

Documentazione Anti Theft Boat

Damiano Vincenzo Coppola, Angelo Spadola, Enrico Sorbello, Santi Lisi

16 maggio 2025

Sommario

Il progetto propone una soluzione nel campo della sorveglianza delle imbarcazioni ormeggiate. Il problema consiste nel furto silenzioso dell'imbarcazione (a motore spento), per la rivendita di pezzi. Questo comporta una limitazione dal punto di vista dell'alimentazione, quindi è necessario un sistema indipendente e a bassi consumi. Verranno utilizzati sensori inerziali e un flussometro per il monitoraggio. Una volta che verrà rilevata un'anomalia, si innescherà un meccanismo di allerta che grazie a servizi web riuscirà ad eseguire un Tracciamento dell'imbarcazione. Per i controlli viene utilizzato un modello "ad hoc" che riconosce la differenza tra una manomissione e l'andamento delle onde.

1 Introduzione

Nel contesto della sicurezza nautica, i furti a scopo di smantellamento e rivendita di componenti rappresentano una minaccia concreta e sempre più diffusa. Veri e propri esperti riescono, in pochi minuti o ore, a sottrarre un'imbarcazione, eludere i controlli portuali e disattivare sistemi di sicurezza tradizionali, come i dispositivi GPS, portando a termine il furto in totale tranquillità. Per contrastare questo fenomeno, proponiamo una soluzione innovativa pensata per ambienti non alimentati, che agisce come un allarme silenzioso e offre un sistema di tracciamento in tempo reale estremamente efficace. A livello hardware, il sistema impiega due sensori inerziali per monitorare i movimenti dell'imbarcazione. I dati raccolti da uno dei sensori vengono elaborati da un modello di rilevamento anomalie, capace di identificare comportamenti sospetti. Inoltre, è stato integrato un flussometro, scelto per il suo basso consumo energetico e per la facilità di integrazione senza necessità di componenti di terze parti. Per ragioni di sicurezza e affidabilità, il sistema non prevede un meccanismo fisico per disattivare l'allarme (come ad esempio un pulsante). L'unico modo per interromperne il funzionamento è tramite comunicazione LoRaWAN, rendendo così estremamente difficile la manomissione da parte di malintenzionati.

2 Struttura e tecnologie

2.1 Hardware

Per essere più fedeli possibili alle premesse, sono state effettuate le seguenti scelte hardware tenendo in considerazione costi e spreco energetico:

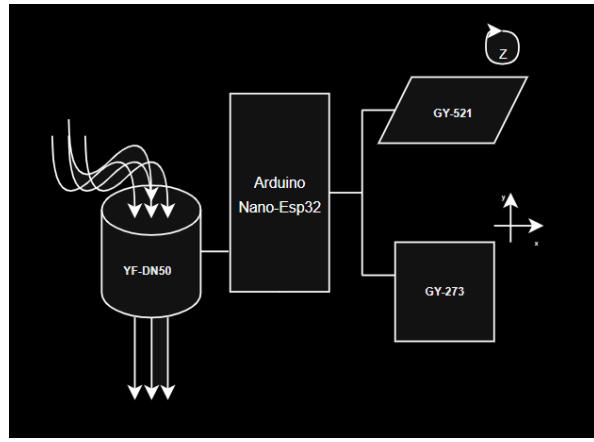


Figura 1: schema dispositivo

4 Block Diagram

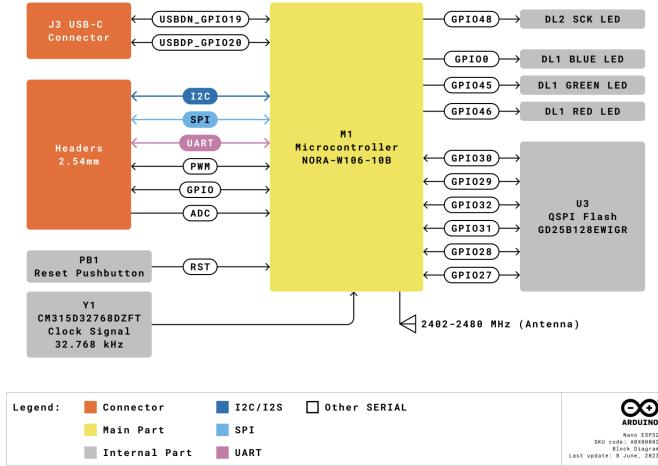


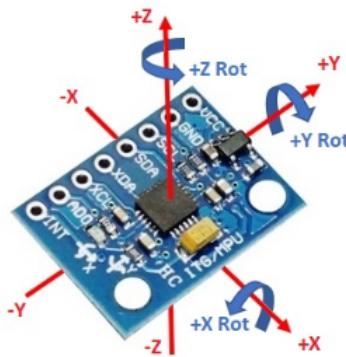
Figura 2: ESP32 Diagram

2.1.1 Arduino nano ESP32

La scelta di questo microcontrollore è stata guidata principalmente dalle sue caratteristiche orientate al mondo IoT, come la connettività **Bluetooth** e **Wi-Fi**, nonché dalla presenza del modulo **u-blox® NORA-W106**, basato sul processore **ESP32-S3**. Quest'ultimo è un **dual-core Xtensa® LX7** a 32 bit, con frequenza di clock fino a **240 MHz**, dotato di acceleratori per operazioni AI/ML, e con supporto per la crittografia hardware, rendendolo adatto ad applicazioni di tipo IoT che richiedono sicurezza, elaborazione locale e comunicazione wireless.

2.1.2 GY-521 - Giroscopio

il modulo **GY-521**, basato sul sensore **MPU-6050**, integra un giroscopio e un accelerometro a 3 assi, ed è ampiamente utilizzato per il rilevamento del movimento e dell'orientamento nei sistemi embedded. La comunicazione avviene tramite interfaccia **I²C**, semplificando l'integrazione con microcontrollori. Nel contesto di questo progetto, viene utilizzato esclusivamente l'**asse Z** del giroscopio, una scelta che consente di ridurre sia il consumo di risorse computazionali sia la complessità del codice. Inoltre, l'interfaccia I²C consente di porre il dispositivo in modalità *sleep* quando non è in uso: in questo modo, durante i periodi in cui il sistema d'allarme è disattivato, i consumi energetici risultano praticamente trascurabili.



2.1.3 YF-DN50 - Flussometro

Il sensore **YF-DN50** è un flussometro a turbina solitamente impiegato in impianti idraulici per il monitoraggio della portata. In questo progetto è stato **adattato** per misurare la velocità dell'acqua, fornendo un'indicazione indiretta della velocità della barca.

Il sensore è in grado di rilevare una portata massima di **300 litri al minuto**. Il principio di funzionamento si basa sulla rotazione della turbina interna: a ogni giro viene generato un segnale digitale di tipo *RISING edge*. Tali impulsi vengono acquisiti, convertiti e successivamente elaborati per calcolare il **volumen d'acqua transitato** in un determinato intervallo di tempo, da cui si ricava la **velocità della barca**, espressa in nodi (*kn*).



Grazie alla semplicità del segnale in uscita, il sensore risulta facilmente integrabile con microcontrollori, e l'elaborazione dei dati può essere gestita con un impatto minimo sulle risorse computazionali.

2.1.4 GY-273 - Magnetometro

Il modulo **GY-273** è basato sul sensore **HMC5883L**, un magnetometro digitale a 3 assi in grado di rilevare il campo magnetico terrestre, e viene comunemente impiegato come bussola elettronica.

Nel contesto di questo progetto, il sensore risulta fondamentale nella **fase di allerta**, in quanto consente di monitorare la direzione della barca. Il sistema confronta la direzione istantanea con quella di riferimento, rilevando eventuali variazioni anomale che potrebbero indicare uno spostamento indesiderato.

