

Astra Labs CAN-SBX – Hivemind

Application Form Answers

Project title: Learning at the Edge of Space: Federated Learning on a Student-Built Microcontroller Swarm

Lay abstract

Future satellite swarms will need to learn on their own, because they cannot send all their data to the ground. We will test this on a stratospheric balloon. Our payload carries 30 small computers, each built by students in an Astra Labs Picosat workshop or at [REDACTED] with its own sensors for temperature, pressure, motion, direction, chip temperature, and current, all powered from one central battery. During the flight, each computer learns to predict its own sensor readings a few minutes ahead, using what the other sensors tell it. Every few minutes, the computers share what they have learned, not their raw data, with a central computer that combines it and sends back an improved model. This is called federated learning. We will measure whether learning together beats learning alone, how close it gets to sending everything to one computer, what each approach costs in communication and computing time, and how cold, low air pressure, and failing nodes affect it.

Team

Field	Answer
Team name	Astra Labs
Institution	Seneca Polytechnic
Team Lead	[REDACTED]
Faculty Advisor	

REDACTED · team contact details

Field	Answer
-------	--------



REDACTED · team contacts and member list

Constraint	Our design	Status
600 mm × 300 mm cylinder	30 nodes on a 5-level ring rack (6 per level) around a central computer bay	<i>Preliminary fit by hand calculation (Section 2.3); confirm in CAD</i>
5 kg maximum	~3.4 kg estimated (Appendix A)	Meets, ~33% margin
No pointing	Experiment works at any orientation; orientation is logged as data	<i>Verify by test</i>
Down to −56 °C	Insulated enclosure, central pack in an insulated compartment next to the Pi, cold-chamber tested	<i>Verify by test: thermal chamber test of the full system at −56 °C to confirm it survives and keeps operating</i>
Daytime, ~3 h, up to 30 km	Experiment runs from power-on through landing; no float phase needed. Power system sized for at least 5 h of operation (the 4.5 h mission profile plus margin)	<i>Verify by test: full-duration bench test logged by onboard battery monitoring system</i>

Constraint	Our design	Status
No hazardous materials or lasers	Only lithium-ion cells: one 12-cell (3S4P 18650) central pack with BMS (Hazard Sheet)	<i>SEDS (7 Oct 2026): own Li-ion cells need vacuum/thermal proof at the campaign; CSA onboard power is the alternative</i>
Safe on power loss	Nothing moves or heats dangerously; nodes simply stop	Meets
No human or animal subjects	None	Meets

Section 1. Introduction

1.1 Motivation

Spacecraft now collect far more data than they can send to the ground, and satellite constellations are growing from dozens of spacecraft to thousands. Learning has to move on board: each spacecraft improving its own models in orbit instead of waiting for engineers on the ground. Federated learning lets many devices train a shared model while keeping their raw data local, exchanging only what each has learned [1], [3]. Studies of federated learning across satellite constellations show it can cut communication and keep learning going when links drop out [9], [10].

Those studies are simulations. Real spacecraft run on small, power-limited processors in cold, near-vacuum conditions, and cheap commercial microcontrollers are increasingly flown on CubeSats. Nobody yet knows how well on-board learning holds up when the processors doing it are cold, low on battery, and occasionally failing. This affects constellation operators, CubeSat teams, and Canadian small businesses building low-cost spacecraft, as well as future orbital computing platforms where heat cannot be removed by air.

The stratosphere is the cheapest place to find out. At 30 km, the air pressure is about 1% of sea level, so electronics lose almost all convective cooling, and temperatures fall to about -56°C [13]. Conditions change continuously through the flight, so a model trained on the ground meets situations it has never seen: exactly the problem on-board learning is meant to solve.

Why this team, why now. This experiment grew out of Astra Labs' own work. Our Project Pioneer rocket was designed to carry 30 picosatellites, built by students in our Picosat workshop, to 2.8 km. Pioneer did not fly: it failed pre-launch inspection because of technical faults in the payload and the rocket's assembly. That lesson shaped this proposal, which puts bench, thermal, and vacuum testing of every node ahead of flight (Section 3.1). CAN-SBX lets us fly the same swarm concept higher and for longer. Planning its data link brought us straight to the problem above: 30 nodes produce far more data than a small radio can send. With guidance from Seneca's AI Lab, we turned that limit into the experiment itself, so each node learns on board and shares only what it has learned. This builds on the idea of running AI models in space that our team lead first proposed in his first-semester project deck.

1.2 Novelty of Experiment

Federated averaging (FedAvg) is the standard method for combining models trained on separate devices [1]. Later work handles devices that see different data or respond late [2], lets each device keep a personal layer [4], and compresses updates to save bandwidth [5].

Training on microcontrollers is newer. Llisterri Giménez et al. ran federated learning across Arduino boards on the ground [6]; Lin et al. showed neural networks can be trained in under 256 KB of memory [7]; Ren et al. demonstrated online learning on a microcontroller in the field [8]. Federated learning for satellites has been studied in simulation only [9], [10], [19], [20]. The one in-orbit demonstration of a re-trainable model retrained it on the ground and uploaded it to a single satellite, with no federated learning [21]. Closest to our work within CAN-SBX itself, the University of Waterloo's ASTRA team (2025) validated an adaptive machine-learning model for filtering cosmic noise in high-frequency data transmission on a stratospheric balloon [22]. That work tested one adaptive model; to our knowledge, it

did not train a model across many devices or use federated learning, which is the question our flight answers.

The gap: no study we have found flies federated learning on real microcontroller hardware through a stratospheric environment and measures what cold, low pressure, failures, and limited bandwidth do to it. Our flight provides that measurement, compares it against four alternatives on the same data, and releases the dataset openly.

1.3 Goals

1. **Primary:** Demonstrate federated learning on a swarm of 30 low-cost microcontrollers during a stratospheric balloon flight.
2. Measure how federated learning compares with a ground-trained model, each node learning alone, a centralized model, and physics and statistical baselines.
3. Characterize how near-space conditions affect low-cost microcontrollers and sensors, and publish the dataset.
4. *Secondary, if resources allow:* test whether the other sensors can stand in for a failed sensor (virtual sensing, SO4), and whether a per-node personal layer improves forecast accuracy (scored with the same method as SO2).

1.4 Importance to Canada's Space Sector

The CSA's 2026–27 Departmental Plan lists space technology and innovation (including funding through the Space Technology Development Program and support for small and medium enterprises) and youth engagement and capacity building among its priorities [\[15\]](#). This project serves both:

- **Technology:** flight data on on-board learning with commercial microcontrollers, directly useful to Canadian small businesses and education providers that build low-cost picosatellite and CubeSat kits and run student outreach programs, as well as university CubeSat teams. Measurable deliverables, released within two months of the flight: an open-source, flight-tested node design and ESP32-S3 on-device training firmware that kit makers can adapt for classroom use, the measured time and energy cost of each federated round in near-space conditions, and the flight dataset. We will report downloads, forks, and adoptions by kit makers or outreach programs.
- **Capacity building:** the Plan expects stratospheric balloon flights to let about 50 students design payloads [\[15\]](#); our workshop extends this by having students build the 30 flight nodes themselves.
- **Open data:** a public dataset of synchronized readings from 30 identical nodes from the ground to 30 km.

1.5 Relevance to the High-Altitude Environment

The experiment needs conditions the nodes were never trained on, changing continuously, with all stresses acting at once: pressure falling from 1013 hPa to about 12 hPa, temperature passing the tropopause minimum near -56°C , solar heating without convective cooling, and an uncontrolled, rotating payload [\[13\]](#). Chambers can reproduce cold or vacuum separately, but not all of these together on 30 nodes with real sun and rotation. A balloon flight can.

Under 3 hours is enough. The balloon climbs at about 5 m/s, so the ascent to 30 km takes about 100 minutes. With one training round every 2 minutes (120 s), that gives about 50 rounds on the way up and 10 to 15 during the ~30-minute descent. Compute is not the limit: a node finishes a round in about 1.3 s ([Section 2.5](#)). Phase 1 bench rehearsals will confirm the model improves measurably within 20 rounds. Every learning mode runs in parallel as its own FreeRTOS task on the ESP32-S3's two cores (timekeeping and CAN run above training priority) on the same data, so all comparisons come from every minute of the flight.

