**Telemetry Documentation**

**Section 1 - Network Overview**

Helium Console

The project currently uses the Helium Console for telemetry. Found at: <https://console.helium.com/> The Helium Console is a web-based tool for the deployment and management of LoraWAN devices. The Console allows the registration and monitoring of up to 10 Class A devices which are then hosted on the Helium network. The Helium Console also provides prebuilt connections known as Integrations to route device uplinks to external systems and services. Data transfer is “credit based,” and the Helium Network charges for data transfer credits using a pay as you need model. In-depth documentation for the Helium Console can be found here: <https://docs.helium.com/console/api>

Helium vs TTN

Previous iterations of the project employed The Things Network for telemetry, but the newest iteration (V4) changed to use The Helium Console during development. The change was due to TTN’s limited coverage in the test area. Connection to TTN was entirely dependent on distance to the configured in-house gateway. Even when a sensor connected to a different gateway, TTN readout would not register the connection. This was strange as the LoraWAN network is a mesh-network that allows connection through any gateway. The Helium Console provides wider coverage and less complicated setup and thus was favored over TTN.

There are some disadvantages to using Helium over TTN. The Helium Console only supports Class A devices whereas TTN supports Class A, B, and C devices. TTN also allows for asynchronous data reading as well as downlink sending. However, Helium operates on an uplink to uplink basis for sending downlinks and reading data. TTN allows manual location setting and monitoring for devices that The Helium Console does not support.

Despite all this, The Helium Console is more advantageous for the project than TTN. As this project uses Class A devices to gather data, many of the benefits that TTN provides are outside the necessities of the project. Furthermore, Helium provides case-by-case data transfer for devices with its “Flows” system and built-in Integrations, making the process from testing to deployment more flexible. It also offers easy and accessible debugging for uplinks, downlinks, functions and integrations. All in all, The Helium Console offers simplicity and functionality, and is incredibly easy to build on.

**Section 2 - Helium Overview**

Nodes

The Helium Console is divided into four sections, three of which have sublevels for management of devices and data. The Nodes section has three subsections.

