Diseño e implementación de un sistema de adquisición transmisión y visualización de datos basado en CanSat
Design and Implementation of a CanSat-Based System for Data Acquisition Transmission and Visualization

Trabajo de Fin de Máster
Curso 2024–2025

Autor
Sergio García Sánchez

Director
Adrián Riesco Rodríguez

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Diseño e implementación de un sistema de adquisición transmisión y visualización de datos basado en CanSat
Design and Implementation of a CanSat-Based System for Data Acquisition Transmission and Visualization

Trabajo de Fin de Máster en Ingeniería Informática
Departamento de XXXXXXXXXXXXX

Autor
Sergio García Sánchez

Director
Adrián Riesco Rodríguez

Convocatoria: *Febrero/Junio/Septiembre 2025*
Calificación: *Nota*

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

DIA de MES de AÑO

Dedicatoria

A Pedro Pablo y Marco Antonio, por crear TeXiS
e iluminar nuestro camino

Agradecimientos

A Guillermo, por el tiempo empleado en hacer estas plantillas. A Adrián, Enrique y Nacho, por sus comentarios para mejorar lo que hicimos. Y a Narciso, a quien no le ha hecho falta el Anillo Único para coordinarnos a todos.

Resumen

Diseño e implementación de un sistema de adquisición transmisión y visualización de datos basado en CanSat

Un resumen en castellano de media página, incluyendo el título en castellano. A continuación, se escribirá una lista de no más de 10 palabras clave.

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Design and Implementation of a CanSat-Based System for Data Acquisition Transmission and Visualization

An abstract in English, half a page long, including the title in English. Below, a list with no more than 10 keywords.

Keywords

10 keywords max., separated by commas.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este Trabajo de Fin de Máster presenta el diseño e implementación de un sistema completo de adquisición, transmisión y visualización de datos en tiempo real inspirado en el concepto CanSat. El proyecto está formado por la construcción de un dispositivo tipo CanSat, con diferentes sensores, GPS, cámara y comunicación por wifi o radio, además, una plataforma web opensource encargada de visualizar los datos recogidos en tiempo real. Esta plataforma se ha diseñado como una herramienta genérica y reutilizable de forma que se pueda adaptar fácilmente a otros proyectos similares. A lo largo del documento se describen el contexto del trabajo, la motivación, los objetivos planteados, la planificación seguida y la estructura de la memoria.

1.1. Contexto

El proyecto CanSat propuesto por el profesor Robert J. Twiggs en 1998 Japan Aerospace Exploration Agency (2003) comienza como un proyecto educativo basado en la simulación de un nanosatélite del tamaño de una lata de refresco y de un peso alrededor de los 350 gramos, el objetivo es ayudar a los alumnos de distintos niveles a entender todas las fases de desarrollo de un satélite, desde la elección de la misión científica hasta la integración del sistema completo, incluyendo los sensores necesarios para obtener los datos necesarios para dicha misión, la electronica necesaria para usar dichos sensores y enviarlos por radio a una estación de tierra, la visualización de los datos, el diseño de una carcasa 3d capaz de aguantar la fuerza del lanzamiento y el diseño de un paracaídas.

Los CanSat no son puestos en órbita, pero son lanzados por cohetes a escala, globos aerostáticos o drones, esto los somete a distintas fuerzas externas como aceleración, vibraciones o posibles impactos, lo que hace que el CanSat tenga que tener una estructura resistente. La misión del CanSat es usar los sensores para recoger datos durante el descenso y transmitirlos a la estación de tierra.

Durante los últimos años la popularidad del proyecto ha ido aumentando, llegando a crear competiciones nacionales e internacionales lideradas por agencias espaciales como la agencia espacial europea European Space Agency (2024) con el objetivo de promover el interés por el sector aeroespacial y por las carreras STEM en general desde pequeños.

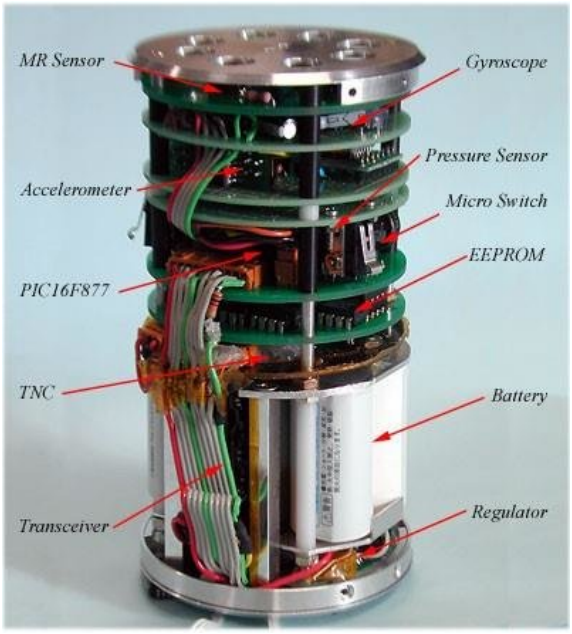

Figura 1.1: Diagrama básico de un CanSat. Fuente: ResearchGate (2018)

1.2. Motivación

Debido a que estos proyectos suelen estar más enfocados en la parte electronica(recogida y transmisión de datos) y no tanto en la visualización de datos, esta última parte suele quedar más descuidada, implementandose solo soluciones básicas como visualización de datos por consola o gráficas simples. Además, estas soluciones son específicas para un CanSat concreto, lo que obliga a implementar soluciones desde cero.

