

## General Description

The AnyLeaf CAN ExpressLRS (ELRS) receiver is a 2.4GHz radio receiver intended for use on small remotely-piloted vehicles. Suitable uses include on quadcopters, fixed-wing aircraft, boats, and wheeled or tracked vehicles. When paired with an ELRS transmitter, they provide a control link between the pilot and aircraft: Transmitters send control channel data (eg commanded pitch, roll, yaw and throttle settings) to receivers. Receivers process this data, and pass it to flight controllers or other control systems over CAN. They transmit telemetry data back to the transmitter. In addition to control channel data, ELRS receivers pass link statistics to their flight controllers, which includes data about signal strength.

This device uses 2 independent radio pathways, which each include a power amplifiers (PA), and low-noise amplifier. (LNA), connected to Semtech SX1280 or SX1281 LoRa radios. Each pathway has its own antenna connection.

This datasheet includes a mix of information generic to ExpressLRS, specific to DroneCAN, and specific to this device. The official ExpressLRS documentation is available at this website:

<https://www.expresslrs.org/3.0/quick-start/getting-started/>

## Specifications

- **Dimensions:** 49 × 49 × 17 (height) mm. 70mm width with tabs
- **Mounting holes:** 2 × M4, spaced 60.4mm
- **Weight:** 28 grams
- **Power input:** 5V, via CAN, USB-C, or UART
- **Operating frequency:** 2.4GHz
- **Node MCU:** Stm32G431. 170Mhz Cortex-M4
- **Radio MCU:** ESP32-PICO-V3
- **Radio Receiver:** 2 × Semtech SX1280, with DC-DC power
- **Low-noise Amplifier / Power Amplifiers:** 2 × SkyWorks RFX2401C
- **Power Amplifier amplification:** +22 dBm
- **Radio timing source:** 52Mhz temperature-compensated crystal oscillator (TCXO)
- **Node update capabilities:** USB-C, CAN
- **Radio Update capabilities:** WiFi, UART
- **Pin header to flight controller:** JST GH, 1.5mm pitch, 4 pins
- **Antenna connector:** 2 × IPEX MHF-1

- **Bus compatibility:** DroneCAN
- **CAN transceiver:** NXP TJA1051TK/3
- **CAN version:** CAN-FD capable
- **CAN headers:** 2 × JST GH, 1.25mm pitch
- **UART header:** JST GH, 1.25mm pitch
- **Max CAN datarate:** 5Mbps
- **Flight controller firmware compatibility:** Ardupilot, and PX4. Compatible with any firmware that supports the applicable *DroneCAN* message types.

This device uses the DroneCAN protocol, and is compatible with any flightcontroller that implements the applicable DroneCAN messages.

## Integrating with your aircraft

This device connects with aircraft systems using a 4-pin connection header; this powers the device, and allows two-way communication over CAN. It uses a JST-GH header, with connections labeled on the enclosure for 5V power, CAN data high, CAN data low, and ground. Because CAN is a bus, multiple peripherals can use these same wires for power and data, and routing can be set up in a way that makes sense for a given aircraft geometry. This device (and many CAN devices) includes two CAN connectors: This can be used to simplify wiring: For example, run one CAN cable from the flight controller to one of this device's connectors. Run another wire from this device's second connector to another CAN device in the same area of the aircraft.

## Protocol description

This device is compatible with DroneCAN, and can exist on busses that include other DroneCAN devices. It periodically broadcasts information from its onboard sensors. The broadcast rate of this information is customization, either using PC software available on the AnyLeaf website, using the USB-C connection on the device, or via a CAN configuration message.

Most messages broadcast by this device are included in the Dronecan *List of standard data types*: [https://dronecan.github.io/Specification/7. List of standard data types/](https://dronecan.github.io/Specification/7.List%20of%20standard%20data%20types/).

The standard data type use allows for compatibility with any flight control firmware that supports the DroneCAN standard. (For example, Ardupilot, and PX4.)

### Messages periodically broadcast:

- Control channel data (dronecan.sensors.rc.RCInput)
- Link statistics(dronecan.sensors.rc.LinkStats)

- Node status (uavcan.protocol.NodeStatus)

**Messages accepted:**

- Node info request (uavcan.protocol.GetNodeInfo)
- Dynamic node ID allocation (uavcan.protocol.dynamic\_node\_id/Allocation)
- Node restart (uavcan.protocol.RestartNode)
- Get/Set parameters (uavcan.protocol.param.GetSet)

**DroneCAN protocol**

This device uses the DroneCAN protocol. Information about its wire protocol can be found in its specification here; most notably in chapter 4:

<https://dronecan.github.io/Specification/4. CAN bus transport layer/>

The CAN ID format is as follows. Most messages broadcast by this device, including all sensor readings, use the *Message frame* format.

