Documentation for OGNbase
==========

By Moshe Braner

This version last updated July 17 2024 to fit software version MB144

https://github.com/moshe-braner/Open-Glider-Network-Groundstation/tree/main/ognbase

HISTORY

FLARM, the traffic awareness and collision avoidance system which has been adopted by most glider (and many other aircraft) pilots in Europe, operates in the 866-928 MHz range (depending on what is legal in each region of the world - 470 MHz in China). It transmits in low power (20 milliwatts), in short bursts of a few milliseconds, once or twice per second. It was developed before the latest developments in hardware integration and modulation schemes, and is oriented towards collision avoidance, not longe-range communications.

The Open Glider Network (OGN) has encouraged the establishment of many ground stations, especially in Europe, that listen to signals from FLARM and other devices, and feed the data to servers on the internet. Despite FLARM's low power, these stations can receive signals from tens of km away.

The "Internet of Things" (IoT) community has prompted manufacturers to develop inexpensive low-power hardware that integrates significant computing power and communications capabilities including WiFi, Bluetooth, GNSS (such as GPS), and radios that operate in the unlicensed bands in the 400-1000 MHz range in a variety of modulation schemes.

Linar Yusupov (https://github.com/lyusupov/SoftRF) developed SoftRF, which (among other things) sends, receives and interprets signals to and from aircraft in the same protocol as used by FLARM, and some other protocols. SoftRF runs on several types of the IoT hardware devices.

Manuel Rösel developed the "OGNbase" software, based on SoftRF, that adds the ability to send the received traffic data to OGN via the WiFi capability of the hardware (and an available internet access point). OGNbase offers a simpler alternative to the Linux-based OGN ground stations. It also offers the ability to split the operation into two stations, a remote relay station that listens to the FLARM signals and relays them - by radio - to the other, "base" station which has access to the internet. The advantages and disadvantages of this "relay" system will be explained later in this document.

Moshe Braner (https://github.com/moshe-braner/Open-Glider-Network-Groundstation) further developed OGNbase to allow it to work in regions (USA, Canada and Australia) where there are many channels in the band, and FLARM performs "frequency hopping" between the channels, based on the exact time to a fraction of a second. And also modified OGNbase to interpret the new radio protocol in 2024.

Future development may include support for other radio protocols besides FLARM, e.g., ADS-L, OGNTP, P3I, FANET, even ADS-B. Some of those are already supported by SoftRF, but not by OGNbase. It may be possible for OGNbase to switch protocols on the fly, either to try and receive data from aircraft using different protocols, or to use a different protocol for relaying data than the protocol originally sent from the aircraft. E.g., using FANET protocol, which uses LORA

modulation, may make communication between the remote and base stations more reliable over longer distances.

HARDWARE AND SOFTWARE PLATFORMS

At this time OGNbase runs specifically on devices based on the ESP32 "system on a chip" CPU. Namely three devices from the LilyGo (TTGO) company, the "T3S3" or the LORA32 "Paxcounter" (without GNSS) or the "T-Beam" (with a GNSS module). These devices use so little power that the "remote relay" station can run on a solar panel about 20 cm (8 inches) across. They can go into "sleep mode" at night to conserve power, and automatically wake up the next day.

Software for the ESP32 is "embedded", i.e., runs on the bare hardware without an operating system, although it incorporates software libraries from the hardware manufacturers and from third parties. The software is developed on other systems (e.g., Linux or Windows) and then copied ("flashed") into the flash memory of the IoT devices, via a USB cable, Wifi, or an SD card. The size of this compiled firmware is about a megabyte, as compared with gigabytes for the "image" installed on a Raspberry Pi for a conventional OGN ground station (which includes the Linux operating system within the image).

The SoftRF and OGNbase software is completely open-source, unlike the conventional OGN ground station software which includes some closed-source components.

WHY (OR WHY NOT) USE OGNBASE

Besides the small embedded open-source aspect, OGNbase offers the "relay" capability. The ability to place the remote station on a hill where there is good reception, without the need for utility power and internet access, may facilitate better reception, especially in regions where locations that are not hilltops have limited reception due to trees and other obstructions. Most hilltops that do have power and internet lines are "noisy" in the sense that cellphone towers and other facilities are usually already installed there, and transmit strong local signals that make it difficult for an inexpensive receiver to decode the weak signals from far-away FLARMs.

