Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DE COMPUTADORES

Sistema de captura y procesado de datos IMU en tiempo real para estimación de movimiento vertical

Estudante: Mateo Rivela Santos

Dirección: Dirección por asignar

A Coruña, febrero de 2026.

A mi familia, por su apoyo constante en cada etapa de este camino, y a todas las personas que me animaron a seguir aprendiendo cuando el resultado no salía a la primera.

Agradecimientos

Agradezco a mi entorno personal por la paciencia y el apoyo durante el desarrollo de este trabajo. Agradezco también al profesorado de la FIC por la base técnica y metodológica que ha hecho posible este proyecto, y a las personas que me ayudaron con pruebas, validaciones y revisión de resultados.

Resumen

Este trabajo presenta el desarrollo de un sistema de captura y procesado de datos IMU utilizando un dispositivo Android y un entorno de análisis en Python. La solución implementada permite adquirir muestras por red local, almacenar sesiones temporales y aplicar un pipeline de procesado orientado a estimar movimiento vertical relativo a partir de aceleración.

El procesado incluye estimación de frecuencia de muestreo, separación de componente gravitatoria mediante filtrado pasa-bajos, filtrado en banda para aislar la dinámica de interés y cálculo de altura relativa por integración espectral. Para reducir ruido y falsos positivos se añadieron mecanismos de umbral mínimo de cambio y confirmación temporal por persistencia de muestras consecutivas.

Los resultados obtenidos muestran que la arquitectura propuesta es viable y reproducible, y que la selección de parámetros de filtrado condiciona de forma decisiva la sensibilidad del sistema. Como trabajo futuro se plantea incorporar corrección por orientación del dispositivo y automatizar la calibración de umbrales para distintos escenarios de medida.

Abstract

Este traballo presenta o desenvolvemento dun sistema de captura e procesado de datos IMU empregando un dispositivo Android e unha contorna de análise en Python. A solución implementada permite adquirir mostras por rede local, almacenar sesións temporais e aplicar unha cadea de procesado orientada a estimar movemento vertical relativo a partir da aceleración.

O procesado inclúe estimación da frecuencia de mostraxe, separación da compoñente gravitatoria mediante filtrado pasa-baixos, filtrado en banda para illar a dinámica de interese e cálculo da altura relativa por integración espectral. Para reducir ruído e falsos positivos engadíronse mecanismos de limiar mínimo de cambio e confirmación temporal por persistencia de mostras consecutivas.

Os resultados obtidos mostran que a arquitectura proposta é viable e reproducible, e que a selección de parámetros de filtrado condiciona de maneira decisiva a sensibilidade do sistema. Como traballo futuro proponse incorporar corrección por orientación do dispositivo e automatizar a calibración de limiares para distintos escenarios de medida.

Palabras clave:

- Unidad de medida inercial (IMU)
- Android
- Procesado de señal
- Acelerómetro
- Estimación de altura relativa
- Filtrado en frecuencia
- Análisis temporal

Keywords:

- Unidade de medida inercial (IMU)
- Android
- Procesado de sinal
- Acelerómetro
- Estimación de altura relativa
- Filtrado en frecuencia
- Análise temporal

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este trabajo fin de grado aborda el diseño e implementación de una boya instrumentada capaz de caracterizar el oleaje mediante medidas inerciales. El desarrollo se planteó en dos fases: una fase inicial con Android como prototipo rápido de validación y una fase posterior con plataforma embebida ESP32 + IMU + LoRa. El objetivo principal es estimar movimiento vertical relativo a partir de aceleración medida en el eje z, aplicando técnicas de filtrado y análisis en frecuencia para reducir ruido y obtener métricas interpretables.

1.1 Contexto y motivación

En entornos reales, las señales de aceleración incluyen ruido, variaciones de orientación, pequeñas derivas y componentes gravitatorias que dificultan la interpretación directa de la medida [1]. Durante el desarrollo se observó que una aproximación naive basada en leer directamente az produce falsas variaciones de altura cuando el dispositivo está en reposo o cuando cambia ligeramente la inclinación. Esta limitación motiva la construcción de un pipeline de procesado robusto y parametrizable.

En la fase actual, el sistema desarrollado permite:

- Capturar datos IMU desde Android por HTTP en red local.
- Guardar sesiones temporales de medida para análisis offline.
- Procesar las señales con distintos filtros y estrategias de validación temporal.
- Visualizar la evolución de cada etapa para entender el impacto de cada parámetro.