**Message frame**

Field name	Priority					Message type ID												Service not message											
	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																		0	1...127										
CAN ID bytes	3					2						1						0											

**Anonymous message frame**

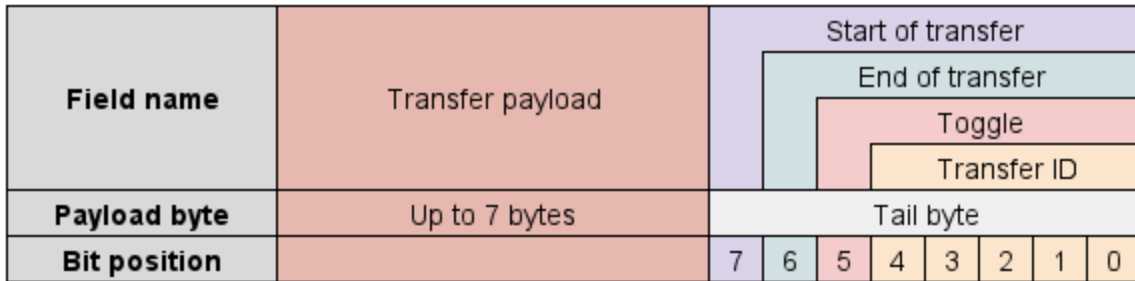
Field name	Priority					Discriminator										Lower bits of message type ID		Service not message											
	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																		0	0										
CAN ID bytes	3					2						1						0											

**Service frame**

Field name	Priority					Service type ID								Request not response							Service not message								
	28	27	26	25	24	23	22	21	20	19	18	17	16	15	Destination node ID							7	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values														1...127							1	1...127							
CAN ID bytes	3					2								1							0								

The first two bytes of every message is the CRC; information on decoding it is found in the DroneCAN Spec linked above. The final byte in each frame called is the tail byte; this uses the following format, and contains information describing if a payload is contained in a single frame, or split across multiple ones:

## CAN payload



Payload contents for the message types this device broadcasts are described below.

### Control channel data format:

This device broadcasts GNSS fixes using the DroneCAN *RCInput* message. This uses the following bit-aligned payload format:

Bits	Description	Data type
<b>0-15</b>	Status	Enum: 0: OK; 1: Good signal; 2: Link lost
<b>16-24</b>	Link quality, ranging between 0 and 255. A value of 255 means 100% of recently-transmitted Over-The-Air (OTA) control-data packets were successfully received. A value of 0 means none were. This is a good proxy for signal strength.	8-bit unsigned integer
<b>24-28</b>	ID of the device; currently unused	4-bit unsigned integer
<b>28+</b>	These represent control channel data. Each channel uses 12 bits, with no padding between channels. There are 4 primary channels, and an additional channel for every aux channel configured. (Described below) In FD-mode, these are prepended with a 6-bit array length field, containing the number of channels transmitted.	Array of 12-bit unsigned integers.

### Node status format:

This device periodically broadcasts the DroneCAN Node Status message. This message reports the following information:

Bits	Description	Data type
<b>0-31</b>	Timestamp since node start in $\mu$ s	32-bit unsigned integer
<b>32-34</b>	Node health	Enum: 0: OK; 1: Warning; 2: Error; 3: Critical.
<b>35-38</b>	Node mode	Enum: Operational; 1: Initialization; 2:

This device responds to DroneCAN *GetNodeInfo* and *Restart* requests. The node info response contains the node status message above, and additional information about software version, hardware version, and the node name. The serialization format for this info is somewhat complicated, and is beyond the scope of this datasheet.

This device responds to dynamic node ID allocation requests, if the *Dynamic node ID allocation* setting, described below, is enabled. (It is enabled by default)

## Configurable parameters

The following parameters can be customized. These settings are stored in non-volatile memory, and take effect after the device is restarted. They can be configured from the AnyLeaf *CAN Preflight* software, Mission Planner, or QgroundControl.

### Node Id

Either hard-sets the node ID, or specifies the desired node ID to send to the ID allocator. See the *Dynamic node ID allocation* setting for details on this. Defaults to 70.