On the other hand, it appears that the reception range of OGNbase is not quite as good as that of the SDR-based receivers. There are basic reasons for that, as the type of radio within the devices OGNbase runs on requires a more perfect signal. And data packets that fail the CRC check are rejected. Bit error correction, recently added to OGNbase, may increase its range somewhat. This feature uses the redundancy of having the additional 16 bit CRC in the radio packet to try and repair some bit errors in marginal reception. It seems to repair a significant percentage of the imperfect packets. In any case, any type of receiver should be equipped with as good an antenna as possible, in a location as high and open as possible, and with a pre-amplifier (LNA) and filter (SAW) between the antenna and the receiver.

HOW TO USE OGNBASE

Hardware choices

The LilyGo (TTGO) "T3S3" and "Paxcounter" do not have a GNSS module, but can be used by getting the time data from internet (NTP) servers. The recommended current product is the T3S3.

It is available with either the sx1276 or sx1262 radio module - the latter is theoretically a better receiver. Either the 915 or 868 MHz version will do, they are identical AFAIK other than the little antenna which will not be used anyway.
https://www.lilygo.cc/products/t3s3-v1-0?variant=42740446331061
https://www.lilygo.cc/products/t3s3-v1-0?variant=42586879721653
https://www.lilygo.cc/products/t3s3-v1-0?variant=42740446298293
https://www.lilygo.cc/products/t3s3-v1-0?variant=42586879688885

The LilyGo "T-Beam" includes a GNSS (GPS) module and can also be used for the SoftRF FLARM-compatible collision avoidance system. It also has an integrated holder for an 18650 li-ion battery. Be sure to get the model for the frequency used in your region, e.g., 915 MHz in North America and 868 MHz in Europe. Only the plain (older) "T-Beam" is supported by OGNbase at this time, not the "T-Beam Supreme".

Note: to turn off the T-Beam, press and hold the button closest to the USB jack until the dim red LED turns off.

Note: One needs to install different firmware binaries on the different hardware models.

Other hardware models could potentially be used, but have not been tested at this point. See the long list of models that SoftRF runs on: https://github.com/lyusupov/SoftRF #models

The remote station of a relay setup can run on solar power. The T-Beam needs an 18650 lithium ion battery, and includes a circuit for regulating the charging of the battery from a 5V source via the USB jack. The T3S3 has a jack for connecting an off-board lithium battery, and includes a charge controller chip. Alternatively, can power the board using a 5V USB "power bank", which in turn is charged from the solar panel, but then the OGNbase software cannot detect a medium-low-battery situation. In either case, a step-down module from the voltage of the solar panel to 5 volts should be placed between the solar panel and the USB charging jack. A solar charging current of only 100 mA or so during sunlight hours should suffice, especially if the software is set up to put the processor into sleep mode at night or when there is no glider traffic.

A better antenna is needed than the tiny whip that is included with these boards. See OGN guide pages for recommendations. For example: https://www.amazon.com/Signalplus-Omni-Directional-Antenna-824-960MHZ-Outdoor-Verizon/dp/B0917CPP84
For a standalone station, or the remote station of a relay setup, an omnidirectional antenna is needed, to be able to receive signals from gliders in any direction. For a base station with one remote station, a directional antenna pointed towards the remote station may work better than an omnidirectional once. Either way, the antenna is much more expensive than the circuit board! Now that the "Helium mining" craze is fading, there are used and closeout antennas on the market.

Installing firmware for the first time

The T-Beam often arrives from the factory with SoftRF installed. That allows installation of OGNbase, just like updated versions of SoftRF, via Wifi. See instructions here:
https://github.com/lyusupov/SoftRF/wiki/Firmware-update-%28Web-method%29 #esp32

Remember to use the correct OGNbase .bin file for the hardware, there are a T3S3 version, T-Beam version, and a "TTGO" version for the Paxcounter.

On the T3S3, Paxcounter, or a T-Beam without a working version of SoftRF or OGNbase already installed, OGNbase needs to be initially installed via the USB jack. For the T-Beam or paxcounter, follow the same instructions as for SoftRF, here: https://github.com/lyusupov/SoftRF/blob/master/software/firmware/binaries/README.md #esp32 - substituting the OGNbase .bin file for the SoftRF.ino.bin file. A ZIP file with the needed files and more detailed instructions is available here: https://github.com/moshe-braner/Open-Glider-Network-Groundstation/tree/main/ognbase/binaries
For the T3S3 need different files and flash addresses, see the file T3S3_flashing.zip in that same folder. The T3S3 does not have a USB-serial adapter chip like the older boards, instead the ESP32-S3 processor has a built-in USB CDC interface. Windows 10 knows to assign it a "COM port" that can then be selected within the flashing tool.