Por ello, surge la necesidad de crear una plataforma común y reutilizable que permita visualizar en tiempo real los datos enviados por el CanSat y recibidos por la antena de manera más visual y profesional sin depender de un hardware concreto. Con la creación de esta plataforma se facilitaría el análisis de los datos durante las pruebas y el lanzamiento, además puede servir como base para futuras implementaciones específicas, ayudando a que los alumnos integren gráficas avanzadas sin tener que desarrollar una plataforma completa.

1.3. Objetivos

El objetivo principal de este proyecto consiste en diseñar y desarrollar desde cero un satélite tipo CanSat y la implementación una plataforma de visualización de datos reutilizable. Para ello se han dividido los objetivos en dos partes diferenciadas: investigación de las tecnologías actuales y desarrollo del CanSat.

- **Investigación**
  - Comparación de los distintos microcontroladores y microcomputadores existentes para determinar cuál se ajusta mejor a los objetivos de nuestro desarrollo.
  - Estudio de los distintos sensores disponibles en el mercado y su comunicación con el microcomputador.

    - Análisis de las diferentes opciones de alimentación para el CanSat y de la posibilidad de cargarse con paneles solares.
    - Comparación de las herramientas actuales de visualización de datos.

- **Desarrollo**, dividido en dos partes: la creación del hardware del CanSat y la plataforma de visualización.
    - CanSat
        - Creación de un CanSat que cumpla con las medidas básicas (66mm x 115mm) y peso (entre 300g y 350g).
        - Integración de receptor GPS
        - Cámara para transmisión de video en tiempo real.
        - Sensor de presión, temperatura y altitud.
        - Giroscopio para obtener la orientación del dispositivo.
        - Batería con posibilidad de carga mediante paneles solares.
        - Retransmisión de los datos por WIFI si el dispositivo tiene conexión a internet.
        - Retransmisión de los datos por radio en caso de que no tenga conexión.
        - Desarrollo de un receptor de radio en la estación terrestre.
    - Plataforma de visualización
        - Visualización en tiempo real de toda la telemetría recibida.
        - Visualización del último valor de cada elemento de la telemetría.
        - Gráficas en tiempo real de los valores recibidos.
        - Visualización en tiempo real de las imágenes transmitidas.
        - Mapa con la ubicación exacta del CanSat.
        - Modelo 3D con la orientación real del CanSat.
        - Descarga de los datos de telemetría para un rango de fechas concreto.

1.4. Plan de trabajo

Durante el desarrollo del proyecto se ha seguido un plan de trabajo con el objetivo de organizar las tareas y alcanzar los objetivos mencionados anteriormente. Este plan de trabajo se ha dividido en los siguientes puntos:

- Revisión y comparación de los microprocesadores Arduino y ESP32 y del microcomputador Raspberry Pi Zero 2 en función de sus capacidades y compatibilidad con los sensores y módulos de comunicación.
- Selección de los sensores (GPS, altímetro y giroscopio), módulo de comunicación (LoRa), cámara y sistema de alimentación.
- Ensamblaje inicial de los distintos componentes electrónicos usando placas de prototipado, lo que facilita las pruebas individuales de cada componente.
- Definición de la arquitectura general del sistema compuesta por tres componentes principales:

    - Código embebido en la Raspberry Pi encargado de leer los sensores y transmisión de datos.
    - Receptor de radio en tierra encargado de la recepción de los datos enviados por el CanSat.
    - Plataforma de visualización en tiempo real.
- Implementación del código embebido en la Raspberry usando Python y las distintas librerías para interactuar con cada sensor.
- Implementación de la plataforma de visualización en tiempo real basada en eventos, usando un sistema de colas RabbitMQ, un backend en Java y Spring Boot, un frontend en Flutter y una base de datos PostgreSQL para guardar los datos de telemetría.
- Soldar los componentes electrónicos en una placa definitiva de un tamaño que cumpla con las medidas reglamentarias de CanSat.
- Realización de pruebas de integración una vez todo el sistema esté terminando, incluyendo pruebas de duración de batería y de alcance de comunicaciones por radio.
- Redacción de la memoria documentando los pasos seguidos en el proyecto y los resultados obtenidos.

1.5. Organización de la memoria

La memoria se compone de los siguientes capítulos:

Capítulo 2

Trabajo relacionado

En este capítulo vamos a contextualizar el concepto CanSat y se revisarán trabajos relacionados con este tipo de sistemas. Primero se describirá en detalle que es un CanSat y las principales iniciativas educativas que los promueven, algunas competiciones educativas y normativa para participar. A continuación, se presentarán las soluciones más habituales de adquisición y transmisión de datos, así como las herramientas existentes para visualización de telemetría en tiempo real.

2.1. Proyectos educativos y competiciones CanSat

Desde su creación en 1998 por Robert J. Twiggs, el concepto CanSat ha sido muy utilizado para referirse a satélites suborbitales de bajo coste, ligeros y de tamaño contenido. El objetivo de estos satélites está mayormente asociado al ámbito educativo, donde se usan para dar una introducción a los estudiantes al desarrollo real de una misión espacial, teniendo en cuenta todas las fases de una misión científica real, definición de los objetivos científicos, elección de los componentes, diseño de la arquitectura, ensamblaje de los componentes, diseño de la estación de tierra y visualización y procesado de los datos.

En la actualidad, varias instituciones y agencias espaciales realizan competiciones anuales dirigidas a estudiantes de todas las edades para desarrollar sus propios CanSat. Estas competiciones se basan en lanzar el CanSat con un cohete a escala, desde un drone o globo aerostático y recibir datos durante el tiempo de descenso.

