user documentation

# abnormal activity detection using ambient sound

an IoT-based solution

team 34, ESW

# Contents

# 1. Overview

This is an IoT-based solution for detecting probable abnormal activities in otherwise unmonitored environments, particularly signs of distress, such as characteristic high amplitude sounds that last for an extended duration.

Typical use cases involve installing the device in poorly lit, less accessible and rarely visited public places, such as deserted roads, fields and bridges. The device can also be used as part of a home security system, to detect attempted break-ins at night for example. For a detailed overview on use cases, see *Appendix-1*.

The system consists of the on-site device, our servers and agents for delivering alerts. The device collects ambient sound data, sending it to our servers, where it is analysed in real time for abnormalities and outlying data points. In case something out-of-the-ordinary is detected, alerts can be delivered via several means, including a hooter on the device itself, text messages via a bot on the Telegram instant messaging application and a live dashboard.

# 2. What's in the box

The device consists of the following components:

- ESP32-WROOM-32D Microcontroller
- Max 4466 Microphone module
- Piezoelectric Hooter
- Power Reserve

which are enclosed in a custom-designed, 3D-printed enclosure to keep the elements out and the external appearance clean and aesthetically pleasing.

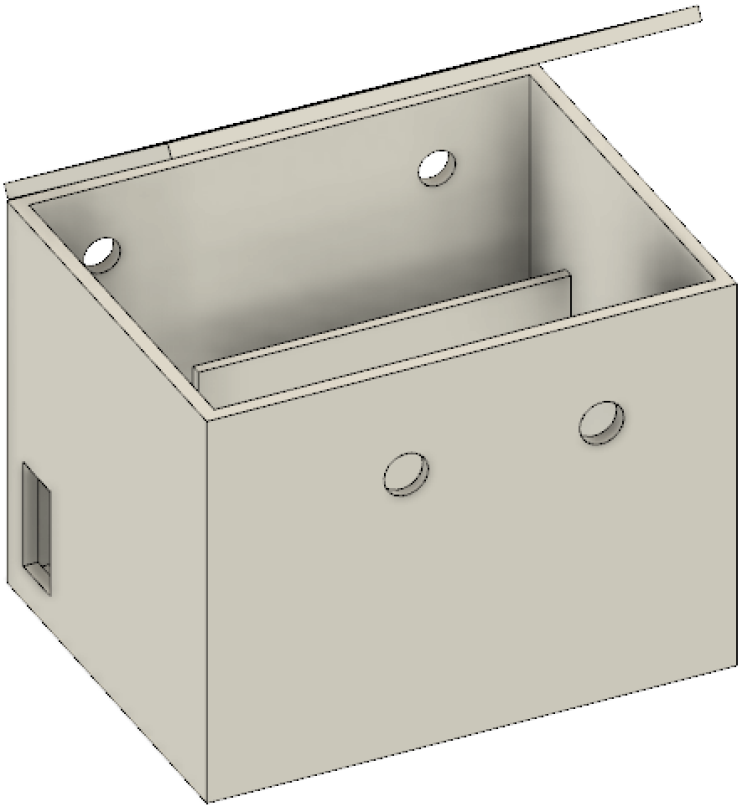


**Figure 1** The device used for collecting ambient sound data

## 2.1. The Microcontroller

An ESP32-WROOM-32D microcontroller acts as the brain of the device. It can be found attached to the base of the device. It is a low-power system-on-a-chip microcontroller with integrated WiFi connectivity.

This allows the device to run on a miniscule amount of power, improving efficiency and decreasing thermal emissions, which leads to longevity.

The integrated WiFi is used to send the recorded data to our servers every 3 seconds for analysis.

## 2.2. The Microphone Module

The device uses a Max 4466 microphone module to record ambient sound data. Measurements are taken approximately every 2.5 ms, and consist of the relative ambient sound intensity values.

For privacy concerns related to recording ambient sound data, see *Appendix-2*.

## 2.3. The Hooter Module

The device comes with a built-in piezoelectric hooter to raise alerts. It is capable of generating an alert tone of 4.0 ± 0.5 kHz at upto 85 dBs.

In case the use case requires, the hooter can be disabled using our mobile application.

## 2.4. The Power Reserve

The device is equipped with a 5000 mAh power reserve to keep it running in case of a power failure. However, this requires that internet connectivity be available during this period.

The reserve is capable of powering the device for ~4 hours on a full charge, which takes ~3 hours.

# 3. Technical Specifications

## 3.1. Physical Specifications

- External Dimensions: 143 mm × 111 mm × 110 mm
- Operating temperature range: 0 °C – 50 °C

## 3.2. Electrical Specifications

- Input voltage: 5 V
- Input current: 2 A
- Battery type: Li-ion
- Battery capacity: 5000 mAh
- Charging time: ~3 hours
- Backup time: ~4 hours

## 3.3. Operational Specifications

- Sampling frequency: 400 Hz
- Data transfer frequency: 0.33 Hz
- Sensitive audio frequency range: 20 Hz – 20 kHz
- Hooter output frequency: 4.0 ± 0.5 kHz

- Hooter output amplitude: ~85 dB
- Wireless technologies: WLAN, oneM2M
- Operating system: freeRTOS

# 4. Operating Instructions

## 4.1. Installation

The device is ready-to-use, and only needs a 5V/2A power supply via a standard Micro-USB (type B) port, and internet connectivity via WLAN.

1. Place the device in a suitable location. The mounting holes on the back can be used for vertical/wall mounts.
2. Connect and turn on the power supply.
3. The device will automatically scan and connect to available WiFi networks. Once connection to our cloud is established, an alert will be sent via our Telegram Bot.

The device, being an embedded system, requires no user intervention to operate once setup has been completed.

## 4.2. Using the Telegram Bot

Instant messaging has taken the world by storm, and has emperically proven to be one of the quickest methods to deliver small pieces of important information. This is why our system provides the option of receiving instant alerts as soon as an anomalous activity is detected via the Telegram instant messaging application.

As soon as an anomaly is detected, the bot sends a timestamped alert with a graph of the recorded data with the anomaly region demarcated.