An alternative to the Windows-based ESP tool is a WebSerial-based version available here: https://esp.huhn.me/
Reportedly it will work with a Mac, for example.

After firmware installation OGNbase will start up waiting for upload of a configuration file. The next two sections describe the configuration file, and the web interface that allows uploading it and other files. Note that a later update of the OGNbase firmware normally does not require uploading the configuration file again. The configuration file, and other small files, remain intact in the small file-storage area (called "SPIFFS") reserved in the flash memory of the device. But, some firmware version updates require the config file and the index.html file to be updated too, in which case, if possible, should upload these files first and then do the firmware update.

Configuration file options

The configuration file is called config.json and is in the JSON format. Start with the config.json provided here:
https://github.com/moshe-braner/Open-Glider-Network-Groundstation/tree/main/ognbase/data
and change only what you need to. The file is plain text, you can edit it with any text editor. Be careful not to delete (or add) commas or quotes. Many settings are either on or off, which in this file are either stated as "true" or "false", or as "1" or "0". The file is divided into sections, such as "wifi", "coordinates", etc.

The "wifi" section has two subsections, one where you can list the names of the (one or more) wifi networks that you will want the device to recognize, and the other where you list the password for those networks. When turned on, the device will attempt to connect to the listed network(s). If none are in range, it will create its own network (after a couple of minutes). Here is an example of that section:

```
  "wifi":{
    "ssid":[
      "my_wifi_ssid",
      "xxxxxxx",
      "xxxxxxx",
      "xxxxxxx",
      "xxxxxxx"
    ],
    "pass":[
      "my_wifi_password",
      "xxxxxxx",
```

```
      "xxxxxxx",
      "xxxxxxx",
      "xxxxxxx"
    ],
    "timeout":12000
  },
```

The "coordinates" section states the location of the station, using lat/lon (in decimal degrees), altitude, and geoid separation (both in meters). You can look up the geoid separation for your location here: http://geographiclib.sourceforge.net/cgi-bin/GeoidEval Here is an example of this section:

```
  "coordinates":{
    "lat":43.78917,
    "lon":-72.35194,
    "alt":582,
    "geoidsep":-28,
    "mobile":false
  },
```

If "mobile" is set to true, then the position is not fixed, rather, the station can move, and the location will be continuously updated using GNSS. If you want to automatically set the station location when you turn OGNbase on, but not move it while operating, then leave "mobile" as false, and set both "lat" and "lon" to zero. Either way, for automatic coordinates, the hardware device used must include GNSS, and "gnsstime" must be set to 1 (see below in the relay section). Note that the web interface (see below) does NOT update the displayed lat/lon after they are adjusted based on GNSS, it keeps displaying the lat/lon that was in the configuration file.

The "radio" section includes settings regarding the reception of radio signals from aircraft. The "protocol" setting is optional, since it is always "legacy", meaning the radio message format used by FLARMs. OGNbase receives and reports both the new (March 2024) protocol and the older protocol. Here is an example of this section (skipping the "protocol"):

```
  "radio":{
    "band":1,
    "bec":true
  },
```

The "band" must be set to the right band for the region your station is in, otherwise it won't be listening on the right frequencies. The codes for the regions are:

```
 RF_BAND_EU   = 1,  /* 868.2 MHz band */
 RF_BAND_US   = 2,  /* 915 MHz band */
 RF_BAND_AU   = 3,  /* 921 MHz band */
 RF_BAND_NZ   = 4,  /* 869.250 MHz band */
 RF_BAND_RU   = 5,  /* 868.8 MHz band */
 RF_BAND_CN   = 6,  /* 470 MHz band */
 RF_BAND_UK   = 7,  /* 869.52 MHz band */
 RF_BAND_IN   = 8,  /* 866.0 MHz band */
 RF_BAND_IL   = 9,  /* 916.2 MHz band */
 RF_BAND_KR   = 10  /* 920.9 MHz band */
```

The example configuration above ("band":1) chose the European band. The example also enables use of Bit Error Correction (which is also enabled by default).

