Sensor Data Dashboard

Overview

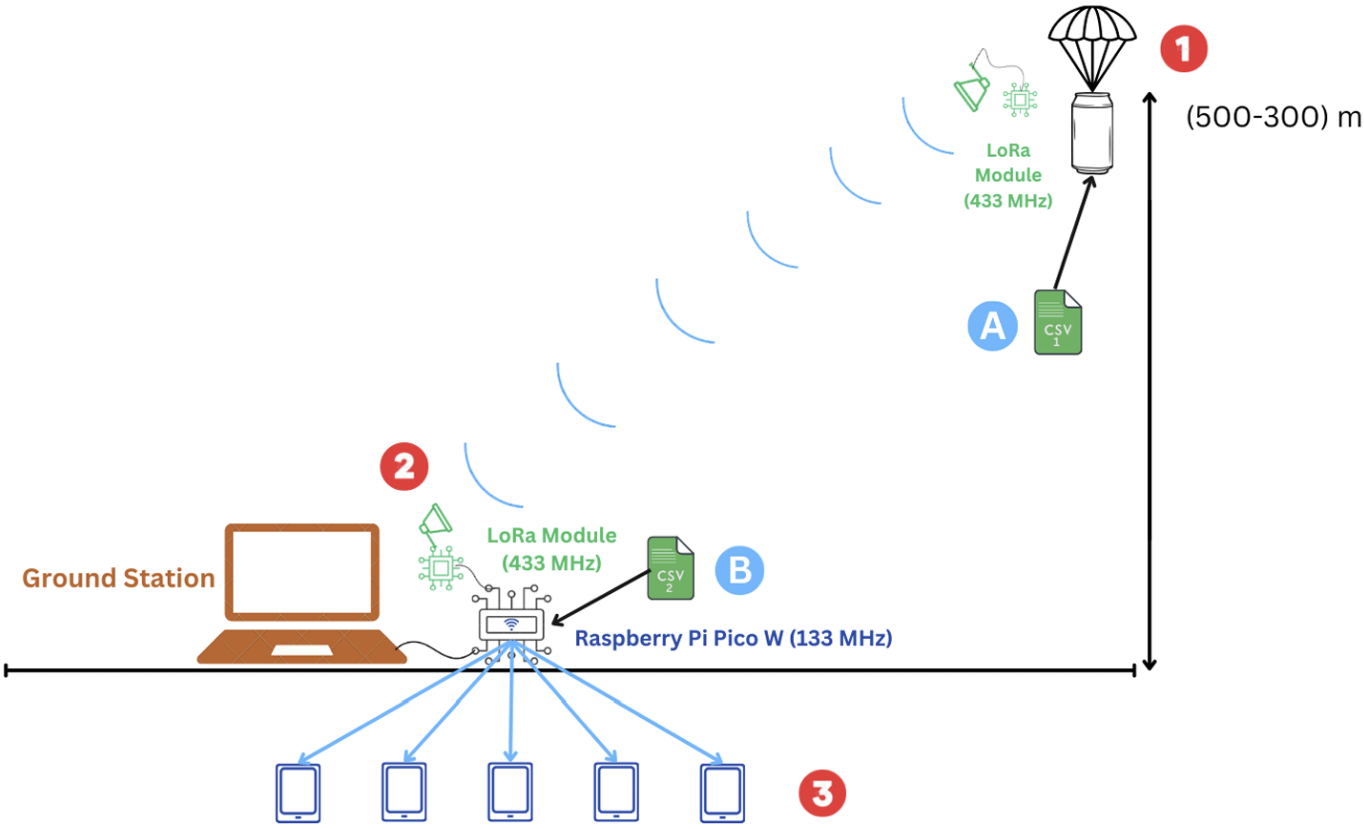
This is a real-time data visualization tool developed to graphically interpret atmospheric sensor data collected during descent. This allows for real-time monitoring of atmospheric data which aid to validate data authenticity. This tool was specifically designed for a CanSat project.

