SPLASH

Sea Parameter Logging And Sensor Hub

Lamprinidis Iasonas, Zoidis Vasileios,
Michalainas Ioannis, Tzanetis Savvas

★ TABLE OF CONTENTS ★

01

Introduction

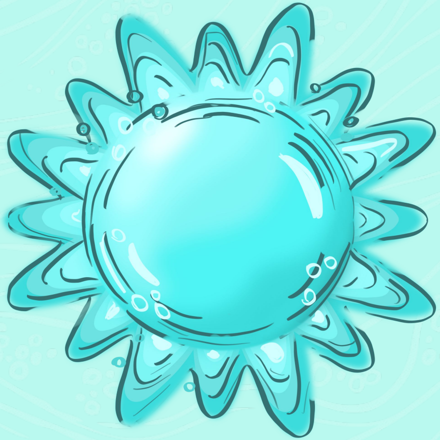
Iasonas Lamprinidis

Goal And Premise

Objective:
To develop an autonomous meteorological station in the form of a floating buoy, designed to collect and record oceanographic data such as wave height and wave period.

Functionality:
By applying mathematical and statistical analysis methods, the system will dynamically assess wave-related risk levels to enable timely alerts and enhance maritime safety.

Our vision



We imagined a fully autonomous weather and wave monitoring system using solar-powered drones in a mesh network. Each drone follows a pre-set route, collects data, and transmits it to a central, self-propelled, hydrodynamic vessel — no external support needed. The system is smart, self-charging, and designed for real-time analysis.

02

Arduino

Vasileios Zoidis

Objectives

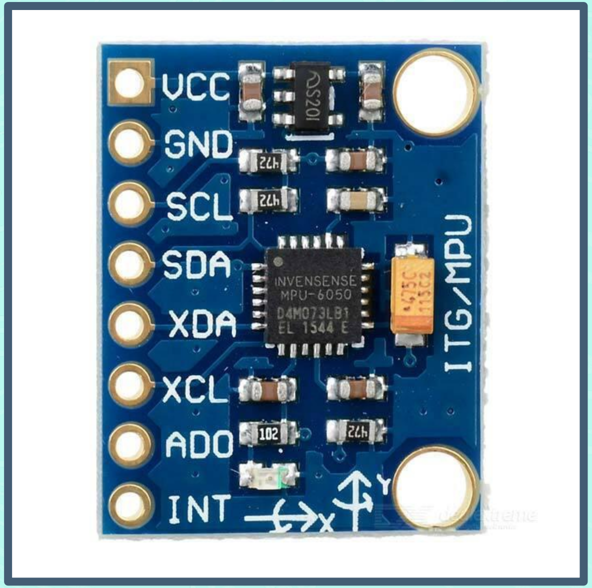
- Design a distributed sensor node that gathers environmental data
- Implement wireless communication between sensor and receiver using RF modules
- Display gathered information on a centralized platform
- Optimize energy use through staggered data transmission and low-power components

Sensor Selection

BMP280:

Measures temperature and pressure with high precision and stability

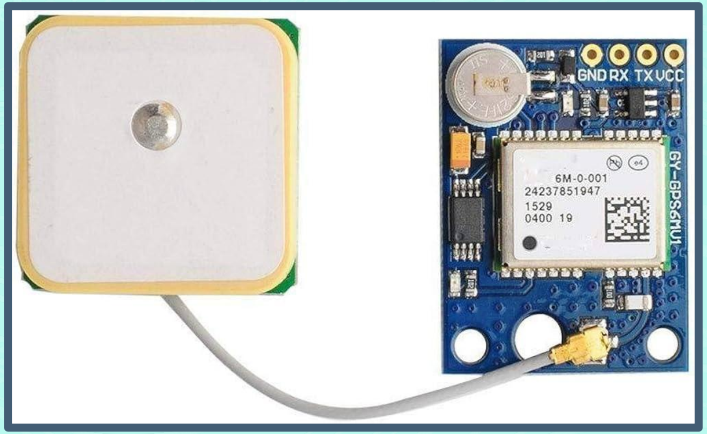
Sensor Selection



MPU6050:

A 6-axis accelerometer and used to estimate wave impact and surface motion dynamics

Sensor Selection



NEO-6M:

A GPS module, known for its compact size, affordability, and reliable performance.

Communication

We use two arduinos:

- The **transmitter** arduino captures and transmits the live data using the RFM22 antenna.

- The **receiver** arduino passes on the data through the serial connection to the backend.

Power Efficiency

We use a powerbank, so to be power efficient:

- Selective data transmission, sending full packets only once per second
- Configurable transmission power
- Power-efficient sensors

03

Backend

Ioannis Michalainas

Data Ingestion

Two modes:

- **DEMO**, with simulated data for testing purposes
- **DEPLOY**, real sensor data

High Frequency: accel_z

Low frequency: lat, lon, signal, temp, pressure, accel_z

Wave Analysis (1/2)

We normalize the z axis accelerometer data

```
lsb_to_ms2 = 9.81 / 16384
data = np.array(accelerometer_data) * lsb_to_ms2
data = data - np.mean(data)  # remove DC offset
```

Then, using FFT and heuristics we extract the wave frequency and height.

```
fft = np.fft.rfft(data)
fft_freqs = np.fft.rfftfreq(n, d=1.0/sampling_rate)
magnitudes = np.abs(fft)
dominant_freq = fft_freqs[np.argmax(valid_magnitudes)]
```

```
std_accel = np.std(data)
scaling_factor = 0.25
wave_height = scaling_factor*std_accel / (dominant_freq**2)
```

Wave Analysis (2/2)