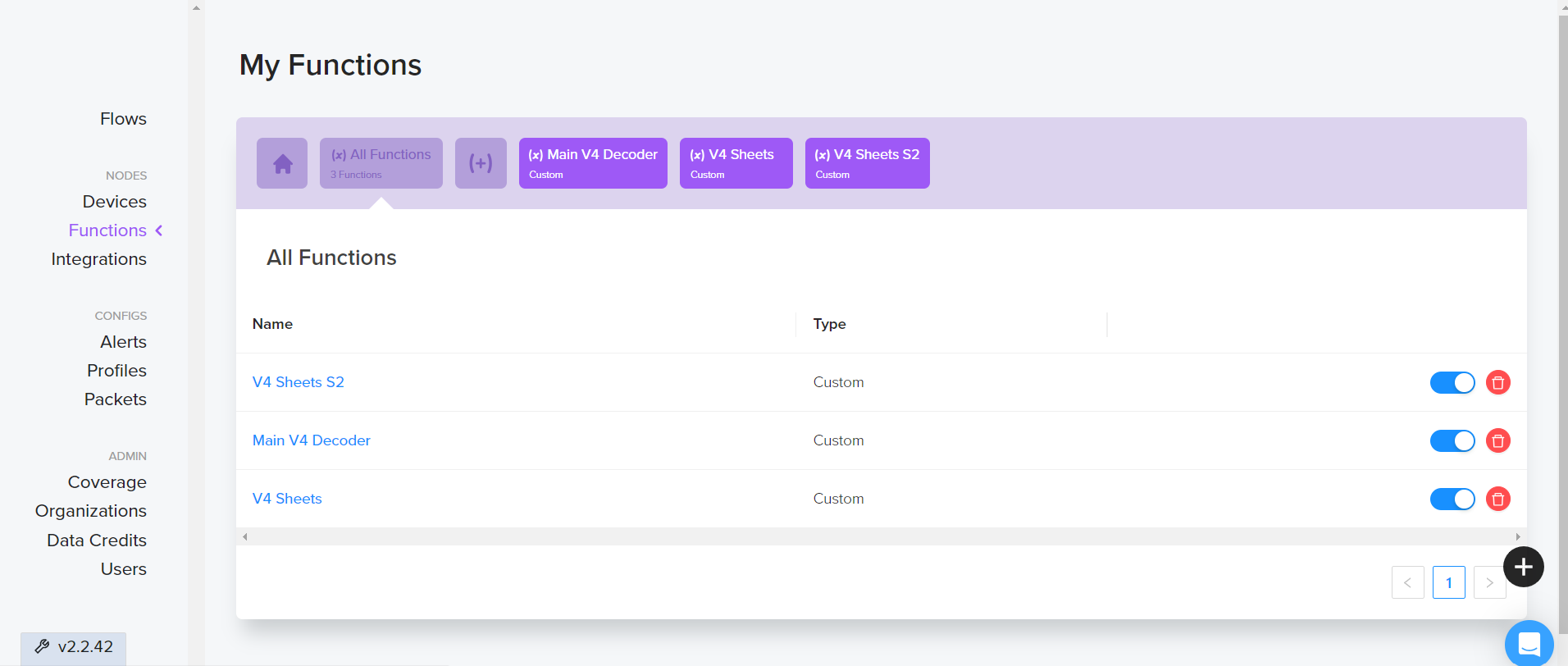
*Devices:*



***Figure: Devices***

The Devices panel is where registration and management of the devices takes place. Each device listing reports activation date and last heard date. Each device page reports device specifics (DevEUI, AppEUI, AppKey, Packets Sent, etc.) along with live data about uplinks and downlinks between the device and Helium.