1.6 Research Hypothesis

During a stratospheric flight, a forecasting model trained cooperatively by 30 microcontroller nodes through federated learning will predict each node's sensor readings 1 to 5 minutes ahead with lower error than both the same model trained only on the ground and models trained by each node alone, and within 10% of a model trained centrally on all raw data. We expect this because the nodes sit at different heights in the rack and at different distances from the warm central computer and the cold enclosure wall, so each sees a different mix of self-heating, cold soak, and rotation; pooling what they learn exposes every node to more of the new flight conditions than it sees by itself. We also expect cold and low pressure to change the hardware itself: sensor offsets will drift, and node current will shift as boards cool, most strongly on the nodes nearest the enclosure wall. We test this (Goal 3) by comparing each node's sensor error, current, and round time against its measured temperature and rack position.

Section 2. Experiment Concept Design

2.1 Scientific Objectives

Each node runs one small neural network that reads the last 60 s of its own sensors and predicts all six sensor channels 1, 3 and 5 minutes ahead. Every node keeps separate copies of that model (frozen, local-only and federated, plus a personal-layer copy for the secondary goal) running side by side on the same data, so the learning modes are compared under identical conditions.

Obj.	Scientific objective	Measurement technique	Goal
SO1	Can low-cost microcontrollers complete federated training rounds in near-space conditions?	Log every round: participants, duration, bytes moved, failures	1
SO2	Does federated learning forecast better than a ground-trained model and than nodes learning alone, and how close does it come to centralized learning?	Forecast error of each learning mode against measured readings, per sensor and horizon	2
SO3	How do cold, low pressure, and rotation affect low-cost microcontrollers and sensors?	Continuous temperature, pressure, attitude, chip temperature, and current on all 30 nodes	3
SO4 (<i>secondary</i>)	Can the other sensors stand in for a failed sensor (virtual sensing)?	Blank one sensor type on several nodes; compare model estimates with the true readings	2

Learning modes compared: frozen (ground-trained, never updated) · local-only (each node learns alone) · federated (merged every round by FedAvg [\[11\]](#)) · centralized (one model on the central computer, trained on the full raw data stream from all 30 nodes) · baselines: persistence, linear trend, and the standard-atmosphere formula [\[13\]](#).

2.2 Science Traceability Matrix

Scientific objective	Measurement objective	Measurement requirement	Instrument	Instrument requirement
SO1	Completion of federated training rounds	Every round logged with start/end time, participating node IDs, and bytes transferred, launch to landing	Node firmware + central computer logger over 2 CAN buses	Round timing to 10 ms; 100% of rounds logged; CAN at 500 kbit/s
SO2	Forecast error of each learning mode	Mean absolute error per mode, sensor and horizon (+1, +3, +5 min) over at least 20 rounds	On-node models; central-computer scoring	Forecasts logged every 5 s per node; time-synced to GPS within 1 s
SO2, SO3	Node temperature	−40 to +60 °C inside the enclosure	MS5611 (on-chip temperature)	±1 °C, sampled every 5 s
SO2, SO3	Static pressure and altitude	1013 hPa down to ~12 hPa (30 km) [13]	MS5611 barometer [16]	Range 10–1200 hPa; resolution ≤ 0.1 hPa
SO2, SO3	Rotation and heading	Rotation rate and heading through ascent and descent	IAM-20380 gyroscope + ADXL355 accelerometer + MMC5983MA magnetometer	±250 °/s range; heading ±5° after calibration; sampled at 10 Hz, averaged per 5 s
SO2, SO3	Node current and bus voltage	0–400 mA and ~9–12.6 V per node	INA219	±10 mV; ±1 mA
SO2, SO3	True altitude and time	0–30 km, for flight phases and time sync	GPS on central computer	Valid fix above 18 km (high-altitude/airborne mode)
SO4	Virtual-sensor estimation error	Estimate vs. true reading while a	Same as above	Same as above

Scientific objective	Measurement objective	Measurement requirement	Instrument	Instrument requirement
		sensor is blanked for 10 min		

Mission requirements: all 30 nodes and the central computer must be powered, armed, and logging before launch, with the same ground-trained model loaded. Sampling every 5 s and training rounds every 2 minutes (120 s) must run from launch through ascent, burst, and descent. All raw data, forecasts, and model weights must be stored on each node's flash and on the central SSD, and the payload recovered for the full dataset; a ~70-byte summary is radioed down every minute in case it is not.

Each requirement follows from its objective: SO2 needs ground-truth readings at least as often as the forecasts they score (5 s), so every sensor is sampled at that rate or faster; the 10–1200 hPa range covers the full flight to 30 km; GPS time sync lets 30 separate logs be compared second by second.

2.3 System Architecture

Subsystem	Main components	Key specification	Interfaces
Structure	Ring rack, 5 levels × 6 nodes, central bay for computer and battery	Fits 600 mm × 300 mm cylinder; 10 G vertical / 7.5 G lateral × 2 safety factor	Mechanical: nodes screwed into slots; rack bolted to gondola interface
Sensor nodes (×30)	One custom PCB with bare chips (no dev boards): ESP32-S3 (4 MB flash, 2 MB PSRAM), MS5611, IAM-20380, ADXL355, MMC5983MA, INA219, SN65HVD230 CAN transceiver	~35 g each (no flight cell); 5 s sampling; on-device training in C (benchmarked at 1.3 s per round)	Power: bus from the central pack; data: CAN; locking connector (CAN-H, CAN-L, GND, bus power)
Node power (×30)	Bus power from the central pack through a polyfuse and INA219; buck regulator to 3.3 V; removable 18650 bench cell, removed with its holder before flight	~0.3 W per node in flight at 240 MHz (estimate; to be measured on the new board, and again at 80 MHz to save power); peak about 3 W worst case with the LED matrix at full brightness, so the matrix stays off in flight	Bus power from the power spine; INA219 in series

Subsystem	Main components	Key specification	Interfaces
Central computer	Raspberry Pi 5, USB SSD, GPS, 2 × MCP2518FD CAN controllers, SX1262 LoRa radio	Scheduler, FedAvg merge, centralized model, scoring, logging	SPI0 to both CAN controllers; SPI1 to the LoRa radio; USB to SSD; UART to GPS
Network	2 CAN buses × 15 nodes, twisted pair, 120 Ω at each end	500 kbit/s (1 Mbit/s if cold tests allow); one broadcast reaches all 15 nodes	Data between nodes and central computer
Communications	SX1262 LoRa radio, 902–928 MHz ISM, ground station receiver	~70 bytes/minute summary	SPI to central computer; RF to ground
Central power	3S4P 18650 Li-ion pack with BMS (~130 Wh rated, ~65 Wh usable in the cold), fuse, 5 V / 5 A regulator for the Pi 5 and its SSD, two arming keys (Pi, then nodes), five Pi-controlled floor switches	~15 W total in flight (30 nodes + Pi 5)	Power to central computer, radio, GPS, and all 30 nodes through the power spine
Thermal	Foam enclosure, aluminum heat spreader for central computer, central pack in a Styrofoam-insulated compartment next to the Pi	Keeps cells and electronics within operating range at −56 °C outside	Conduction from computer to enclosure wall

If the central computer fails, nodes notice the missing 1 Hz time beacon after 120 s and keep learning locally while logging, so the dataset survives. A polyfuse on every node and five Pi-controlled floor switches isolate a failed node or floor; the central pack is the one shared point of failure. Preliminary fit (hand calculation, to be confirmed in CAD and shown in Appendix E): each level holds six 89 mm round node boards mounted on the flats of a hexagon, about 180 mm across the corners, leaving about 50 mm for structure and foam on each side inside the 300 mm diameter. The central bay inside the hexagon is about 120 mm across, enough for the Raspberry Pi 5 (85 × 56 mm) on one level and the 3S4P pack (about 72 × 54 × 65 mm) on another. Five levels at 110 mm pitch use 550 of the 600 mm height. If the CSA does not accept our pack at the campaign, the payload runs from the CSA onboard power system through a converter into the same power spine, with no change to the nodes.

