



NOISETRACKR

4th June Report

Agenda



Motivation and Goals



Architecture



Service Structure



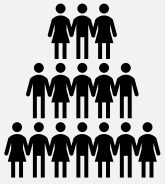
Development Plan



Showcase of Prototype



Introduction to the Problem



Touristic cities and municipalities such as Venice tend to suffer from overtourism and consequentially from **environmental problems**. Currently, they look for solutions such as a entering fees.

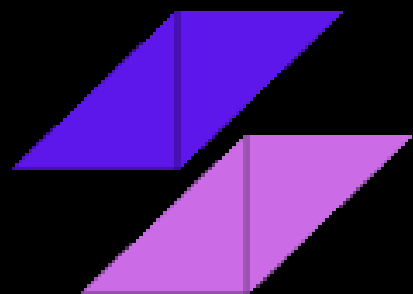


Cities such as Memphis suffer from **gun violence** and **gang related** incidents. Police and Forensic personnel would like to have information such as when a gun was fired for investigation purposes.



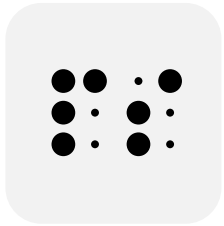
Housemasters and Universities would like to **identify which** blocks or apartments are causing noise and at **what time** of the day.

Solution: NoiseTrackr



NOISETRACKR

Goal



An application which allows users to analyze and monitor the **amount** of noise generated noise at **selected regions** within a given **timeframe**.



In addition to previous knowing the past and the present, users are also given **noise forecast** for the selected regions.

ForgeAI – Requirements of MVP


 Users receive noise insights through **graphs** and **time-series** + **bar charts**.

 Users can select the **timeframe** for their analysis.,

 Users may select **one or multiple** regions at a time for analysis.

 Users have access to ML methods to **forecast future noise** in the selected regions.

 Users can be alerted if a selected region is **exceeds** a predefined noise level.

 The IoT devices should use **LoraWAN**.

Architecture of Forge AI



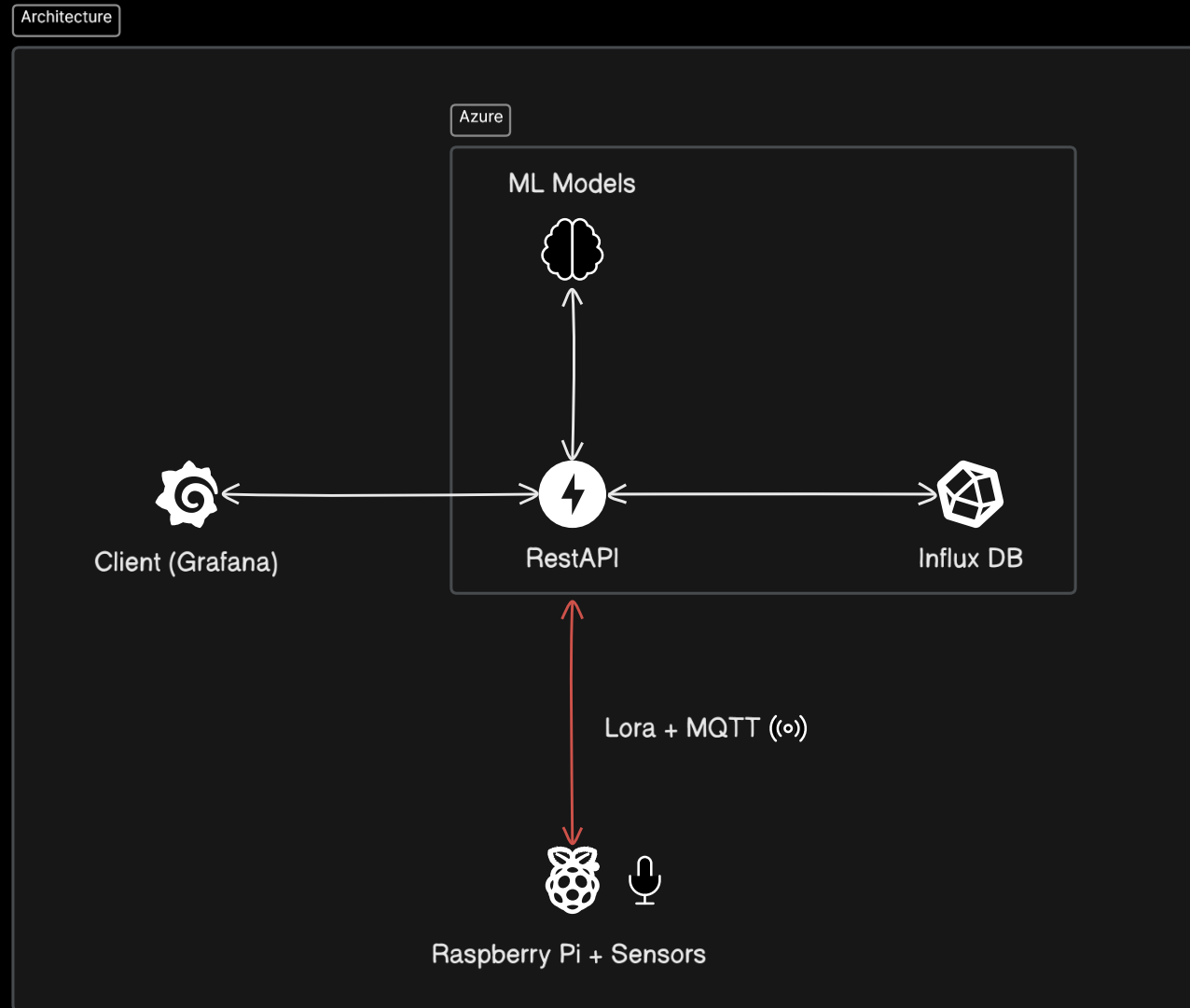
We use [Lora Sensors](#), [LoraWAN](#) and [Azure](#).