Algunas de estas competiciones son la American CanSat Competition (2025), organizada por la American Astronautical Society (2025) dirigida a estudiantes universitarios y European Cansat Competition (2025) organizada por la European Space Agency (2025) y dirigida a grupos de estudiantes de entre 14 y 19 años. Esta competición organiza torneos regionales y una final a nivel europeo con los mejores representantes de cada país.

Los requisitos que debe cumplir un CanSat para participar en una competición europea son:

- Deben tener una altura máxima de 115mm y un diámetro máximo de 66mm.
- Un peso entre 300 y 350 gramos.
- Deben cumplir una misión principal basada en medir la presión y temperatura del

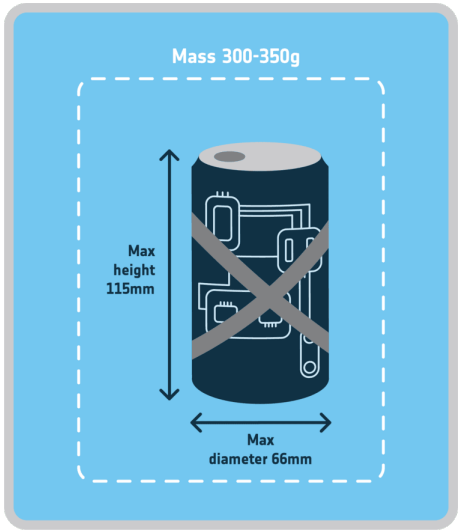

Figura 2.1: Medidas máximas de un CanSat para una competición europea

aire y enviarlo a la estación de tierra al menos una vez por segundo.

- Una misión secundaria elegida por cada equipo.
- No deben incluir materiales inflamables, explosivos o peligrosos para el medio ambiente.
- Debe estar alimentado por batería o paneles solares que lo mantengan encendido durante al menos cuatro horas.
- La batería debe ser accesible y fácil de reemplazar o cargar.
- Incluir un interruptor de encendido y apagado accesible.
- Incluir un paracaídas que facilite la recuperación del CanSat después del lanzamiento y que garantice un tiempo de vuelo máximo de 120 segundos.
- El ratio de descenso debe estar entre 8 y 11m/s.
- Soportar una aceleración de hasta 20g.
- El coste total del CanSat no puede superar los 500€.

2.2. Sistemas de adquisición y transmisión de datos para CanSat

Los sistemas de adquisición y transmisión de datos utilizados en los CanSat siguen una arquitectura similar a la empleada en las misiones espaciales reales. Esta arquitectura se basa en dos flujos de comunicación realizados entre el satélite y la estación de tierra:

- **Uplink:** Enviar telecomandos al satélite desde tierra, en el caso de los CanSat esto es poco común.
- **Downlink:** Recibir telemetría desde el satélite a la estación de tierra.

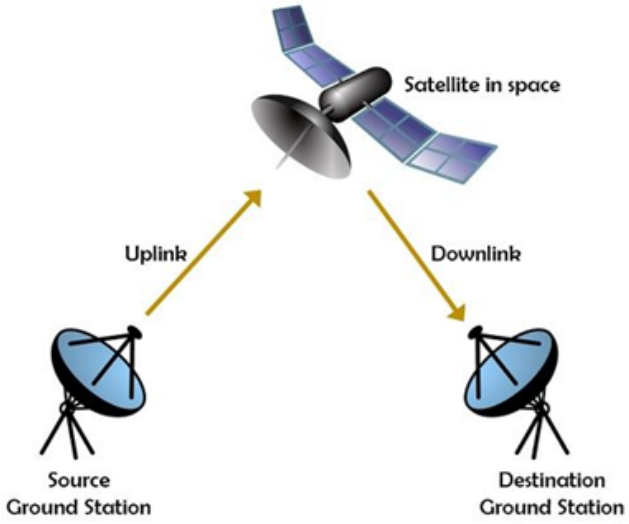

Figura 2.2: Ejemplo de comunicación Uplink/Downlink

Para llevar a cabo esta comunicación, los CanSat deben incluir una serie de componentes básicos:

- **Ordenador a bordo (OBC, On-Board Computer):** Encargado de la comunicación con los sensores y el módulo de transmisión, Puede estar basado en microcontroladores (como Arduino o ESP32) o microcomputadores (como Raspberry Pi).
- **Módulo de comunicación:** se encarga de la transmisión de datos mediante tecnologías como LoRa Corporation (2015), XBee International (2024) o Wi-Fi, dependiendo del alcance necesario, la elección de la frecuencia de transmisión depende tanto del módulo que se utilice como de la normativa de cada país, además, debe poderse cambiar con facilidad para no interferir con otros CanSat.
- **Sensores:** permiten adquirir información relevante sobre el entorno, como presión atmosférica, temperatura, orientación (IMU) o localización geográfica (GPS).
- **Fuente de alimentación:** normalmente una batería recargable, capaz de mantener operativo el sistema durante toda la misión. En algunos casos puede complementarse con pequeños paneles solares.

Estos componentes forman la base mínima que un CanSat necesita para poder llevar a cabo la misión científica y comunicar los datos con la estación de tierra.

