S-ART Training Program

Bloque 0 — Fundamentos y Herramientas

Student Astronomical Radio Telescope (S-ART)

Propósito del Bloque

El objetivo de este bloque es transformar a un participante sin experiencia previa en alguien capaz de:

- Entender qué es un CubeSat y cómo funciona una misión espacial.
- Trabajar en un repositorio colaborativo usando Git y GitHub.
- Generar e interpretar datos básicos de telemetría.

Regla fundamental del curso:

> Si no está en el repositorio, no existe.
> Si no se entiende, no está bien hecho.
> Si no se puede explicar, no se ha aprendido.

1 B0.1 — Introducción a CubeSats

1.1 Objetivo

En este bloque vas a adquirir una visión de alto nivel sobre qué es un CubeSat y cómo se entiende como **un sistema completo**. Para ello, leerás los recursos indicados y responderás a una serie de preguntas que te obligan a conectar:

- El concepto de CubeSat
- Los subsistemas principales
- El flujo de datos desde el payload hasta Tierra
- La misión S-ART como caso de estudio

1.2 Recurso principal

Lectura obligatoria:

- **NASA CubeSat 101** Link al libro

Charla introductoria:

- **Conferencia Julio Gallegos:** Ver charla en Google Drive

S-ART Mission Statement:

- **S-ART Mission Statement** Ver en Google Drive

1.3 Conceptos clave

- Qué es un CubeSat
- Estándar 1U, 3U, etc.
- Subsistemas:
  - OBC (On-Board Computer)
  - EPS (Electrical Power System)
  - TT&C (Telemetry, Tracking & Command)
  - AOCS (Attitude and Orbit Control System)
  - Payload
- Flujo de datos: Payload → OBC → TT&C → Tierra

1.4 Actividad

Entrega B0.1 — CubeSat Fundamentals

Responder:

1. ¿Qué es un CubeSat y por qué existe este estándar?
2. Describe los subsistemas principales.
3. ¿Qué papel tiene el payload?
4. Explica el flujo de datos hasta Tierra.
5. ¿Cómo encaja S-ART como misión?
6. Idea una misión para tu propio CubeSat y descríbelo brevemente.
7. Propón un nombre para tu misión y justifícalo.
8. ¿Qué desafíos crees que enfrentaría tu misión en términos de diseño y operación?
9. ¿Qué tipo de datos esperas recibir de tu CubeSat y cómo los usarías?

Extra: Realizar un diagrama del satélite.

La actividad se considera completada cuando entregues los **entregables** indicados y realices una **explicación oral breve** (2–3 minutos) donde presentes el concepto de tu misión.

1.5 Entregables

- `B0/docs/cubesat_notes/<nombre>_<apellido>.pdf`
- `B0/docs/cubesat_notes/<nombre>_<apellido>_diagram.png`
- `B0/docs/cubesat_notes/<nombre>_<apellido>_presentacion.pptx`
- `B0/docs/cubesat_notes/<nombre>_<apellido>_presentacion.pdf`

Explicación oral (2–3 minutos)

Tu explicación oral debe cubrir, de forma simple y clara:

1. **Nombre de la misión** y por qué lo has elegido.
2. **Objetivo principal**.
3. **Payload** propuesto (qué dato produce o mide).
4. **Dato esperado** (qué esperas recibir) y **para qué lo usarías**.
5. **Desafío principal** (elige uno) y una mitigación básica.

1.6 Criterios de aceptación

- Explicación clara y original
- Diagrama comprensible
- Capacidad de explicación oral

2 B0.2 — Git y GitHub aplicado a S-ART

2.1 Objetivo

En este bloque aprenderás el flujo básico de colaboración usado en proyectos reales: **trabajar en una rama, hacer commits claros, abrir una Pull Request (PR) y responder a un code review**. La idea no es "usar comandos por usar", sino adquirir un método de trabajo que hace que el equipo pueda:

- trabajar en paralelo sin romper el repositorio,
- revisar cambios antes de integrarlos,
- y dejar evidencia/documentación de lo que se ha hecho.

2.2 Conceptos clave

- Repositorio
- Rama (`branch`)
- Commit
- Pull Request (PR)
- Code review

2.3 Recursos

- GitHub Skills: `https://skills.github.com/`
- Learn Git Branching: `https://learngitbranching.js.org/`
- Repositorio del curso: `https://github.com/S-ART-manifold/Curso_de_formacion`

2.4 Actividad

Entrega B0.2 — First Contribution

1. Clonar el repositorio S-ART
2. Crear una rama:

```
git checkout -b feature/<nombre>-cubesat-notes
```

3. Añadir la entrega B0.1
4. Realizar commits claros
5. Subir cambios:

```
git push -u origin feature/<nombre>-cubesat-notes
```

6. Abrir Pull Request
7. Responder al review

2.5 Estructura del repositorio

```
repo-root/
B0/
  data/
  src/
    telemetry/
  outputs/
  docs/
    00_onboarding/
    cubesat_notes/
```

```
      submissions/
    runbooks/
    tm_reports/
B1/
B2/
shared/
  templates/
  runbooks/
```

2.6 Criterios de aceptación

- PR aprobado
- Uso correcto de ramas
- Commits claros
- Respuesta adecuada al review

Esta parte se considera completada cuando tu entrega B0.1 está añadida al repositorio mediante una **Pull Request aprobada** (con al menos 1 revisión) y has demostrado que sabes iterar sobre tu PR cuando te piden cambios.