2.4 Block Diagram

All 30 nodes run from one central pack and talk to the Pi over CAN

Payload block diagram: electrical power, data and mechanical interfaces

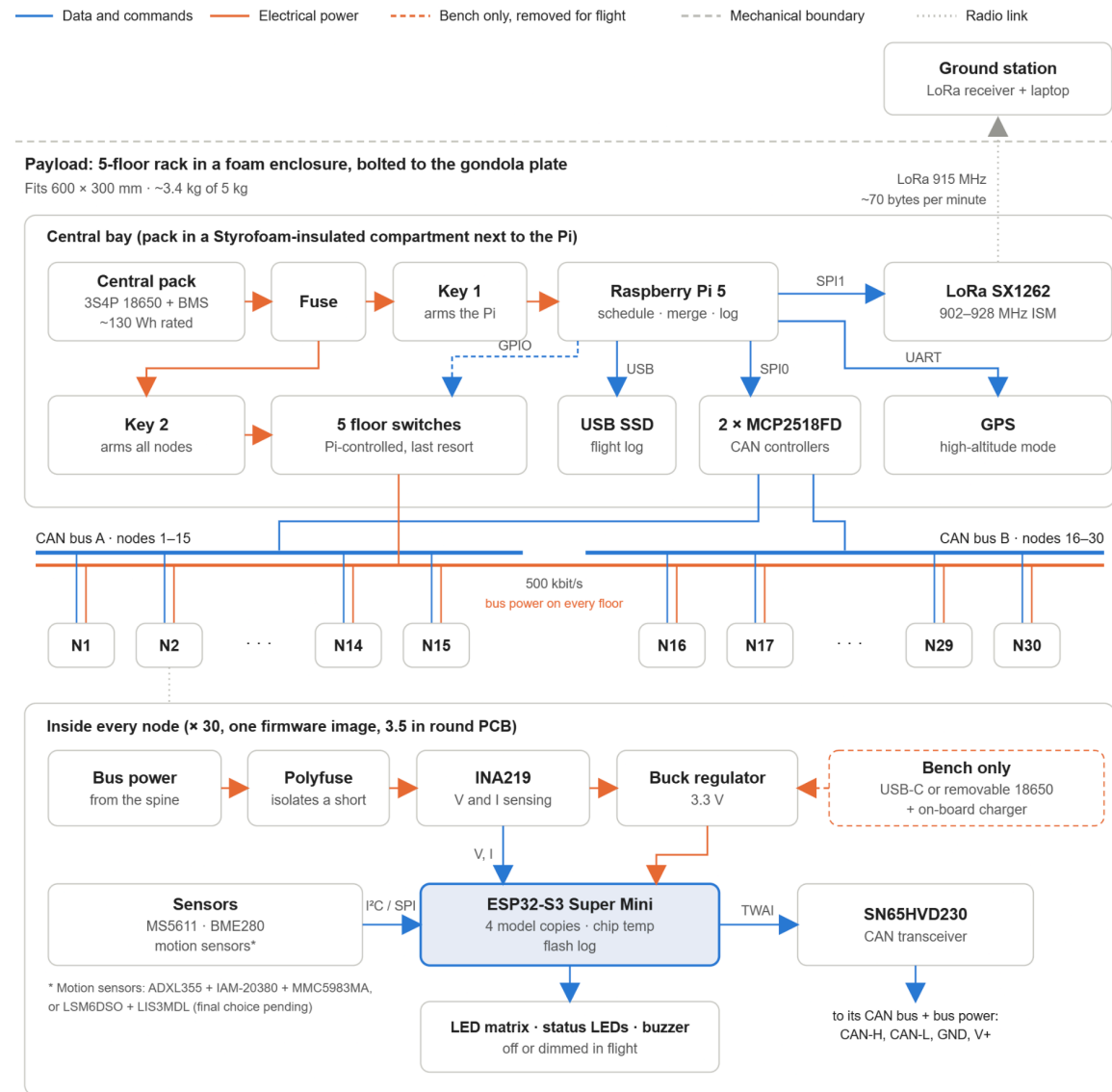


Figure 1. Payload block diagram · power, data and mechanical interfaces

Power comes from one central pack through two arming keys and a power spine to each floor; data crosses the two CAN buses, and only the central computer uses radio. The two arming keys are the electrical inhibits. The flight sensor set is the one in Sections 2.2 and 2.5: MS5611, IAM-20380, ADXL355, MMC5983MA and INA219. The other sensors named in the figure's footnote were alternatives considered during selection and are not flown.

2.5 AI Model

Every ESP32 node and the Raspberry Pi run the same small neural network (a multilayer perceptron). Keeping one model everywhere is what makes the federated and centralized results directly comparable.

Item	Design
Inputs	The last 60 s (12 samples) of 8 input channels, 96 values in total: 7 sensor channels plus 1 physics feature, the expected outside air temperature from the standard atmosphere at the measured pressure [13]
Sensor channels	Board temperature and pressure (MS5611), rotation rate (IAM-20380 gyroscope), acceleration (ADXL355), heading (MMC5983MA), ESP32-S3 chip temperature (internal sensor), and node current (INA219)
Outputs	6 channels (temperature, pressure, rotation rate, heading, chip temperature, current) predicted 1, 3, and 5 minutes (18 values)
Layers	96 => 24 => 12 => 18, ReLU activations
Size	2,862 parameters: 11.4 KB at 32-bit, 2.9 KB at 8-bit (measured 8-bit error under 0.4% of range)
Loss and optimizer	Mean squared error on normalized values; stochastic gradient descent, one example at a time, updated in place (starting learning rate 0.01, tuned on the bench)
Training per round	3 passes over the node's most recent 240 labelled examples (20 minutes of data)
Code	Hand-written forward and backward pass in plain C on the ESP32-S3, float32 only (TensorFlow Lite Micro can run models but not train them, and double is software-emulated and 38× slower); Python and NumPy on the Raspberry Pi

Why this size. These numbers are measured, not estimated. We benchmarked this exact network on an ESP32-S3 (2 cores at 240 MHz) with flight-like mock data. One training example takes 0.41 ms and one round takes 0.29 s per copy. With all node tasks running together (sensors at 100 Hz, 10× the flight rate, CAN, inference, logging), a round for the three trainable copies averaged 1.31 s against a 120 s budget, with no missed sensor deadlines in a 60 s mission rehearsal. One trainable copy needs 12.6 KB, so four copies (frozen, local, federated, and a personal-layer copy for the secondary goal) use about 51 KB of internal RAM; PSRAM is not needed. At 80 MHz, a round still takes 0.89 s, so nodes can run slower to save power. The same chip trains a model about 13× larger (38,898 parameters) in under 10 s per round,

which leaves room if real data needs it. Still unproven: real sensors, the CAN bus with transceivers, power draw, cold, and multi-hour runs (Section 3.1, Risk TR1). Training time for the centralized model on the Pi is estimated at under 1 s and is benchmarked next. If the Pi 5 cannot run the merge, the centralized model and logging together on the bench, we move to an NVIDIA Jetson.

Physics first, AI for the rest. The standard-atmosphere input already explains most of how temperature changes with altitude, including the reversal above the tropopause that ground tests can never show. The network only has to learn what the formula misses: heat from our own electronics, sunlight on one side of the payload, and how our electronics heat up under load [11], [12].

Virtual sensing. In 20% of training examples, one randomly chosen input channel is blanked out. The model therefore learns to estimate a missing sensor from the others, which we test in flight (SO4).

2.6 Training Data: How We Obtain It

The model needs examples of how our payload and the atmosphere behave before it ever flies. We collect them from six sources (Figure 2). Ground tests teach the model how our hardware behaves; weather-balloon data and a simulator cover the 0 to 30 km atmosphere that no ground test can reach.

Where the training data comes from

Ground tests teach the model how OUR payload behaves. Weather-balloon data and the simulator teach the 0 to 30 km atmosphere that ground tests cannot reach.