The bot also sends an alert if our servers receive no data from the device for over 10 seconds, a probable indication that something is wrong with the device.

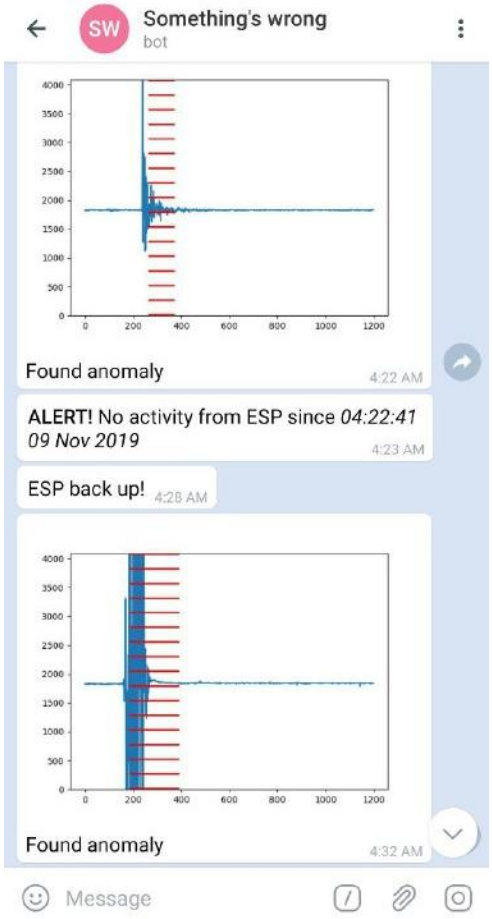



**Figure 2** Receiving alerts via the Telegram bot

To set up the Telegram bot:

1. Create and account and register on Telegram
2. Subscribe to the bot with your chat ID

## 4.3. Using the Live Dashboard

The live dashboard is a web-based application that allows for continuous monitoring of the data. A live graph pictorially depicts the current situation, while weekly metrics offer useful insights. Any anomalies being detected triggers alerts on the dashboard. To view the dashboard, visit our website from your computer or mobile device.

We recommend displaying the dashboard in the security control centre of your premises for continuous monitoring.

## 4.4. Using the Mobile Application

The mobile application is meant to be used as a controller for the device. This allows for the hooter to be disabled if required. Note that even with the hooter disabled, anomaly detection works normally, and alerts will continue to be delivered through all other means.

To use the Mobile application, install the APK bundled with the device. It is already registered to that particular device. The easy-to-use interface gives the option of enabling or disabling the hooter.

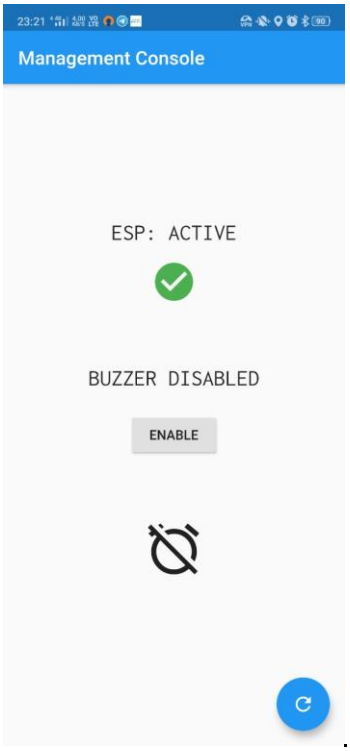



**Figure 3** The Mobile application

# 5. Behind the scenes

Our system is an integrated end-to-end solution to the problem of monitoring otherwise unmonitored areas. This inevitably implies that it involves a complex interplay between several different technologies and platforms, however, here's a brief overview of what goes on behind the scenes to keep your premises safe.

## 5.1. The Base State

The system constantly samples the ambient sound and sends the data to the server for real-time analysis. For as long as the diagnosis is negative, no action is taken. This is known as the *Base State* of the system.

### On the Device

The device samples the ambient audio every 2.5 ms. Every 3 s, the data gathered is compressed and uploaded to our servers. The device then waits for a response, which is usually instantaneous.

### On the Cloud

The data collected from the device is analysed using a *sliding window* algorithm that looks out for significant deviations from the mean that last longer than a certain duration. This allows us to compensate for inherently loud environments and reduce the number of false positives. It is only if an anomaly is diagnosed that further action is taken.

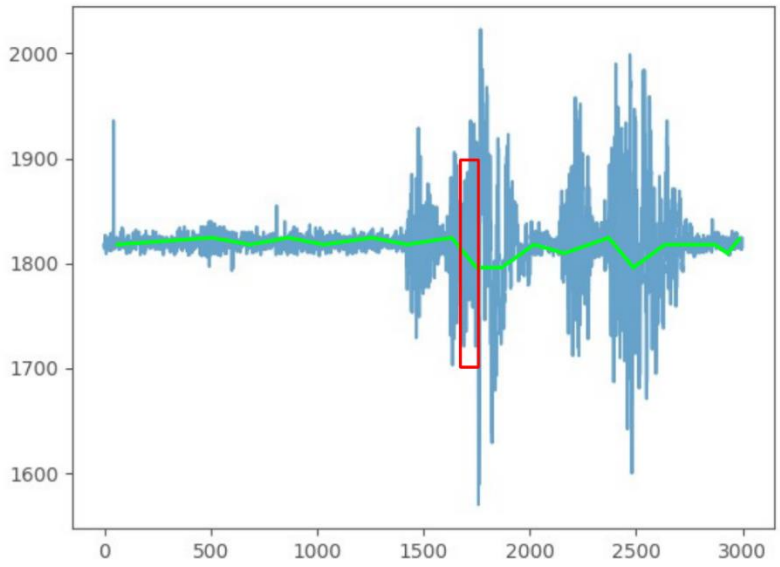



**Figure 4** The sliding window algorithm

The cloud also powers the live dashboard by providing data and statistical analysis.

## 5.2. The Active State

The system switches to the *Active State* as soon as the algorithm detects an anomaly that requires attention.

