Anyleaf ExpressLRS Receiver (CAN)

December 2023

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 tranceiver:** 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/.

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)

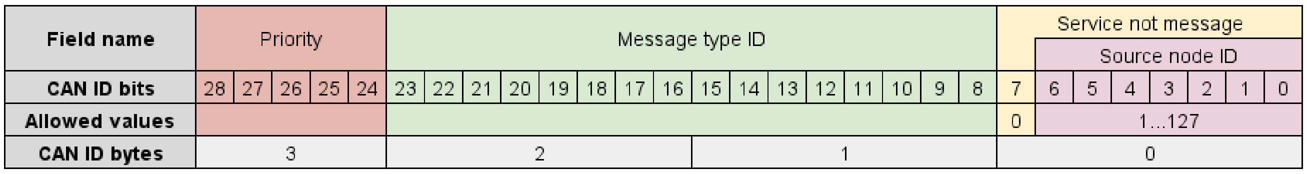
- Get/Set parameters (uavcan.protocol.param.GetSet)

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/

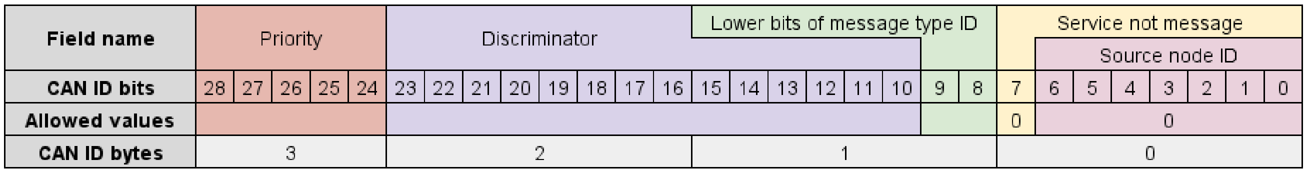
The CAN ID format is as follows. Most messages broadcast by this device, including all sensor readings, use the *Message frame* format.

Message frame



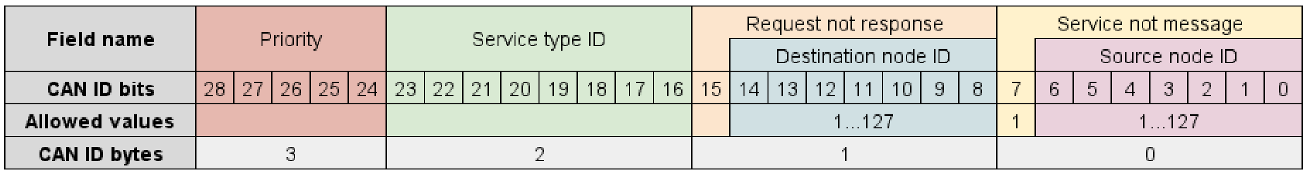
Field name	Priority					Message type ID																Service not message							
																							Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																						0	1...127						
CAN ID bytes	3					2								1				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

Anonymous message frame



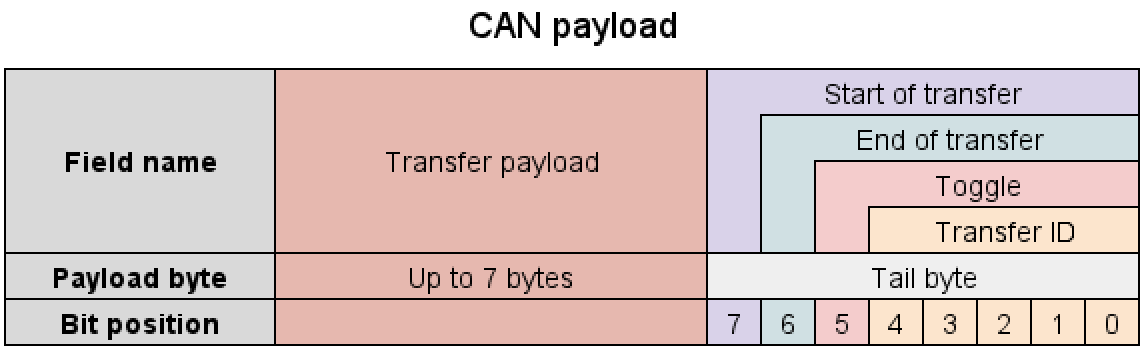
Field name	Priority					Discriminator								Lower bits of message type ID								Service not message							
																							Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																						0	0						
CAN ID bytes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