Overview of libraries used

Library/ Tool	Type	Pros	Specific Use cases	Reasons for choice
network	Built-in (MicroPython)	- Simple API for network configurations - Allows Wi-fi setup for both access points and client modes - Allows connection status monitoring	- configuring the Pico W as an access point - server Wi-Fi connection management	- easy to use - compatible with the Pico W's built-in wireless module
time	Built-in (MicroPython)	-lightweight -supports timing functions in constrained environments	- ensures adequate timing during server setup (event synchronization) - allows timing of flight	- easy to use
socket	Built-in (MicroPython)	-provides low-level socket communication, ideal for light-weight servers - consumes minimal memory	- server setup to handle incoming HTTP requests - server setup to send HTTP responses.	- provides direct control over network - built-into MicroPython, so no external dependencies

chart.umd.js	External (JavaScript)	- highly customizable for interactive, responsive charts - supports a variety of charts (line, scatter, etc.) - lightweight and quick for client-side rendering	- allows us to render and visualize our data points in real-time	- the umd version is lightweight as all unnecessary spaces are removed, immensely reducing file size. - offers a lot of chart features without requiring a heavy JavaScript framework - responsive charts automatically adjust to different screen sizes
regression.min.js	External (JavaScript)	- simple and lightweight - supports a range of regression models (linear, polynomial, exponential) - fast client processing without server-side load - easily compatible libraries like chart.js	- allows analysis for sensor data trends and predictions - ease of data trend comparison between our datasets and other historical datasets.	- reduces computational burden on the server (Pi Pico W), making it more efficient

Broadcasting System



Refinements:

1. Once the CanSat is in the air, it will start collecting data and broadcasting it, via a LoRa module, to a ground station. A second LoRa module, positioned at the ground station, picks up these signals and transmits the data to a Raspberry Pi Pico W.

2. Because the frequencies of the Raspberry Pi Pico W and the LoRa module are 433MHz and 133MHz, respectively, there should be minimal interference between the two.

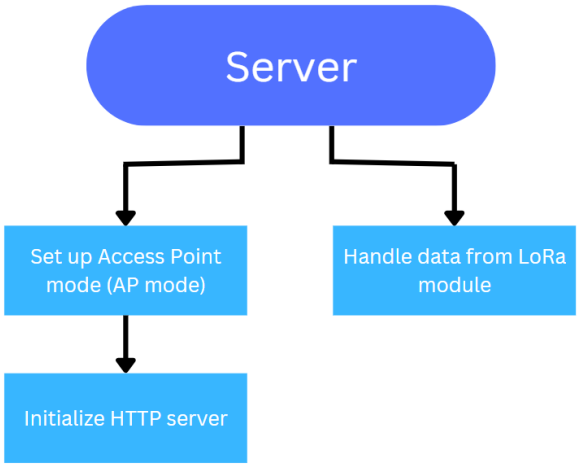
3. The Raspberry Pi Pico W will wirelessly transmit this data to all the clients connected, allowing real-time data visualization.

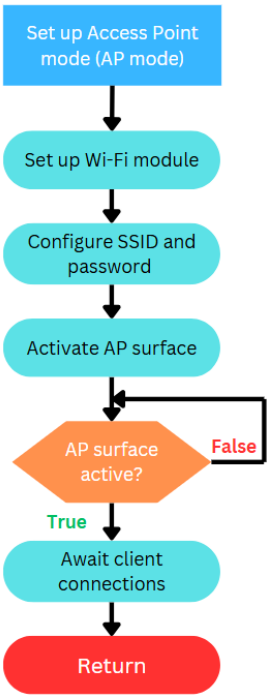
A. CSV 1 will be stored locally on the disk drive of the microcontroller located within the CanSat; this is to ensure that we will, still, have the data even if the connection between the two LoRa modules is broken.