2.3. Soluciones existentes para telemetría y visualización de datos

Normalmente, los proyectos CanSat están más enfocados en el diseño electrónico y la comunicación por radio, por lo que se suele desarrollar menos la parte de visualización de

datos, utilizando por lo general herramientas genéricas o limitadas al ordenador en el que está corriendo la estación de tierra, algunas de estas herramientas son:

- **SerialPlot**:herramienta ligera que permite graficar en tiempo real los datos recibidos por puerto serie. Es muy usada durante el desarrollo por su sencillez y rapidez para validar sensores, aunque no permite guardar datos ni personalizar la interfaz.
- **Matplotlib** Hunter (2003), **PyQtGraph** o **Tkinter**: librerías desarrolladas en python, permiten construir interfaces personalizadas, pero no están pensadas para ser accesibles a través de una interfaz web y requieren conocimientos de programación hasta para los casos más simples.
- **Excel** o **Sheets**: se utilizan exportando directamente los datos recibidos, no requieren conocimientos de programación pero no permiten la visualización en tiempo real.
- **LabVIEW National Instruments** entorno gráfico profesional diseñado para adquisición y visualización de datos en sistemas embebidos e instrumentación industrial. Permite construir interfaces complejas mediante programación visual y cuenta con soporte nativo para muchos dispositivos de hardware. Aunque es muy potente, tiene un coste elevado de licencia y una curva de aprendizaje considerable, por lo que raramente se utiliza en proyectos educativos como CanSat.

En resumen, la gran mayoría de herramientas son o muy básicas o demasiado potentes para el uso necesario en un CanSat, además ninguna de estas es específica para el tipo de datos y gráficos habituales en este tipo de proyectos.

Capítulo 3

Fundamentos teóricos

Este capítulo se enfoca en analizar en profundidad los aspectos técnicos necesarios para llevar a cabo el desarrollo del CanSat. Se incluye una comparativa entre los distintos microcontroladores y microcomputadores disponibles en el mercado, evaluando su precio, disponibilidad y adecuación a los requisitos técnicos del proyecto. Además, se analizan los sensores y módulos de comunicación más comunes, así como los principales protocolos utilizados para la comunicación entre el microcontrolador y los sensores, como I2C y UART. También se estudian distintas opciones para la retransmisión de vídeo en tiempo real desde el CanSat. Por último, se exploran soluciones para la visualización de los datos a través de una interfaz web en tiempo real.

3.1. Comparativa de microcontroladores y microcomputadores

Uno de los componentes principales de un CanSat o de cualquier sistema embebido en general es su microcontrolador o microcomputador, es el encargado de comunicarse con los sensores, procesar los datos y transmitirlo a través del módulo de comunicación, también es el encargado de procesar el video y retransmitirlo en tiempo real(si el hardware lo permite).

Primero conviene aclarar la diferencia entre microcontrolador y microcomputador:

- **Microcontrolador:** Circuito integrado que combina procesador, memoria y periféricos de entrada/salida en un solo chip. Está diseñado para realizar tareas específicas con bajo consumo energético y recursos limitados. Es común en aplicaciones como lectura de sensores, control de motores o gestión de comunicaciones básicas. Ejemplos comunes son Arduino Uno o ESP32
- **Microcomputador:** Sistema completo en una sola placa (Single-Board Computer) que integra procesador, memoria, almacenamiento y puertos de expansión. Es capaz de ejecutar sistemas operativos completos (como Linux) y realizar tareas más complejas, como procesamiento de imágenes, servidor web o interfaces gráficas. Un ejemplo típico es la Raspberry Pi Zero.

Actualmente, en el mercado se pueden encontrar varios de estos microcontroladores o microcomputadores de bajo coste, tamaño y peso reducido y bajo consumo. En esta sección

nos vamos a centrar en el análisis de tres de las opciones más populares:

- **Raspberry Pi Zero 2 W:** La Raspberry Pi Zero 2 es un Single Board Computer de bajo costo lanzado en Reino Unido por la Raspberry Pi Foundation, de toda la familia Raspberry Pi se ha elegido el modelo Zero 2 por ser el de tamaño más reducido pero con una potencia adecuada para el desarrollo de un proyecto como el CanSat. Las características más relevantes para este proyecto son:
  - CPU Arm Cortex-A53 de cuatro núcleos y 64 bits a 1 GHz.
  - SDRAM de 512 MB.
  - LAN inalámbrica de 2,4 GHz 802.11 b/g/n.
  - Bluetooth 4.2, Bluetooth Low Energy (BLE), antena integrada.
  - Ranura para tarjeta microSD.
  - Conector de cámara CSI-2.
  - Cabecera GPIO de 40 pines para conexión de periféricos.
  - 1 × SPI
  - 2 × interfaces I^2C
  - 1 × UART
  - Consumo típico de energía: entre 0.7 W y 1.5 W dependiendo de la carga de trabajo.
  - Precio aproximado: 15–20€.