Service frame



Field name	Priority					Service type ID								Request not response								Service not message							
															Destination node ID								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:



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
0-15	Status	Enum: 0: OK; 1: Good signal; 2: Link lost
16-24	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
24-28	ID of the device; currently unused	4-bit unsigned integer
28+	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
0-31	Timestamp since node start in μs	32-bit unsigned integer
32-34	Node health	Enum: 0: OK; 1:Warning; 2: Error; 3: Critical.
35-38	Node mode	Enum: Operational; 1: Initialization;2:

		Maintenance; 3: Software update; 7: Offline

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

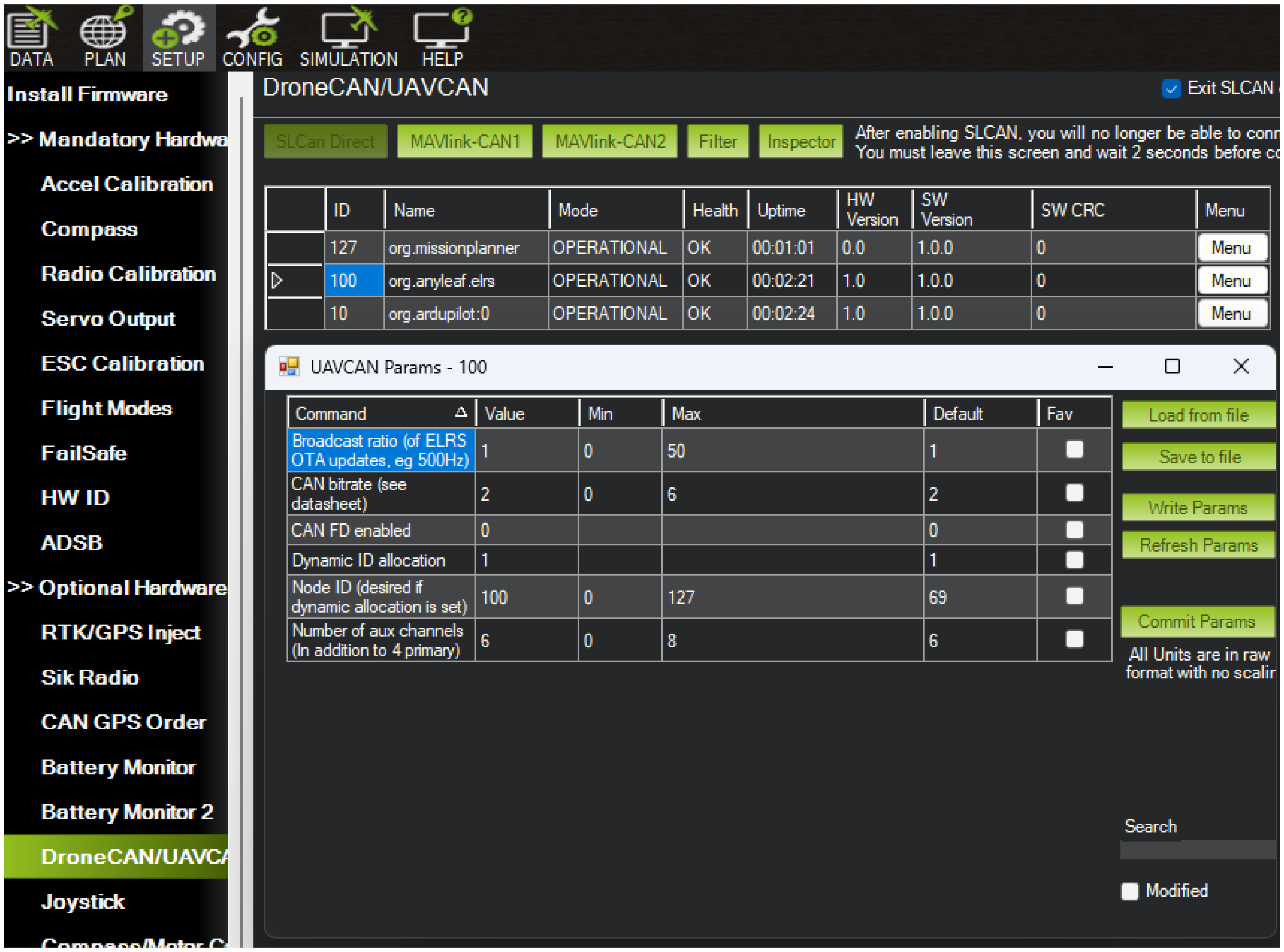
This sets, in conjunction with radio configuration, the rate control channel data and link stats messages are broadcast over CAN. The default of 1 broadcasts every message received. A value of 2 means every other message is broadcast. A value of 0 means no control data is broadcast. The baseline rate 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 note 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:



After enabling SLCAN, you will no longer be able to conn
You must leave this screen and wait 2 seconds before c

	ID	Name	Mode	Health	Uptime	HW Version	SW Version	SW CRC	Menu
	127	org.missionplanner	OPERATIONAL	OK	00:01:01	0.0	1.0.0	0	Menu
▷	100	org.anyleaf.elrs	OPERATIONAL	OK	00:02:21	1.0	1.0.0	0	Menu
	10	org.ardupilot:0	OPERATIONAL	OK	00:02:24	1.0	1.0.0	0	Menu

Command	Value	Min	Max	Default	Fav
Broadcast ratio (of ELRS OTA updates, eg 500Hz)	1	0	50	1	☐
CAN bitrate (see datasheet)	2	0	6	2	☐
CAN FD enabled	0			0	☐
Dynamic ID allocation	1			1	☐
Node ID (desired if dynamic allocation is set)	100	0	127	69	☐
Number of aux channels (In addition to 4 primary)	6	0	8	6	☐

All Units are in raw format with no scalir

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.

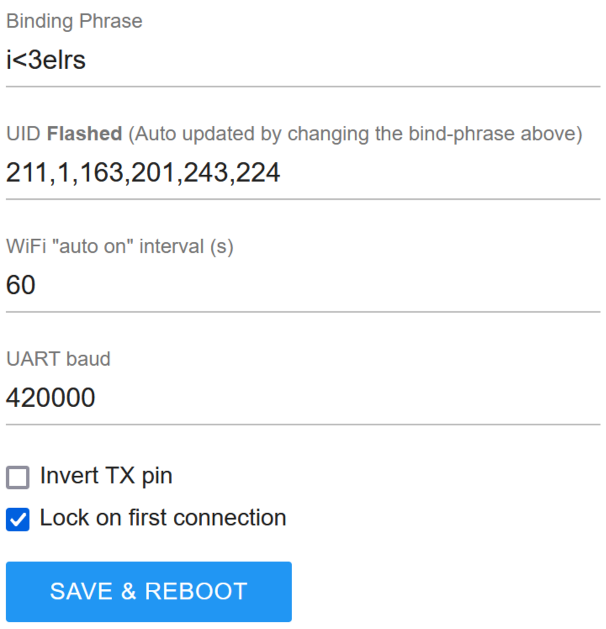
Binding to a transmitter

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

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.

Configuring Ardupilot CAN settings

There are several CAN settings in ArduPilot that may need to be enabled or modified to make this device work. These are accessed by selecting the *Config* button at the top left of Mission Planner, then selected *Full Parameter List* from the menu at the left. You can then use the search window at the right to find these settings. The following settings are the most relevant:

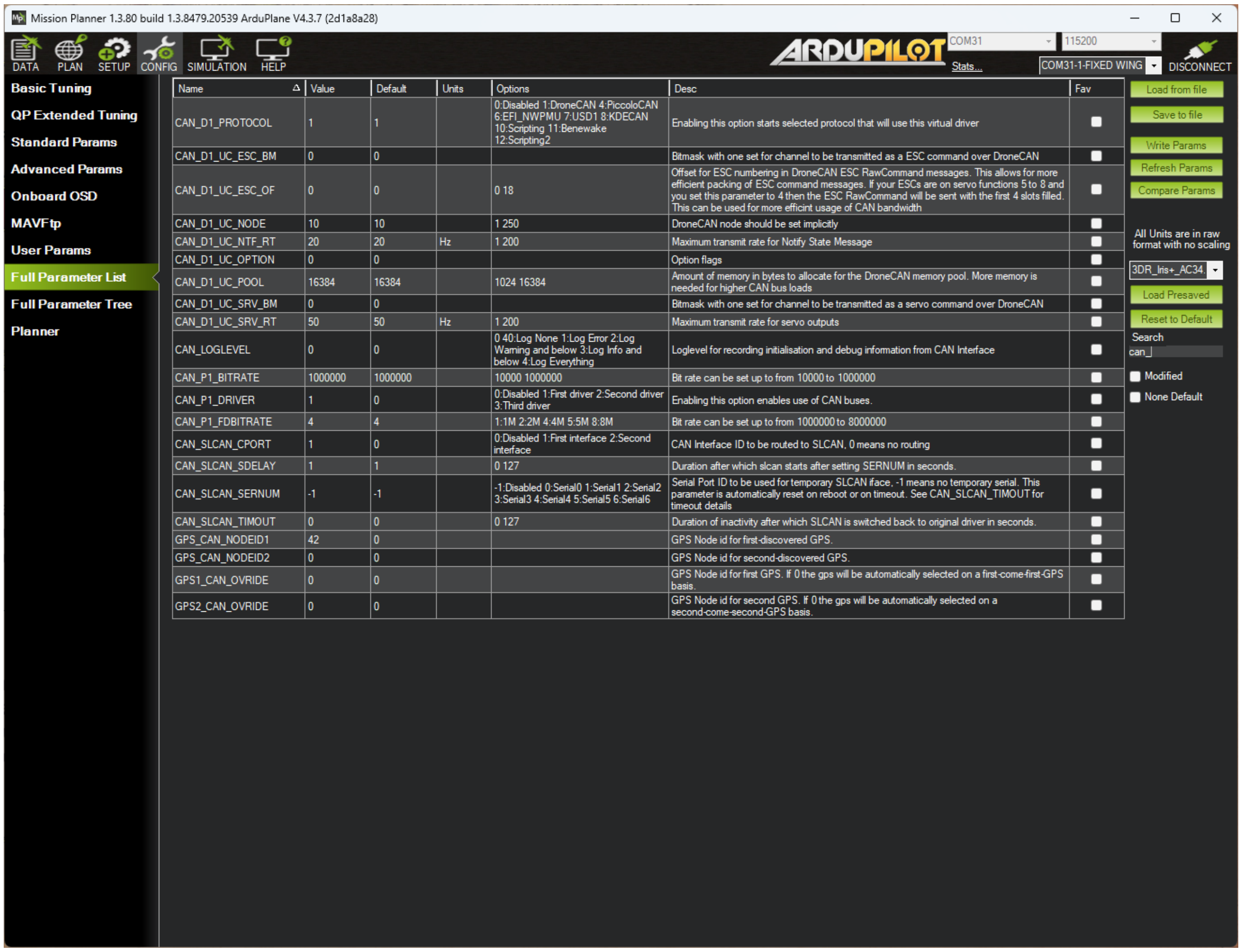
CAN_P1_DRIVER = 1 ("Enables use of CAN buses")

If using FD mode or a different bit rate from the default, adjust these settings as required:

CAN_D1_UC_OPTION: enable FD (Value of 4, if no other flags here are set.)
CAN_P1_FDBITRATE: enum for FD bit rate. 1 for 1Mb. Defaults to 4 for 4mb. Select 5 for this device's maximum of 5Mbps.
CAN_P1_BITRATE = 1000000 Bitrate if using classic (non-FD) mode.

If troubleshooting, confirm CAN_D1_PROTOCOL = 1 (This selects DroneCAN as the CAN protocol). This should be set by default. You can set CAN_SLCAN_CPORT = 1 to enable reading CAN messages from a PC, eg using the DroneCAN GUI software.

