, data was collected by running the python script "Project_Store_Data.py" (Appendix 1: Python Code). Both sensors were configured to send constant readings to the serial port while plugged in, which were read by the python script. The python code used to configure the Raspberry Pi Pico to send the noise level readings is available in

Appendix 2: Raspberry Pi Pico Code. The mobile monitoring station could be left in a single location or moved while data was being collected. When the user wished to end data collection, they could stop the program which would trigger the data to be committed to the database "Noise and Position.db" where it could later be referenced. This database used the "sqlite3" module available in python to store data. This database would store readings for the current time, the latitude, the longitude, and the noise level reading.

To view and map the data, the user could then run the python script "Project_Load_Data.py" which would access the "Noise and Position.db" database and group the datapoints by their latitude and longitude coordinates. Within the program a grid was created with areas of 10 meters by 10 meters, breaking down the existing latitude and longitude coordinates into areas to group by. With the datapoints grouped by the grid cells, the median sound value for each cell was selected and displayed on the map. Open Street Map was used as a base layer for the user's reference.[10] A user can zoom in, scroll, shift, and hover over points in the map to see more detail. The points on the map were color-coded to display higher noise levels as darker and quieter noise levels as lighter. From this map it was easy for a user to tell where the noisiest areas were since they were easily distinguished from the lighter (quieter) points.

[10] Boeing, G. (2017). OSMNX: New methods for acquiring, constructing, analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*, *65*, 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004

E-R Diagram

Figure 3 displays the E-R diagram for the "Noise and Position.db" database:

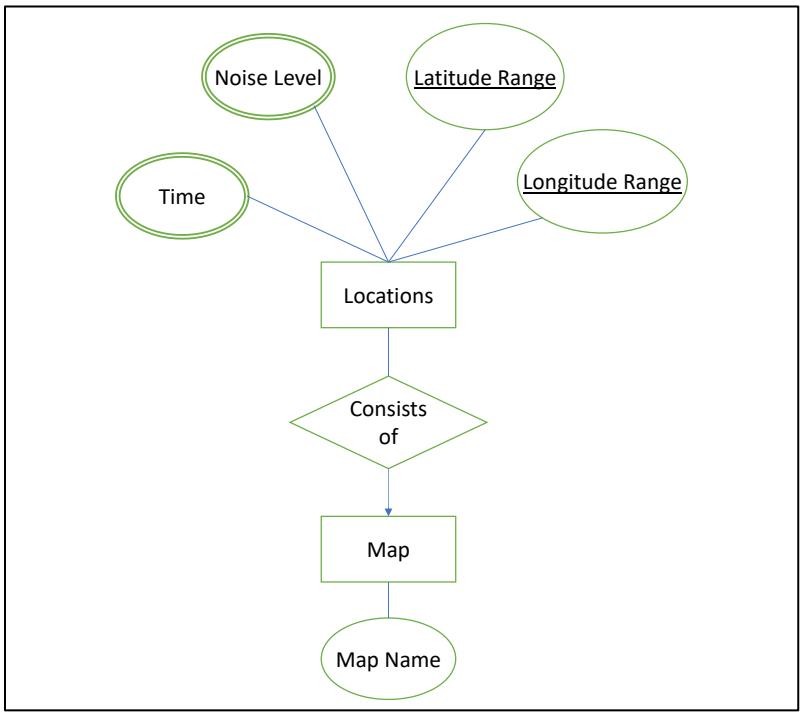

Figure 3. Entity-Relationship Diagram describing the relationship between Locations and the Map

From Figure 3 we can see that the Locations have a many-to-one relation with the Map, which consists of all the locations. The Locations have attributes of the latitude range, the longitude range, as well as the noise and timestamp readings. Since we took multiple noise level readings at several timestamps, these attributes are represented with a double-line. The key for the Locations entity sets is the latitude range and the longitude range, where each latitude and longitude range combination will uniquely identify a Location. The Locations correspond to the grid cells defined by 10 meter by 10 meter squares overlaid on the world map.