The "aprs" section is about the OGN server that OGNbase will send APRS data packets to, how it will identify itself, and some options on the messaging. Here is an example of this section:

```
  "aprs":{
    "callsign":"GileMtn",
    "server":"aprs.glidernet.org",
    "port":14580,
    "debug":false,
    "debugport":12000,
    "range":200
    "hiderelayed":true
  },
```

It states that the name of the station is "GileMtn" - **see here for guidelines on naming stations: http://wiki.glidernet.org/receiver-naming-convention**

The "range" is the maximum distance (in km) to be reported on - an aircraft farther than that from the station will not be reported. If "hiderelayed" is set to true then traffic air-relayed by SoftRF is disassociated from this station - reported with "relayed" as the station name instead. That is so as to not affect this station's range statistics. If set to false, then "relayed" will appear at the end of the otherwise-normal APRS message from this station.

The "sleep" section configures the deep sleep options. Here is an example of this section:

```
  "sleep":{
    "mode":1,
    "rxidle":60,
    "wakeuptimer":30,
    "morning":10,
    "evening":17
  },
```

If "mode" is set to 1, OGNbase will sometimes put the device into deep sleep. That is especially useful in a remote relay station running on a small solar panel, to save power. It will enter sleep mode after "rxidle" minutes of no traffic received, and will sleep for about "wakeuptimer" minutes before waking up again - 60 and 30 minutes in this example. The wakeup time may be automatically adjusted to fall on the beginning of an hour (or on the half-hour), to try and get both stations to wake up at the same time (if both are set to sleep mode 1). If the remote station sleeps while the base station stays awake, the base station will send status messages to OGN every few minutes, which will say zero battery voltage and "time_not_synched".

If it is late in the day when it goes to sleep ("evening"), or if the battery voltage is low (<3.65), it will sleep until "morning", instead of "wakeuptimer" minutes. (If battery is low and it is late in the day it will sleep later into the next morning.) Note that as long as there is traffic being received, the evening sleep will be postponed (unless battery is low). And if time-relay is selected, the rxidle timer does not advance until the (first) time-synch has happened. In this example "morning" is 10:00 in the morning, standard time (not daylight savings time), and "evening" is at 5pm (17:00). These times are local - approximately - they are simply computed based on the longitude of the station. Adjust these times to fit your actual needs.

If "rxidle" is set to 0, the station stays awake all day (unless the battery charge is low), even if there is no traffic, but (if "mode" is > 0) sleeps at night, from "evening" until "morning". If "wakeuptimer" is set to 0, then if and when the station goes to sleep, it always sleeps until the next "morning". E.g., if rxidle=180 and wakeuptimer=0, the station will listen for traffic for 3 hours starting from "morning". If no traffic, or once 3 hours have gone by with no further traffic, it will then sleep until the next "morning".

In a base station only, and only when it is sending the time to, or receiving the time from, the remote station (see relay mode section below), if sleep "mode" is set to 2 ("follow remote"), the base station will go to sleep a few minutes after the remote station sleeps, and wake up at about the same time as the remote station. If the base station does sleep, it will not be sending status messages to OGN. The last few status messages before sleeping will display the length of time the remote station is planning to sleep, and then the sleep intention of the base station.

The "private" section allows selecting a port other than 80 for the web server to listen on. This can allow remote access to the web interface via http://externalipaddress:portnumber. (Need to also set up, in your router settings, public access through this port to the (static) IP address of the device.) Choose a unique port number between 49152 and 65535. For example:

```
  "private":{
    "webport":65432
  },
```

The "ognrelay" section selects the operation mode of OGNbase. See a later part of this document about relay modes. The following example turns off all the relay settings, i.e., it selects single-station operation, without using GNSS (GPS), i.e., gets time from NTP servers on the internet:

```
  "ognrelay":{
                "enable":0,
                "basestation":0,
                "relaytime":0,
                "reversetime":0,
                "gnsstime":0,
                "relaykey":54321
  },
```

Some of the other sections in the configuration file are about things I do not understand, and I hope that Manuel will document. E.g., remote logging and debugging. You can leave all of those things disabled and just have OGNbase report directly to OGN.