Como siguiente fase, se plantea migrar la captura a ESP32 para aumentar la frecuencia de muestreo efectiva hasta valores cercanos a 1000 Hz en adquisición local, y transportar telemetría por LoRa.

1.2 Objetivos

1.2.1 Objetivo general

Implementar y validar un sistema extremo a extremo para estimación de movimiento vertical relativo a partir de datos IMU, priorizando reproducibilidad experimental y capacidad de ajuste de parámetros.

1.2.2 Objetivos específicos

- Diseñar una arquitectura de comunicación Android–servidor basada en HTTP para validación temprana.
- Implementar un módulo de captura temporal con persistencia en CSV/NPZ.
- Construir un pipeline de procesado con separación de gravedad, filtrado en banda e integración espectral.
- Incorporar mecanismos de reducción de ruido: umbral mínimo y confirmación por persistencia temporal.
- Evaluar el comportamiento del sistema con sesiones de prueba y documentar parámetros efectivos.
- Definir y ejecutar una comparación entre plataforma Android y plataforma ESP32+LoRa en términos de calidad de medida y viabilidad de despliegue en boya.

1.3 Alcance y limitaciones

El alcance del trabajo se centra en estimación de movimiento vertical *relativo* sobre sesiones cortas de captura. No se plantea una estimación absoluta de posición global ni navegación inercial completa, dado que ello requeriría fusión avanzada con magnetómetro/GNSS y modelos de error más complejos [2].

Las limitaciones principales detectadas son:

- Sensibilidad a orientación del dispositivo.
- Dependencia fuerte de la banda de frecuencias elegida.
- Atenuación de movimientos lentos cuando los filtros son demasiado restrictivos.
- Frecuencia de muestreo limitada en Android (aprox. 50 Hz en el prototipo actual).

1.4 Metodología de desarrollo

El trabajo se realizó de forma incremental:

1. Fase 1: prototipo Android para captura por red local y validación del pipeline.
2. Captura estructurada de sesiones de 30 segundos.
3. Procesado offline con métricas de validación y visualización.
4. Ajuste de parámetros con experimentación controlada.
5. Fase 2: migración prevista a ESP32+IMU+LoRa y comparación de plataformas.
6. Documentación de resultados y consolidación de pipeline.

1.5 Estructura de la memoria

La memoria se organiza de la siguiente forma:

- En el Capítulo 2 se describe el diseño técnico, la implementación del sistema, el pipeline de procesado y los resultados de pruebas.
- En el Capítulo 3 se presentan conclusiones, aportaciones y líneas futuras.
- En el Apéndice A se incluye material de soporte y reproducibilidad experimental.

Capítulo 2

Desarrollo del sistema y validación

En este capítulo se detalla la arquitectura implementada, el flujo de captura y el procesado de señal aplicado para estimar movimiento vertical relativo.

2.1 Arquitectura general

La solución se compone de dos bloques:

- **Cliente Android**: adquiere datos de acelerómetro y giroscopio, guarda una copia local en CSV y envía muestras por HTTP en formato JSON.
- **Servidor de captura/procesado**: recibe muestras en POST `/data`, almacena sesiones temporales y ejecuta análisis offline con scripts en Python.

El intercambio mínimo para captura es `{"t": ..., "az": ...}`, aunque el sistema admite también `ax`, `ay`, `gx`, `gy`, `gz`. La base de adquisición de sensores en Android se apoyó en la documentación oficial de la plataforma [3].

2.2 Captura de datos

Se desarrolló el script `capture_http_imu.py`, con las siguientes características:

- Escucha configurable (por defecto `0.0.0.0:8000`).
- Ventana temporal de captura (30 segundos en las pruebas base).
- Corrección de monotonía temporal para evitar `dt <= 0`.
- Exportación en NPZ cuando hay `numpy` o en CSV en modo fallback.

El flujo de red se validó tanto en entorno Linux como en escenario Windows/WSL, utilizando redirección de puertos cuando fue necesario.

2.3 Pipeline de procesado

El procesado principal se implementó en dos scripts: `process_imu_session.py` y `process_imu_step_by_step.py`. Las etapas son:

2.3.1 Estimación de frecuencia de muestreo

A partir del vector temporal `t`, se calcula `fs` mediante la mediana de diferencias positivas `dt`. Este valor gobierna el resto de filtros y transformadas.

2.3.2 Separación de gravedad

Se aplica un filtro pasa-bajos de primer orden para estimar componente lenta g_{est}, y se calcula aceleración dinámica:

$$a_{z,dyn} = a_z - g_{est}$$

2.3.3 Filtrado en banda

Sobre $a_{z,dyn}$ se aplica un band-pass de primer orden $[f_{min}, f_{max}]$ para aislar la banda de interés.