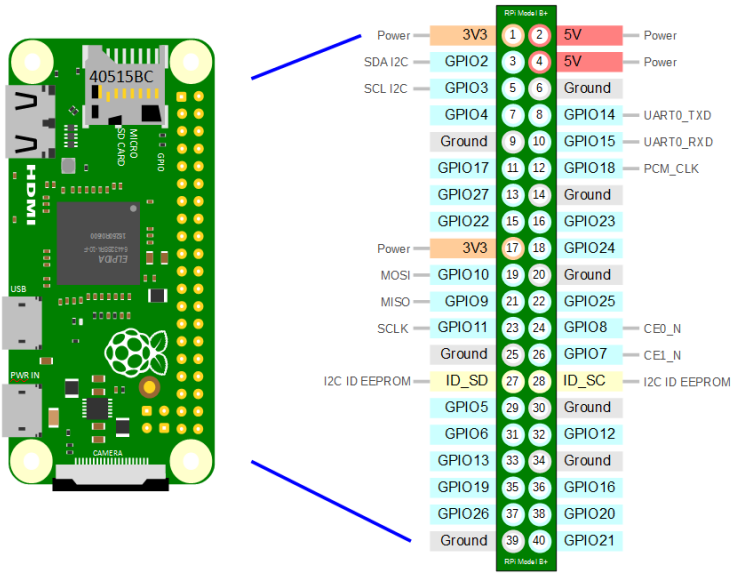

Figura 3.1: Distribución de pines GPIO en Raspberry Pi Zero 2

Como se puede ver en las características la Raspberry Pi Zero 2 cumple con los requisitos necesarios para este proyecto, cuenta con un procesador y memoria adecuados, soporte para cámara (útil para la retransmisión de video en tiempo real), conectividad inalámbrica mediante WiFi y compatibilidad con los protocolos I^2C y UART mediante los pines GPIO necesarios para la conexión directa de sensores.

- **ESP32:** Es un microcontrolador de bajo coste desarrollado por Espressif Systems que combina bajo coste con buena capacidad de procesamiento y conectividad inalámbrica integrada, lo que lo convierte en una de las opciones más populares para proyectos embebidos, incluyendo CanSat.

  A diferencia de un microcomputador como la Raspberry Pi, el ESP32 no ejecuta un sistema operativo generalista, pero su bajo consumo energético y la integración de múltiples periféricos lo hacen convierten en muy buena opción cuando se busca eficiencia y simplicidad del sistema.

  Las características más relevantes para este proyecto son:

  - CPU dual-core Tensilica Xtensa LX6 a 240 MHz.
  - 520 KB de SRAM interna.
  - Memoria flash externa: normalmente 4 MB (dependiendo del modelo).
  - Conectividad Wi-Fi 802.11 b/g/n.
  - Bluetooth 4.2 y BLE.
  - 4 × SPI
  - 2 × interfaces I^2C
  - 3 × UART
  - Hasta 34 pines GPIO (según versión del módulo).
  - Consumo típico: entre 0.2 W y 0.6 W, dependiendo del modo de operación.
  - Precio aproximado: 4–8€ ESP32.

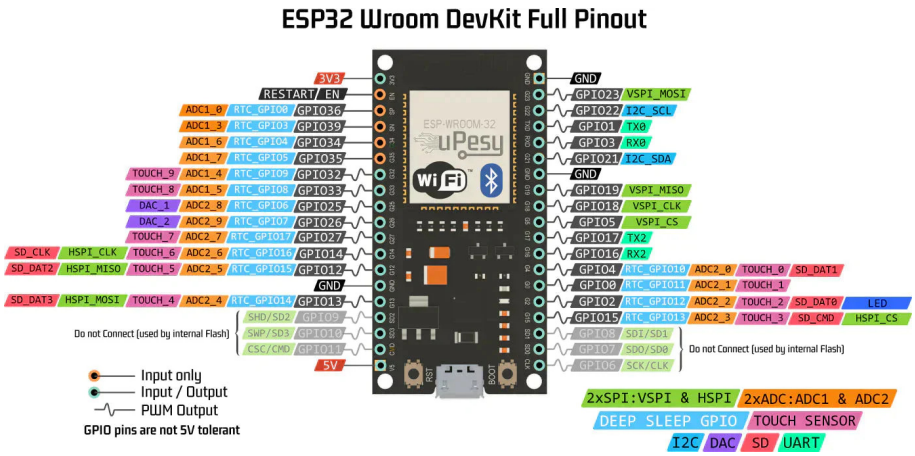

Figura 3.2: Distribución de pines GPIO en ESP32

Gracias a su bajo consumo, potencia y multiples conexiones, el ESP32 permite integrar sensores fácilmente a través de interfaces estándar y puede encargarse tanto de la adquisición como de la transmisión de datos por radio o Wi-Fi. Además, su bajo consumo lo hace especialmente adecuado para sistemas alimentados por batería en entornos con restricciones energéticas. También existen variantes como el ESP32-CAM que integran una cámara de tipo OV2640, lo que permite capturar imágenes y transmitir video mediante Wi-Fi aunque con un rendimiento y resolución inferior a la de Raspberry Pi.

- **Nano:** Es un microcontrolador compacto de bajo coste basado en el chip ATmega328P, es el más utilizado en entornos educativos gracias a su simplicidad, por lo que cuenta con una amplia comunidad detrás y desarrollo de librerias. A diferencia de la Raspberry Pi Zero 2 o el ESP32, el Arduino Nano no cuenta con conectividad inalámbrica ni capacidad de procesamiento avanzada, pero es suficiente para gestionar sensores básicos y transmitir datos mediante un módulo externo de radio.

  Las características más relevantes para este proyecto son:

  - Microcontrolador ATmega328P.
  - Frecuencia de reloj: 16 MHz.
  - Memoria flash: 32 KB (2 KB utilizados por el bootloader).
  - SRAM: 2 KB.
  - EEPROM: 1 KB.
  - 22 pines GPIO (14 digitales, 8 analógicos).
  - 1 × interfaz I²C.
  - 1 × UART.
  - 1 × SPI.
  - Consumo típico: entre 0.05 W y 0.2 W.
  - Precio aproximado: 27€ en la web oficial.