Validation and Experiments

To validate our design, we conducted experiments on Carnegie Mellon University's campus on December 7th and December 8th, 2023. We moved the cart with both sensors connected to a laptop computer from the sidewalk alongside Frew Street to the North end of the Cohon University Center. Along the way we continually collected measurements from both sensors, while also pausing at times to conduct more specific measurements in certain locations. Figure 4 displays the trajectory of our experiments across the campus, with annotations for specific sections of our path:

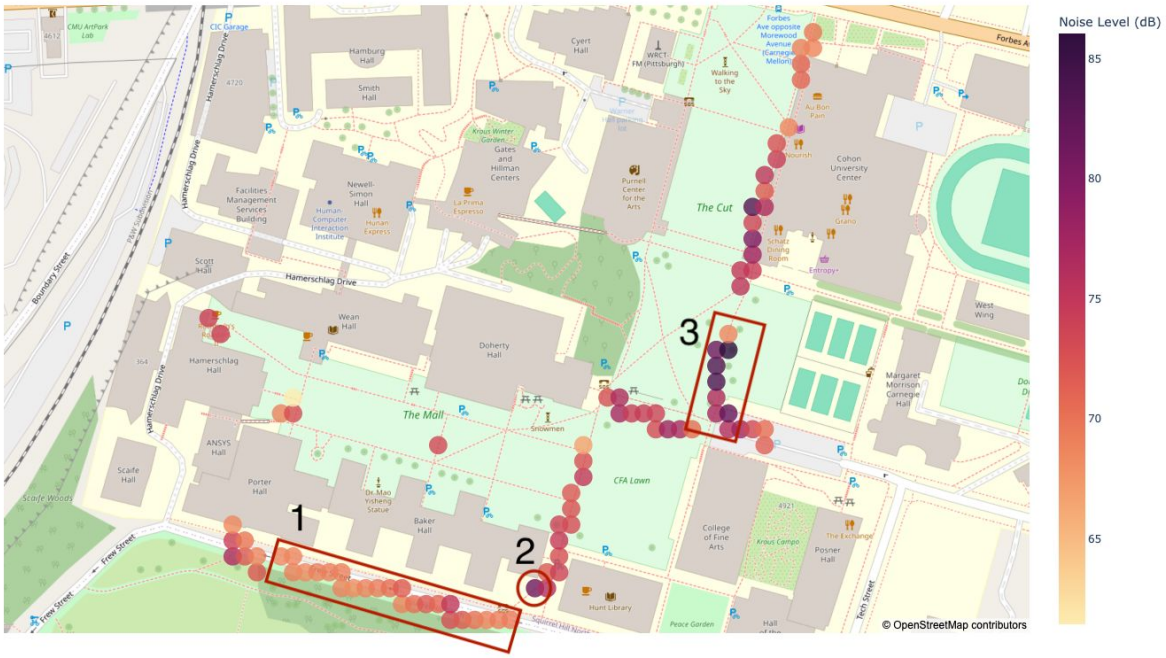

Figure 4. The trajectory of our measurement system across CMU's campus on December 7th and December 8th, 2023

Figure 4 shows the noise levels we collected as we walked through the campus. We took specific observations at three points along our path to validate our results. In section 1, we moved along the relatively smooth sidewalk alongside Frew Street at a constant walking speed. Frew Street was observed to have relatively low foot and vehicle traffic, resulting in low sound levels. These observations are corroborated with the results displayed in the map, which show consistently low levels (below 70 dB) along the entirety of section 1. In section 2 we intentionally held the measurement device in place and played loud music directly into the noise sensor. This was intended to display a deliberately high measurement on the map to validate it could register and display high noise levels. Our test was corroborated, as this point does display a higher median sound level than many of the surrounding points. In section 3 we observed that our cart was moving over a very rough surface (a sidewalk in need of repair) and was therefore making a significant amount of noise directly below the noise sensor. The elevated readings in this area validate our observations in this section.

Conclusion and Future Work

Based on our experimentation and validation, our prototype was successful in measuring differences in sound levels across an area about the size of a residential neighborhood. We were able to observe changes in the reading of our noise sensors that corresponded with the relative loudness or softness of the local noise levels. The interactive map was able to display this information to the user in a way that was easy to understand and draw further conclusions from.

Areas with high noise levels are clearly visible on the map, indicating that with more data these locations would be apparent to end users of the sensor devices.