### On the Cloud

The cloud is where the proceedings of the active state are initiated. This involves sending a response to the waiting device to sound the hooter (provided it hasn't been disabled), computing, rendering and sending the graph via Telegram along with an alert and pushing an alert to the live dashboard. These tasks are performed parallelly to keep the response times low.

### On the Device

As soon as it receives a positive response, the device sounds the hooter continuously for 2 seconds before it resumes collecting and uploading data to continue keeping a watch.

# 6. Appendix-1: Use Cases

Situations of distress are fairly common in times when, and at places where, there is nobody around to help, such as at night, in places obscured from general view.

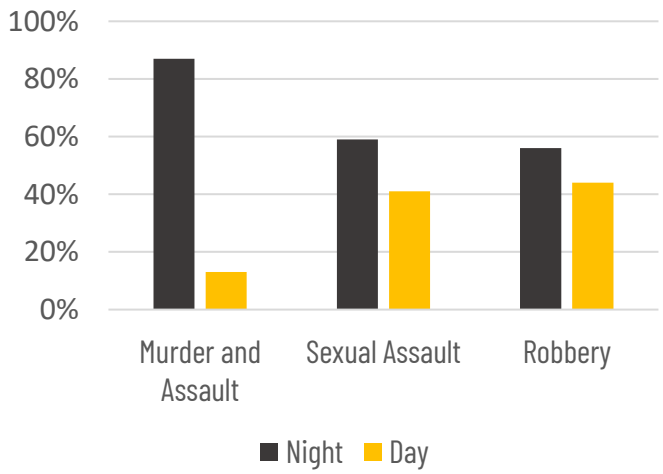



**Figure 5** Percentage of crimes in public places categorised by time of day. Source: FBI UCR 2017

These situations could range from sudden medical emergencies such as a cardiac arrest to instances of crime such as assaults, robbery, crimes against women or even animal attacks.

Naturally, the instinctive response of the victim in these scenarios is to shout for help, however the calls often go unheard. It is this gap that this system was designed to bridge. It needs no extra actuation – it acts on the first instinctive

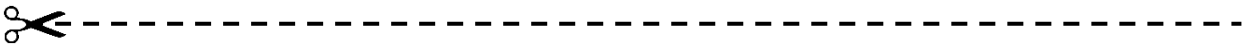
response and connectivity to the internet means alerts can be sent instantly to nearby security agencies. The use of audio data, as opposed to video, allows the device to function on low power, consume less network bandwidth, and makes the analysis of the collected data easier and faster.

# 7. Appendix-2: Privacy

We understand that the data we collect is sensitive and we take safeguarding it very seriously.

It starts with *what* data we collect. The device only samples the ambient sound *intensity* levels at a relative low frequency. This is just a string of 400 numbers per second, and is *insufficient* to generate back the sounds. This is by design, as part of our commitment to collect only the minimum we require to power our services, and also helps us keep the bandwidth requirements low, allowing the device to function in areas of poor network coverage. The data is then stored on secure servers and is only used to show insights and improve our algorithms.

We recommend a notice such as this one be prominently displayed next to the device:



## NOTICE

This device is part of a security system and monitors the ambient sounds. The sound data being collected is only used for analysis and is insufficient to generate the sound wave back.

In case you have any further concerns, kindly contact *<contact information>* .

developer documentation

# abnormal activity detection using ambient sound

an IoT-based solution

team 34, ESW

# Contents

# 1. Abstract

Abnormal activities and situations of distress are fairly common in times when, and at places where, there is nobody around to help, such as at night. These situations could range from sudden medical emergencies such as a cardiac arrest to instances of crime such as murder, robbery and sexual assault, or even animal attacks. Helpless situations like these are where our device is of service. The device does not need any additional actuation, as it detects one of the most widespread signals of distress, screams and shouts. Any positive identification causes the device to sound an alarm locally, and connectivity to the internet means an alert can be sent to the relevant safety agencies. The device is small, low-power consuming and can be deployed in remote places.

This document aims to detail the device and its operation. In the developer document, we describe the project in a developer-oriented manner. This guide would help a developer get started with setting up a similar device in their region too, effectively as well as correctly. In the user document, we describe the device with the user in mind, and how they should operate the device at their place. The robustness and simplicity of the device ensures that troubleshooting the device is as simple as switching it on and off again, so the document mainly focuses on a layman view of approaching the device. In the IoT projects component document, we describe all the major components that we have used in this device for completeness as well as to make sure there is a common page for reference.

Keywords: review, remote devices, hazard prevention, user guidelines, audio analytics, scream detection

# 2. Introduction

## 2.1. Problem Statement and Requirements

Situations of distress commonly occur at times, and in places, where there is nobody around to help, such as at night. These situations could range from sudden medical emergencies such as a cardiac arrest to instances of crimes such as murder, assault, crimes against women or even animal attacks.

Naturally, the instinctive reaction of the victim is to shout for help, but because of the time and place, the calls often go unheard.