Anche questo modulo comunica tramite interfaccia **I²C**, consentendo una gestione efficiente dei consumi: quando l'allarme è disattivato, il dispositivo può essere messo in modalità *sleep*, riducendo ulteriormente l'assorbimento energetico complessivo del sistema.

2.1.5 Gateway TTN - LoRaWAN

Il gateway utilizzato è compatibile con il protocollo **LoRaWAN** ed è collegata all'infrastruttura di **The Things Network (TTN)**. Questo dispositivo ha il compito di ricevere i pacchetti inviati via radio dal nodo a bordo dell'imbarcazione e inoltrarli tramite connessione a Internet (Ethernet o Wi-Fi) verso la rete TTN, da cui possono essere successivamente elaborati o integrati in altre piattaforme cloud.

L'utilizzo di una gateway LoRaWAN permette una **comunicazione a lungo raggio e a basso consumo energetico**, ideale per ambienti remoti o privi di copertura cellulare. La scelta di TTN

consente inoltre di sfruttare un'infrastruttura open-source, scalabile e gratuita, con accesso facilitato ai dati attraverso webhook o API.

2.2 Struttura e funzionamento

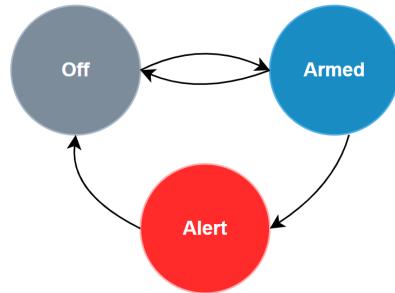
La struttura del sistema è volutamente semplice, consentendo un'installazione e una configurazione rapide. L'imbarcazione comunica con la **gateway LoRaWAN**, inviando messaggi di allerta che vengono raccolti e inoltrati a un **server remoto**. Quest'ultimo elabora i dati ricevuti e li rende disponibili all'utente finale attraverso un'applicazione web dedicata, visualizzandoli in tempo reale all'interno della propria area riservata.

2.2.1 Logica interna

Il dispositivo una volta configurato applica una semplice logica a tre stati: Spento, Allarme acceso e Allerta. In partenza il dispositivo è *"Spento"*. Tramite comando dell'utente (dall'applicativo web) è possibile attivare l'allarme. Una volta *armato* il dispositivo attiverà i vari sensori ed effettuerà le letture. Ogni sensore ha un controllo specifico, per colmare tutte le varie casistiche:

- Il **flussometro** rileva un incremento significativo della velocità dell'imbarcazione, indicando una possibile messa in moto non autorizzata.
- L'**accelerometro** verifica se l'imbarcazione si sta orientando verso una direzione al di fuori del range angolare registrato al momento dell'ormeggio. Questo controllo è particolarmente efficace per rilevare manovre lente ma ampie.
- Il **modello di rilevamento anomalie** analizza i dati del giroscopio, e se identifica un movimento sospetto all'interno del suo range di riferimento, lo interpreta come un'azione forzata.

Se anche solo uno dei sensori rileva un problema allora si attiva l'allarme. Infine, è possibile spegnere allarme e allerta tramite l'applicativo.