2.3.4 Periodo dominante

Se estima T_p mediante FFT, seleccionando el pico espectral dominante en banda válida [4].

2.3.5 Altura relativa por integración espectral

La altura relativa $h(t)$ se obtiene en frecuencia aplicando:

$$H(f) = \frac{A(f)}{(2\pi f)^2}$$

tras enmascarar frecuencias fuera de banda para reducir deriva e inestabilidad en bajas frecuencias.

2.4 Reducción de ruido y validación temporal

Durante las pruebas se detectaron microvariaciones en reposo. Para reducir falsos positivos se añadieron dos mecanismos:

- **Deadband de altura** (`--height-min-change`): cambios menores que un umbral se ignoran.

- **Confirmación por persistencia**: una subida o bajada solo se confirma si se mantiene durante N muestras consecutivas con signo coherente. Los parámetros usados son `--height-step-min` y `--confirm-samples`.

Este enfoque mejora la robustez frente a ruido impulsivo de una sola muestra.

2.5 Resultados experimentales

Con sesiones de aproximadamente 30 segundos, se obtuvieron resultados consistentes con el movimiento realizado y se observó que la selección de parámetros afecta de forma crítica a la amplitud estimada.

Una configuración sensible que mostró buen comportamiento fue:

- `g_fc = 0.03`
- `fmin = 0.02`
- `fmax = 1.20`
- `height_min_change = 0.00005`
- `height_step_min = 0.000001`
- `confirm_samples = 6`

Se constató que valores estrictos (umbrales altos y confirmación larga) eliminan prácticamente toda la señal útil en capturas suaves, mientras que una configuración demasiado laxa aumenta sensibilidad al ruido.

2.5.1 Gráficas de validación

La Figura 2.1 muestra la vista completa del pipeline sobre la sesión de prueba principal. Incluye señal cruda, eliminación de gravedad, filtrado en banda y altura estimada.

La Figura 2.2 corresponde a un ajuste estricto, con valores:

- `height_min_change=0.005`
- `height_step_min=0.00008`
- `confirm_samples=20`

Con esta configuración, la estimación queda prácticamente anulada por filtrado de ruido.

La Figura 2.3 muestra la configuración sensible seleccionada durante las pruebas. Parámetros utilizados:

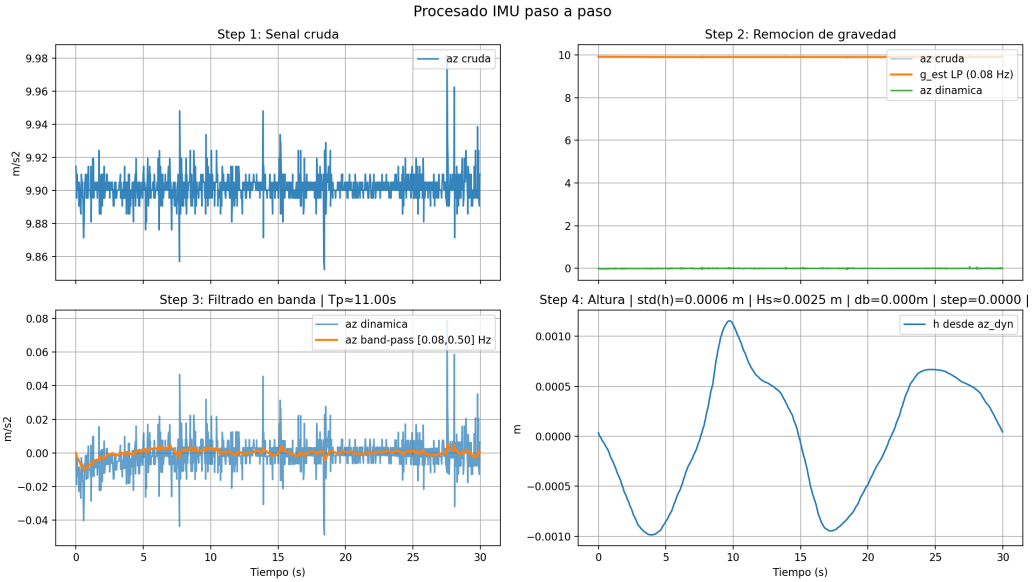

Figura 2.1: Pipeline completo de procesado IMU en modo `all`.