To continue this work in the future, we would first develop a database outside of python using an MQTT (Message Queuing Telemetry Transport) server that could store data sent via Wi-Fi or the internet. This would allow the sensor devices to be totally separated from a computer or laptop, making them significantly easier to user and move. However, this would likely require additional power to the sensors and a connection to the internet. We would configure the sensor to connect to Wi-Fi if available and send information automatically. If no internet connection was available, the sensor would temporarily store readings until a connection could be made.

In addition to making our system easier to use, we would further develop the information storage and retrieval systems to allow more insights to be drawn. For example, we would use the stored timestamp information to group noise level readings both spatially and temporally, allowing a user to look for noise levels in an area at specific times of the day. The exact time range that would be grouped together may shift with the amount of data available to ensure the data quality is high. For locations with enough noise measurements, granularity of 1 hour or even 15 minutes may be possible. Areas without enough measurements may not display meaningful information at this level of detail, so a warning to a user about the data quality may be necessary to display.

While testing our current prototype on the sidewalks surrounding Carnegie Mellon University, we observed elevated readings while traversing over pavement that was older and rougher than newer sidewalk. We suspect that these readings were a result of the noise from the cart that our prototype rested one. With these observations in mind, we also propose that our prototype, with some small modifications and further testing, could be used to measure the smoothness or roughness of a pavement surface. For this use case we would require the cart to move at a constant velocity over the surface, and for there to be minimal surrounding noise to interfere with the readings. Elevated readings would likely indicate a rough surface, causing higher vibrations in the cart and therefore higher noise to be read by the sound sensor. Smooth surfaces would likely register low readings since there would be less vibration and noise from the cart. As we had limited time to design and test our prototype, we left this use case explored no further. However, this remains another potential application of the combination of noise and location sensors.

References

Boeing, G. (2017). OSMNX: New methods for acquiring, constructing, analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*, *65*, 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004

Centers for Disease Control and Prevention. (2018b, December 11). *Loud noise can cause hearing loss*. Centers for Disease Control and Prevention. https://www.cdc.gov/nceh/hearing_loss/default.html

Columbus P-7 Pro submeter (0.5 meter accuracy) USB and Bluetooth GNSS receiver. GPSWebShop. (n.d.). https://gpswebshop.com/products/columbus-p-7-pro-submeter-0-5-meter-accuracy-usb-and-bluetooth-gnss-receiver

Electret Condenser Microphones Explained. Alan Butcher Components. (n.d.). https://www.abcomponents.co.uk/electret-condenser-microphones/

Meet Speck. Speck by Airviz. (n.d.). https://www.specksensor.com/

Passchier-Vermeer, W., & Passchier, W. F. (2000). Noise exposure and public health. *Environmental Health Perspectives*, *108*(suppl 1), 123–131. https://doi.org/10.1289/ehp.00108s1123

Sound Intensity. HyperPhysics Concepts. (n.d.). http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/intens.html

Sygrove, C. (2023, January 2). *Decibel Chart: All You Need to Know*. MDHearing. https://www.mdhearingaid.com/blog/decibel-chart/

Taidacent Industrial Noise Decibel Detection Module Sound Sensor to Measure Sound Levels in Decibels RS485 5V Uart Dust Noise Meter. Amazon.com. (n.d.). https://www.amazon.com/Taidacent-Industrial-Decibel-Detection-Measurement/dp/B07QMLYV5B

US Department of Transportation. (2017, August 24). *Highway Traffic Noise Analysis and Abatement Policy and Guidance*. FHWA. https://www.fhwa.dot.gov/environMent/noise/regulations_and_guidance/polguide/polguide01.cfm