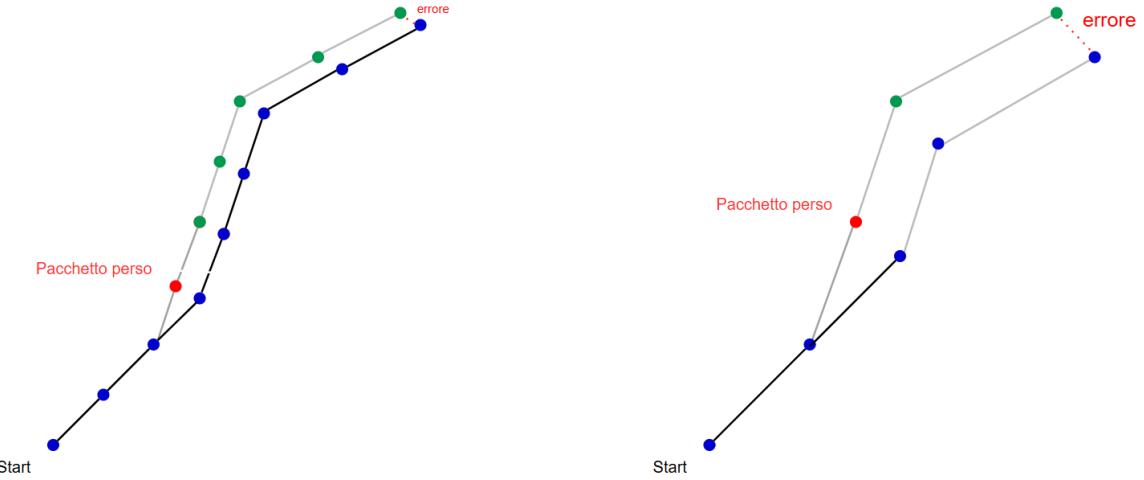


2.2.2 Comunicazioni dal dispositivo

Il dispositivo per motivi relativi alla **classe A LoraWan**, comunica costantemente con il backend. Il motivo è che per avere la finestra di ricezione attiva e quindi ricevere messaggi, il dispositivo deve prima comunicare. C'è un discorso a parte sulla *frequenza* dei messaggi, ma ne parleremo dopo. Per evitare rallentamenti nei controlli che devono tenere traccia di un contesto dinamico, la parte che si occupa delle comunicazioni è stata spostato su un altro **core**. Ogni stato ha il suo messaggio specifico, così che sia facile anche da fuori conoscere lo stato interno del dispositivo. La parte fondamentale di questa sezione è nel caso di allerta. I sensori come Flussometro e magnetometro forniranno direzione e velocità e una volta comunicati sarà possibile calcolare la nuova posizione.

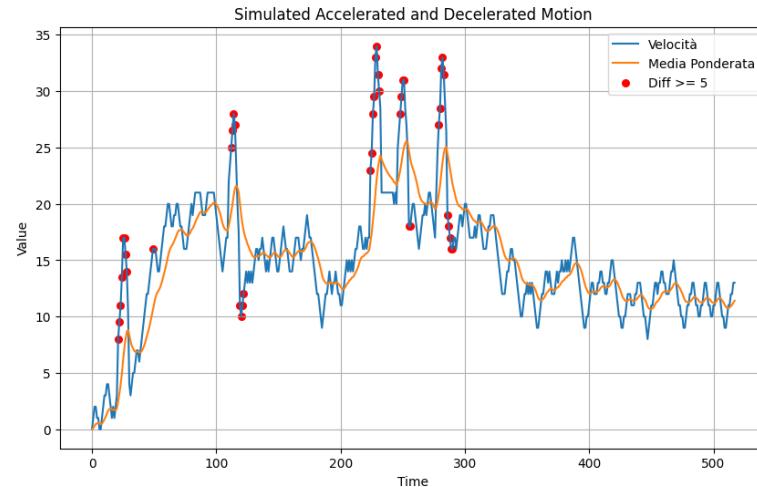
2.2.3 Intervalli di comunicazione

Nella casualità in cui un pacchetto si perda, il calcolo della posizione è più **robusto** se l'intervallo è piccolo, ma ciò comporta un maggior sforzo del dispositivo e un aumento significativo del carico da gestire per il backend.



3 Gestione dei dati

Controlli

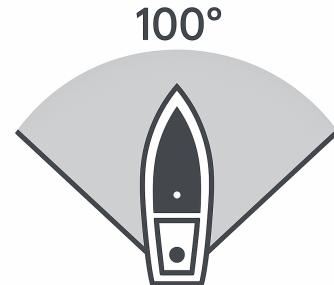


Flussometro

Il flussometro restituisce continuamente un segnale di *RISING* che viene convertito in volume. Conoscendo il volume e il tempo possiamo conoscere la velocità dell'acqua. Una volta ottenuta la velocità in *nodi* vengono continuamente fatti *campionamenti* e viene mantenuta una media ponderata. Qualora la differenza fosse maggiore di 5 o comunque venga superata una certa soglia l'allarme si attiva.

