1. PASSIVE ACOUSTIC SURVEILLANCE

Passive acoustic surveillance system that we are working on is a TÜBİTAK-2209A project. As its name suggest, we aim to design a acoustic surveillance with 8 MEMS microphone. The microphones are put into circular uniform area and each microphone is responsible for listening a specific direction. The PCB board also includes IMU sensor for navigation, GPS module for marking the board movements onto the map, SD card for backup, LoRa module for wireless comm and other tools for various purposes. I'm responsible for end-to-end software development so that I'm gonna give the software-specific details of this system.

1.1. Libraries

I've used many type of library in this project. Also I have designed two custom libraries. I will explain these in separately.

Libraries for embedded firmware:

- CMSIS (Common Microcontroller Software Interface Standard)
- HAL (Hardware Abstraction Layer)
- FreeRTOS Kernel

Libraries for ground station:

- GNU C (for standard POSIX-compatible API)
- SQLite3 (for database integration)
- Shumate (for mapping)
- Cairo (for 2D plotting)
- GTK 4 and Adwaita (for UI and main application)
- GNU GSL (for numeric computations)
- Custom DSP toolkit (will be explain below)
- Custom linear algebra toolkit (will be explain below)

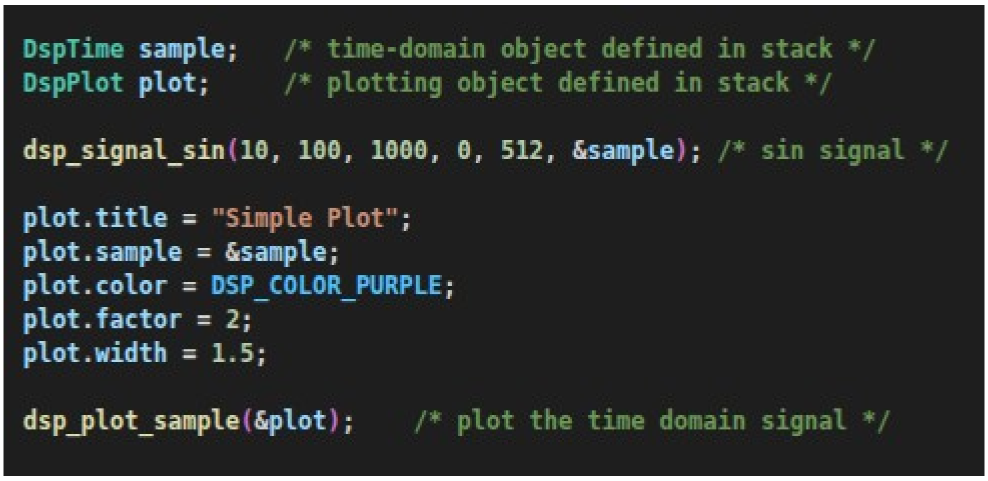
1.1.1. DSP (Digital Signal Processing) Library

I've designed a custom DSP library for this project. The main reason of this choice is that passive acoustic surveillance requires some kind of special algorithms like beamforming, direction of arrival or specific filter designs. So this custom library was designed with these considerations.

This library includes the following modules:

- core (for the main time and frequency domain operations)
- filter (for FIR and IIR – based filter designs)
- signal (for common signal generations)
- transform (for time-frequency domain transformations)
- plot (for 2D signal plotting)
- window (for windowing operations)
- beamform (for beamforming operations)
- arrival (for direction/angle of arrival operations)

Each module is grouped with associated prefix like *dsp_transform_dft(), dsp_filter_fir_low_pass()* or so on. Also each data in any kind of signal is being represented as *double* and dynamic allocation memory is not being used at all. So that each signal is defined in *stack*. This is required for deterministic behavior. Figure 1.1. and Figure 1.2. shows the example usage.



```c
DspTime sample;    /* time-domain object defined in stack */
DspPlot plot;      /* plotting object defined in stack */

dsp_signal_sin(10, 100, 1000, 0, 512, &sample); /* sin signal */

plot.title = "Simple Plot";
plot.sample = &sample;
plot.color = DSP_COLOR_PURPLE;
plot.factor = 2;
plot.width = 1.5;

dsp_plot_sample(&plot);    /* plot the time domain signal */
```