### Dynamic node ID allocation

If set to true, node ID is determined by the DroneCAN dynamic node ID allocation process, and ID is 0 until assigned. (Broadcasts are anonymous, as defined in DroneCAN. In this case, the *Node ID* settings determines the desired ID to send to the allocator. If false, the *Node Id* settings is hard set as the ID. Defaults to true.

### FD mode

Enable this to support frame-lengths up to 64 bytes. If disabled, only 8-byte frames are supported. You should only enable this if your flight controller supports and is configured to use FD mode. Defaults to disabled.

### CAN bit rate

Select the data bit rate to use. This has discrete settings available: 250kbps, 500kpbs, 1Mbps, 2Mbps, 4Mbps, and 5Mbps. You should only enable values higher than 1Mbps if your flight controller supports and is configured to use FD mode. Defaults to 1Mbps.

### Update ratio

This sets, in conjunction with radio configuration, the rate control channel data and link stats messages are broadcast over CAN. The default of 1 broadcasts every message received. A value of 2 means every other message is broadcast. A value of 0 means no control data is broadcast. The baseline rate is configured on the transmitter, and is usually 500Hz or 1kHz. So, with the

radio set to broadcast at 500Hz, and a broadcast rate configured to be 1, control messages are broadcast over CAN at 500Hz.

### **Number of auxillary channels**

This receiver outputs 4 12-bit control-channel data packets, usually associated with pitch, roll, yaw, and throttle. Up to 8 additional 4-bit ("aux") channels are transmitted according to the ELRS spec. This setting controls how many of these to broadcast over CAN.

### **Enable or disable link stats**

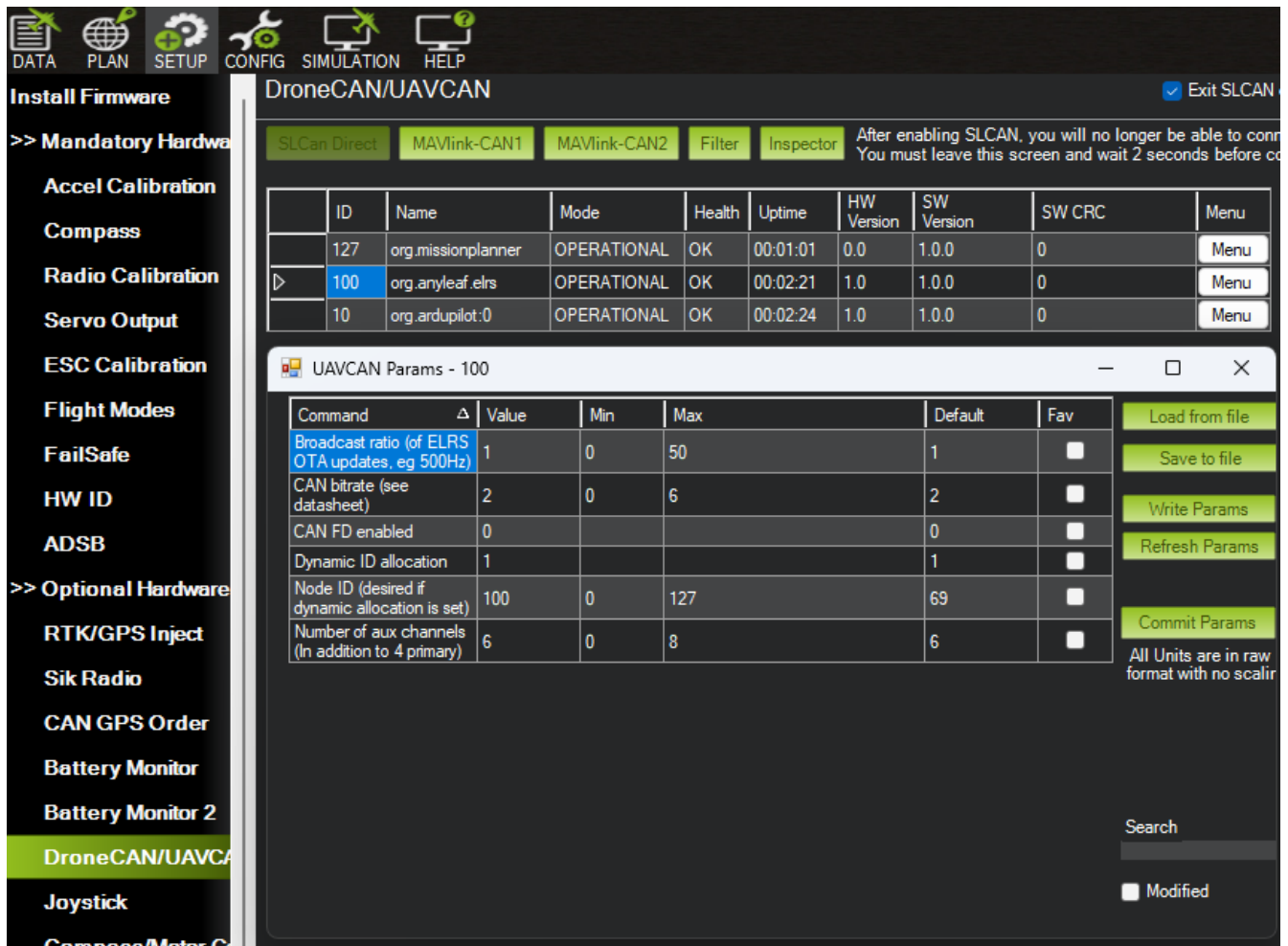
This device is capable of broadcasting detailed link statistics, including both uplink and downlink link quality, RSSI, and signal-to-noise ratio, for both radios. This is not a standard DroneCAN DSDL message, so you may disable this if it's not supported by your flight controller firmware. Note that using non-official messages can cause DroneCAN GUI to crash, which may drive disabling link stats from this device.