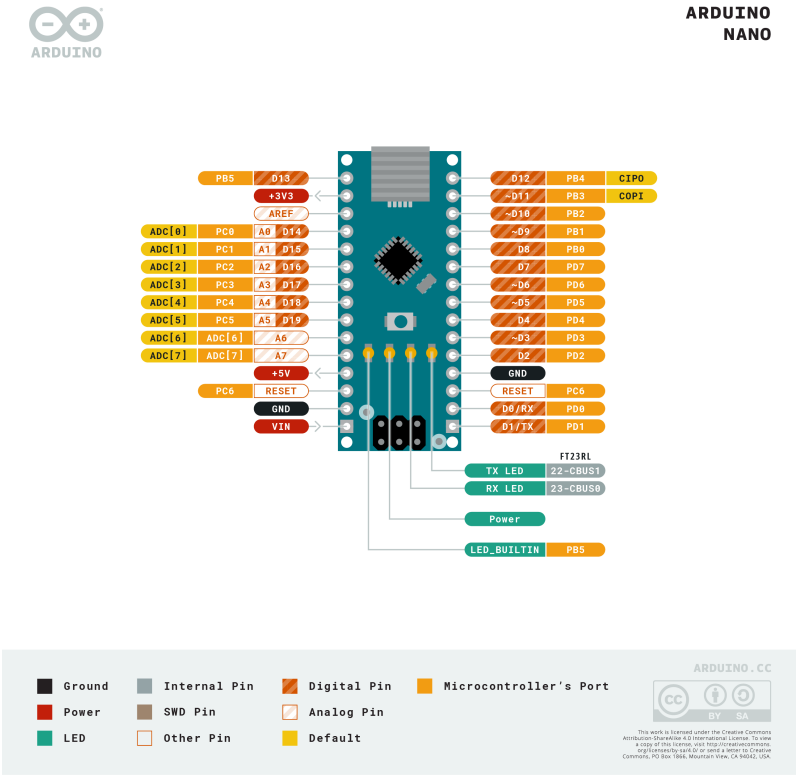

Figura 3.3: Distribución de pines en Arduino Nano

Aunque no tiene las capacidades de procesamiento de una Raspberry Pi ni la conectividad integrada del ESP32, el Arduino Nano puede ser una solución válida para

CanSat muy simples, en los que se priorice el consumo mínimo y no se necesiten funcionalidades avanzadas como WiFi o procesamiento de vídeo, sin embargo, al no tener soporte para cámaras, no cumple con los requisitos técnicos necesarios para este proyecto.

Característica	**Raspberry Pi Zero 2**	**ESP32**	**Arduino Nano**
Procesador	ARM Cortex-A53 (4×, 1 GHz)	Xtensa LX6 (2×, 240 MHz)	ATmega328P (1×, 16 MHz)
Memoria	512 MB SDRAM	520 KB SRAM + 4 MB Flash	2 KB SRAM + 32 KB Flash
Wi-Fi	Sí	Sí	No
Bluetooth	4.2 + BLE	4.2 + BLE	No
SPI	1	4	1
I²C	2	2	1
UART	1	3	1
Compatibilidad cámara	CSI (cámara oficial)	OV2640 (ESP32-CAM)	No
Consumo típico	0.7–1.5 W	0.2–0.6 W	0.05–0.2 W
Precio estimado	15–20 €	4–8 €	25–30 €

Tabla 3.1: Comparativa de microcontroladores y microcomputadores

3.2. Interfaces de comunicación serie

Una vez analizados sobre los distintos microcontroladores y microprocesadores que existen en el mercado para este tipo de proyectos, es importante entender los distintos tipos de interfaces de comunicación que utilizan para interactuar con sensores y otros módulos externos y como funcionan. En esta sección se presentan tres de los más relevantes: SPI, I²C y UART

- **SPI:** La interfaz SPI (Serial Peripheral Interface) Dhaker (2018) es una interfaz síncrona y full dúplex basada en una arquitectura maestro-esclavo, los dos dispositivos, maestro y esclavo pueden transmitir datos simultáneamente sincronizados con una señal de reloj. La interfaz SPI utiliza cuatro señales:
  - Señal de reloj (CLK)
  - Selección de chip (CS)
  - Salida del maestro hacia el esclavo (MOSI)
  - Salida del esclavo hacia el maestro (MISO)

  El maestro genera la señal de reloj y controla el intercambio de datos. El pin CS selecciona el estado activo y los pines MISO y MOSI transportan datos en ambas direcciones. Para iniciar la comunicación, el maestro activa el pin CS y empieza a emitir la señal de reloj, al ser una interfaz full dúplex, maestro y esclavo pueden enviar y recibir datos simultáneamente. Esta interfaz tiene un solo maestro y puede tener uno o múltiples esclavos. En configuraciones con múltiples esclavos se pueden conectar de dos modos:
  - **Modo regular:** Cada nodo tiene su propia línea de CS
  - **Modo cadena (daisy-chain):** Todos los nodos comparten el mismo reloj y CS y los datos se propagan de un esclavo al siguiente. De esta manera se reduce el número de GPIO necesarios en el maestro, aunque aumenta el número de ciclos de reloj requeridos para llegar a cada esclavo.