Figure 1.1. Code snippet for Plotting Sine

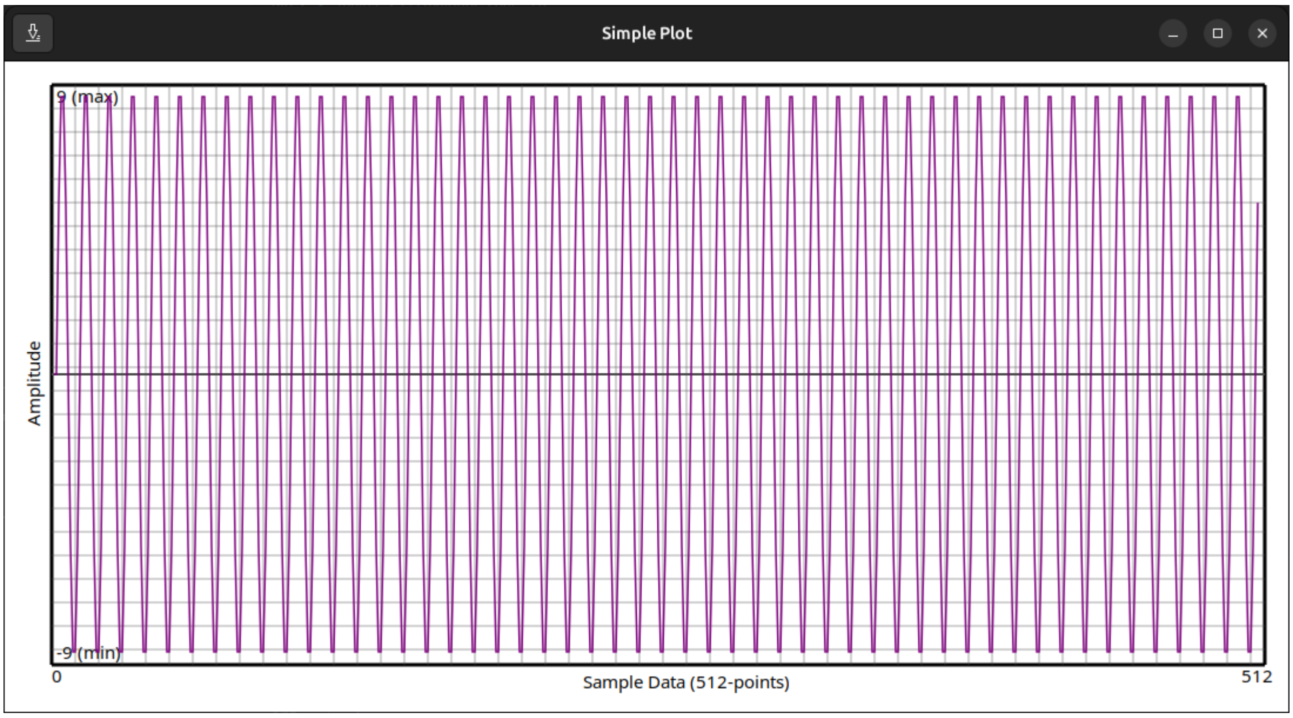


Figure 1.2. Result of the Code Snippet

This custom library was directly embedded into the ground station codebase. After the microphone outputs are captured from the serial port, the ground station makes required analysis using this custom library.