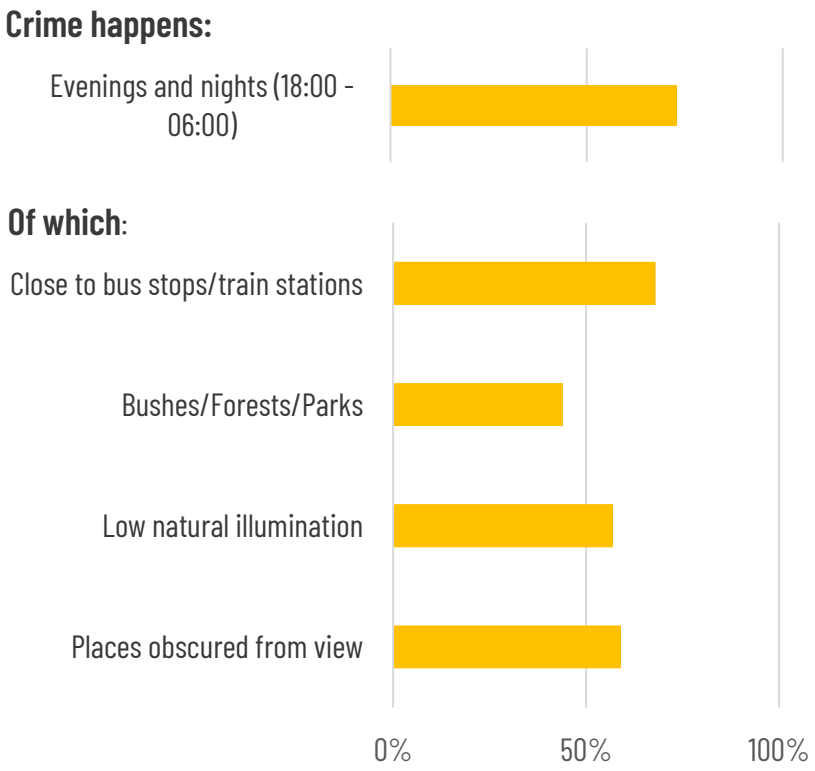



**Figure 1** Space-time signatures of risky places. Source: Journal of Environmental Psychology, 2014

This is especially relevant in the context of crimes against women, as a major question looms over the safety of our cities for women.

And it is helpless situations like these that we intend to improve by building a solution that satisfies the following requirements:

*Detect and act on any kind of abnormal activity to improve security*

- Should be able to **autonomously collect, analyse** and **categorise** the data as anomalous or normal in real time
- Should be able to **send alerts** and sound alarms if an anomaly is detected
- Should be an embedded system with **minimal power, bandwidth** and **maintenance** requirements
- Should be able to **statistically analyse** the data and display trends and insights

## 2.2. Purpose of the System

The main purpose of this system is to **improve security** in the premises where it is deployed, i.e. the IIIT campus to start with. While security cameras already

exist, they require constant monitoring, thereby increasing the man power required to ensure security and safety. This system intends to **augment the already existing security arrangements** by autonomously sensing for anomalous activity and reporting it to the personnel on duty, also **easing up their work load**.

## 2.3. Overview of the Solution

Our solution to the problem is an **IoT based system** consisting of an **embedded system, a central communication and processing server, a live dashboard web application, an mobile application** and a **Telegram bot**. The device collects ambient *sound* intensity data and relatively low frequencies and sends the data to a server for analysis. A *sliding window* algorithm is used to analyse the data on the server. In case an anomaly is detected, the device is instructed to sound a hooter and alerts are sent via the Telegram bot and displayed on the live dashboard. The mobile application can be used to control some aspects of the embedded device.

# 3. Design Aspects

## 3.1.System Specifications

In order to satisfy the requirements of the problem statement, and be a practical solution deployable in the real world, the system should possess the following properties:

- The system should be able to **detect anomalous activities** using certain signatures. It should **minimise the number of both false positives and negatives**.
- The system should be able to **function in all weather** and visibility conditions.
- Because this is a **security-critical application**, the system should work in **real time**, minimising the time taken from the occurrence of the abnormal activity to it being reported.
- The system should be **robust** and **self-diagnosing**. It should report the failure of any one particular part and try to isolate the damage, **degrading gracefully**.

As we shall see, our solution has been designed to fulfil all these specifications.

## 3.2. Stakeholders

The biggest stakeholders in this system are the residents of the premises where the solution is deployed. This is because it is their security and safety that hangs in the balance. The system should be so designed such that they can have confidence and faith in its capabilities. Other stakeholders include the security personnel who will benefit from the system and will be responsible for acting on the alerts. The owners of the premises where the system is installed also hold an important stake in the system because the safety of the people there is their responsibility.

## 3.3. Design Decisions

This section details the decisions that were taken while designing the building this system and the rationale behind them.

### Using ambient sounds

The first decision to be taken was the selection of the primary source of information about the surroundings.

The first obvious choice is live video. However, in the context of this system, it posed several challenges. Firstly, categorising incidents around as normal or abnormal is difficult as it would depend heavily on the type of activity going on, the perspective and viewing angle of the camera and the obstructions in between. This would adversely affect the number of false positives and negatives reported, and the system would fail to meet the specifications. Recording and transmitting video data would also require higher network bandwidth, again violating the problem statement requirements. Further, analysis of the recorded data using computer vision algorithms would be slow and processing intensive, thereby preventing the system from responding in real time. It might also raise privacy concerns.

This is why we decided to focus on ambient audio. This allows the system to react to the first instinctive reaction of every victim: the call for help. This simplifies the problem of data analysis and categorisation and allows the device to listen to activities all around, under all weather and visibility conditions. It also reduces the amount of data that needs to be transmitted over the network and speeds up analysis by the server – as the server can now

employ simpler machine learning or conventional algorithms to remove outliers and categorise the data. It will also most likely sound less intrusive to most people.

## How much data to collect

We decided against recording, transmitting and storing entire sound files in favour of only sampling the ambient audio intensity at set intervals. This was done to reduce the amount of data being collected and transmitted while maintaining an acceptable success rate. It also simplifies the processing and storage of data on the server and makes it conventionally impossible to recover the original sounds, respecting people's privacy.

## Component Selection

Having made these crucial decisions, next in the order of business was to select the various components that make up the embedded system.

ESP32-WROOM-32D was an easy choice for the microcontroller, because of the availability of an integrated ADC and Wi-Fi antenna. Its dual core architecture and power saving modes also help improve the efficiency of the system. The availability of software libraries and integration with the Arduino IDE simplify the programming process.

The MAX 4466 microphone was selected because of its integrated filtering and amplification circuitry. Its output is also readily available on sampling the corresponding pin, without requiring any libraries or communication protocols. This perfectly fits in with the design choice of only sampling the ambient sound intensity values at set intervals.

The piezoelectric hooter and the power reserve were selected based on their cost and working life.

## Server considerations

It was clear from the start that a server would be required to analyse the recorded data. This is because performing this computation on the microcontroller (which only has one core available for user processes) would block the processor and prevent the system from recording more data, leading to missed events and false negatives.

It was decided that a flask server running a flask web application would act as the central communication hub and data pool. This makes the system scalable and extendable, as several nodes can now send data to this server, and alerts can be delivered through several means, and new ones can be added.

While Python3 was the language of choice for the server because it is a high level language with several libraries, the algorithm and key sections which handle large amounts of data were implemented in C++ to improve performance. Multi-threading was extensively used in order to meet the real-time requirements.