3 Python Primeros Pasos — Nivelación (opcional)

3.1 Objetivo

Esta sección está pensada para participantes sin experiencia previa en programación.

El objetivo es adquirir los conocimientos mínimos necesarios para poder completar el Bloque B0.3.

3.2 Contenidos mínimos

Se espera que el participante entienda:

- Variables y tipos básicos
- Estructuras de control (if, for)
- Funciones simples
- Ejecución de scripts desde terminal

3.3 Recursos recomendados

- Python for Everybody (Coursera)
- W3Schools Python: `https://www.w3schools.com/python/`

3.4 Actividad recomendada

Antes de comenzar B0.3, el alumno debería ser capaz de ejecutar un script sencillo como:

```
for i in range(10):
    print(i)
```

Es casi obligatoria la realización de los bloques introductorios de w3schools y sobre todo el apartado de matplotlib.

3.5 Nota

No es necesario dominar Python. Solo se requiere la capacidad de leer, modificar y ejecutar scripts básicos.

4 B0.3 — Análisis de telemetría (laboratorio guiado)

Para esta práctica trabajaremos como si fuésemos el equipo de operaciones de una misión espacial.

La misión **CASE-01** es un CubeSat en órbita que está realizando una fase temprana de comisionado y pruebas de subsistemas: se verifican consumo/voltaje de batería (EPS), temperatura interna (térmico) y calidad de enlace (TT&C).

Durante una ventana de operación, el satélite ha transmitido un paquete de telemetría con varias variables clave. No tenemos acceso directo al satélite: solo vemos los datos recibidos en Tierra. Nuestra tarea es hacer lo mismo que haría un operador real:

- Comprobar que el dataset es consistente,
- Visualizar tendencias y cambios,
- Detectar eventos anómalos (por umbrales simples),
- y proponer una hipótesis razonable y una acción de respuesta.

Importante: el CSV incluye **comportamiento nominal** y también **eventos no nominales** (simulando fallos o degradaciones temporales). El objetivo no es adivinar "la respuesta correcta", sino justificar tu análisis con evidencia en los datos.

4.1 Objetivo

En este laboratorio aprenderás a trabajar con telemetría como lo haría un equipo de operaciones:

- Cargar un dataset realista (CSV),
- Comprobar que tiene sentido (validación mínima),
- Visualizarlo (plots),
- Detectar anomalías simples por umbral,
- y proponer una interpretación y una acción.

No se evalúa programación avanzada. Se evalúa que sepas **leer datos, representarlos y razonar**.

4.2 Qué vas a usar

Dataset oficial (proporcionado por el curso):

```
B0/data/tm_case_01.csv
```

Scripts base (proporcionados por el curso):

```
B0/src/telemetry/tm_plot.py
B0/src/telemetry/tm_analyze.py
```

Salida esperada:

```
B0/outputs/tm_plot.png
```

4.3 Contexto del dataset

Cada fila del CSV es una medida en un instante de tiempo. Columnas:

- `timestamp_utc`: instante temporal (UTC).
- `batt_v`: voltaje de batería (EPS).
- `temp_c`: temperatura (térmico).
- `rssi_dbm`: potencia de señal recibida (comunicaciones).

4.4 Laboratorio paso a paso (Runbook)

Paso 0 — Preparación (5 min)

Abre una terminal en la raíz del repositorio (donde está `README.md`). Comprueba que existe el dataset:

```
ls B0/data/
```

Deberías ver `tm_case_01.csv`.

Paso 1 — Crear entorno Python (10 min)

Crea un entorno virtual e instala dependencias:

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Checkpoint 1: Si funciona, `python3 --version` y `pip --version` no dan error.

Paso 2 — Abrir el CSV y entender su estructura (5 min)

Antes de programar, mira el CSV:

```
head -n 5 B0/data/tm_case_01.csv
```

Checkpoint 2: La primera línea debe ser:

```
timestamp_utc,batt_v,temp_c,rssi_dbm
```

Paso 3 — Generar la gráfica (10 min)

Ejecuta el script de plots:

```
python3 B0/src/telemetry/tm_plot.py --in B0/data/tm_case_01.csv \
  --out B0/outputs/tm_plot.png
```

Checkpoint 3:

- Debe crearse el archivo `B0/outputs/tm_plot.png`.
- Debes ver tres curvas (batería, temperatura, RSSI) y una leyenda.

Si no se crea la imagen, revisa:

- que el entorno está activado (`source .venv/bin/activate`),
- que el path del CSV es correcto.