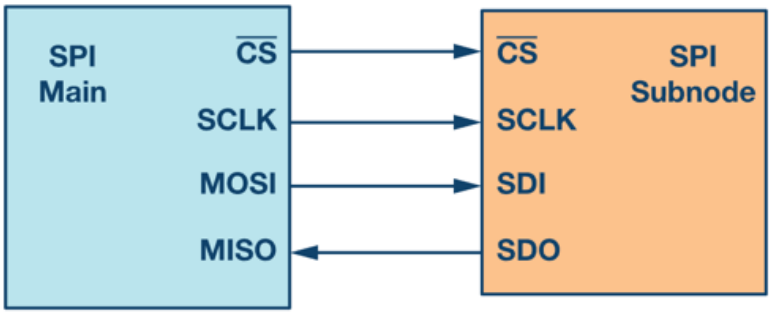

Figura 3.4: Configuración SPI con un maestro y un esclavo

La velocidad de transferencia puede variar dependiendo del hardware utilizado, lo habitual es entre 1 y 10Mbps, aunque algunos dispositivos permiten velocidades superiores.

- **I²C:** La interfaz I²C (Inter-Integrated Circuit) Semiconductors (2014) es un bus de comunicación síncrono y half dúplex basado en una arquitectura maestro-esclavo, que utiliza solo dos líneas para comunicarse con múltiples dispositivos, originalmente fue desarrollada por Philips Semiconductors en 1982. Esta interfaz utiliza solo dos señales:
  - Línea de datos (SDA)
  - Línea de reloj (SCL)

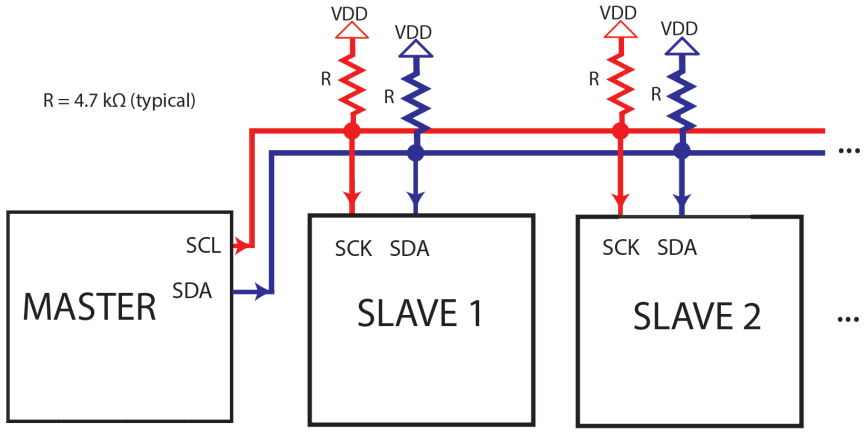

Figura 3.5: Configuración I²C con un maestro y varios esclavos

Ambas líneas son bidireccionales aunque al ser half dúplex la comunicación se realiza en un sentido a la vez. Un dispositivo actúa como maestro, iniciando la comunicación y generando la señal de reloj, mientras que uno o más esclavos responden. Cada dispositivo conectado al bus tiene una dirección única.

La comunicación se inicia cuando el maestro envía una condición de inicio, la dirección de uno de los dispositivos conectados al bus y un bit indicando si va a leer o escribir. Después de transmitir cada byte, el receptor envía una señal de reconocimiento (ACK). La transmisión termina cuando se envía una condición de parada.

El bus I²C permite velocidades de transferencia de hasta 100 kbit/s en modo estándar (Standard-mode), 400 kbit/s en modo rápido (Fast-mode), 1 Mbit/s en modo fast-mode plus (Fm+), y hasta 3.4 Mbit/s en modo high-speed (Hs-mode), dependiendo de las capacidades del hardware.

- **UART:** La interfaz UART (Universal Asynchronous Receiver-Transmitter) AG (2018) es una interfaz de comunicación asíncrona basada en una arquitectura punto a punto, que permite la transmisión de datos en serie entre dos dispositivos. A diferencia de las interfaces anteriores, que eran síncronas, UART es asíncrona, por lo que no utiliza una señal de reloj compartida, sino que cada dispositivo funciona con una velocidad de transmisión acordada previamente y común para los dos (baud rate). UART emplea dos líneas de comunicación:

  - Transmisión de datos (TX)
  - Recepción de datos (RX)

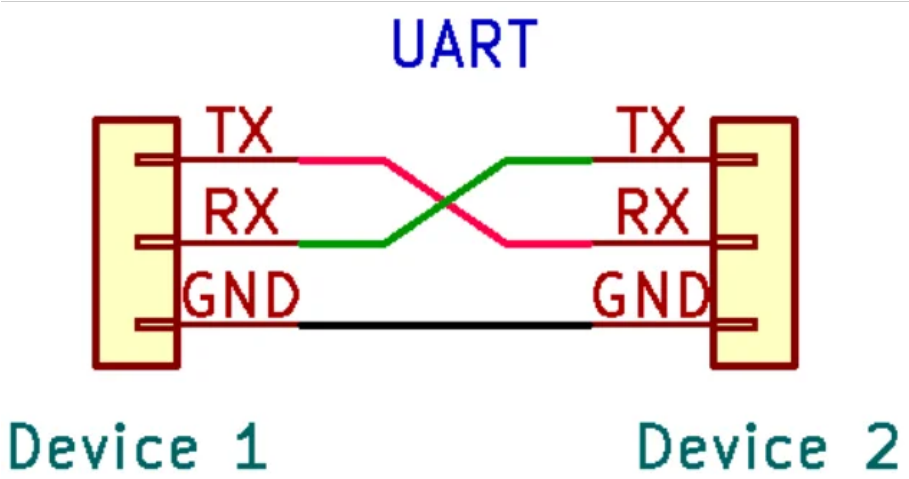

Figura 3.6: Ejemplo de conexión UART

La línea TX de un dispositivo debe ir conectada a la línea RX del otro dispositivo, y la línea RX a TX.