### **Enable or disable UART port**

The UART port on this device may be enabled to use this device without CAN. Note that if the port is enabled, the CAN reception is disabled.

## **Configuring and updating using Mission Planner**

It's possible to view and set the parameters listed above using *Mission Planner* or *QgroundControl*, through the standard DroneCAN parameter interface. To do this in Mission Planner, connect the device to a CAN-capable Flight-controller running ArduPilot or PX4. After connecting, select *Setup, Optional Hardware, DroneCAN/UAVAN*. The AnyLeaf device should be listed, as below:



Select *Menu*, then *Parameters*. The parameters described above will be displayed, with their minimum, maximum, and default values. After changing these, click *Write Params* on this window to save to the device.

## Configuring and updating over USB

To configure, update, and view device status over USB, download and run the *AnyLeaf CAN Preflight* software, from the link on the Anyleaf website's page for this product.

## LED status indications

The device uses a full-color LED that provides status indications for the radio connection. The meanings of both can be found at the official ELRs documentation here:

<https://www.expresslrs.org/3.0/quick-start/led-status/>

### Status LED indications:

LED Indication	Status
Rainbow fade effect	Starting Up
Green heartbeat	Web update mode enabled
Slow blink 500ms on/off	Waiting for connection from transmitter
Red flashing 100ms on/off	Radio chip not detected
Orange Double blink then pause	Binding mode enabled
Orange Triple blink then pause	Connected to transmitter but mismatched model-match configuration
Solid single color	Connected to a transmitter, color indicates packet rate

When connected, color indicates the current packet rate and RF protocol (selected on the transmitter), as follows:

**Red: 1kHz FLRC**

**Orange: 500Hz FLRC**

**Yellow: 500Hz Deja Vu**

**Light green: 250Hz Deja Vu**

**Dark green: 500Hz LoRa**

**Teal: 333Hz LoRa**

**Light blue: 250Hz LoRa**

**Dark blue: 150Hz LoRa**

**Purple: 100Hz LoRa**

**Pink: 50Hz LoRa**

These rates and protocols are selected on the transmitter, in its ELRS settings. (Eg *SYS* button, *Tools* tab, *ExpressLRS* LUA, on Radiomaster controllers. *Note* that FLRC provides lower latency, while LoRa provides longer range. Déjà Vu sends the same packet multiple times at 1kHz using FLRC, which provides resistance to interference.

To use F1000, you may need to increase your radio controller's baud rate from its default. For example, on Radiomaster controllers, this is adjustable by pressing the *SYS* button, then navigating to the *Hardware* tab. Setting the rate above 400kHz may be required.

## Binding to a transmitter

Reference the ExpressLRS documentation here for details on binding a transmitter:

<https://www.expresslrs.org/3.0/quick-start/binding/>



By default, the receiver is set up in manual bind mode. After power is applied, the status LED will blink orange, indicating it is ready to bind. To bind from this mode, activate the *Bind* feature using your transmitter's ELRS LUA script.