The below image shows most of the ArduPilot CAN settings:



Name	Value	Default	Units	Options	Desc	Fav
CAN_D1_PROTOCOL	1	1		0:Disabled 1:DroneCAN 4:PiccoloCAN 6:EFI_NWPMU 7:USD1 8:KDECAN 10:Scripting 11:Benewake 12:Scripting2	Enabling this option starts selected protocol that will use this virtual driver	☐
CAN_D1_UC_ESC_BM	0	0			Bitmask with one set for channel to be transmitted as a ESC command over DroneCAN	☐
CAN_D1_UC_ESC_OF	0	0		0 18	Offset for ESC numbering in DroneCAN ESC RawCommand messages. This allows for more efficient packing of ESC command messages. If your ESCs are on servo functions 5 to 8 and you set this parameter to 4 then the ESC RawCommand will be sent with the first 4 slots filled. This can be used for more efficint usage of CAN bandwidth	☐
CAN_D1_UC_NODE	10	10		1 250	DroneCAN node should be set implicitly	☐
CAN_D1_UC_NTF_RT	20	20	Hz	1 200	Maximum transmit rate for Notify State Message	☐
CAN_D1_UC_OPTION	0	0			Option flags	☐
CAN_D1_UC_POOL	16384	16384		1024 16384	Amount of memory in bytes to allocate for the DroneCAN memory pool. More memory is needed for higher CAN bus loads	☐
CAN_D1_UC_SRV_BM	0	0			Bitmask with one set for channel to be transmitted as a servo command over DroneCAN	☐
CAN_D1_UC_SRV_RT	50	50	Hz	1 200	Maximum transmit rate for servo outputs	☐
CAN_LOGLEVEL	0	0		0 40:Log None 1:Log Error 2:Log Waming and below 3:Log Info and below 4:Log Everything	Loglevel for recording initialisation and debug information from CAN Interface	☐
CAN_P1_BITRATE	1000000	1000000		10000 1000000	Bit rate can be set up to from 10000 to 1000000	☐
CAN_P1_DRIVER	1	0		0:Disabled 1:First driver 2:Second driver 3:Third driver	Enabling this option enables use of CAN buses.	☐
CAN_P1_FDBITRATE	4	4		1:1M 2:2M 4:4M 5:5M 8:8M	Bit rate can be set up to from 1000000 to 8000000	☐
CAN_SLCAN_CPORT	1	0		0:Disabled 1:First interface 2:Second interface	CAN Interface ID to be routed to SLCAN, 0 means no routing	☐
CAN_SLCAN_SDELAY	1	1		0 127	Duration after which slcan starts after setting SERNUM in seconds.	☐
CAN_SLCAN_SERNUM	-1	-1		-1:Disabled 0:Serial0 1:Serial1 2:Serial2 3:Serial3 4:Serial4 5:Serial5 6:Serial6	Serial Port ID to be used for temporary SLCAN iface, -1 means no temporary serial. This parameter is automatically reset on reboot or on timeout. See CAN_SLCAN_TIMOUT for timeout details	☐
CAN_SLCAN_TIMOUT	0	0		0 127	Duration of inactivity after which SLCAN is switched back to original driver in seconds.	☐
GPS_CAN_NODEID1	42	0			GPS Node id for first-discovered GPS.	☐
GPS_CAN_NODEID2	0	0			GPS Node id for second-discovered GPS.	☐
GPS1_CAN_OVRIDE	0	0			GPS Node id for first GPS. If 0 the gps will be automatically selected on a first-come-first-GPS basis.	☐
GPS2_CAN_OVRIDE	0	0			GPS Node id for second GPS. If 0 the gps will be automatically selected on a second-come-second-GPS basis.	☐

For more information, reference the ArduPilot CAN setup documentation:
https://ardupilot.org/copter/docs/common-canbus-setup-advanced.html

FD CAN and Classic CAN selection

ArduPilot, PX4, and this device all default to using classic CAN; this is a good choice for compatibility. If any nodes on a given bus do not support FD mode, classic is the only viable option for that bus. If all devices support FD mode, selecting it, with the maximum bitrate supported by all nodes on the bus is the best option. Enabling FD mode with a high bitrate may be required if there are many devices on the bus, or a device is sending high-bandwidth data. This device, if configured to send AHRS or fused position data at a high rate, may saturate a low-bitrate bus.

This device supports FD mode, with a datarate up to 5Mbps. If all other devices on the bus support this, this is the recommended setting. Important: This device and any devices on the bus it communicates with (notably the flight controller) must be configured the same way in regards to FD mode vice classic mode, and datarate.