LoraWAN offers [low power consumption](#) and the ability to deploy within regions with [low signal reception](#).

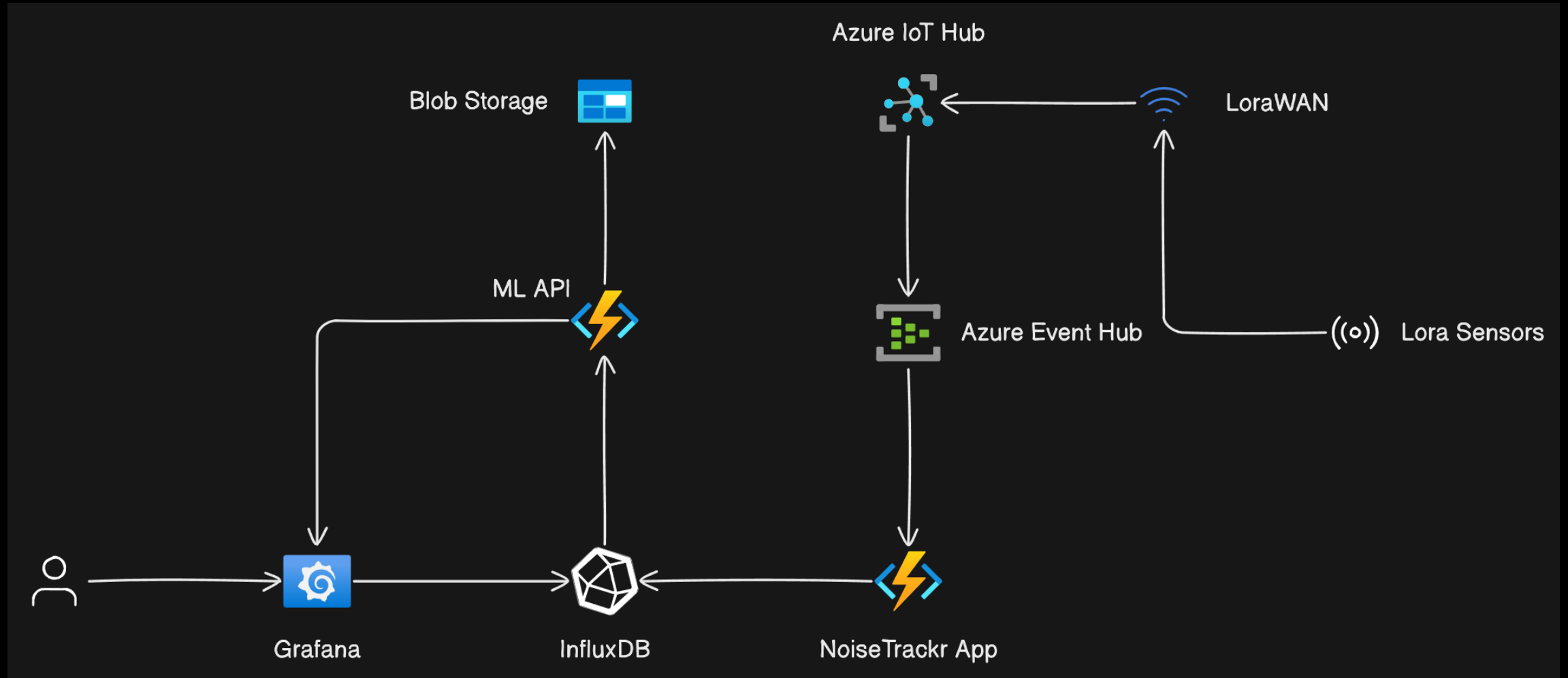
Lora sensors generally have a lower energy consumption than using microcontrollers such as [raspberry pi](#). + It is overall [plug and play](#).

Azure is used to handle the connection of multiple IoT devices and machine learning models [efficiently](#). Furthermore, it holds the [Grafana instance](#).