## 3.4. System Entities

The system consists of the following entities:

- The *embedded system* equipped with sensors to detect abnormal activities and actuators to sound alarms
- The *central communication server* responsible for receiving, analysing and categorising the data as normal or anomalous and sending out alerts
- The *live dashboard web application* responsible for displaying live data, trends and insights and alerts in case an anomaly is detected
- The *Telegram bot notification system* responsible for displaying instant alerts to authorised and subscribed users.
- The *controller mobile application* which allows administrators to control certain functionality on the embedded system
- The *oneM2M server* which receives and logs timely reports about anomalies

### Entity Interaction

In order to achieve what the requirements and specifications demand, different components of the system depend on and interact with each other.

- *Embedded system and central server:*
  The embedded system communicates to the server over the internet, access to which is provided by the WLAN connection. This interaction consists of POST requests and responses being exchanged, allowing for bi-directional communication. The embedded system regularly sends

the collected data to the server for analysis, and the server responds with whether that data is anomalous or not.

- *Central server and live dashboard*
  The live dashboard constantly receives live data and results of the statistical analysis from the server for display. This is done by exchanging GET requests and responses.
- *Central server and Telegram bot notification system*
  The central server communicates to the Telegram API to send messages to authorised and subscribed personnel whenever an anomaly occurs or something in the system is not working as expected.
- *Central server and control Mobile application*
  The mobile application functions by exchanging GET requests and responses with the server. This allows the application to verify the status of the embedded system and request the server to make changes requested by the user. The server verifies that the requests are coming from legitimate and authorised users and takes action to execute them.
- *Embedded system and oneM2M server*
  The embedded system communicates with the oneM2M server over the internet, access to which is provided by the WLAN connection. The exchange consists of POST requests, in which a log indicating if some anomalies occurred in the past minute is sent to the oneM2M server.
- *The system and its users*
  The system interacts with its users, primarily the security personnel through the numerous user interfaces, namely the live dashboard, Telegram bot notification system and the control mobile application. These can be used to view insights, trends and live data, receive alerts if an abnormal activity is detected or something goes wrong in the system and to control the system.

## 3.5. Conceptual Flow

This section details the flow of events and cycle of states that the system loops through at a conceptual level.

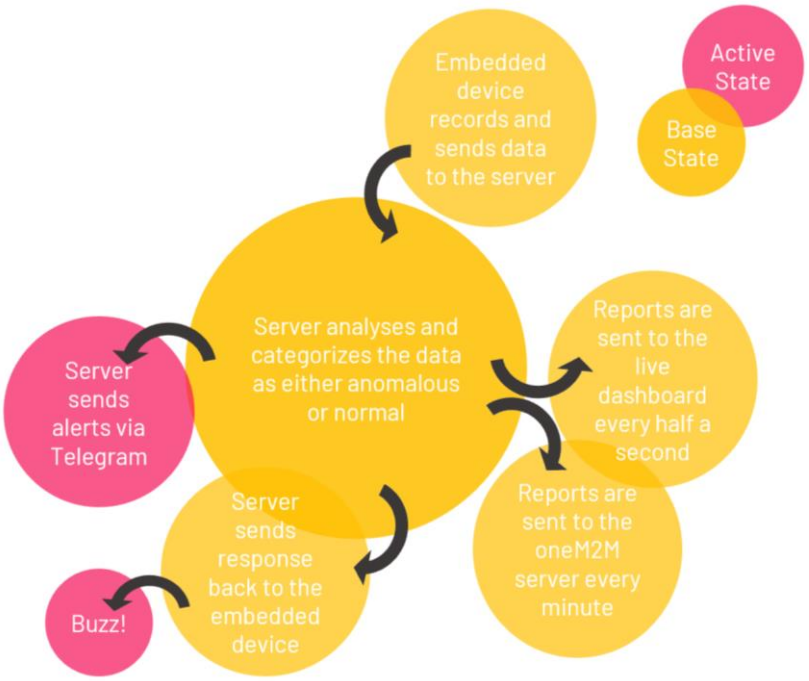



**Figure 2** Conceptual Flow

## It starts on the node

The node, on bootup connects to Wi-Fi and powers the microphone which constantly records the ambient audio intensity at 400 Hz. After recording for exactly 3 seconds, it compresses the recorded values as follows:

To remove the overhead of separators we make the values self-punctuating by restricting each value to exactly 3 digits. Because a 12-bit ADC corresponds to a range of 0-4095, all values less than 100 are left padded with 0's, while all values greater than 999 have their least significant bit stripped. This decreases the size of the network payload by 40%. This compression was necessary because the limited bandwidth of the provided network meant that the time taken to upload the data was otherwise prohibitively long. Because of the presence of only a single user-accessible core on the microcontroller, time spent uploading data is time spent not listening to the surroundings, which is not acceptable in a security-critical system.

The recorded values are then sent as a string via a POST request to the server. The node then synchronously waits for the server's response, a binary value indicating whether the sent data was anomalous or not, and sounds the alarm for 2 seconds if it was. Several optimizations on the server ensure that the response arrives almost instantaneously, even on limited bandwidth networks.

The node also communicates with the oneM2M server, sending it information about the activities in the last minute.

## It all comes together on the server

The server consists of a Flask backend, which is responsible for bringing all the different components together. First is the order of business is analysing the data being sent.

Upon receiving the values, it unpacks them and runs the analyser on this input, which reports if the values given to it seem anomalous or not. The server then performs the following tasks in separate threads to reduce the total time taken in a cycle:

- Send a response back to ESP
- Append the current values to the hourly log
- If an anomaly was detected, generate and send a graph containing the anomalous regions to the Telegram bot via the Telegram API

The flask server runs on a T-Mux session.

## The Analysing Algorithm

The algorithm was implemented in C++ for improved performance. The compiled binary is executed on the server.

The primary logic behind the algorithm is as follows:

- It keeps track of the running sum of a certain number of a previous values. The exact count is computed experimentally.
- Every new value, it is compared against the existing running average. If the deviation is beyond a certain absolute value, then that time instant is marked as probably anomalous. Again, the threshold is determined experimentally.
- If the number of anomalous values in a window exceeds a certain percentage, then the entire window is categorized as anomalous. The exact threshold is determined experimentally.