Magnetometro

Essendo un dispositivo I2C la lettura dei valori viene fatta nei registri del dispositivo. Precisamente vengono letti i valori degli assi *x* e *y*, con i quali viene calcolata l'arctangente con la funzione *atan2()* (a differenza di *atan()* risolve le divisioni per 0) per calcolare l'angolo magnetico. Infine viene convertito in gradi e normalizzato per ottenere lo 0 a nord e un range di valori [0,359]. Quando l'allarme si attiva viene campionato un valore *g* l'angolo attuale e normalizzato, se non rientra tra [-50, +50] si attiva l'allarme.

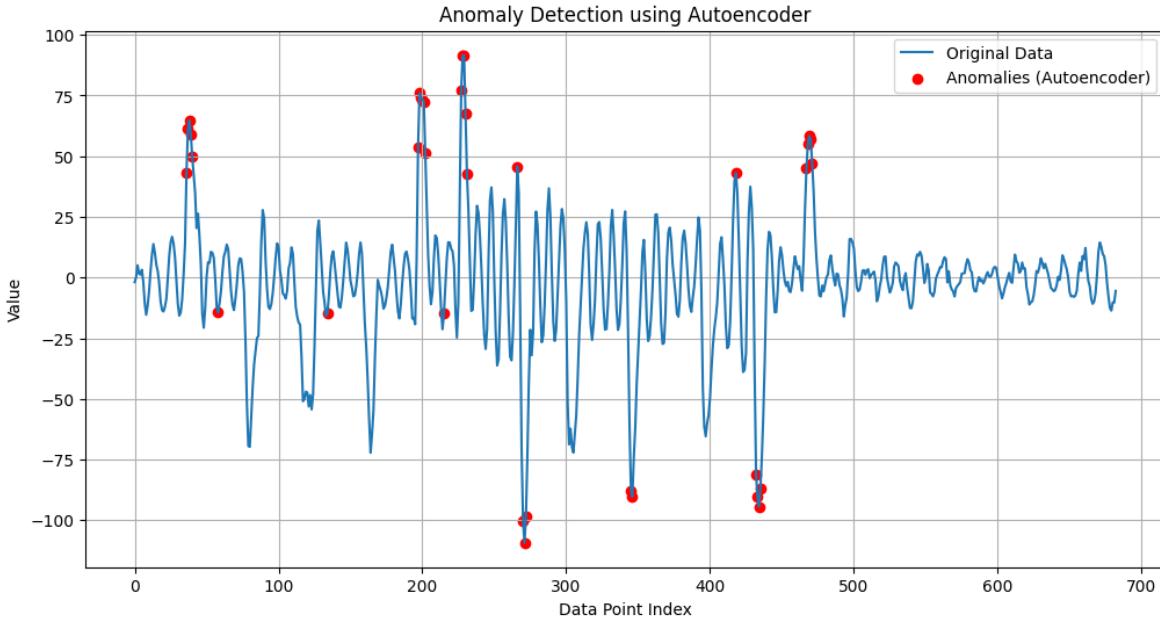


Giroscopio

Ogni secondo vengono effettuate k misurazioni, dove k è la dimensione dell'array che viene usato come input nel **modello**. Ogni misurazione viene normalizzato col valore 131.0 per settare la sensibilità a $250^\circ/\text{s}$ (una sensibilità più alta è inutile). Anche questo è un dispositivo I2C quindi leggiamo i dati dai registri. Ora vedremo come è stato allenato il modello.