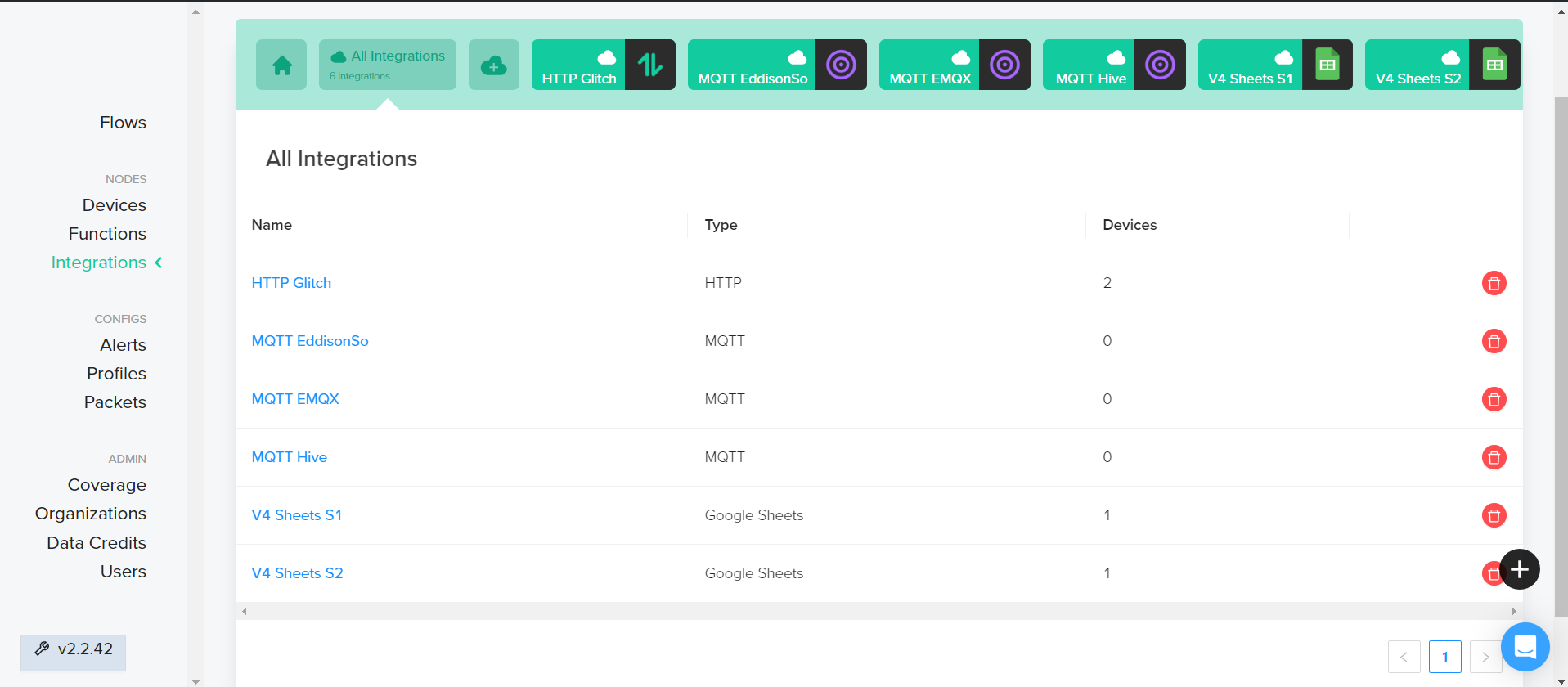
*Functions:*



***Figure: Functions***

The Functions panel is where functions that decode raw data are written and managed. Functions are the middle communicators between devices and integrations. Each function can receive input from multiple devices, decode that input, then output to multiple integrations. Functions are programmed in JavaScript and there are provided examples when creating a function.