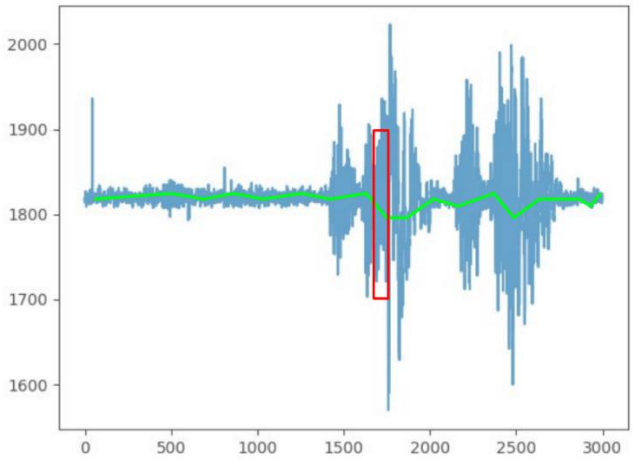



**Figure 3** The sliding window algorithm

Characteristics:

- *Outlier Detection and Removal*: The algorithm requires several values to be anomalous in order to characterize the data as abnormal. This prevents very short duration spikes and outliers from triggering the system, as they are probably caused due to erroneous sensor readings.

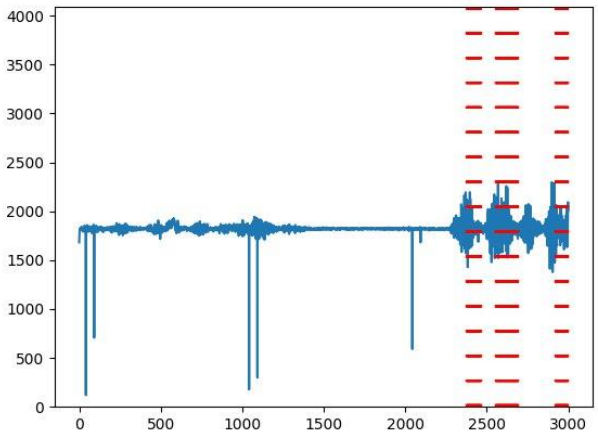



**Figure 4** Ignoring outliers

- *Adaptation:* The algorithm uses a running mean of a sliding window for comparison. This allows the algorithm to get adjusted to the surrounding noise. Thus, if the surroundings are inherently loud, the average is higher, and it takes a greater spike in intensity for the algorithm to characterize the data as abnormal. However when the device is placed in

normally quiet surroundings, even shorter spikes in intensity will be treated as abnormal.

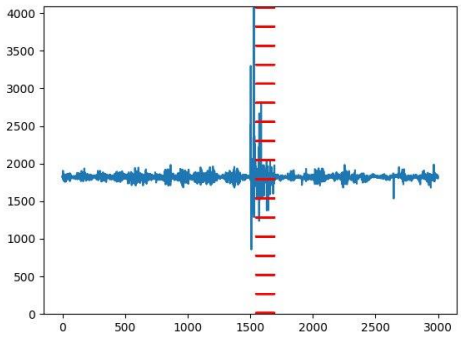



**Figure 5** Loud surroundings

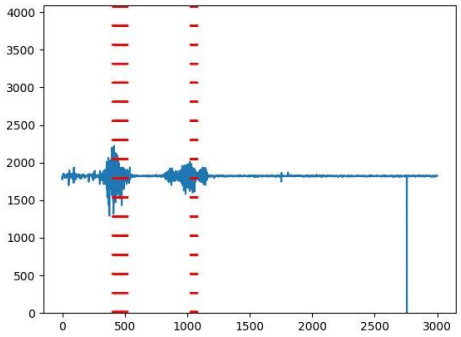



**Figure 6** Quiet surroundings

- *Envelope Detection:* Because the algorithm looks at the absolute difference between the mean and the current value, it treats values higher or lower than the mean equally. This essentially causes the algorithm to only look at the envelope of the curve, and not at the individual data points.

## Robustness is built in

The Linux utility *cron* is used to perform periodic checks and housekeeping activities on the server.

- We keep track of the latest activity from the node. Every minute, we check if this value is not more than 10 s (this margin was added to ensure that a slow network doesn't trigger this). If it is more than 10 s, a message is sent via the telegram bot which contains details like the last activity from ESP. Whenever the next activity is detected, it also sends a message that it has started working again.

- Every minute we check if the flask server itself is running or not. If not, the Telegram bot is used to send an alert. Also, when the flask server is back up again, it sends a message
- Every hour, the log file to which the flask server writes the values is renamed and moved to another location to keep the logs organized.

### The control Mobile application

An Mobile application is used to control the node. It allows disabling the hooter if required. It also displays the current status of ESP and flask server.

## 3.6. Data Analysis

Data analysis is an integral part of the design of this system. The algorithm described above, which lies at the heart of the system, analyses the data in real time to detect whether or not it is an abnormal activity which determines what the rest of the system does.

The system also computes several other global statistical parameters, such as the total number of anomalies today, the total number of anomalies this week, how anomalies this week compare with the last, the distribution of the anomalies through the day and throughout the week. These trends are then displayed on the live dashboard to provide a useful and actionable digest to the security personnel.

## 3.7. User Interface Design

The system is designed to be autonomous, keeping user interaction to a minimum. However, the authorised user is always in full control, and the interactions between the user and the system that do exist are designed to be beautiful, intuitive and easy to use. The live dashboard is a responsive and clean website built using HTML, CSS and JavaScript that shows all the key statistics and live data at a glance. Keeping up with this ideology, the control application is also designed to be simple and intuitive.

## 3.8. Operational Requirements

To function, the system needs the following:

- The server needs to be powered and running at all times

- The node needs a constant power supply. However, to ensure robustness, it has been equipped with a power reserve that can power it for ~4 hours
- The node needs to be connected to a wireless network in order to communicate with the server
- The node needs to be placed at an appropriate location, oriented such that the microphone and hooter are not obstructed

# 4. Technical Aspects

## 4.1. Hardware Specifications

As described above, the components that make up the embedded device were chosen after a careful comparison of their features. Listed below are their key technical specifications.