3.1 Rete Neurale

La sfida più difficile per il rilevamento di manomissioni è legata alla natura *instabile* dell'imbarcazione. Le onde causano continuamente disturbi ed è facile mal interpretare le lettura se venissero fatti controlli statici. Quindi si è pensato di sfruttare un **classificatore** per riconoscere **anomalie** nelle misurazioni del giroscopio. Per fare ciò sono stati raccolti manualmente i dati simulando una manomissioni e periodi di calma. Questi dati sono stati utilizzati in un **autoencoder** specifico per l'**anomaly detection** e grazie a questo *etichettati*. Successivamente sono state prelevate le anomalie insieme ai valori circostanti, formando per ogni anomalia un array di k elementi. Sono stati creati dai dati rimanenti i dati con etichetta "0" e sono stati usati per allenare il modello. Infine il modello del classificatore è stato convertito in linguaggio C grazie alla libreria python tinymlgen per essere importato sulla board del microcontrollore che essendo un sistema embedded possiede risorse limitate. La dimensione della rete finale è **2540** parametri. Questa strategia è molto potente, perché siamo riusciti a: etichettare in modo automatico i dati, allenare un modello su schemi specifici e a creare una rete di piccole dimensioni (ideale per dispositivi ridotti).



Parametri e altro

3.1.1 L'Autoencoder

Viene addestrato utilizzando i dati del giroscopio. **Definizione dell'autoencoder:**

- **Layer di input:** attivazione ReLU, dimensione input: 1;
- **Layer di codifica:** 8 neuroni e attivazione ReLU;
- **Layer di decodifica:** 1 neurone e attivazione lineare per la ricostruzione dell'input.

Come funzione di loss viene utilizzato il **Mean Squared Error (MSE)**, e l'addestramento viene eseguito per 30 **epoch**. Dopo l'addestramento, viene calcolato l'MSE per ciascun punto del dataset. Si stabilisce una **threshold al 95%** dei valori MSE: tutti i punti che superano questa soglia vengono identificati come **anomalie**.

3.1.2 Il Classificatore

Viene addestrato con i dati estrapolati e etichettati dall'autoencoder. **Architettura della rete neurale:**

- **Layer 1 (Input):** attivazione ReLU, dimensione input: 5;
- **Layer 2 (Hidden):** 10 neuroni , attivazione ReLU;
- **Layer 3 (Output):** attivazione Softmax (per classificazione binaria).

Come funzione di loss la **categorical_crossentropy** (con l'**accuracy** come metrica). L'addestramento in 30 epoch. Raggiunge il 97.8% di **accuracy** e 0.15 di **loss**.

4 Trasmissione dati

4.1 Rete LoRaWAN

4.1.1 Gateway

Un **gateway LoRaWAN** viene messo a disposizione dei dispositivi per ricevere i dati da essi trasmessi. Questi dati vengono successivamente recuperati dallo **stack di The Things Network (TTN)** e resi disponibili tramite un'applicazione web dedicata.

Il gateway svolge anche una funzione attiva nel sistema, permettendo di **monitorare lo stato dei dispositivi** e di **inviare comandi** per l'attivazione o l'armamento dell'allarme attraverso **messaggi downlink** indirizzati ai singoli device.

Per la gestione della comunicazione e il controllo dei dispositivi viene utilizzata un'architettura ibrida che combina **MQTT** e **HTTP**, sfruttando script sviluppati in **Python** per l'integrazione, l'elaborazione dei dati e l'automazione delle operazioni di sistema.

4.1.2 Registrazione Dispositivo

La registrazione e configurazione del dispositivo all'interno dell'infrastruttura di **The Things Network (TTN)** avviene tramite l'utilizzo delle sue **API REST**. Per completare la procedura di attivazione, è necessario fornire alcuni parametri fondamentali:

```
device_id  
dev_eui  
app_key  
join_eui
```

La registrazione del dispositivo avviene attraverso tre chiamate HTTP ai seguenti endpoint:

- `api/v3/ns/applications/app/devices/deviceid`
- `api/v3/as/applications/app/devices/deviceid`
- `api/v3/js/applications/app/devices/deviceid`