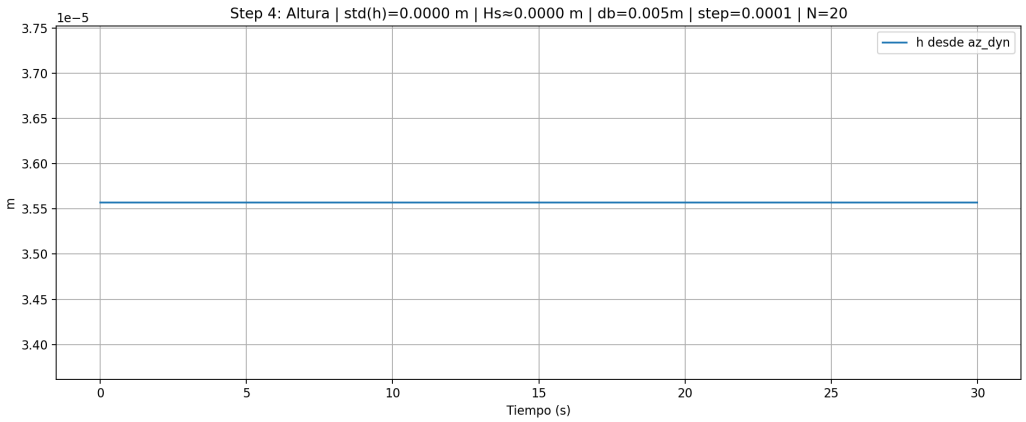

Figura 2.2: Estimación de altura en modo estricto: se minimizan falsos positivos, pero también se pierde señal útil.

- `g_fc=0.03`
- `fmin=0.02`
- `fmax=1.20`
- `height_min_change=0.00005`
- `height_step_min=0.000001`
- `confirm_samples=6`

Esta configuración ofrece mejor compromiso entre detección y robustez.

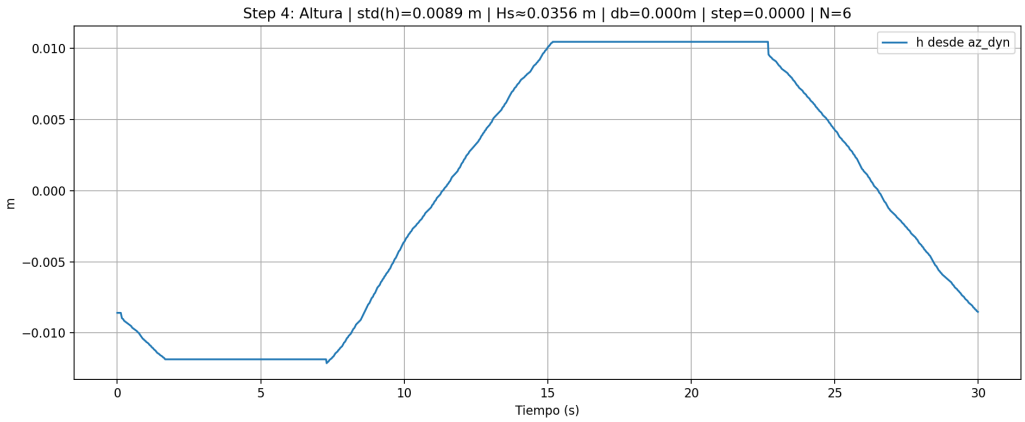

Figura 2.3: Estimación de altura con ajuste sensible: mayor capacidad de detección manteniendo control de ruido.

2.6 Trazabilidad y documentación

Como apoyo a la memoria se generó una bitácora técnica en `INFORME_IMU.md`, donde se registran decisiones, comandos de prueba, resultados de calibración y estado del pipeline por fecha.

2.7 Comparación de plataformas (Fase 1 vs Fase 2)

El objetivo del TFG no se limita al prototipo Android. La estrategia completa contempla:

- **Fase 1 (actual):** validación funcional y metodológica con Android ($\approx$ 50 Hz).
- **Fase 2 (prevista):** implementación en ESP32 + IMU + LoRa, con adquisición local de mayor frecuencia (hasta $\approx$ 1000 Hz según sensor/configuración).

Para justificar la migración de plataforma se definieron los siguientes criterios de comparación:

- Frecuencia de muestreo efectiva y estabilidad temporal.
- Calidad de señal tras procesado (T_p, std(h), H_s).
- Latencia extremo a extremo y continuidad de transmisión.
- Consumo energético y autonomía estimada en escenario de boya.
- Robustez de enlace en campo (pérdida de paquetes y alcance útil).