The project link: https://github.com/CanGulmez/Digital-Signal-Processing

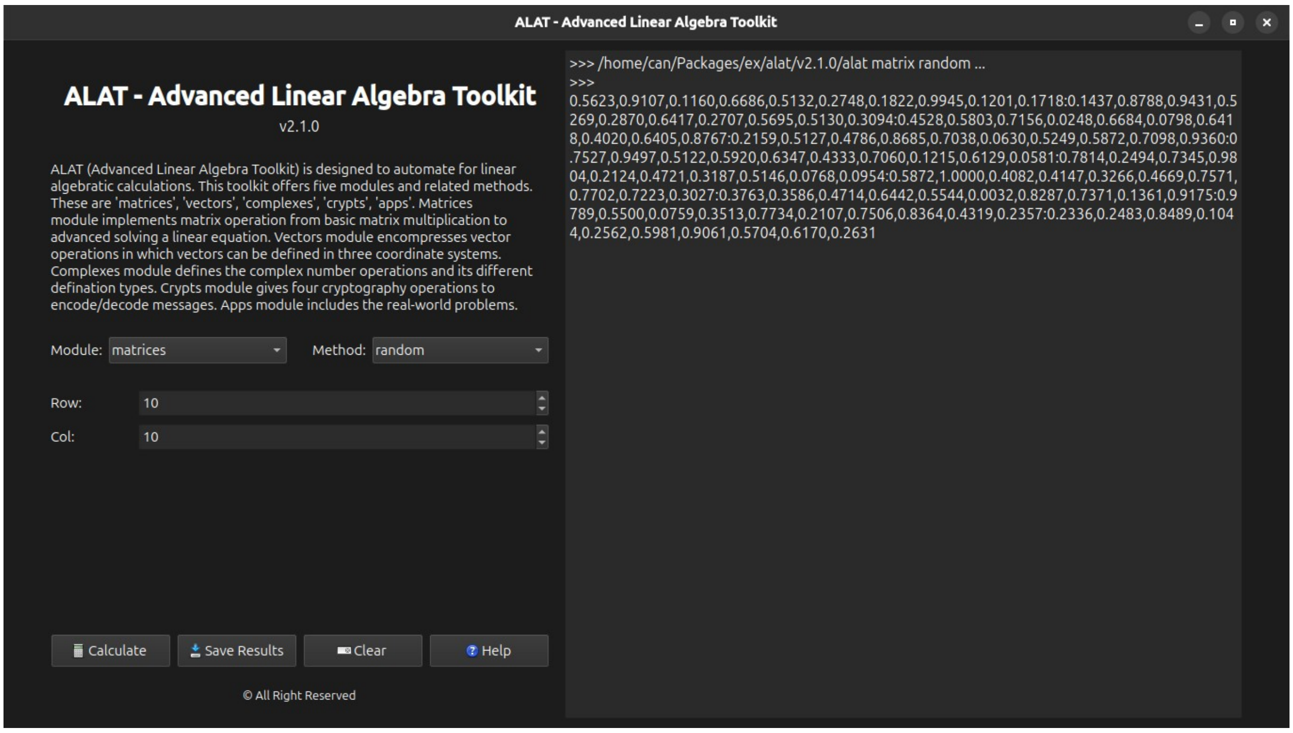
1.1.2. ALAT (Advanced Linear Algebra Toolkit) Library

I've wrote this custom linear algebra library independently from passive acoustic surveillance project. But I noticed that this project also requires the numeric computations. After that I've included this library into the this project.

As being custom DSP library, each data was defined as *double* and the library object is being defined in stack. So that this behavior provides determinism also.
This custom library includes the following modules:

- matrices (for matrix operations)
- vectors (for vector operations)
- complexes (for complex-number operations)
- crypts (for encoding/decoding messages)
- apps (for some real-world applications)

This library also includes the GUI script to use the library easily. Figure 1.3 shows an example usage.



ALAT - Advanced Linear Algebra Toolkit

v2.1.0

ALAT (Advanced Linear Algebra Toolkit) is designed to automate for linear algebratic calculations. This toolkit offers five modules and related methods. These are 'matrices', 'vectors', 'complexes', 'crypts', 'apps'. Matrices module implements matrix operation from basic matrix multiplication to advanced solving a linear equation. Vectors module encompresses vector operations in which vectors can be defined in three coordinate systems. Complexes module defines the complex number operations and its different defination types. Crypts module gives four cryptography operations to encode/decode messages. Apps module includes the real-world problems.