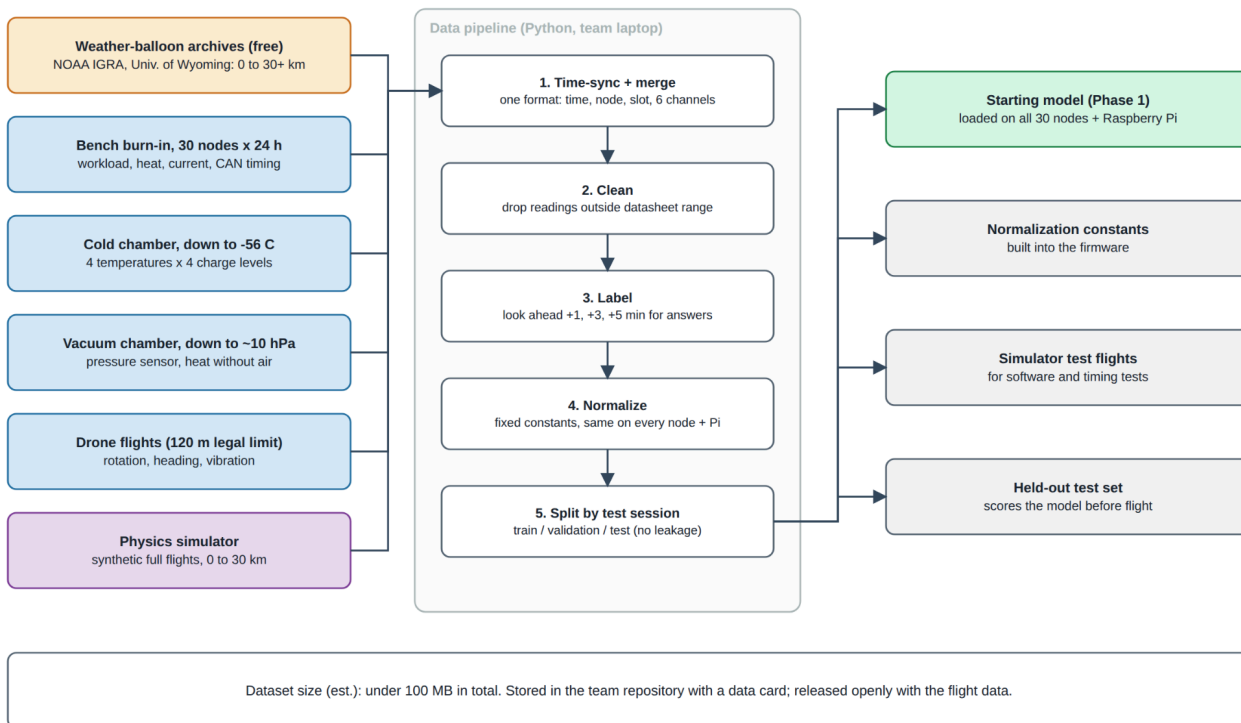


Figure 2. Training data sources and the data pipeline.

#	Source	What it provides	Amount (est.)	When	Owner

1	Public weather-balloon archives: NOAA IGRA [14] and the University of Wyoming sounding archive	Real pressure, temperature, and humidity from the ground to 30+ km, including the tropopause	Several hundred profiles from the stations nearest the launch site	Any time (free download)	Software lead
2	Bench burn-in of all 30 nodes in the rack	Normal behaviour: heat vs. workload, current draw, CAN timing, node-to-node differences	24 h per node (~17,000 samples per node)	Feb to Mar 2027	Electrical
3	Cold chamber (down to -56 °C)	Cold effects on sensors, cells and training speed	4 temperatures × 4 charge levels = 16 runs of ~1 h	Mar to Apr 2027	Electrical + Software
4	Vacuum chamber (down to ~10 hPa)	Pressure sensor at flight pressures; heat build-up without air	2 to 3 runs	Apr 2027	Software + Mechanical
5	Drone flights (within the 120 m Transport Canada limit)	Rotation, heading and vibration while moving	~10 short flights	Apr to May 2027	Team
6	Physics simulator (Python)	Synthetic full flights: weather-balloon profiles + a simple thermal model + sensor noise measured on the bench	Thousands of flights	From Nov 2026	Software lead

Data pipeline. Every source goes through the same five steps before training:

1. **Time-sync and merge** into one format: time, node ID, rack position, the 6 channels, and the test condition.
2. **Clean** by removing readings outside each sensor's datasheet range.
3. **Label** each example by looking ahead +1, +3, and +5 minutes for the true values.
4. **Normalize** with fixed constants computed once from ground data and built into the firmware, identical on every node and on the Pi.

5. **Split by test session** (not by random rows) into training, validation and test sets, so the model is never scored on data it has trained on.

The full ground dataset is estimated at under 100 MB. It is kept in the team repository with a short data card and released openly with the flight data.

2.7 Training Phases

Training happens in four phases, each ending in a pass/fail gate (Figure 3).

Four training phases, each with a pass/fail gate

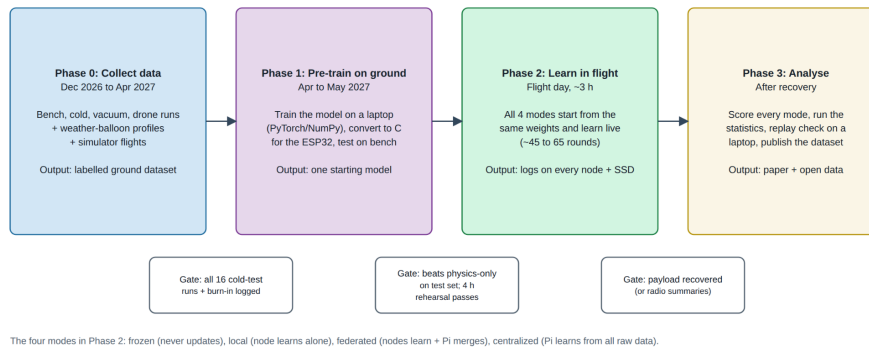


Figure 3. The four training phases and their gates.

Phase	When	What happens	Output	Gate to continue
0. Collect data	Dec 2026 to Apr 2027	Bench, cold, vacuum, and drone runs; download weather-balloon profiles; run the simulator	Labelled ground dataset	All 16 cold-test runs and the burn-in logged
1. Pre-train on ground	Apr to May 2027	Train the model on a laptop, convert it to C, verify it gives identical answers on the ESP32 and the Pi	One starting model	Beats the physics-only baseline on the held-out test set; 4-hour flight rehearsal passes
2. Learn in flight	Flight day (~3 h)	All four learning modes start from the same weights and learn live (Section 2.8)	Logs on every node and the Pi's SSD	Payload recovered (or radio summaries received)
3. Analyse	After recovery	Score every mode, run the statistics (Section 2.9),	Paper and open dataset	n/a

		replay check on a laptop, publish		
--	--	--------------------------------------	--	--

Learning modes in flight. Frozen: never updated after Phase 1. Local: each node learns from its own data only. Federated: each node learns, then the Pi merges all nodes every round. Centralized: the Pi learns from all 30 nodes' raw data.

2.8 In-Flight Learning: Federated and Centralized in Parallel

The core of the experiment is comparing federated learning with centralized learning on the same flight (Figure 4). The Raspberry Pi 5 on the gondola does both jobs at once: it merges the nodes' federated updates, and it trains its own centralized model on the raw data. Without it, the centralized comparison could not happen during the flight, because LoRa's bandwidth is far too small to send the raw data to the ground and train there.

In flight: federated and centralized learning run side by side on the same data

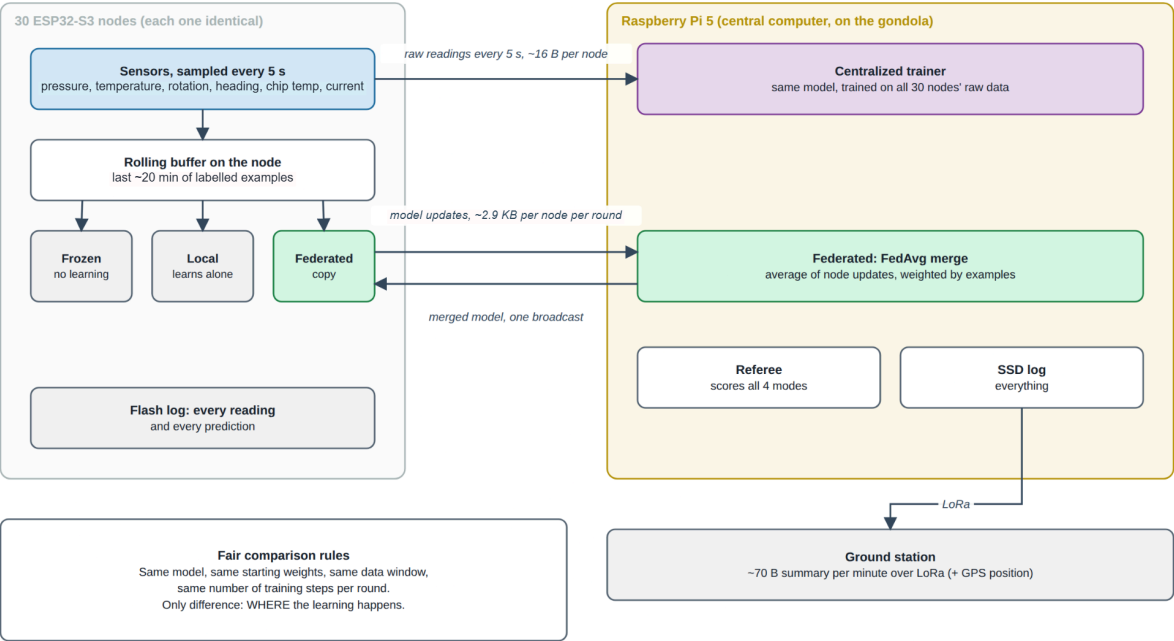


Figure 4. Federated and centralized learning run side by side on the same data.