Knowing f and H, and assuming β we calculate the vertical and horizontal runup

$$T = \frac{1}{f} \qquad L_0 = \frac{gT^2}{2\pi} \qquad \xi = \frac{\beta}{\sqrt{\frac{H}{L}}}$$

$$R_2 = \begin{cases} 0.043\sqrt{H_s L_0}, & \text{if } \xi < 0.3 \\ 1.1\left(0.35\beta\sqrt{H_s L_0} + 0.5\sqrt{H_s L_0\left(0.563\beta^2 + 0.004\right)}\right), & \text{if } \xi \geq 0.3 \end{cases}$$

$$I = \frac{R_2}{\beta}$$

If R2 > 0.9 the wave is deemed dangerous (VE), otherwise safe (AE)

API ENDPOINTS

The API provides four endpoints:

- **status**: online, battery level, coordinates and signal strength
- **info**: temperature and pressure
- **waves**: runup, inundation and danger zone
- **weather**: wind speed and direction and general weather condition

These endpoint are called upon by the frontend, in order to display this information in a meaningful and aesthetic manner

04

Frontend

Savvas Tzanetis

Tools Used

The application is built using several tools and frameworks, more specifically:

- **React** with the help of **Typescript**
- **Recharts:** Powers the data visualization for the wave height
- **Lucide React:** Supplies the iconography used throughout the interface
- **Framer Motion:** Provides fluid animations and transitions for the UI elements
- **Tailwind CSS:** Implements custom styling, creating consistent design patterns across components

Look and Feel

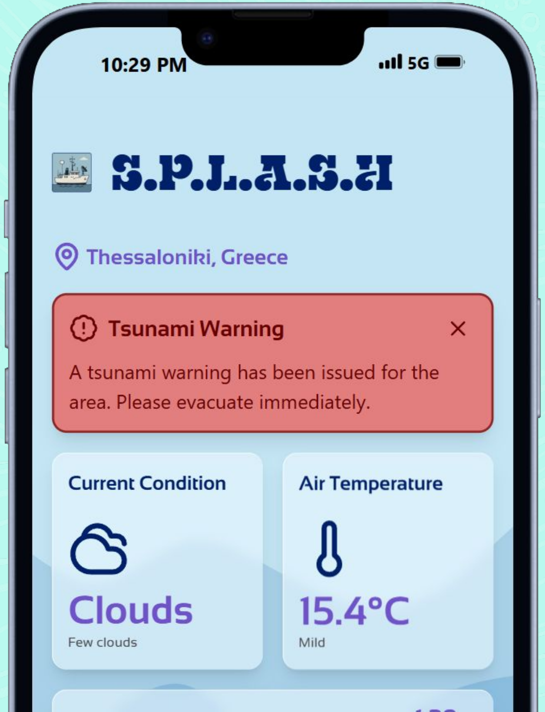
The layout of the web app consists of several independent widgets to display information:

- **Current Condition:** Changes it's icon as well as color depending on the current weather
- **Air Temperature:** Displays the current temperature as measured by the drone
- **Pressure:** Shows the atmospheric pressure as measured by the drone
- **Wind:** Provides the wind speed and direction
- **Avg Dist to Shore:** Displays the projected distance the average wave react to the shore
- **Drone Status**: Gives information about the drones battery and signal strength
- **Wave Height**: Shows a graph the current wave in 1 second intervals

Implementing Notifications

The user is notified with in-app alerts about extreme weather conditions and/or connection issues with the backend server. These alerts are categorized based on their priority:

- **Low priority**: Low priority alerts are indicated with a **yellow** color
- **High priority**: High priority alerts are indicated with a **red** color



API Requests

Each widget handles API requests independently with different time intervals depending on their needs. For example:

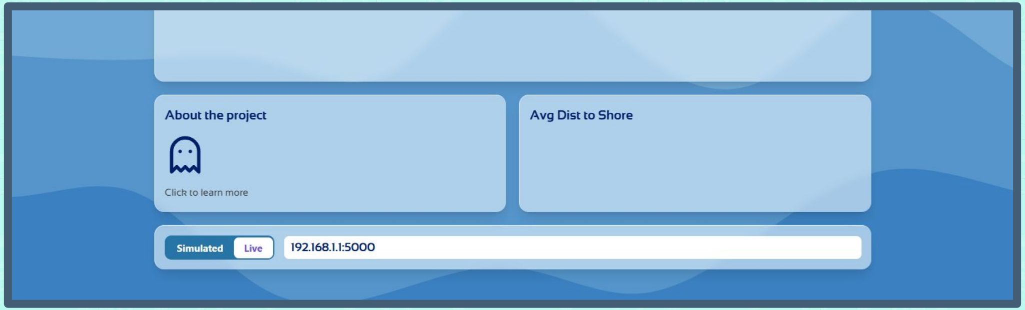
- The **Current Condition** widget makes API requests every 3 minutes as it relies on an external API, limiting the number of requests the system can make daily
- The **Wave Height** widget, makes a request every second as this makes its corresponding graph more responsive and informative

Other widgets and alerts have different request intervals, most of them being set at *30 seconds.*

Simulated Mode

Considering the difficulties associated with setting up a drone and backend infrastructure for a live demo, we have also set up the ability for the user to test our application with the help of an always online backend server.

This setting can be changed any time at the bottom of the page from the widget shown below:



Video Evidence*

Demo: https://www.youtube.com/watch?v=GD0r5xGcenA

Spot: https://www.youtube.com/watch?v=TnTkD0Kx9Sg

Repo: https://github.com/ioannisam/SPLASH

Thanks for your Attention