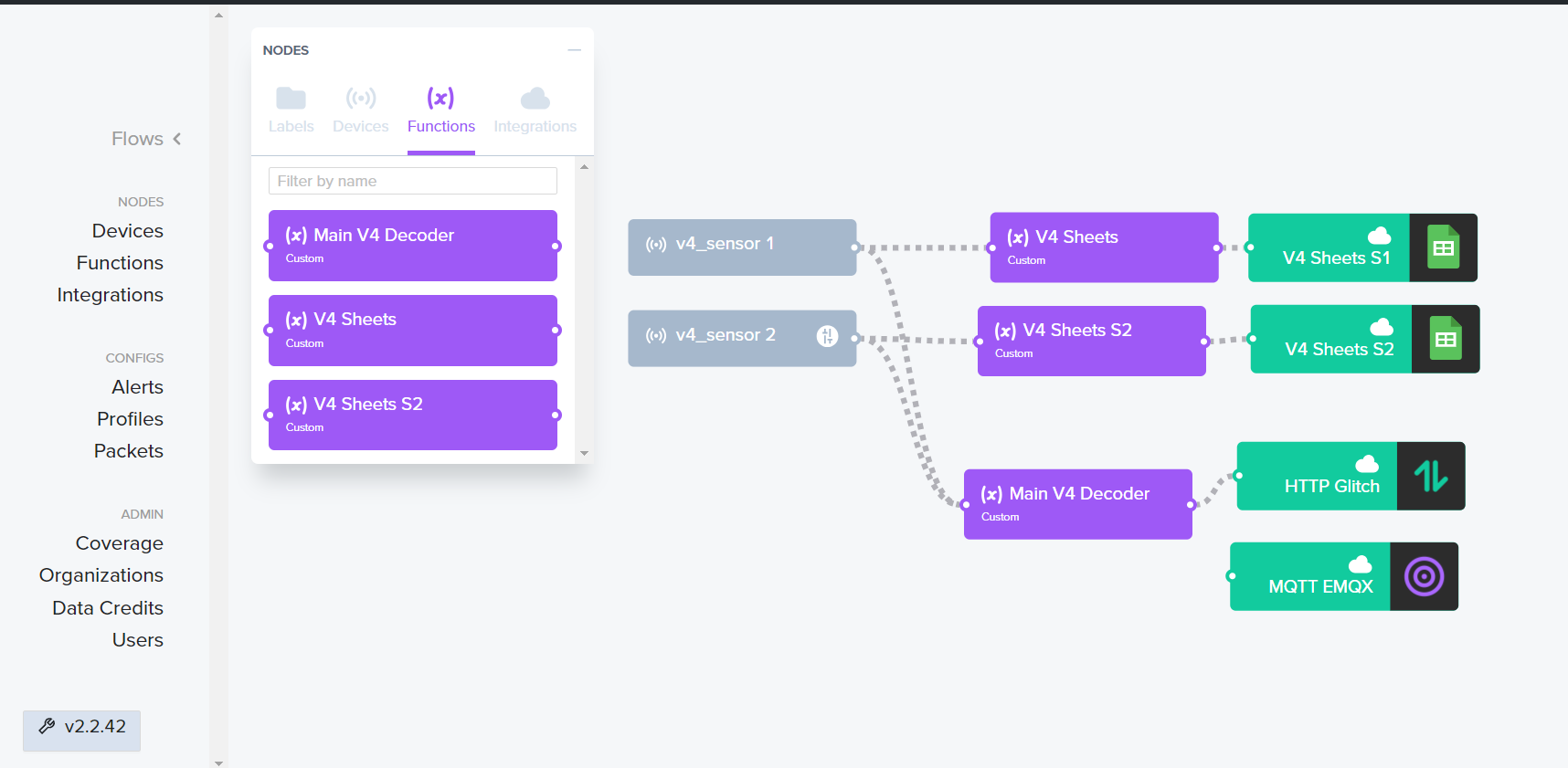
*Integrations:*



***Figure: Integration***

The Integrations panel is where external systems and services are added. The Helium Console has many pre-built integrations, among which are basic HTTP and MQTT integration. The project employs a Google Sheet integration as well as a basic “POST” HTTP integration. The Google Sheet integration routes uplinks to a Google Form *linked* to a Google Sheet and has instructions for setup.

Flows



***Figure: Flows***

The Flows panel is where sensors are connected to functions and integrations. Flows are constructed in a node-based fashion, and each node is connected from endpoint to endpoint. Nodes are connected by edges that can be clicked on, observed, and severed.

Admin

*Coverage:*

* The Coverage panel contains a list of hotspots that sensors in the application have pinged recently, as well as a map where those hotspots are. Different hotspots can be followed for updates or specifically preferred for packet routing.

*Organizations:*

* The Organizations panel contains organizations a user is associated with as well as a summary of the devices deployed in each one.

*Data Credits:*

* The Data Credits panel contains an organization’s data credits, data transfer bytes according to purchased credits, and a payment history. New Credits can also be purchased from here.

*Users:*

* The Users panel is where an organization’s members are added and managed. In order to use the Helium Console, every user must be affiliated with an organization. Either a user creates their own organization on signup, or they are invited to a pre-existing organization.

**Section 3 - Active Integrations**

Google Sheets

The primary, active integration associated with the project is a Google Sheet in which data from two sensors is collected. Each sensor publishes to a different Google Form linked to different sheets in the Google Sheet. Date, Sensor Error, Voltage, Distance to Ground, Accumulation, and Event Accumulation are recorded.

Website

The second, active integration is an HTTP integration that posts to a website for public visualization found here: <https://ismart-labs-floodsensors.glitch.me/> The website is a full-stack application hosted (for the moment) on Glitch for $10 a month.

On the backend, the website uses a node.js/express server and employs an “update on uplink” method for serving data to the front-end. The server receives POST requests from Helium and appends the uplinks to a sensor in a JSON file. The server also has GET methods built in for status reporting.

On the frontend, the website uses mapbox-gl for visualization of sensor location and status. A custom map was created in mapbox and was added to the website through mapbox APIs. The website also uses ej2 Charts for visualization of data points. Data charts report data from the time of access up to an hour before.