Sebbene i payload richiesti da TTN siano molto estesi, i campi che devono essere specificati manualmente sono contenuti nella sezione **ids**, comune a tutte le richieste. Un esempio di struttura JSON utilizzata per la configurazione è il seguente:

```
1  {  
2      "ids": {  
3          "join_eui": "0000000000000000",  
4          "dev_eui": "deveui",  
5          "device_id": device,  
6          "application_ids": { "application_id": "app" }  
7      }  
8  }
```

Questa procedura, automatizzata tramite script Python, consente di registrare dinamicamente nuovi dispositivi all'interno dell'applicazione TTN, semplificando la scalabilità del sistema e riducendo la possibilità di errori manuali.

4.2 Uplink e Downlink del Dispositivo

Il comportamento del dispositivo varia in funzione dello **stato operativo** in cui si trova. In linea generale, l'interazione con il dispositivo avviene tramite messaggi **uplink** e **downlink**, gestiti attraverso il protocollo **MQTT**.

Per il monitoraggio, il sistema si iscrive al *topic* MQTT generico per ricevere tutti i payload inviati *dai dispositivi* in uplink. Inversamente, l'invio dei comandi verso *un dispositivo* avviene tramite downlink utilizzando il seguente topic:

```
v3/app@ttn/devices/{deviceid}/down/replace
```

il "frm_payload" di downlink deve contenere uno dei due comandi riconosciuti dal dispositivo, i quali vengono interpretati in base allo stato corrente:

- "on" — per armare il dispositivo
- "off" — per disattivare il dispositivo o l'allarme in corso

il payload¹ di downlink ha il seguente formato:

```
1 "downlinks": [
2   {
3     "f_port": 1,
4     "frm_payload": base64_payload,
5     "priority": "NORMAL"
6   }
7 ]
```

4.2.1 Stato Spento

Quando un dispositivo si trova nello stato di **spento**, esso continua a trasmettere ciclicamente messaggi uplink contenenti il payload "spento". Solamente a seguito della ricezione di un comando downlink con payload "on", il dispositivo cambia il proprio stato, passando alla modalità **armato**.

4.2.2 Stato Armato

In questa modalità, il dispositivo monitora costantemente i dati provenienti dai **sensori precedentemente descritti** (giroscopio, magnetometro, flussometro, ecc.). Qualora venga rilevata un'anomalia significativa rispetto ai parametri attesi o di riferimento, il dispositivo entra automaticamente nello stato di **allarme attivo (on)**.

In alternativa, se il dispositivo riceve un messaggio **downlink** con payload "off", passerà allo stato di **spento**, interrompendo il monitoraggio attivo e tornando in una modalità a basso consumo.

4.2.3 Stato di Allarme

Nel caso in cui il dispositivo rilevi delle anomalie tramite i sensori, esso cambia stato ed entra in modalità di allerta. In questa modalità, il dispositivo invia tramite uplink i dati rilevati dal flussometro e dal magnetometro. Queste informazioni includono la velocità e la direzione del dispositivo, analogamente come per lo stato di armato se un downlink "off" viene trasmesso il dispositivo torna in stato spento .

Il campo **frm_payload** è strutturato nel seguente modo:

```
"frm_payload": "base64encoded("velocità;direzione")"
```

Il carattere ; viene utilizzato come separatore per facilitare l'interpretazione dei dati da parte dello script Python.

¹nota che frm_payload è presente anche nell' uplink

5 Gestione dei Dati in Modalità Allerta

Durante la fase di allerta, l'applicativo si occupa di filtrare i dati ricevuti in base al *Device ID*, registrando le informazioni di sensoristica relative a ciascun dispositivo all'interno di una tabella dedicata.

Questa procedura consente di tracciare in modo ordinato i dati relativi alla velocità e alla direzione, utili per il calcolo della posizione. A partire da una posizione iniziale generica, è infatti possibile stimare la latitudine e la longitudine attuali del dispositivo, elaborando i dati ricevuti nel tempo.