Module: matrices Method: random

Row: 10

Col: 10

```
>>> /home/can/Packages/ex/alat/v2.1.0/alat matrix random ...
>>>
0.5623,0.9107,0.1160,0.6686,0.5132,0.2748,0.1822,0.9945,0.1201,0.1718:0.1437,0.8788,0.9431,0.5
269,0.2870,0.6417,0.2707,0.5695,0.5130,0.3094:0.4528,0.5803,0.7156,0.0248,0.6684,0.0798,0.641
8,0.4020,0.6405,0.8767:0.2159,0.5127,0.4786,0.8685,0.7038,0.0630,0.5249,0.5872,0.7098,0.9360:0
.7527,0.9497,0.5122,0.5920,0.6347,0.4333,0.7060,0.1215,0.6129,0.0581:0.7814,0.2494,0.7345,0.98
04,0.2124,0.4721,0.3187,0.5146,0.0768,0.0954:0.5872,1.0000,0.4082,0.4147,0.3266,0.4669,0.7571,
0.7702,0.7223,0.3027:0.3763,0.3586,0.4714,0.6442,0.5544,0.0032,0.8287,0.7371,0.1361,0.9175:0.9
789,0.5500,0.0759,0.3513,0.7734,0.2107,0.7506,0.8364,0.4319,0.2357:0.2336,0.2483,0.8489,0.104
4,0.2562,0.5981,0.9061,0.5704,0.6170,0.2631
```

Figure 1.3. Creating a Random Matrix

This library is not being directly embedded into the ground station. Because it has separate CLI and GUI tools. So it can be thought the complementary program to the main ground station.

The project link: https://github.com/CanGulmez/ALAT

1.2. Embedded Firmware

Embedded firmware was written for STM32H7x series microcontroller using FreeRTOS kernel. It's designed in modular design. This is achieved over abstraction hardware layer. Its codebase includes the hardware configuration macros like *initGPIO*, *initSPI*, *initDFSDM*, or so on. The embedded firmware includes these component's settings:

- LoRa module (for wireless communication)
- GPS module (for marking the movements of board onto map)
- IMU sensor (for detecting acceleration and rotation of board)
- SD card (for backup of critical data)
- MEMS microphone (for collecting the audio data around board)

Ground station supports three communication ports (I will explain later) but current embedded firmware sends the payload data over wireless, LoRa, communication. It uses UART peripheral. Embedded firmware parses the NMEA sentences coming from GPS module and then extract the predefined data using these sentences. These data are UTC time, latitude, longitude, fix quality, number of satellite used, altitude, status, speed over ground (knots), course over ground (degrees) and data. It uses also UART peripheral.

The board includes the IMU sensor too. It’s used for navigation in the ground station. Embedded firmware reads and interprets this sensor and then put the interpreted data (in case, accelerometer and gyroscope outputs) into payload data structure. It’s done over SPI interface. In ground station, these are shown in 2D plot.

SD card is used to store the critical data when the power is off suddenly or abnormally. Embedded firmware writes the data with a fixed interval time. It’s widely used and recommended techniques in the critical system designs.

The most important mission of this system is to collect the audio data around the board simultaneously coming from MEMS microphones. It’s done overDFSDM peripheral. The embedded firmware just is responsible for packing these data into payload structure. The required analysis and operations will be taken in ground station. Because it is computationally heavy process for microconroller.

As can be seen, there are many tasks to cooperate the embedded firmware have to do. So the embedded firmware uses the FreeRTOS kernel to create, manage, schedule required tasks. Total 12 tasks are being used for these operations.

1.3. Ground Station

Ground station is responsible for listening the communication port, reading the payload data, processing the payload data as required, giving showing the outputs and giving a control panel to primary user. It has relative bigger and includes many modules. As described earlier, embedded firmware sends the payload data over serial/wireless communication channels. As a complementary with embedded firmware, ground station reads this payload data and let the user to control all the operations in real-time.