Federated path. Each node trains its federated copy on its own data, then sends only the change in its weights (2.9 KB at 8-bit; on the bench, one update crossed the ESP32-S3's CAN controller in 103 ms at 500 kbit/s, so 15 nodes need about 1.5 s per bus) over CAN. The Pi combines them with federated averaging (FedAvg) [\[1\]](#):

$w_{\text{new}} = \sum (n_k / n) \times w_k$, where w_k is node k 's model, n_k its number of training examples and n the total.

The merged model goes back to all nodes on that bus in one broadcast. Nodes never talk over each other: CAN arbitration lets only one frame onto a bus at a time (the lower message ID wins and the other node retries automatically, so nothing is lost), and the Pi gives each node its own upload slot in the round, timed from the 1 Hz GPS time beacon, so updates go out one after another instead of competing.

Centralized path. Every 5 s, each node also sends its raw readings to the Pi (~16 bytes, two CAN frames). For 30 nodes, that is about 12 frames per second, under 1% of the bus capacity. The Pi trains one copy of the same model on everything.

Fair comparison rules. Both modes use the same model, the same starting weights, the same 20-minute data window and the same number of training steps per round. The only difference is where learning happens, so any difference in accuracy comes from federated vs. centralized learning itself. The experiment also measures the communication cost of each: raw data flowing continuously vs. model updates once per round.

One training round (every 2 minutes (120 s), Figure 5): the Pi schedules which nodes train (resting nodes that are too hot or drawing abnormal current); nodes score their last forecasts, train, and send their updates within a 10 s deadline; the Pi merges them, trains the centralized model, scores all four modes, broadcasts the merged model and logs everything. A late node is left out of that round's merge but still receives the new model. If the Pi goes silent for 120 s, nodes keep learning and logging on their own, so no data is lost.

One training round (every 2 to 3 minutes)

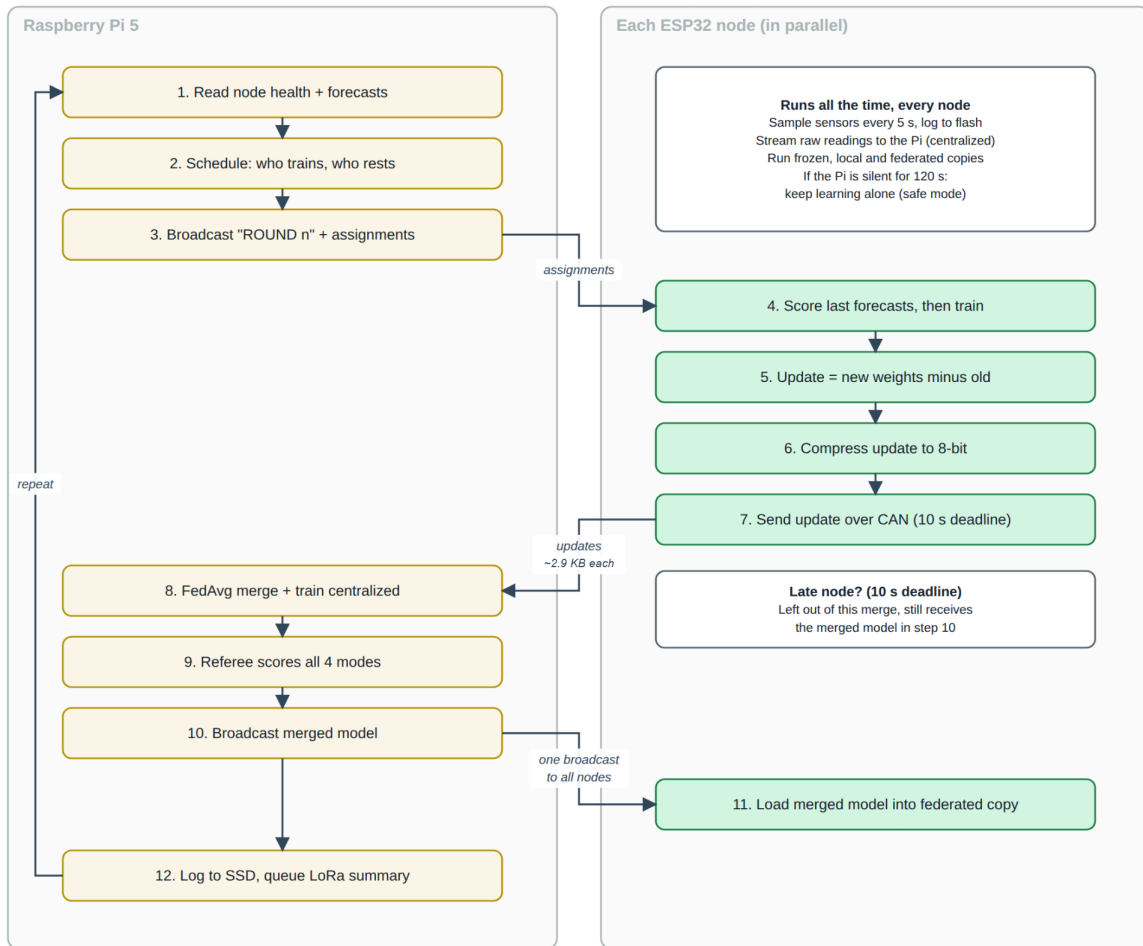


Figure 5. One training round.

Rounds per flight. Ascent to 30 km at ~5 m/s takes ~100 minutes; at one round every 2 minutes (120 s) that gives ~50 rounds, plus ~10 to 15 during the ~30-minute descent. The round length is set by data, not compute: each round needs ~24 new readings per node (one every 5 s) to learn from.

2.9 Evaluation Method

Test-then-train scoring. Every forecast is logged when it is made and scored when the true reading arrives, before that reading is used for training. This standard method for models that learn from a data stream [18] means no mode is ever scored on data it has already learned from.

Metrics.

- Mean absolute error (MAE) per sensor channel and per horizon (+1, +3, +5 min), for each of the four modes and three baselines (persistence, linear trend, standard atmosphere).
- Skill score against persistence: $\text{Skill} = 1 - \text{MAE}(\text{model}) / \text{MAE}(\text{persistence})$. Skill above 0 means the model beats the “nothing changes” forecast.

- Results reported separately for each flight phase: lower ascent, around the tropopause (~9 to 13 km), upper ascent, and descent.

Statistical test. Each of the 30 nodes gives one paired result per comparison (for example, federated error minus local error on the same node). We use a Wilcoxon signed-rank test across the 30 nodes ($\alpha = 0.05$) and report the median difference with a 95% bootstrap confidence interval. The hypothesis in [Section 1.6](#) holds if federated error is lower than frozen and local error, and no more than 10% above centralized error.

Virtual sensing (SO4). One sensor type at a time is blanked on 5 nodes for 10 minutes. The model's estimate of the missing channel is compared with the true reading and with a linear-regression baseline.

System metrics (SO1). Round duration, nodes taking part, bytes moved on each bus, energy per round (from the INA219s), and the Pi's CPU time and temperature while training the centralized model.