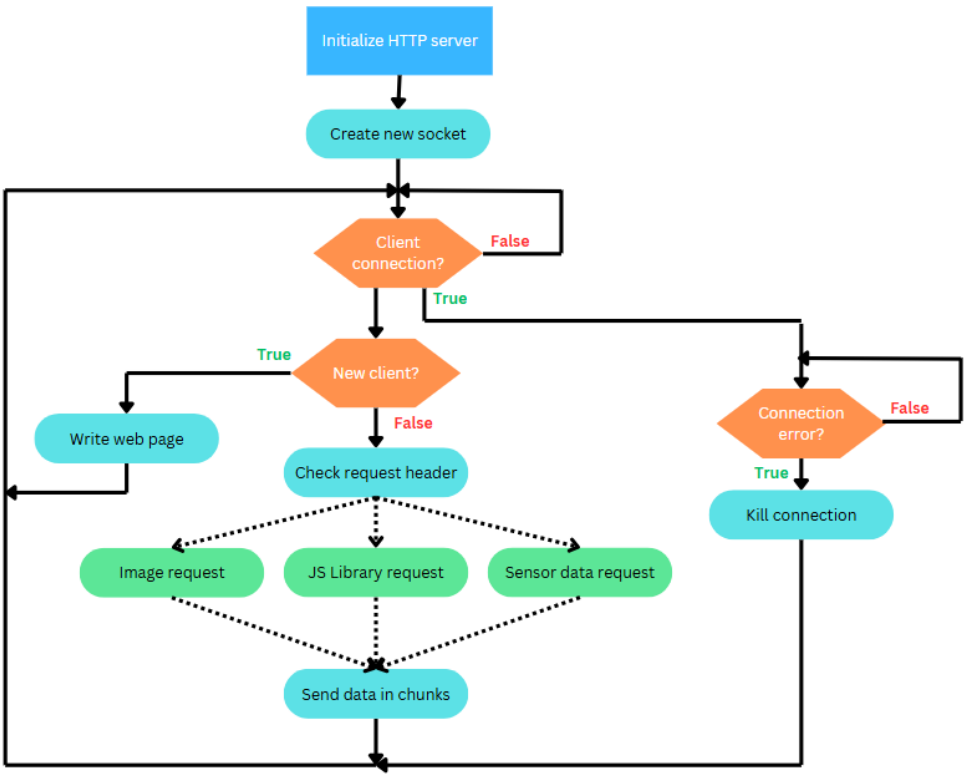
B. CSV 2 will be stored locally on the disk drive of the Raspberry Pi Pico W, positioned at the ground station; this is to ensure that the data will still be obtainable even if the CanSat is unable to be recovered.