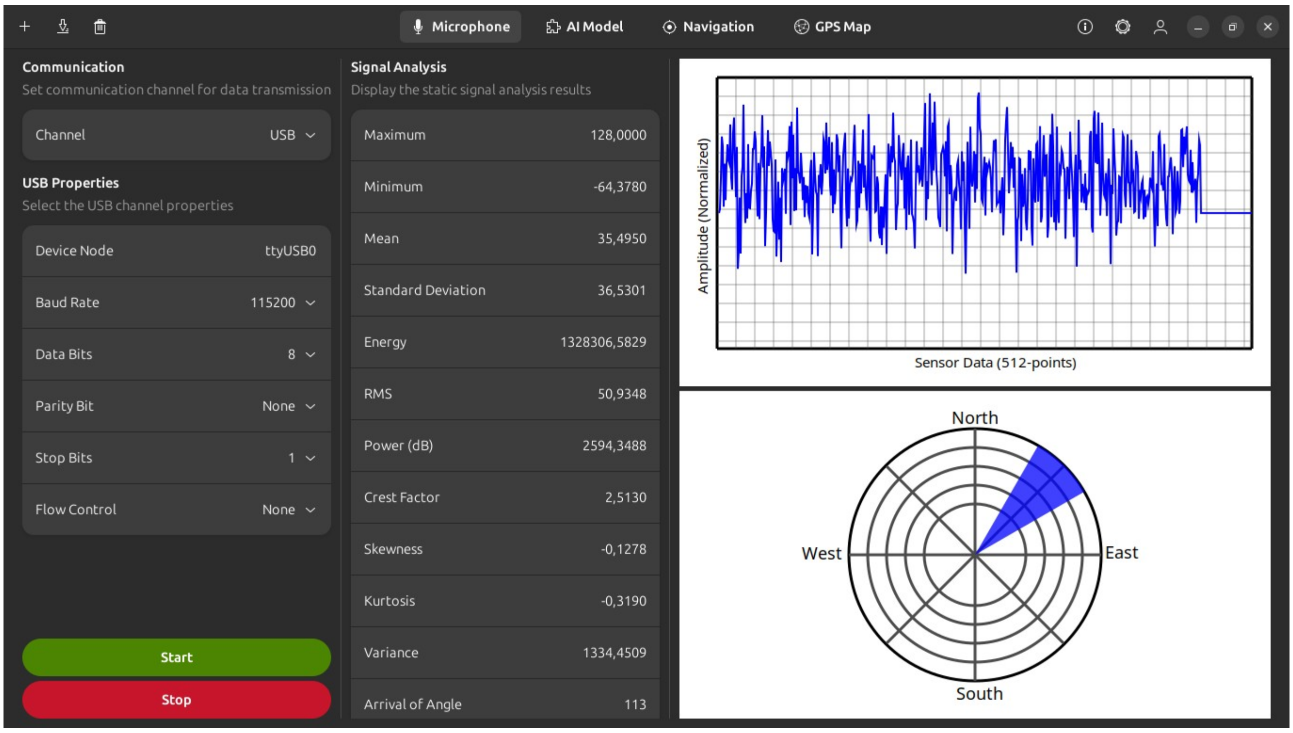
Ground station consists of four module:

- Microphone
- AI Model
- Navigation
- GPS Map

Each module has its own interface and responsibilities. Let’s look at these in details.

1.3.1. Microphone Module

Microphone module consists of three panel. Firstly, it has a channel selection panel. Over there, the payload data will be read with predefined port properties. After started the payload data reading, it will make the required signal analysis and shows the outputs in numerically at center panel and then update the plots. Upper plot shows the amplitude of analyzed signal and lower one shows the direction of the most intensive audio signal. Figure 1.4 shows the interface.



Communication
Set communication channel for data transmission

Channel	USB

USB Properties
Select the USB channel properties

Device Node	ttyUSB0
Baud Rate	115200
Data Bits	8
Parity Bit	None
Stop Bits	1
Flow Control	None