Updating firmware

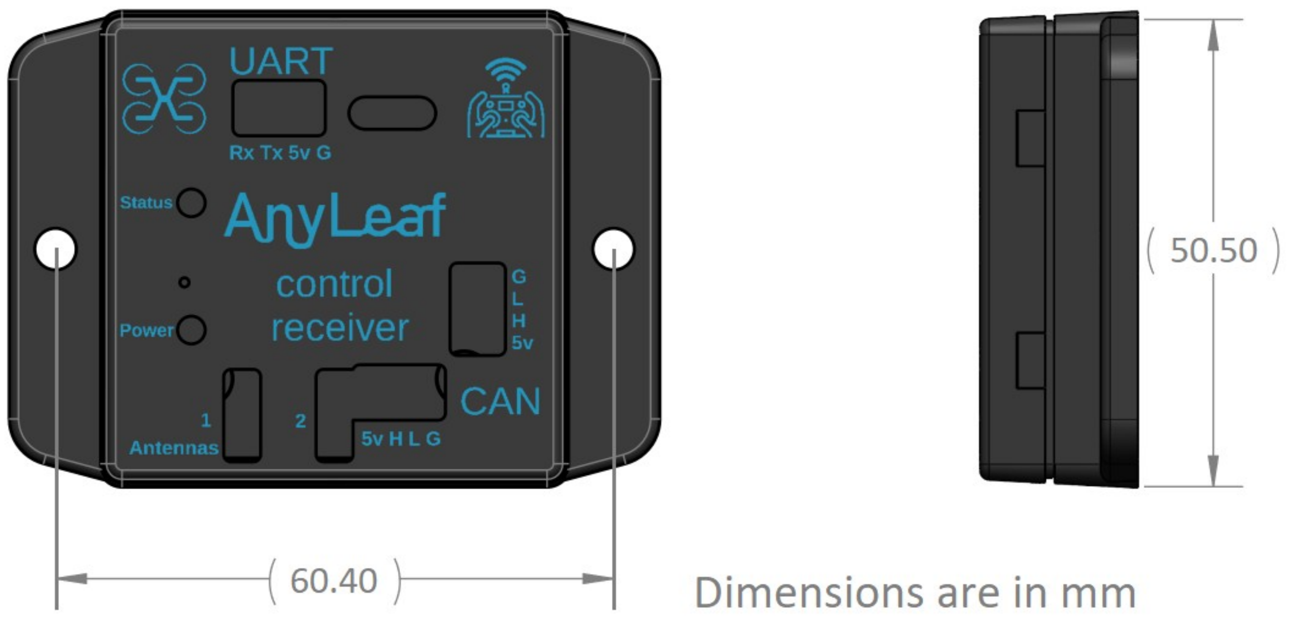
This device’s firmware can be updated over USB. To do so, download firmware from the AnyLeaf website. Open the device’s lid.

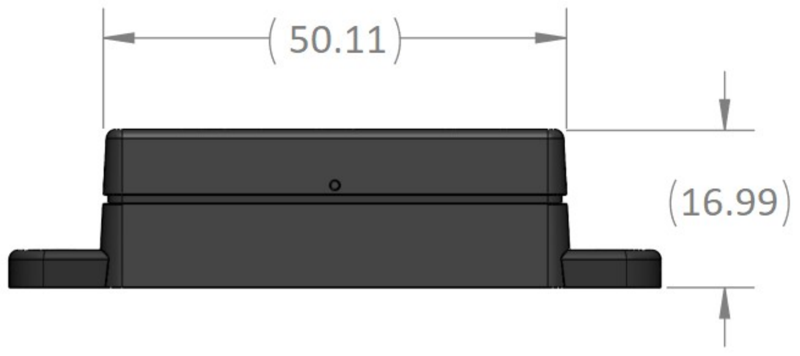
Mounting on smaller frames

If you wish to mount this device on frames that do not accommodate the plastic enclosure, you can remove the bare circuit board and mount it directly. To remove the circuit board, press gently on the front and back of the lid, while pulling it away from the base. Once removed, the PCB can be removed from the enclosure by removing the 4 Phillips screws securing it to the base.

The base circuit board can be mounted using its 28mm-spaced M3 mounting holes, or by applying an adhesive or securing mechanism to its bottom surface.

Dimensions





Support

If you have any questions, or support requests, contact us by email: anyleaf@anyleaf.org.

© 2023 AnyLeaf LLC