### Microcontroller: ESP32-WROOM-32D

- Core: ESP32-D0WD
- SPI: flash 32 Mbits, 3.3 V
- Crystal: 40 MHz
- Antenna: onboard antenna
- Dimensions: (Unit: mm) (18.00±0.10) × (25.50±0.10) × (3.10±0.10)

**Figure 7** ESP32-WROOM-32D

### Microphone: Max 4466

- Supply Voltage: +2.4V to +5.5V
- Power-Supply Rejection Ratio: 112dB

- Common-Mode Rejection Ratio: 126dB
- AVOL: 125dB (RL = 100kΩ)
- Outputs: Rail-to-Rail
- Supply Current: Low 24µA Quiescent


**Figure 8** MAX 4466 Microphone

## Piezoelectric Buzzer

- Pressure level: 72 min. 3Vp-p,2kHz,square wave,10cm dB
- Operating voltage range: 7 Vp-p
- Capacitance: 40 ±30% [120Hz] nF
- Operating Temp. Range: -20°C - +70°C

## Power Bank: Zinq ZN5KP

- Input voltage: 5 V
- Input current: 2 A
- Battery type: Li-ion
- Battery capacity: 5000 mAh
- Charging time: ~3 hours
- Backup time: ~4 hours
- Output voltage: 5 V
- Output current: 1-2 A

## Overall Technical Specifications

*Physical Specifications*

- External Dimensions: 143 mm × 111 mm × 110 mm
- Operating temperature range: 0 ˚C – 50 ˚C

*Electrical Specifications*

- Input voltage: 5 V
- Input current: 2 A
- Battery type: Li-ion
- Battery capacity: 5000 mAh
- Charging time: ~3 hours
- Backup time: ~4 hours

*Operational Specifications*

- Sampling frequency: 400 Hz
- Data transfer frequency: 0.33 Hz
- Sensitive audio frequency range: 20 Hz – 20 kHz
- Hooter output frequency: 4.0 ± 0.5 kHz
- Hooter output amplitude: ~85 dB
- Wireless technologies: WLAN, oneM2M
- Operating system: freeRTOS

## Connections and Interfacing

The various components are connected together as per the circuit diagram show below:

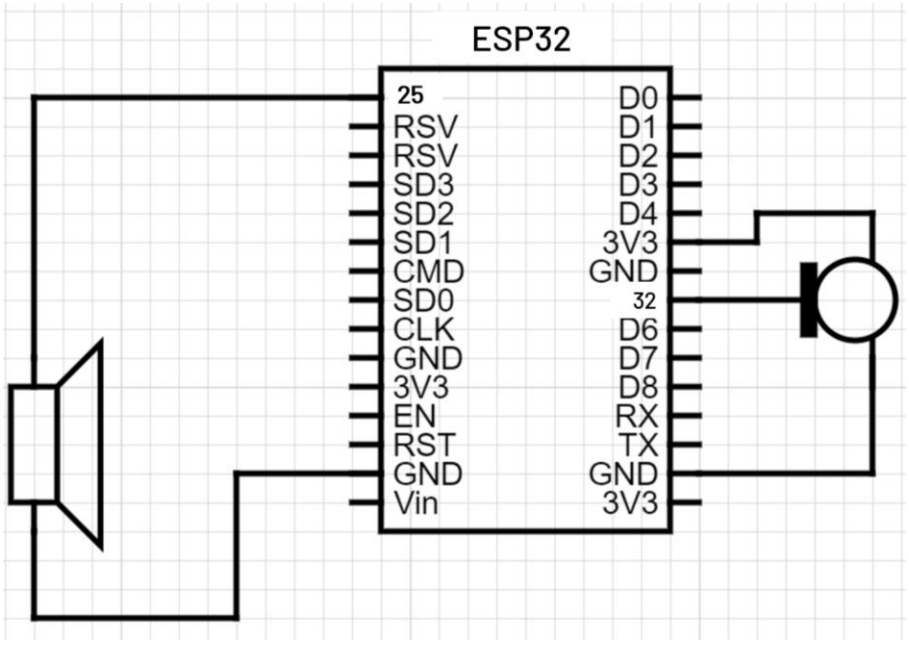



**Figure 9** Circuit Diagram

## 4.2. Communication

All communication is done using GET and POST requests over the internet. This includes communication with the oneM2M server as well as the central communication server. The communication happens as per the entity interactions described above.

### The Telegram API

A telegram bot was created using Telegram's own "BotFather". We got a unique id and an endpoint to which we can send POST requests with some data (text/images). On sending a message to the bot, the person receives a unique chat id, using which we can send the alerts to that person. We maintain a list of all interested stakeholders who will receive the alert.

Alerts are sent whenever one of the following events happen:

- An anomaly is detected
- The flask server goes down / starts working again
- The esp32 node goes offline / starts working again

## 4.3. Software Specifications

### The Flask Backend

*Technical Stack:*
Flask, Python3, Matplotlib, C++, *cron* utility

*Algorithm Specifications:*
Input frequency: 400 Hz, Window size: 100 values or 0.25 seconds, Threshold for anomaly categorization: ±100 units from the mean, Minimum anomaly duration: 24 values or 0.06 seconds

*General Specifications:*
Employs multithreading to improve response times and meet real-time deadlines

*Responsible for:*
Receiving data from the embedded device, decompression, outlier removal, analysis and categorization, sending response to the embedded device, sending

data to the live dashboard, sending alerts to the Telegram bot in case of an anomaly, data logging, statistical analysis, diagnostic checks and sending alerts to the Telegram Bot in case the embedded device or the flask web application is not functioning.

## The Live Dashboard

*Technical Stack:*
HTML, CSS, JavaScript, chart.js

*Specifications:*
Data is updated every 500 ms using AJAX requests

*Information displayed:*
Live data for the last 10 seconds, alert in case of an anomaly, number of anomalies in the current day and the current week, percentage change in the number of anomalies since last week, distribution of the anomalies of the current day across the day, distribution of the anomalies of the current week across the week, logs of all previously collected data.