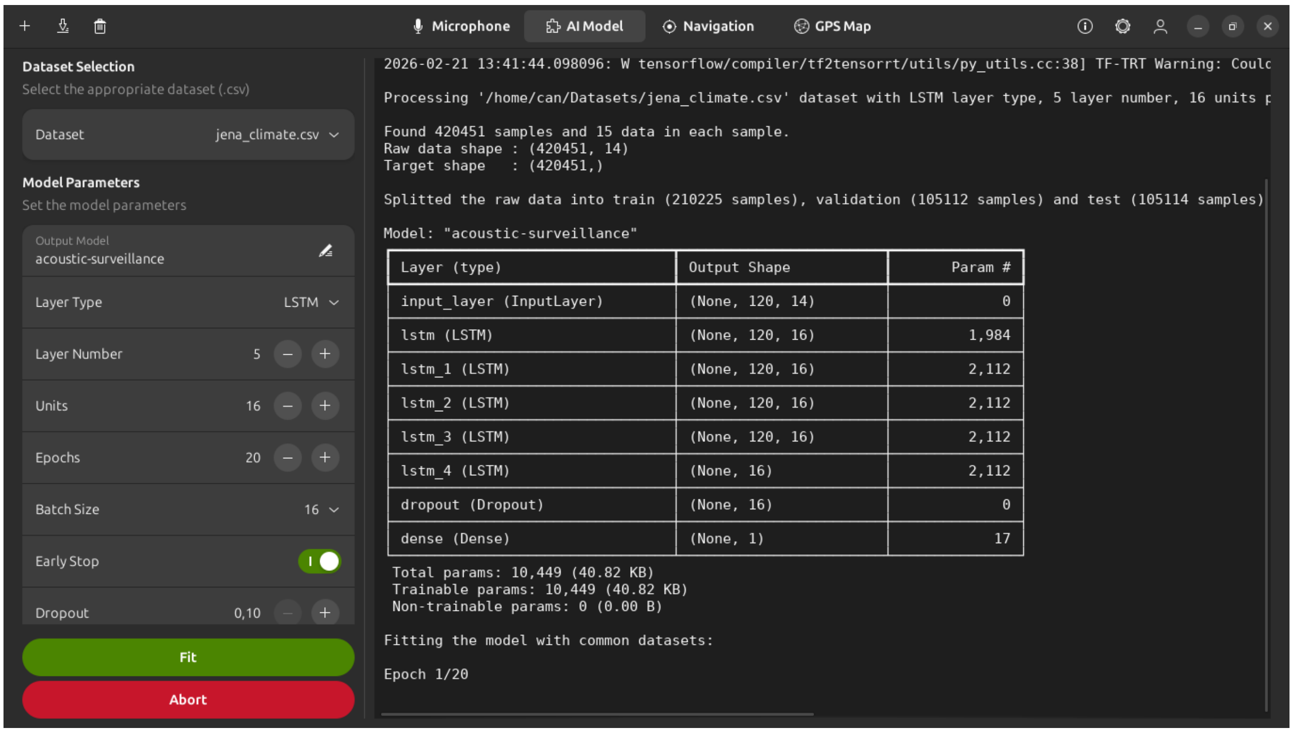
Signal Analysis
Display the static signal analysis results

Maximum	128,0000
Minimum	-64,3780
Mean	35,4950
Standard Deviation	36,5301
Energy	1328306,5829
RMS	50,9348
Power (dB)	2594,3488
Crest Factor	2,5130
Skewness	-0,1278
Kurtosis	-0,3190
Variance	1334,4509
Arrival of Angle	113

Figure 1.4. Microphone Module Interface

1.3.2. AI Model Module

AI model module has a Python 3 script that runs a LSTM/GRU based deep learning model using Tensorflow.keras library. Using the control panel at left side, some model parameters can be selected. After running the model, the outputs are being displayed at left side with a fixed timeout. Figure 1.5 shows the interface.



Dataset Selection
Select the appropriate dataset (.csv)

Dataset	jena_climate.csv

Model Parameters
Set the model parameters

Output Model	acoustic-surveillance
Layer Type	LSTM
Layer Number	5
Units	16
Epochs	20
Batch Size	16
Early Stop	I
Dropout	0,10

```
2026-02-21 13:41:44.098096: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Coulc

Processing '/home/can/Datasets/jena_climate.csv' dataset with LSTM layer type, 5 layer number, 16 units f

Found 420451 samples and 15 data in each sample.
Raw data shape : (420451, 14)
Target shape   : (420451,)

Splitted the raw data into train (210225 samples), validation (105112 samples) and test (105114 samples)

Model: "acoustic-surveillance"
```