La Tabla 2.1 resume la plantilla de evaluación experimental.

Métrica	**Android (Fase 1)**	**ESP32+LoRa (Fase 2)**	**Observación**
Frecuencia efectiva (Hz)	≈ 50	Pendiente de medir	Variabilidad por carga configuración
Jitter temporal	Medido en CSV	Pendiente de medir	Evaluar estabilidad del re
T_p, std(h), H_s	Disponible	Pendiente de comparar	Misma metodología de p cesado
Pérdida de muestras	Baja en LAN	Pendiente de medir	Considerar canal LoRa buffers
Latencia de transmisión	Baja en WiFi LAN	Pendiente de medir	Medir en escenario real
Consumo/autonomía	No optimizado	Objetivo de optimización	Clave para despliegue boya

Cuadro 2.1: Plantilla de comparación entre prototipo Android y plataforma embebida ESP32+LoRa.

Como resultado esperado, la plataforma embebida debería mejorar resolución temporal y control de adquisición, a costa de mayor complejidad de integración hardware y comunicaciones.

Capítulo 3

Conclusiones

El trabajo realizado permite concluir que es viable construir un sistema ligero de captura y procesado IMU con herramientas accesibles, manteniendo una cadena completa desde adquisición en Android hasta análisis técnico en escritorio. Esta fase constituye la validación metodológica inicial de una solución mayor orientada a boya instrumentada.

3.1 Conclusiones principales

- Se implementó una solución funcional extremo a extremo para captura y análisis de aceleración vertical relativa.
- Se consiguió una infraestructura reproducible de pruebas basada en sesiones temporales y scripts parametrizables.
- El análisis por etapas facilitó identificar el impacto de cada filtro y corregir configuraciones demasiado agresivas.
- La incorporación de confirmación temporal de cambios redujo falsos positivos asociados a ruido de muestra aislada.
- Se establecieron criterios de comparación para la transición a una plataforma embebida ESP32+LoRa.

3.2 Aportaciones

Las aportaciones más relevantes del trabajo son:

- Diseño modular de scripts de captura y procesado.
- Soporte de formatos de entrada/salida en distintos entornos de ejecución.

- Estrategia práctica de validación basada en umbral mínimo y persistencia temporal.
- Documentación técnica incremental útil para evolución futura del proyecto.

3.3 Trabajo futuro

Como evolución natural del TFG se proponen las siguientes líneas:

- Implementación completa de la Fase 2 en ESP32 con IMU y transmisión LoRa.
- Comparativa experimental Android vs ESP32+LoRa con métricas homogéneas.
- Proyección de aceleración al eje vertical global usando información de orientación del dispositivo.
- Calibración automática de umbrales a partir de un tramo inicial en reposo.
- Generación automática de informes por sesión con métricas y figuras.
- Validación con campañas experimentales controladas y comparación frente a referencia externa.

Apéndices

Apéndice A

Material adicional

En este apéndice se recoge material de apoyo para reproducir el entorno experimental y el flujo de trabajo técnico del proyecto.

A.1 Scripts principales

- `capture_http_imu.py`: recepción HTTP y guardado de sesiones.
- `process_imu_session.py`: procesado offline con métricas y visualización.
- `process_imu_step_by_step.py`: visualización por etapas del pipeline.

A.2 Comandos de referencia

A.2.1 Captura

```
python3 capture_http_imu.py --host 0.0.0.0 --port 8001 --seconds 30
```

A.2.2 Procesado rápido sin gráfica

```
python3 process_imu_session.py session_YYYYMMDD_HHMMSS_30s.csv
    --no-plot
```

A.2.3 Procesado paso a paso con validación temporal

```
python3 process_imu_step_by_step.py session_YYYYMMDD_HHMMSS_30s.csv
    \
  --mode step4 --g-fc 0.03 --fmin 0.02 --fmax 1.20 \
  --height-min-change 0.00005 --height-step-min 0.000001
    --confirm-samples 6
```

A.3 Datos de prueba

Se trabajó con múltiples sesiones de 30 segundos en formato CSV, incluyendo capturas de reposo y capturas con movimiento vertical manual. La bitácora de resultados y ajustes se mantiene en `INFORME_IMU.md`.

Bibliografía

[1] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. The Institution of Engineering and Technology, 2004.

[2] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.

[3] Android Developers, "Sensor overview," https://developer.android.com/guide/topics/sensors/sensors_overview, 2026, accedido el 18 de febrero de 2026.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Prentice Hall, 2010.