Section 3. Concept of Operation

3.1 Equipment Requirements

Equipment	Needed for	Procurement plan	Status
Thermal chamber to -60 °C	Charge × temperature battery tests; cold soak of every node; CAN signal checks in the cold	<i>Primary: request access to a thermal chamber</i> [REDACTED] <i>contact by 15 Nov 2026 for test slots in Mar–Apr 2027. No agreement is in place yet. Team-run backup if [REDACTED] is unavailable: an insulated dry-ice box (about -70 °C) for full-node cold-soak runs, plus a chest freezer (-20 °C) for long burn-in, both logged with thermocouples</i>	<i>To confirm</i>
Vacuum chamber to ~10 hPa	MS5611 behaviour at flight pressures; central computer heat rejection without air	<i>Primary: request access to a vacuum chamber</i> [REDACTED] <i>with the thermal request, for Apr 2027. No agreement is in place yet. Team-run backup: a small acrylic-lid vacuum chamber and two-stage rotary pump [REDACTED]</i> [REDACTED] [REDACTED] <i>which reaches the 10 hPa needed for MS5611 checks and single-node heat-build-up tests</i>	<i>To request</i>

Equipment	Needed for	Procurement plan	Status
Bench power supplies, multimeter, oscilloscope, CAN analyser	Node bring-up, current calibration, bus debugging	Astra Labs and Seneca lab equipment	Available
Soldering stations, ESD-safe benches	Student node-building workshop	Seneca lab and Seneca Research	<i>To book</i>
Lithium-ion charger with cell-level protection, fire-safe charging bag	Safe charging of the central pack and the bench cells	Purchase	
Drone (under Transport Canada rules, ~120 m ceiling)	Rotation, heading and integration testing in motion	<i>Team or partner</i>	<i>To confirm</i>

Pre-flight training data also comes from bench burn-in runs and public weather-balloon profiles up to 30 km from NOAA's Integrated Global Radiosonde Archive [\[14\]](#).

3.2 Environmental Requirements

Environmental risk	Effect on the experiment	Mitigation and engineering controls
Cold, down to -56°C outside	Cell capacity drop and voltage sag; sensor drift; slower training	Insulated enclosure; central pack in an insulated compartment next to the Pi, warmed by it; high-capacity cold-rated cells, usable energy confirmed in cold tests; cold-chamber tests at 4 charge levels $\times$ 4 temperatures
Low pressure, ~12 hPa at 30 km	Almost no convective cooling; pressure sensor near its 10 hPa limit	Heat spreader conducts central computer heat to the wall; capped power mode; MS5611 rated to 10 hPa, verified in vacuum chamber; GPS altitude as a cross-check
Uncontrolled rotation and swinging	Uneven solar heating; antenna nulls in downlink	Experiment works at any orientation; rotation and heading logged as model inputs; all data stored on board

Environmental risk	Effect on the experiment	Mitigation and engineering controls
Launch, burst and landing loads	Loose nodes or connectors; cell damage	Rack analysed to 10 G vertical and 7.5 G lateral with a safety factor of 2; locking connectors; pack in a clamped holder
Integration site (outdoor, Timmins, limited checkout time)	Moisture, cold soak before launch, rushed arming	Enclosure sealed against moisture but vented through a filtered hole, so the pressure sensors read outside air; status LEDs visible through a window; 15-minute checkout procedure rehearsed on the bench
Radio interference with platform telemetry	Downlink conflicts	Licence-free 902–928 MHz ISM band, low duty cycle; the CSA platform only does tracking (no telemetry), and SEDS does not fly teams with clashing frequencies together

3.3 In-flight Operations

1. **Before launch:** charge the central pack on site and remove all bench cells and their holders; load the ground-trained model on all nodes; insert arming key 1 (central computer), then key 2 (nodes); confirm via status LEDs and the ground station that all 30 nodes report; hand over to the integration crew.
2. **Ascent (~100 min):** nodes sample every 5 s; a training round runs every 2 minutes (120 s) (about 50 rounds); the central computer merges, scores every learning mode, and radios a summary each minute. Stress tests (5-node dropout, sensor blanking, compressed updates) run once each during ascent.
3. **Burst near 30 km and descent (~30 min): the experiment keeps running,** adding 10–15 rounds in the fastest-changing conditions.
4. **Recovery:** remove both arming keys; copy node flash and central SSD; inspect the pack and structure; compare against the downlinked summaries.

If the payload is not recovered, the per-minute downlink still carries round counts, participating nodes, forecast skill per learning mode, and node health, which is enough to show whether federated learning ran in flight and how it compared (Goals 1 and 2 partially met). Raw data for Goal 3 and the full dataset would be lost.

Section 4. Project Plan

4.1 Funding Strategy

REDACTED · budget and funding plan

REDACTED · funding sources

4.2 Outreach Strategy

Outreach is built into the experiment: students build the 30 flight nodes themselves. Events reach every audience level (K-12, post-secondary, general public and academic), each with a number we will report.

Event	Audience	Date	Owner	Measurable outcome
Three-day node-building workshop at [REDACTED]	Post-secondary students	<i>Jan–Feb 2027</i>	Outreach Lead	30+ students trained; 30 flight nodes pass acceptance tests
Seneca open house booth with a live node demo	General public	<i>Spring 2027</i>	<i>Outreach Lead</i>	200+ visitors; sign-ups to the mailing list
Social media updates	General public	<i>Monthly from Nov 2026; daily during the Aug 2027 flight campaign</i>	<i>Team Lead</i>	Views and followers before vs. after
Conference paper or poster on flight results	Academic	<i>Fall 2027</i>	<i>Team Lead</i>	1 paper or poster submitted
Open-source release: node design, firmware,	Academic and makers	<i>Within 2 months of flight</i>	<i>Software lead</i>	Repository published;

Event	Audience	Date	Owner	Measurable outcome
-------	----------	------	-------	--------------------

training code,
flight dataset

downloads and
forks tracked



REDACTED · partner details

Moonshot 6 booth (expected Jan to Mar 2027, date still an estimate): live node demo, swarm learning display and a hands-on sensor activity for visiting school groups, for K-12 students and the general public. Astra Labs has run a booth at the last two Moonshots showing our rocketry work. Owner: Outreach Lead. Measurable outcome: number of K-12 visitors and school groups at the booth, and a short exit survey on what students learned.

Canadian CubeSat Conference 2027 (SEDS-Canada, Montreal, March 2027): booth or poster on the node hardware and on-board learning results, for the academic community and student satellite teams. Owner: Team Lead. Measurable outcome: 1 booth or poster, number of teams engaged, and any test-collaboration leads.

Each event gets a SEDS Outreach Activities Report with attendance by audience level, feedback, and photos.

Appendices

Appendix A. Mass budget

Estimated total **3,360 g against the 5,000 g limit, a 33% margin** (Raspberry Pi 5; nodes bus-powered from one central pack). E = estimate, M1 = datasheet.

Component	Est. type	Qty	Unit [g]	Total [g]	Mass fraction	Remarks
Structure and mechanisms				750	22%	
Ring rack (5 levels × 6 slots)	E	1	600	600		Aluminum or printed; to CAD
Fasteners and gondola interface	E	1	150	150		
Payload (sensor nodes)				1,050	31%	
Node board with sensors and CAN transceiver	E	30	35	1,050		No flight cell; mass to confirm on the first board
Power				700	21%	
Power spine, arming keys, floor switches	E	1	100	100		Replaces per-node cells
Central battery pack	E	1	600	600		3S4P 18650 + BMS, ~130 Wh, fused
Data handling and				190	6%	

Component	Est. type	Qty	Unit [g]	Total [g]	Mass fraction	Remarks
communications						
Raspberry Pi 5 and SSD	M1	1	100	100		
Dual CAN controller board	E	1	30	30		
GPS module	M1	1	20	20		High-altitude capable
LoRa radio and antenna	E	1	40	40		
Thermal				470	14%	
Aluminum heat spreader	E	1	120	120		
Enclosure and foam insulation	E	1	350	350		
Miscellaneous				200	6%	
Wiring harness and connectors	E	1	200	200		2 CAN buses + power
Total				3,360	100%	Target 5,000 g · margin 33%

Appendix B. Power budget

Every load runs from the central 3S4P 18650 pack of cold-rated cells (3,000 mAh minimum, ~130 Wh rated). Usable energy assumes the pack loses half its capacity in the cold.