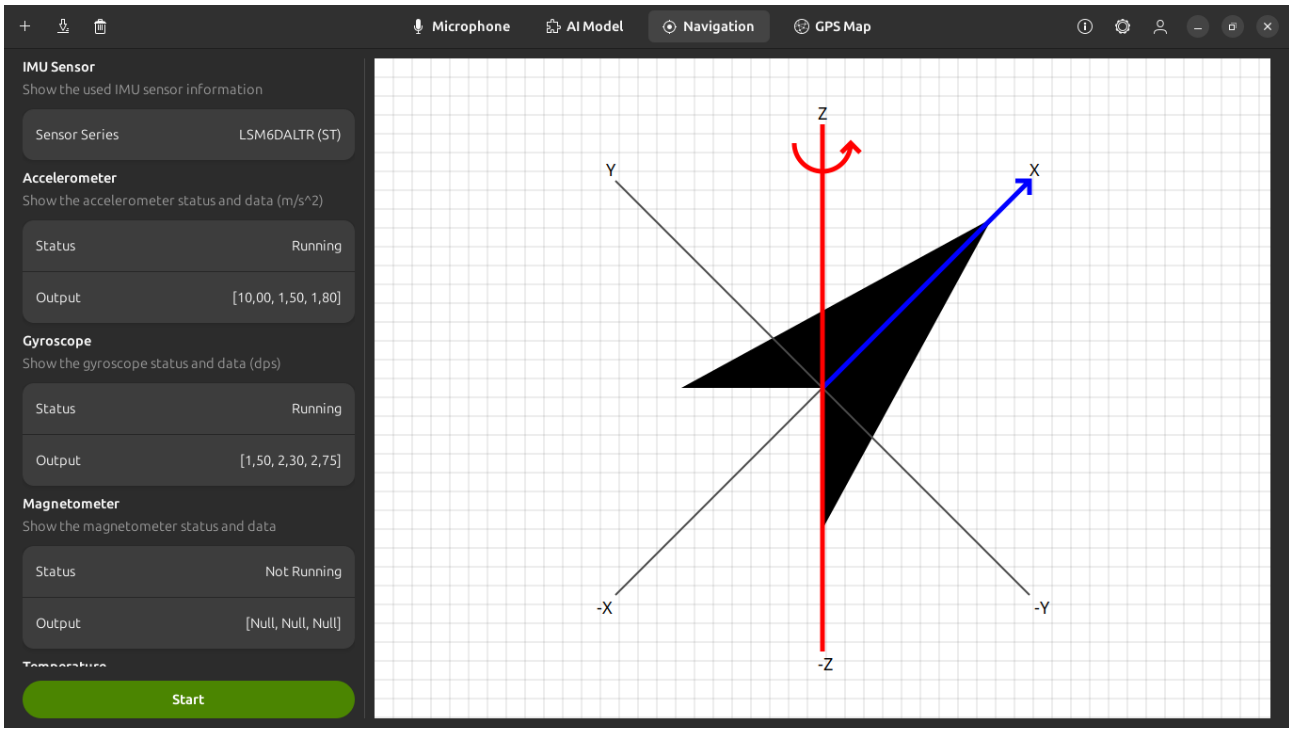
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 120, 14)	0
lstm (LSTM)	(None, 120, 16)	1,984
lstm_1 (LSTM)	(None, 120, 16)	2,112
lstm_2 (LSTM)	(None, 120, 16)	2,112
lstm_3 (LSTM)	(None, 120, 16)	2,112
lstm_4 (LSTM)	(None, 16)	2,112
dropout (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17

```
 Total params: 10,449 (40.82 KB)
 Trainable params: 10,449 (40.82 KB)
 Non-trainable params: 0 (0.00 B)

Fitting the model with common datasets:

Epoch 1/20
```

Figure 1.5. AI Model Module Interface

1.3.3. Navigation Module

Navigation module is responsible for detecting the orientation and rotation of the board using the IMU sensor. These data comes from the gyroscope and accelerometer circuits built-in the used IMU sensor. Figure 1.5 shows the interface.