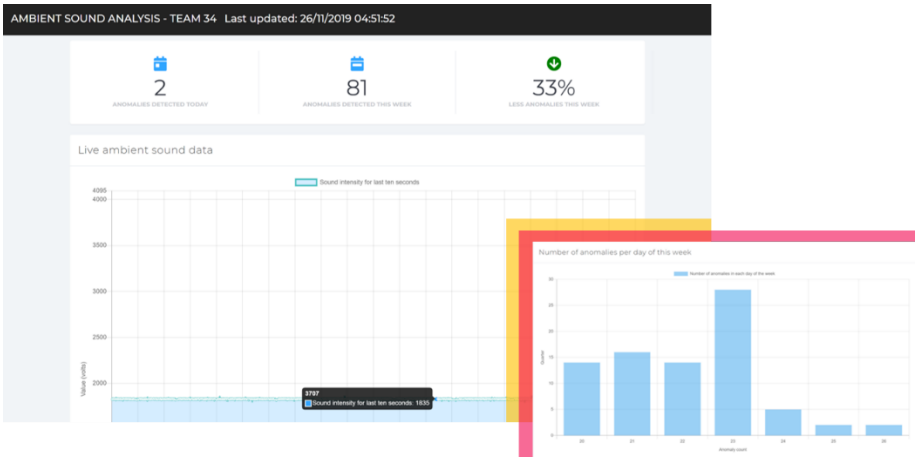



**Figure 10** Live Dashboard

## The Telegram Bot

*Technical Stack:*
Python3, Telegram API

*Information delivered:*
Alert along with a graph (with anomaly region marked) in case of an anomaly, alert whenever the embedded device stops communicating with the server, alert when the device starts functioning again, alert when the flask web application stops working, alert when it starts functioning again.

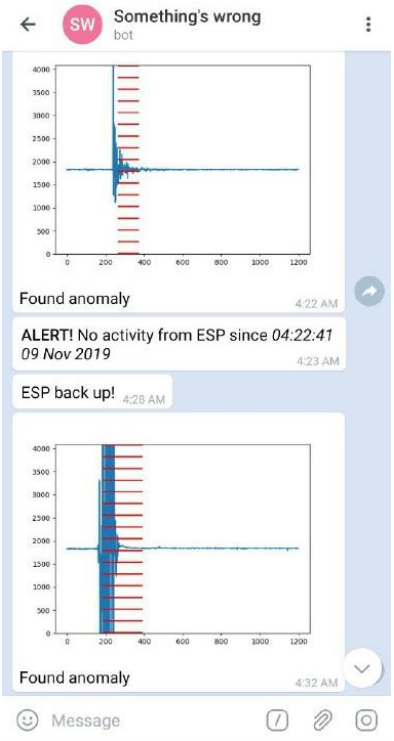



**Figure 11** The Telgram Bot

## The Mobile Application

*Technical Stack:*
Flutter, Dart Programming Language

*Platforms:*
Android, iOS

*Responsible for:*
Displaying the status of the embedded device (functioning or not) and giving the option of disabling and enabling the hooter as required.

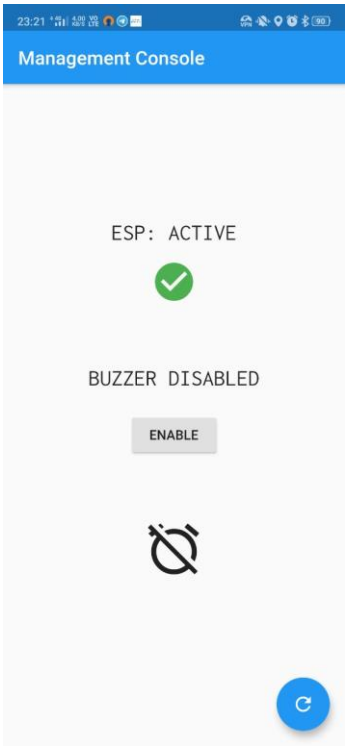



**Figure 12** The control mobile application

## Embedded Software

*Technical Stack:*
Arduino, C, ESP32 libraries

*Specifications:*
Recording frequency: 400 Hz continuously for 3 seconds, compresses by 40%, uploading blackout: <0.5 seconds, Data transfer frequency: once every 3 seconds, hooter duration: 2 seconds

*Responsible for:*
Recording ambient sounds, compressing them and sending them to the server, sounding the hooter in case of an anomaly

## OneM2M Server

*Technical Stack:*
oneM2M

*Specifications:*
Data is sent to the oneM2M server every 1 minute and gets stored in the container corresponding to that node

*Responsible for:*

Storing a per-minute log of whether or not an anomaly occurred in the last minute.

## 4.4. Data Handling Model

All data transfer happens using strings. The values sent by the node to the server are 3600-byte strings.

On the server, all data is stored and manipulated in ASCII-text files. This is because of the large volume of serial data, which makes other options such as a database inefficient.

## 4.5. Integration Framework

The Flask server integrates all the different components together. REST (Representational State Transfer) has been used throughout the system to permit inter-device communication using the server as a middleware. Every API call made by the microcontroller or the web or mobile application is a RESTful API call, which has been chosen for its ease of implementation and reliability.

## 4.6. Data Visualization and Analysis Framework

### The Sliding Window Algorithm

As already discussed, the primary data analysis algorithm is the sliding window algorithm that categorizes the data as normal or anomalous.

### Statistical Analysis

The data is the log files is statistically analysed to evaluate trends and create summaries and digests, which are then visually presented on the live dashboard.

### Matplotlib

A python graphing library, Matplotlib is used to generate the graphs that are sent over the Telegram bot. The graphs show the data around the anomaly for 3 seconds and highlight the anomalous region.

### Chart.js

Chart.js is a JavaScript data-plotting library that is used to generate real-time graphs on the live dashboard. This includes the live data feed, and the distribution of the anomalies across the times of the day and days of the week. The graphs are designed to deliver a broad overview of the information instantly and be aesthetically appealing.