La comunicación es full dúplex, permitiendo la transmisión y recepción de datos de forma simultánea. Cada byte se transmite en una trama que incluye un bit de inicio (start bit), los bits de datos (normalmente 8), un bit opcional de paridad, y uno o más bits de parada (stop bits).

Las velocidades de transmisión habituales van desde 9600 hasta 115200 baudios, aunque se pueden alcanzar velocidades de hasta 1 o 2 Mbps, dependiendo del hardware.

Característica	**SPI**	**I²C**	**UART**
Tipo de comunicación	Síncrona	Síncrona	Asíncrona
Arquitectura	Maestro-esclavo	Maestro-esclavo	Punto a punto
Número de líneas	4 (CLK, CS, MOSI, MISO)	2 (SDA, SCL)	2 (TX, RX)
Full/Half dúplex	Full dúplex	Half dúplex	Full dúplex
Número de dispositivos	1 maestro, varios esclavos	1 maestro, varios esclavos	Solo dos
Velocidad típica	1–10 Mbps	100 kbit/s – 3.4 Mbit/s	9.6 kbit/s – 3 Mbps
Control de dirección	Señal CS por esclavo	Dirección en el protocolo	No necesario
Complejidad del hardware	Media	Baja	Muy baja

Tabla 3.2: Comparativa entre las interfaces SPI, I²C y UART

3.3. Transmisión de datos mediante LoRa

3.4. Sensores embarcados: presión, orientación y GPS

3.5. Captura y transmisión de vídeo en tiempo real

3.6. Visualización de datos en tiempo real: arquitecturas orientadas a eventos

3.7. Modelado 3D de orientación con cuaterniones

Capítulo 4

Diseño e implementación del sistema

4.1. Montaje electrónico del CanSat

4.2. Código embebido en Raspberry Pi

4.3. Integración con RabbitMQ

4.4. Implementación del backend en Spring Boot

4.5. Persistencia de datos con PostgreSQL

4.6. Frontend Flutter para visualización en tiempo real

4.7. Pruebas de integración

Capítulo 5

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Antes de la entrega de actas de cada convocatoria, en el plazo que se indica en el calendario de los trabajos de fin de máster, el estudiante entregará en el Campus Virtual la versión final de la memoria en PDF. En la portada de la misma deberán figurar, como se ha señalado anteriormente, la convocatoria y la calificación obtenida. Asimismo, el estudiante también entregará todo el material que tenga concedido en préstamo a lo largo del curso.

Chapter 6

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter 1.

Chapter 7

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter 5.

Bibliografía

AG, I. T. *Component UART V2.50 - Software Module Datasheet*, 2018.

American Astronautical Society. American astronautical society – advancing space science and exploration. Electronic resource, sitio oficial, 2025.

American CanSat Competition. Cansat competition — design-build-fly aerospace challenges. Electronic resource, official website, 2025.

Corporation, S. Lora: Long range, low power wireless platform. *White Paper*, 2015.

Dhaker, P. Introduction to spi interface. *Analog Dialogue*, vol. 52, 2018.

ESP32. Esp32 series datasheet. `https://www.espressif.com/en/products/socs/esp32`, ????

European Cansat Competition. Cansat – european space agency. Electronic resource, ESA Education, 2025.

European Space Agency. Build your can-sized satellite with cansat 2024–2025. ESA Education, 2024. `https://www.esa.int/Education/Teachers_Corner/Build_your_can-sized_satellite_with_CanSat_2024-2025`.

European Space Agency. European space agency (esa). Sitio web institucional, 2025.

Excel, M. Microsoft excel. `https://www.microsoft.com/en-us/microsoft-365/excel`, ????

Hunter, J. D. Matplotlib: A 2d graphics environment. `https://matplotlib.org/`, 2003. Accedido en julio de 2025.

International, D. Xbee rf modules. `https://www.digi.com/xbee`, 2024.

Japan Aerospace Exploration Agency. What is cansat? `https://stage.tksc.jaxa.jp/taurus/member/miyazaki/old/e/CanSat.html`, 2003. `https://stage.tksc.jaxa.jp/taurus/member/miyazaki/old/e/CanSat.html`.

LabVIEW National Instruments. Labview system design software. `https://www.ni.com/en-us/shop/labview.html`, ????

Nano, A. Arduino nano. `https://store.arduino.cc/products/arduino-nano`, ????

PyQtGraph. Pyqtgraph - scientific graphics and gui library for python. http://www.pyqtgraph.org/, ????

Raspberry Pi Foundation. Raspberry Pi Foundation - Official Website. https://www.raspberrypi.org/, ????

Raspberry Pi Zero 2 W. Raspberry Pi Zero 2 W. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/, ????

ResearchGate. Atmospheric data measurement using can-sat and data logging in ground station - scientific figure on researchgate. https://www.researchgate.net/figure/A-typical-CanSat-1_fig1_326505110, 2018.

Semiconductors, N. Um10204 i²c-bus specification and user manual. https://www.nxp.com/docs/en/user-guide/UM10204.pdf, 2014.

SerialPlot. Serialplot - real-time plotting software. https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-plotter/, ????

Sheets, G. Google sheets. https://www.google.com/sheets/about/, ????

Tkinter. Tkinter — python interface to tcl/tk. https://docs.python.org/3/library/tkinter.html, ????

Apéndice A

Título del Apéndice A

Contenido del apéndice

Apéndice B

Título del Apéndice B