IMU Sensor
Show the used IMU sensor information

Sensor Series	LSM6DALTR (ST)

Accelerometer
Show the accelerometer status and data (m/s^2)

Status	Running
Output	[10,00, 1,50, 1,80]

Gyroscope
Show the gyroscope status and data (dps)

Status	Running
Output	[1,50, 2,30, 2,75]

Magnetometer
Show the magnetometer status and data

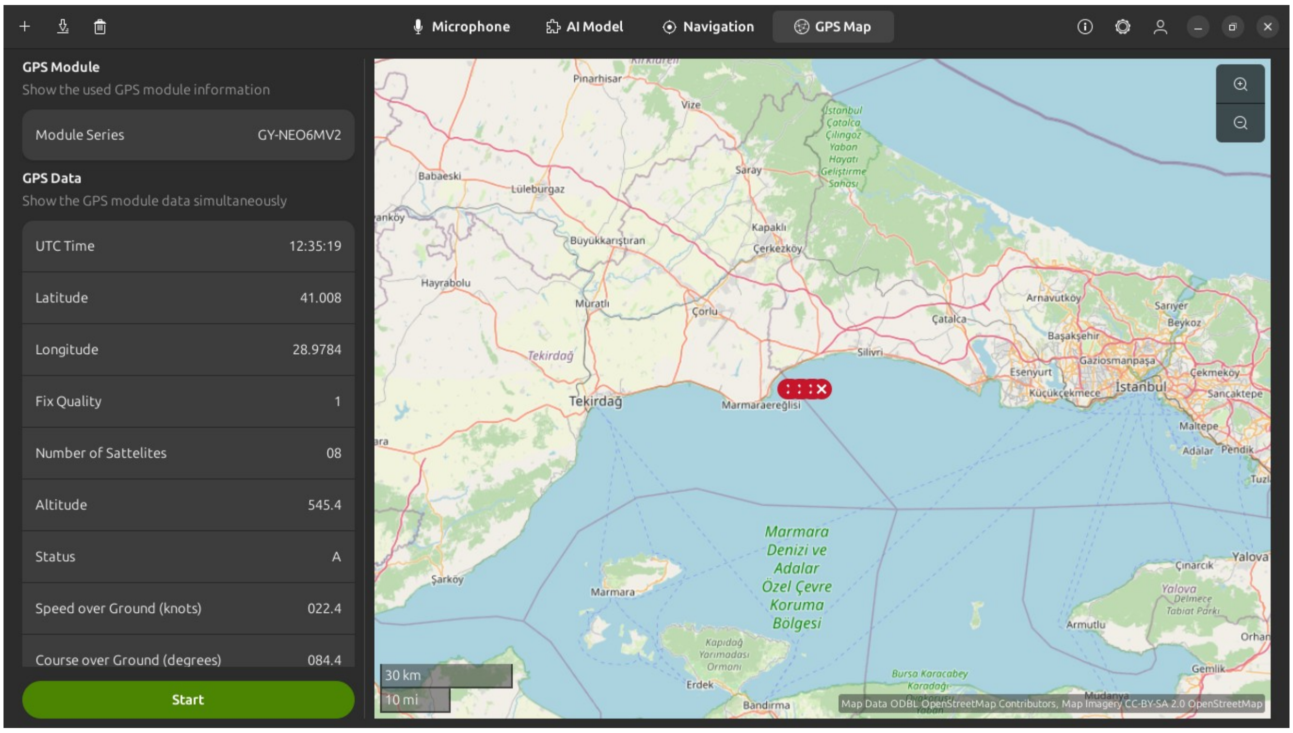
Status	Not Running
Output	[Null, Null, Null]

Temperature

Figure 1.5. Navigation Module Interface

1.3.4. GPS Map Module

GPS Map module shows the movements of the board onto GPS map and mark the latitude-longitude coordinates as the board moves. Figure 1.6 shows the interface.



GPS Module
Show the used GPS module information

Module Series	GY-NEO6MV2

GPS Data
Show the GPS module data simultaneously

UTC Time	12:35:19
Latitude	41.008
Longitude	28.9784
Fix Quality	1
Number of Sattelites	08
Altitude	545.4
Status	A
Speed over Ground (knots)	022.4
Course over Ground (degrees)	084.4

Figure 1.6. GPS Map Module Interface

Apart from these modules, the ground station has a set of sub-systems that work with these modules. First one is the database management system. It has Sqlite3 database that stores the analyzed signal with a fixed timeout.

Other one is the logging system. The ground station has many logging files that keeps the important missing messages coming from the ground station while running the modules.

The project link: https://github.com/CanGulmez/Passive-Acoustic-Surveillance