Appendix

Appendix 1: Python Code

Project_Store_Data.py

```
import atexit
import serial
import sqlite3
from pynmeagps import NMEAReader


# python ≥3.0
def only_numerics(seq):
    seq_type = type(seq)
    return seq_type().join(filter(seq_type.isdigit, seq))


def commitsqlatexit():
    con.commit()
    con.close()


# Set up Database
con = sqlite3.connect('Noise and Position.db')
atexit.register(commitsqlatexit)

# Create cursor to interact with the database
cursor = con.cursor()

# Creating table for noise levels and position
cursor.execute("""
        CREATE TABLE IF NOT EXISTS noise
        (
            id INTEGER PRIMARY KEY,
            timestamp TEXT,
            longitude REAL,
            latitude REAL,
            noise REAL
        )
        """)

ser = serial.Serial("/dev/cu.SLAB_USBtoUART", 57600)
ser2 = serial.Serial("/dev/cu.usbmodem2101", 115200)

gnrmc = 'GNRMC'

id = 0

while True:
    nmr = NMEAReader(ser)
    (raw_data, parsed_data) = nmr.read()
    noise_raw = ser2.readline()
    noise_data = noise_raw.decode().strip()
    if gnrmc in str(parsed_data):
        id = id + 1
        lat = parsed_data.lat
        long = parsed_data.lon
```

```
        print(lat, long, noise_data)
        if lat and long and noise_data:
            time = str(parsed_data.time)
            date = str(parsed_data.date)
            dateAndTime = date + '_' + time
            all_data = [(dateAndTime, long, lat, noise_data)]
            cursor.executemany('INSERT INTO noise (timestamp, longitude,
latitude, noise) VALUES (?, ?, ?, ?)',
                               all_data)
```

Project_Load_Data.py

```
import sqlite3
import plotly.express as px
import pandas as pd
import numpy as np


# Connect to Database
filename = "Noise and Position.db"
con = sqlite3.connect(filename)
cursor = con.cursor()

# Print Data
sql = 'SELECT timestamp, longitude, latitude, noise FROM noise'
cursor.execute(sql)
con.close()

# The original code within the Raspberry Pi Pico incorrectly converted the
reading by dividing by 63353
# The correct factor to divide by is 65535, which is corrected here:
multFactor = 63353 / 65535

# The original code within the Raspberry Pi Pico incorrectly did not
# convert based on the output voltage of 3 V and input voltage of 3.3 V
# The correct factor to multiply by is 3.3/3, which is corrected here:
voltFactor = 3.3 / 3

# Conversion from decimal degrees to 10 meters
gridFactor = 11113.9
# Other constants for the map
markSize = 5
latFactor = 90
longFactor = 180

# try:
cnn = sqlite3.connect(filename)
df = pd.read_sql(sql, cnn)

df['noise'] = df['noise'] * multFactor * voltFactor

# Remove rows with potentially bad readings
# (close to 30, indicating the sensor was likely disconnected and reading 0)
df = df[df['noise'] >= 33]

to_col = lambda x: np.floor((x + latFactor) * gridFactor).astype(int)
```

```
to_row = lambda x: np.floor((x + longFactor) * gridFactor).astype(int)

# Map the latitudes to columns
# Map the longitudes to rows
df['col'] = df.latitude.map(to_col)
df['row'] = df.longitude.map(to_row)
df['longGrid'] = (df['row'] / gridFactor) - longFactor
df['latGrid'] = (df['col'] / gridFactor) - latFactor
df['size'] = markSize

# Group by latitude grid cell and longitude grid cell, then take the median
value
df_grouped = df.groupby(['longGrid', 'latGrid', 'size'],

as_index=False)['noise'].median().rename(columns={'noise': 'Noise Level
(dB)'})

# Reshape dataframe
df_grouped = df_grouped.reset_index()


# Find the max and min lat and long to define map boundaries and calculate
zoom
x1 = min(df['longitude'])
x2 = max(df['longitude'])
y1 = min(df['latitude'])
y2 = max(df['latitude'])

# Calculate center point of the map
center = dict(lon=(x1 + (0.5 * (x2 - x1))), lat=(y1 + (0.5 * (y2 - y1))))

# Calculate zoom level
max_bound = max(abs(x2 - x1), abs(y2 - y1)) * 111
zoom = 16 - np.log(max_bound)

# Plot points on map
fig = px.scatter_mapbox(df_grouped, lon=df_grouped['longGrid'],
lat=df_grouped['latGrid'], zoom=zoom,
                        color=df_grouped['Noise Level (dB)'],
size=df_grouped['size'], opacity=0.8, size_max=15,
                        color_continuous_scale='matter', center=center)

fig.update_traces()
fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(showlegend=False)
fig.show()
cnn.close()
```

Appendix 2: Raspberry Pi Pico Code

main.py

```
from machine import ADC, Pin
import utime

sound_pin = machine.ADC(26)
led = Pin('LED', Pin.OUT)
led(1)

while True:
    read_value = sound_pin.read_u16()
    decibels = 30 + ((read_value/65535)*90)
    print(decibels)
    utime.sleep(0.1)
```