Paso 4 — Detectar anomalías automáticamente (15 min)

Ejecuta el script de análisis:

```
python3 B0/src/telemetry/tm_analyze.py --in B0/data/tm_case_01.csv
```

Este script detecta **eventos** por umbral con duración mínima:

- **EPS_LOW:** `batt_v < 3.5 V` durante ≥ 10 s
- **THERMAL_HIGH:** `temp_c > 45 C` durante ≥ 20 s
- **COMMS_DROP:** `rssi_dbm < -85 dBm` durante ≥ 5 s

Checkpoint 4: El programa debe imprimir una lista de anomalías detectadas (si hay).

Paso 5 — Modificación guiada (para aprender) (15–20 min)

Ahora vas a hacer un cambio pequeño para entender el algoritmo:

1. Abre `B0/src/telemetry/tm_analyze.py`.
2. Localiza los umbrales dentro de `detect_anomalies(...)`.
3. Cambia **solo uno** de estos valores y observa qué ocurre:

Ejemplos:

- Cambia EPS_LOW de `3.5` a `3.6` (deberían salir más eventos o más largos).
- Cambia COMMS_DROP de `-85` a `-90` (deberían salir menos eventos).

Vuelve a ejecutar:

```
python3 B0/src/telemetry/tm_analyze.py --in B0/data/tm_case_01.csv
```

Checkpoint 5: Debes ver que el número/duración de eventos cambia.

Paso 6 — Interpretación (10 min)

Responde en tu entrega (en texto) a lo siguiente:

- **Batería (EPS):** ¿hay un tramo por debajo del umbral? ¿qué implicaría?
- **Comms (TT&C):** ¿hay pérdida de enlace? ¿duración aproximada?
- **Térmico:** ¿se supera el umbral? ¿es puntual o sostenido?
- **Acción de operador:** ¿qué harías primero? (elige 1 acción concreta)

4.5 Qué tienes que entregar

Archivos

- `B0/outputs/tm_plot.png`

Código

- `B0/src/telemetry/tm_plot.py` (sin romperlo)
- `B0/src/telemetry/tm_analyze.py` (con tu modificación guiada o mejoras pequeñas)

Texto de interpretación (obligatorio)

Añade tu interpretación en el **cuerpo del PR** o en un fichero:

```
B0/docs/tm_reports/<nombre>_b03.md
```

Estructura sugerida:

- eventos detectados (lista)
- qué subsistema se ve afectado
- acción como operador

4.6 Criterios de aceptación

- Ejecuta en un entorno limpio siguiendo estos pasos.
- `tm_plot.png` existe y es legible (ejes + leyenda).
- El análisis detecta eventos (o justificas por qué no).
- La interpretación es coherente (subsistema → dato → acción).

4.7 Troubleshooting (errores típicos)

- **ModuleNotFoundError:** falta `pandas` o `matplotlib`. Activa el entorno `.venv` e instala `requirements.txt`.
- **No se crea el PNG:** revisa el path de salida y que existe `B0/outputs/`.
- **Error leyendo timestamp:** asegúrate de no haber modificado el CSV y de usar el path correcto.

Calendario y plazos (3 semanas)

Este bloque está diseñado para completarse en **3 semanas** con un ritmo sostenible. El laboratorio de telemetría (B0.3) se realiza en **un solo día** (sesión guiada), pero el alumno tiene días antes para preparar el entorno y días después para redactar el informe.

Semana 1 — B0.1 con calma (lectura + comprensión)

- **Días 1–5:** Lectura de **NASA CubeSat 101** y visualización de la charla.
- **Días 5–7:** Redacción de respuestas + diagrama opcional + preparación de la explicación oral (2–3 min).

Hito 1 (fin de Semana 1): B0.1 listo en local (documento y materiales preparados).

Semana 2 — B0.2 (Git/GitHub) + entrega por PR

- **Días 8–10:** Recursos de Git (GitHub Skills / Learn Git Branching).
- **Días 10–12:** Crear rama + subir B0.1 + abrir Pull Request (PR).
- **Días 12–14:** Responder al review hasta dejar el PR **aprobado**.

Hito 2 (fin de Semana 2): PR de B0.1 mergeado (o como mínimo aprobado y listo para merge).

Semana 3 — Python Primer + Laboratorio B0.3 (1 día) + informe

- **Días 15–18:** Python Primer (si se necesita): ejecutar scripts, leer CSV, matplotlib.
- **Días 19–20:** Preparación B0.3: entorno listo + comprobar que el dataset existe.
- **Día 21 (sesión única):** Laboratorio guiado B0.3 (plot + análisis + cambio de parámetro).
- **Días 22–23:** Redacción del informe de telemetría y entrega final.

Hito 3 (fin de Semana 3): Bloque 0 completado con:

- `B0/outputs/tm_plot.png`
- `B0/docs/tm_reports/<nombre>_b03.md`

Nota para quien no pueda asistir al laboratorio

El laboratorio B0.3 se puede reproducir siguiendo el runbook: `B0/docs/runbooks/rb_b0_3_tm.md`.