5.1 Identificazione dei Dispositivi in Allarme

Non appena un dispositivo invia un uplink contenente il carattere ; nel campo `frm_payload`, l'applicativo lo classifica automaticamente come dispositivo in stato di allarme. A partire da questo momento, tutti i pacchetti successivi inviati dal dispositivo vengono registrati nel database per consentirne il monitoraggio e l'elaborazione dei dati.

```
1 if ";" in payload_text:
2     direzione, flusso = payload_text.split(":")
3     with app.app_context():
4         device = Device.query.filter_by(device_id=device_id).first()
5         if device and not device.alarm and device.status == 1:
6             print(f"Setting alarm ON for device {device_id}")
7             device.alarm = True
```

5.2 Storage dei dati

Una volta che un dispositivo è stato classificato come in stato di allarme, tutti i suoi messaggi in *uplink* vengono costantemente monitorati e salvati nel database. Questo permette di tenere traccia in tempo reale dei dati di sensoristica e di ricostruire con precisione il comportamento e la posizione stimata del dispositivo durante la fase di allerta.

```
1 if device.alarm:
2     new_packet = AlarmMessage(
3         device_id=device_id,
4         time=timestamp,
5         velocity=flusso,
6         direction=direzione,
7     )
8     db.session.add(new_packet)
```

5.3 Calcolo della Posizione

Prendendo in considerazione i dati dell'ultimo e del penultimo messaggio *uplink* ricevuto da un dispositivo, è possibile stimare la nuova posizione geografica (latitudine e longitudine) del dispositivo attraverso l'applicazione delle formule di navigazione sferica.

Il procedimento si basa su:

Il tempo trascorso tra i due pacchetti, utilizzato per calcolare lo spostamento.

La velocità e la direzione del dispositivo, ricavate dall'ultimo pacchetto.

La posizione iniziale nota (latitudine e longitudine correnti) del dispositivo.

Utilizzando la formula dell'approssimazione dell'arco di cerchio massimo sulla sfera terrestre, si calcolano le nuove coordinate. In particolare, si assume la Terra come una sfera con raggio medio di circa 3440.065 miglia nautiche (valore assegnato alla variabile R).

Il calcolo avviene secondo i seguenti passaggi.

5.3.1 codice

```
1 time1 = last_packet.time time2 = previous_packet.time
2 time1 = time1.rstrip("Z")[:26]
3 time2 = time2.rstrip("Z")[:26]
4
5 dt1 = datetime.datetime.strptime(time1, "%Y-%m-%dT%H:%M:%S.%f")
6 dt2 = datetime.datetime.strptime(time2, "%Y-%m-%dT%H:%M:%S.%f")
7
8 second1 = int(dt1.timestamp())
9 second2 = int(dt2.timestamp())
10
11 time3 = (second1 / 3600) - (second2 / 3600)
12
13 v = last_packet.velocity
14 direction = last_packet.direction
15 R = 3440.065
16
17 distanza = float(v) * time3
18 device = Device.query.filter_by(device_id=deviceid).first()
19
20 latrad = radians(device.current_lat)
21 lonrad = radians(device.current_long)
22
23 distanza_angolare = distanza / R
24 directionrad = radians(int(direction))
25
26
27 new_lat = asin(
28     sin(latrad) * cos(distanza_angolare)
29     + cos(latrad) * sin(distanza_angolare) * cos(directionrad)
30 )
31
32 new_lon = lonrad + atan2(
33     sin(directionrad) * sin(distanza_angolare) * cos(latrad),
34     cos(distanza_angolare) - sin(latrad) * sin(new_lat),
35 )
36
37 new_lat = degrees(new_lat)
38 new_lon = degrees(new_lon)
```

Grazie a questo metodo è possibile aggiornare in modo dinamico la posizione stimata del dispositivo in allarme, assumendo una traiettoria rettilinea a velocità costante tra due rilevamenti consecutivi.