Load	Est. type	Pre-launch , 1 h: power [W]	Flight, 2.5 h: power [W]	Landed until recovery, 1 h: power [W]	Energy used [Wh]	Usable energy [Wh]	Margin
30 sensor nodes	E	4.5	9.0	1.5	28.5	From the central pack	See pack margin below
Central computer, GPS, CAN, radio	E	4.5	6.0	3.4	22.9	65 (of ~130 rated), shared by both rows	13% for the whole pack (51.4 Wh + 10% conversio n loss $\approx$ 56.5 Wh vs 65 Wh)

Depth of discharge at end of flight: *about 87% of usable cold capacity (56.5 of 65 Wh, estimate), to confirm in cold-chamber tests.* Training stops when landing is detected, to save energy for recovery. Extending the landed phase to meet the 5 h requirement adds about 2.7 Wh: about 59 Wh needed against 65 Wh usable, a ~9% margin. The landed figure uses a Raspberry Pi 5 idle draw of about 3 W.

Appendix C. Risk register

P = probability, C = consequence (L/M/H), inherent -> residual after mitigation. TR = technical, HR = human/health, MR = managerial.

Risk #	Risk	P / C	Mitigation	Contingency	Residual P / C
TR1	On-device training too slow or memory-hungry on the ESP32-S3	M / H	Benchmarked on an ESP32-S3 (Oct 2026): 1.31 s per round against a 120 s budget, 51 KB of RAM for four model copies; repeat with real sensors, CAN bus and cold	Nodes run models only; central computer trains (still compares frozen vs centralized)	L / L
TR2	Models fail to beat the persistence baseline	M / M	Physics input feature; tuned on ground data	Report it: a valid negative result; Goals 1 and 3 still met	M / L
TR3	Central pack loses too much capacity in the cold	M / H	Insulated pack compartment next to the Pi; high-capacity cold-rated cells; charge × temperature tests	Fewer rounds, then stop training and keep logging	L / M
TR4	A CAN bus fails and silences many nodes	L / H	Two separate buses; bus-off and listen-only firmware rules	Remaining bus continues with 15 nodes	L / M
TR5	Central computer overheats without convection	M / M	Heat spreader to enclosure wall; capped power mode; vacuum test	Fewer rounds; pause centralized model	L / L

Risk #	Risk	P / C	Mitigation	Contingency	Residual P / C
TR6	Pressure sensor at the edge of its range near 30 km	M / L	MS5611 rated to 10 hPa; vacuum-chamber check	Use GPS altitude above ~28 km	L / L
TR7	GPS stops reporting above 18 km	M / M	Choose a module with a high-altitude (airborne) mode; verify setting	Use pressure altitude and free-running clock	L / L
TR8	Mass or volume grows past the limits	L / H	33% mass margin; CAD fit check early	Fly 20–24 nodes instead of 30	L / M
HR1	Lithium-ion fire or injury (12-cell central pack and bench cells)	L / H	Pack with BMS; fire-safe charging; trained handlers; CSA Hazard Sheet	Fire extinguisher and isolation procedure	L / M
HR2	Burns or cuts during the student workshop	M / L	Supervision, PPE, safety briefing	First aid on site	L / L
MR1	REDACTED				
MR2	Training firmware takes longer than planned	M / H	Start firmware in the first month; weekly bench milestones	Fallback in TR1	L / M

Risk #	Risk	P / C	Mitigation	Contingency	Residual P / C
MR3	CSA does not accept our pack (survivability not proven at the campaign)	M / H	Vacuum and thermal tests before the campaign; bring the test evidence	Switch to the CSA onboard power system	L / M

Appendix D. ESP32-S3 node bench test

One ESP32-S3 trained the full mission model on board in 0.29 s per round per learning copy (1.31 s for all three trainable copies with every other task running, about 90× faster than the 2-minute round needs), and passed every check in a 60 s mission rehearsal (8 October 2026). This is a processor-only result on simulated data: real sensors, the CAN transceiver, cold and flight power are not yet tested.

Setup: ESP32-S3 rev 2 (2 cores at 240 MHz, 4 MB flash, 2 MB QSPI PSRAM), Arduino core 3.3 / ESP-IDF 5.5.5. Forward pass, backpropagation and SGD are written in plain C for the 96-24-12-18 mission network (2,862 parameters). Firmware ESP32S3_Bench v3 ran 15 tests, and repeat runs agreed within 3%. A second sketch, ESP32S3_TrainDemo, tested forecasting.

What it can do (measured)

Test	Result	Against the need
Inference, one forecast	0.15 ms	Forecasts every 5 s
Training step, one example	0.41 ms	n/a
One round, one copy (3 passes × 240 examples)	0.29 s at 240 MHz; 0.89 s at 80 MHz	120 s round budget
Three trainable copies with sensors, CAN, inference and logging running (60 s rehearsal, 47 rounds, 6 FreeRTOS tasks on 2 cores)	1.31 s average, 1.33 s worst	About 90× headroom
Memory per trainable copy	12.6 KB; 4 copies ≈ 51 KB in internal RAM	190 KB free; PSRAM not needed
100 Hz sensor loop while training	6,125 ticks, none late by more than 2 ms	Real-time tasks unaffected
8-bit compression of an update	2.5 ms; 11.4 KB → 2.9 KB; worst error 0.39%	Updates sent as 8-bit
CAN controller self-test (no transceiver)	One 2.9 KB model in 103 ms at 500 kbit/s, 53 ms at 1 Mbit/s; 0 errors	About 1.5 s per bus for 15 nodes

FedAvg merge of 15 models on one node	8.2 ms	A node could merge in safe mode
Flash logging	1,408 KB partition $\approx$ 20 h of logs	Flight about 4 h
Chip self-heating	46.5 °C $\rightarrow$ 55.5 °C after 20 s at full load	Useful heat in the cold

Where it stops (limits found)

- Largest model that trains a round in under 10 s: 38,898 parameters, 13.6× the mission model.
- Largest model that fits in memory: 450,403 parameters (1.7 MB in PSRAM), but one round takes 3.0 min, too slow for 2-minute rounds.
- Models above about 60 KB spill into PSRAM and train 2 to 3× slower per parameter.
- Double precision is software-emulated and 38× slower than float32, so the firmware uses float32 only.
- Plain-C 8-bit maths is slower than float (107 μ s vs 79 μ s per layer), so 8-bit is used only for transport.
- Both cores together give 1.47×, not 2×, because they share one memory bus.
- A 64 KB flash erase stalls both cores for about 20 ms, so log space is pre-erased before launch and in-flight erases stay at 4 KB.

Can it forecast? (simulated ascent)

The board trained in 4.7 s on the first 20 minutes of a simulated 5 m/s ascent, then forecast 16 minutes it had never seen. Skill = $1 - (\text{model error} \div \text{"no change" error})$; above 0 beats assuming nothing changes.