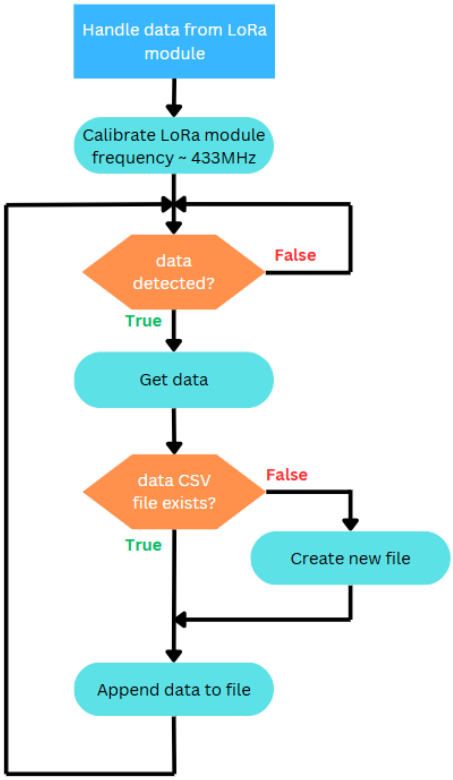
Software

Backend

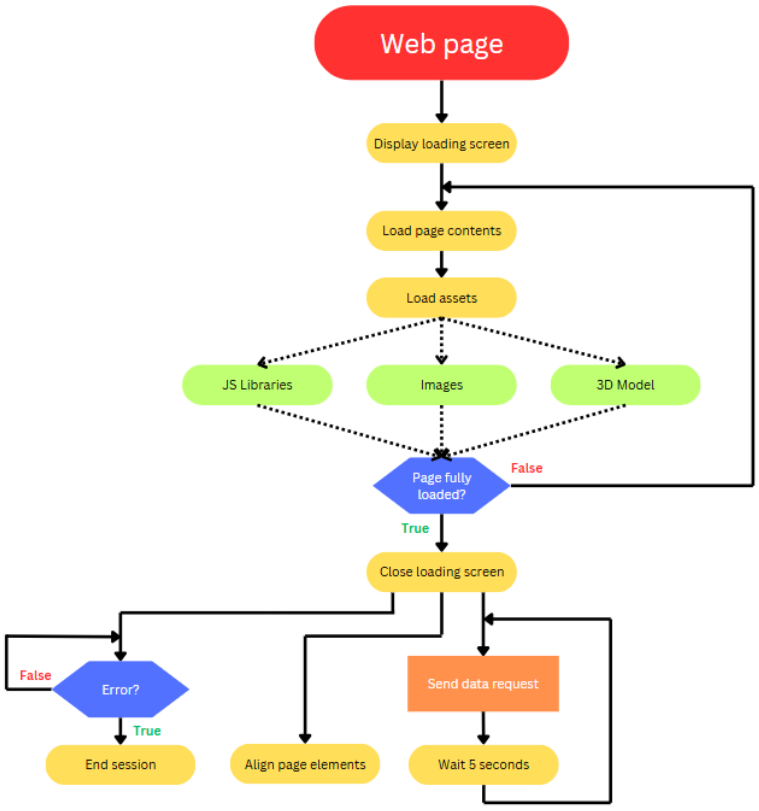


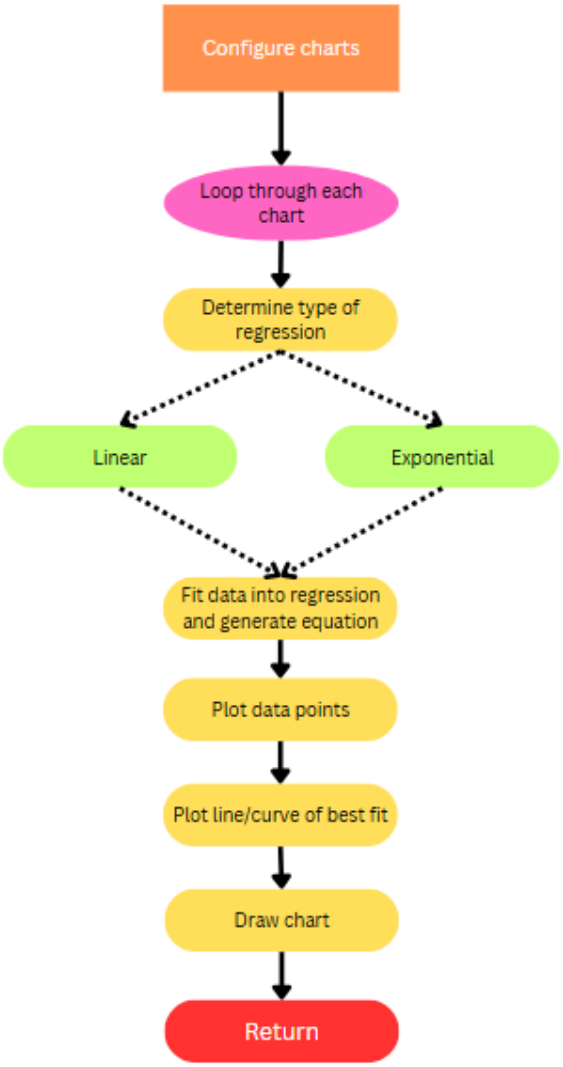


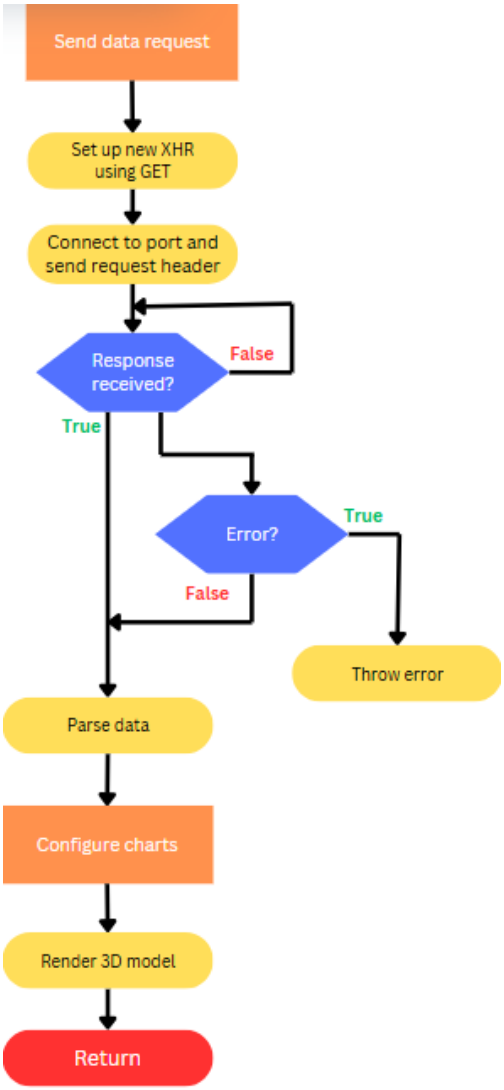




Frontend


Web page
Display loading screen
Load page contents
Load assets
JS Libraries
Images
3D Model
Page fully loaded?
False
True
Close loading screen
False
Error?
True
End session
Align page elements
Send data request
Wait 5 seconds


Configure charts
Loop through each chart
Determine type of regression
Linear
Exponential
Fit data into regression and generate equation
Plot data points
Plot line/curve of best fit
Draw chart
Return


Send data request
Set up new XHR using GET
Connect to port and send request header
Response received?
False
True
Error?
True
False
Throw error
Parse data
Configure charts
Render 3D model
Return