For automatic binding, set up a binding phrase shared by the transmitter and receiver. This can be set, for either, by allowing the transmitter and receiver each to go into WiFi mode. Using a computer or phone with WiFi capability, connect to the *ExpressLRS RX* or *ExpressLRS TX* WiFi networks. The password is *expresslrs*.

Your web browser should load a configuration page automatically upon opening, similar to WiFi login pages at cafes. Once the page loads (10.0.0.1 is the address), set a binding phrase using the web interface on that page. This should be the first field on the page that loads, labeled *Binding Phrase*. Click the *SAVE & REBOOT* button towards the bottom of that page.

Binding Phrase  
i<3elrs

---

UID **Flashed** (Auto updated by changing the bind-phrase above)  
211,1,163,201,243,224

---

WiFi "auto on" interval (s)  
60

---

UART baud  
420000

---

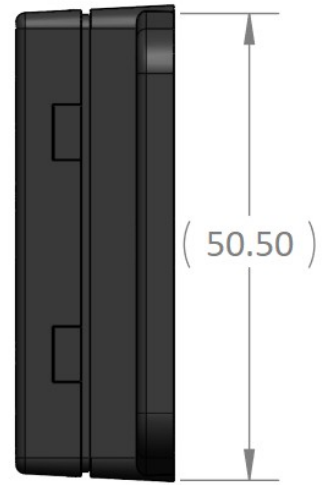
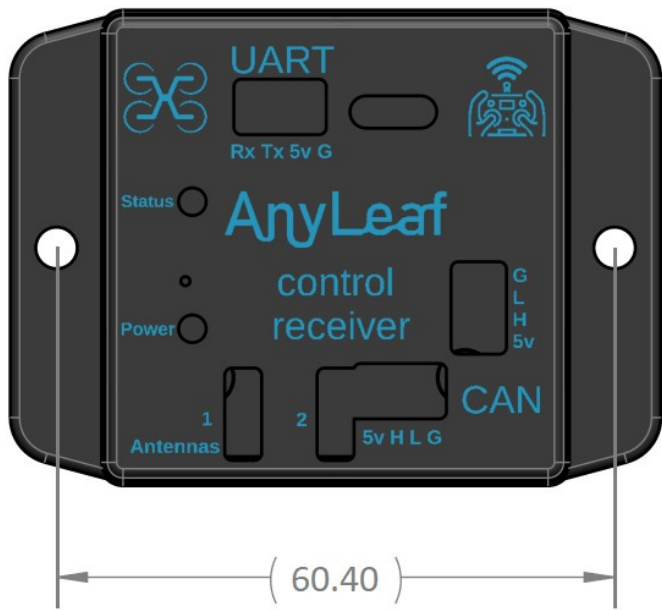
Invert TX pin  
 Lock on first connection

**SAVE & REBOOT**

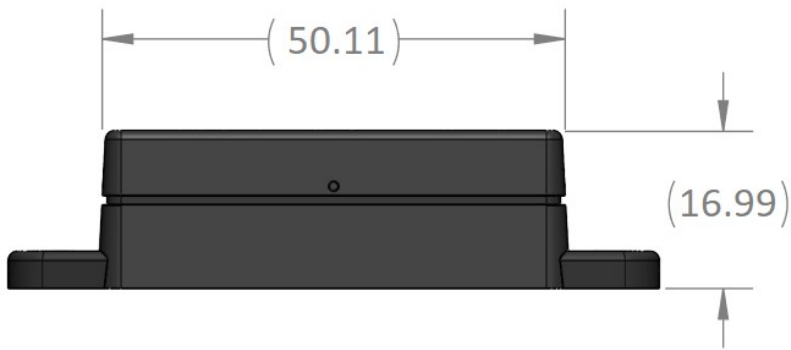
Note that on the single-radio Rx, you can tell it's in WiFi mode by the LED strobing rapidly. On the dual-radio Rx, the LED will pulse yellow and green to indicate it's in WiFi mode. In either case, this occurs after being powered on for 1 minute without binding to a transmitter.

Make sure the bind phrases set on the receiver and transmitter match. The bind phrase should be kept private.

## Dimensions



Dimensions are in mm



## Support

If you have any questions, or support requests, contact us by email: [anyleaf@anyleaf.org](mailto:anyleaf@anyleaf.org).

© 2023 AnyLeaf LLC