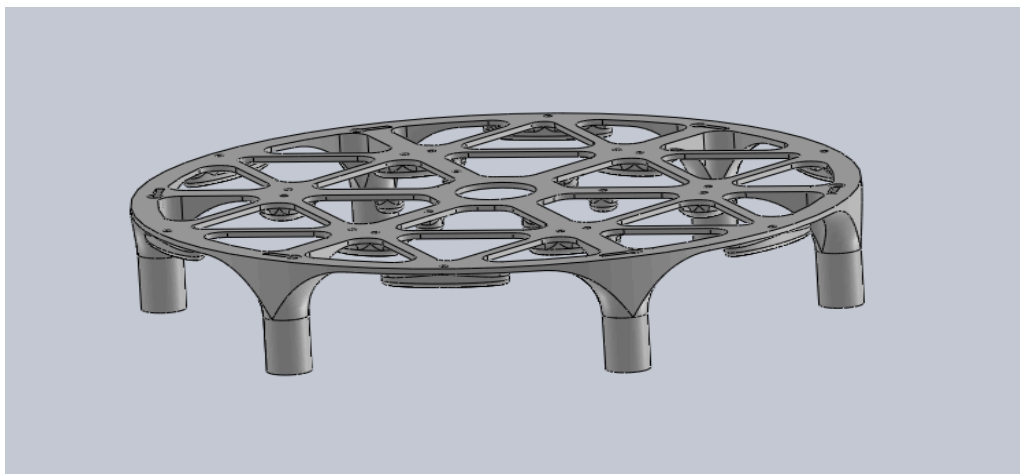
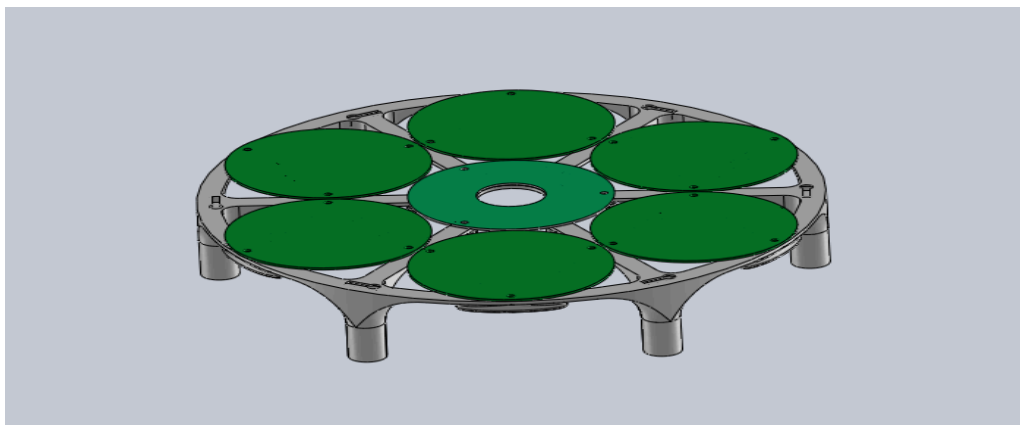
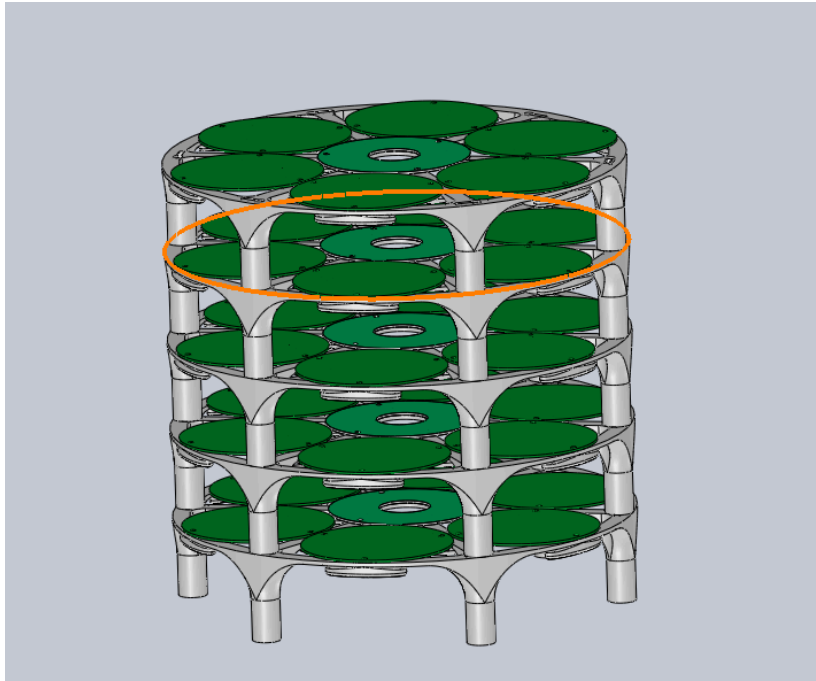
Channel	Skill +1 min	Skill +3 min	Skill +5 min
Node temperature	+0.74	+0.85	+0.87
Pressure	+0.52	+0.77	+0.85
Cell voltage	+0.06	+0.58	+0.51
Heading (sin)	+0.39	−0.10	+0.19
Rotation rate	+0.06	−0.51	−0.09
Current	−0.27	−1.00	−0.51

The model learns the physics-driven channels but fails on rotation and current, which we randomized in the simulation. The flight baselines ([Section 2.9](#)) are built to show this kind of result. The voltage channel has since been replaced with ESP32-S3 chip temperature, so we will re-run the demo.

Not yet tested

Item	Next test
Real sensors on SPI/I ² C	Re-run the rehearsal with real sensor reads
Real CAN bus with transceivers and 15 nodes	Two-node loop with the Raspberry Pi 5, then 4 and 15 nodes
Power draw at 240 vs 80 MHz	USB power meter, idle vs training
Cold (−20 to −40 °C)	Freezer run of the same firmware
Hours-long stability	4 h rehearsal, logging free memory every minute
Forecast skill on real data	Train on radiosonde profiles and bench burn-in logs

Appendix E. Payload structure (CAD)



References

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., Agüera y Arcas, B. (2017). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. AISTATS. [arXiv:1602.05629](https://arxiv.org/abs/1602.05629)
2. Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V. (2020). *Federated Optimization in Heterogeneous Networks*. MLSys. [arXiv:1812.06127](https://arxiv.org/abs/1812.06127)
3. Kairouz, P., McMahan, H. B., et al. (2021). *Advances and Open Problems in Federated Learning*. Foundations and Trends in Machine Learning. [arXiv:1912.04977](https://arxiv.org/abs/1912.04977)
4. Arivazhagan, M. G., Aggarwal, V., Singh, A. K., Choudhary, S. (2019). *Federated Learning with Personalization Layers*. [arXiv:1912.00818](https://arxiv.org/abs/1912.00818)
5. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W. J. (2018). *Deep Gradient Compression*. ICLR. [arXiv:1712.01887](https://arxiv.org/abs/1712.01887)
6. Llisterri Giménez, N., Monfort Grau, M., Pueyo Centelles, R., Freitag, F. (2022). *On-Device Training of Machine Learning Models on Microcontrollers with Federated Learning*. Electronics 11(4):573. [Link](#)
7. Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., Han, S. (2022). *On-Device Training Under 256KB Memory*. NeurIPS. [arXiv:2206.15472](https://arxiv.org/abs/2206.15472)
8. Ren, H., Anicic, D., Runkler, T. (2021). *TinyOL: TinyML with Online-Learning on Microcontrollers*. IJCNN. [arXiv:2103.08295](https://arxiv.org/abs/2103.08295)
9. Matthiesen, B., Razmi, N., Leyva-Mayorga, I., Dekorsy, A., Popovski, P. (2023). *Federated Learning in Satellite Constellations*. IEEE Network. [arXiv:2206.00307](https://arxiv.org/abs/2206.00307)
10. Razmi, N., Matthiesen, B., Dekorsy, A., Popovski, P. (2022). *On-Board Federated Learning for Dense LEO Constellations*. IEEE ICC. [arXiv:2111.12769](https://arxiv.org/abs/2111.12769)
11. Karniadakis, G. E., et al. (2021). *Physics-informed machine learning*. Nature Reviews Physics 3:422–440. [Link](#)
12. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V. (2022). *Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems*. ACM Computing Surveys. [arXiv:2003.04919](https://arxiv.org/abs/2003.04919)
13. NOAA, NASA, USAF (1976). *U.S. Standard Atmosphere, 1976*. NASA TM-X-74335. [NTRS](#)
14. Durre, I., Vose, R. S., Wuertz, D. B. (2006). *Overview of the Integrated Global Radiosonde Archive*. Journal of Climate. [IGRA data \(NOAA\)](#)
15. Canadian Space Agency. *Departmental Plan 2026–2027*. [Link](#)
16. TE Connectivity. *MS5611-01BA03 Barometric Pressure Sensor datasheet (10–1200 mbar)*. [PDF](#)
17. SEDS-Canada. *CAN-SBX Application Guide 2025–2026 and Application Form*. [Guide](#) · [Form](#)
18. Gama, J., Sebastião, R., Rodrigues, P. P. (2013). On evaluating stream learning algorithms. Machine Learning 90(3):317–346.
19. Kim, G. R., Svoboda, F., Lane, N. D. (2025). Bringing Federated Learning to Space. [arXiv:2511.14889](https://arxiv.org/abs/2511.14889)
20. Gómez, P., Meoni, G. (2024). Tackling the Satellite Downlink Bottleneck with Federated Onboard Learning of Image Compression. IEEE/CVF CVPR Workshops, 6809–6818. doi:10.1109/CVPRW63382.2024.00674
21. Mateo-Garcia, G., Veitch-Michaelis, J., Purcell, C., Longepe, N., Reid, S., Anlind, A., Bruhn, F., Parr, J., Mathieu, P. P. (2023). In-orbit demonstration of a re-trainable machine learning payload for processing optical imagery. Scientific Reports 13. doi:10.1038/s41598-023-34436-w

22. SEDS-Canada. CAN-SBX past participants: ASTRA, University of Waterloo (2025).
<https://www.seds.ca/?p=1816>