The previous architecture



The new architecture



The Lora Sensor

Datasheet
Publish Date: 21.08.2019



ERS Sound

 LoRaWAN[™] Wireless Sensor

Description

ERS Sound is an advanced indoor environment sensor. It is enclosed in a room sensor box and is designed to be wall mounted. ERS Sound is completely wireless and powered by two 3.6V AA lithium batteries. Inside you will find internal sensors for measuring sound level, temperature, humidity, light, and motion. The sound module is always on and will detect every sound event.



Applications

- Indoor environment measuring
- Smart buildings
- Workplace management
- Room occupancy
- Sound level control in public spaces, schools, libraries etc.

Product features

- LoRaWAN Certified ^{CM}
- Sound sensor / Peak + Average
- Always on – no missed sound events
- dBa filtering
- Temperature sensor
- Humidity sensor

Reasons for switching to **Function Apps**



Performance

Function apps are **faster** than FastAPI when sending data or fetching the history.



Regularity

Might as well stick to Azure since we're already **using most services** such as Queue, Blob, Storage, and Video Indexer.



Safety

Azure IoT Hub creates connections with devices more **reliably** and **securely**.

How it works – Sending Data to Influx DB



The IoT devices (Arduino) capture data using a noise sensor. When turned on, they constantly send data to Azure, activating an [EventHubTrigger](#).

It wraps all the data from the device + Azure IoT Hub and sends it into the InfluxDB.

```
# Combine all extracted data
influx_data = {
    'measurement': device_id,
    'time': received_at,
    'device_id': device_id,
    'application_id': application_id,
    'received_at': received_at,
    'gateway_id': gateway_id,
    **decoded_payload # Include all decoded payload fields
}

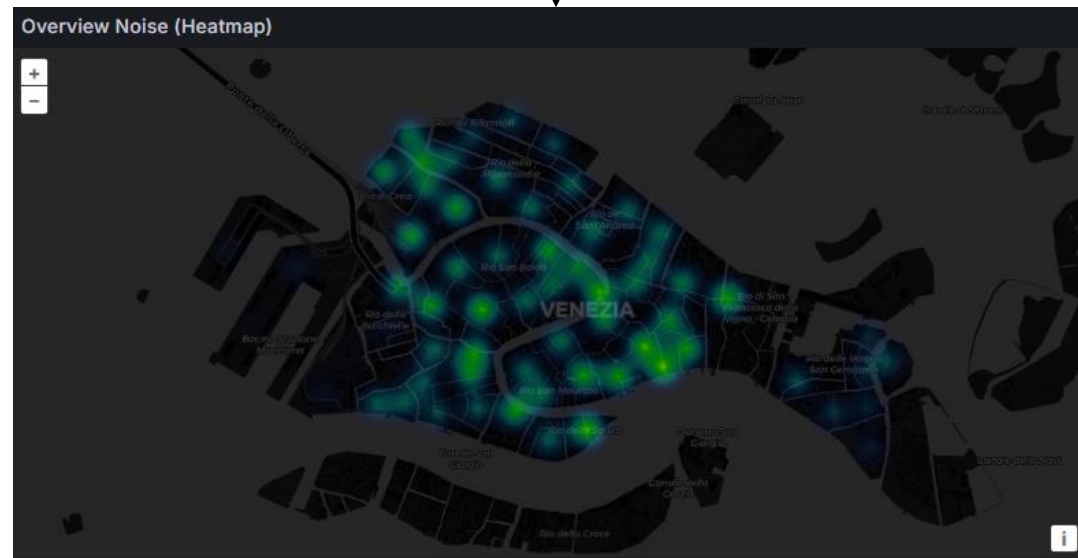
try:
    Datasender(influx_data).start(10.0)
```

How it works – From InfluxDB to Grafana



Data from InfluxDB is queried using **FluxQL**. Grafana then processes this data and displays the graphs.

```
from(bucket: "noisetrackr")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => contains(set: ${Devices:json} , value: r["location"]))
  |> mean()
  |> group()
```



How it works – Sending Data to Influx DB



The IoT devices (Arduino) capture data using a noise sensor. When turned on, they constantly send data to Azure, activating an [EventHubTrigger](#).

It wraps all the data from the device + Azure IoT Hub and sends it into the InfluxDB.

```
# Combine all extracted data
influx_data = {
    'measurement': device_id,
    'time': received_at,
    'device_id': device_id,
    'application_id': application_id,
    'received_at': received_at,
    'gateway_id': gateway_id,
    **decoded_payload # Include all decoded payload fields
}

try:
    Datasender(influx_data).start(10.0)
```

Development Plan

100%

Development of Database and DB Schema + Queries

Gabriel Witte

50%

Implementation of Machine Learning methods

Warren

75%

Gitlab CI/CD

Victor

100%

Setup of IoT devices and communication with backend

Luiza

75%

Development of Grafana frontend

Saruni

50%

Development of Grafana alerts

Saruni

100%

Development of Backend

Victor

Let's try it out!

Showcase of Prototype

Let's try it out!