The Web Interface

After startup, OGNbase tries to connect to a Wifi network, and offers its web interface at an IP address that is chosen by the hosting network. That IP address will be shown on the OLED display of the device, and you can also see it in the DHCP client list of the hosting network.

If no configuration file (naming the wifi network) has been loaded, or if the named network(s) is (are) out of range, OGNbase has no choice but to set up its own network, with the access password 987654321. Note that it may take a couple of minutes before that happens. OGNbase then offers its web interface at the IP address 192.168.1.1.

Either way, a web browser (on a computer, tablet, or smartphone connected to the same network), if pointed to the OGNbase IP address, will connect to the mini web server built into OGNbase, allowing one to view and change settings, upload a configuration file, or even update the firmware.

The web interface shows the OGNbase status page, or its file-upload page, or its statistics page, etc - see sections below.

Uploading files

Before any configuration files are uploaded to the device, the web interface automatically goes to the file-upload page. On that page, the files currently stored are listed. Click the "browse" button on the left, select a file in the browser's file-selection window, click "open" (or whatever that window wants), then click "upload" in the OGNbase file-upload page. If you upload a file it will silently overwrite an already stored file by the same name if it exists. The file will be uploaded immediately, and the status (or upload) page will be displayed again.

The one file you MUST upload is the configuration file, which must be named config.json - it was described in a section above. When you upload it, if a previous version exists within the device, it will be renamed "oldconf.json". The upload page includes a link to revert to the older version if necessary. If the uploaded config.json is not readable, the next boot will automatically use the saved older version. If the newly uploaded file is readable but of the wrong version, it will be used anyway, to try and maintain the wifi connection settings, but you will see a message saying it is the wrong version.

Optionally you can also upload the files "index.html" and "style.css". Once you do, the next time you start up OGNbase its web server will display the OGNbase status page instead of the file-upload page. If the index.html file is of a wrong version, it will be deleted automatically on the next boot.

To upload files after that (for example, if you want to change the configuration), there are two ways:
(1) in the Web interface status page, click the "upload" button, that will take you to the file-upload page.
(2) on the Paxcounter hardware, which has an SD card slot, can put all the files (optionally even a firmware update file, renamed "ognbase.bin", within the "firmware" folder) on the SD card, and upon startup OGNbase will read them from the SD card. See here: https://github.com/moshe-braner/Open-Glider-Network-Groundstation/tree/main/ognbase/sdcard

Other features of the web interface

The status page shows the OGNbase settings in effect, and a few statistics on its operation. Near the top the OGNbase version and the operational mode will be displayed. If there was a problem reading or interpreting the configuration file, a note about that will appear in the "Mode:" line. The buttons and links at the bottom allow resetting the local or remote device, or reaching the statistics page, the file upload page, the firmware update page. The "clear" button removes all the files stored in the device - there is normally no need to do that, since uploading a file overwrites the previous version. But if you accidentally uploaded a wrong file, perhaps a large one that fills the space, this button may be useful. Also a future version of the firmware may be incompatible with the existing uploaded files, so it may be a good idea to clear the files before updating the firmware. (See later section about firmware updates.)

The OGNbase settings visible in the status page can be changed there, and the "save config" button can save them to the stored configuration file within the device and then immediately reboot. But it is best to edit the configuration file outside of the device and then upload it. If you want to download the current configuration file stored on the device, for viewing or editing, click the "Download Config" button on the status page.

If in relay mode, the "Remote Reboot" button sends a radio packet from the base station that tells the remote station to reboot. The reboot only happens under some conditions: The remote station has been running for at least 10 minutes and has already turned off its WiFi server, and the packet arrived ungarbled and passed a security check based on the secret "relay key". Not every click of the button actually sends such a packet. It is safe to press this button a few times, just in case, since the remote station will not reboot again for 10 minutes. Every time this button - or the "Refresh" button - is clicked, the statistics at the top of the status page refresh, if in time-relay mode the number of (remote) GNSS satellites received are displayed, which shows whether the remote station has finished rebooting and establishing communications.

The statistics page, reachable via a button at the bottom of the status page, shows additional statistics, especially relevant to relay mode. These include the number of unique aircraft seen (today or ever), and the number of radio packets currently received per minute. The statistics get refreshed when following that link, when the statistics page is refreshed in your browser, or automatically every 23 seconds. To get back to the status page, can use your browser's "back" button.

Note that the base station can only report statistics from the remote station if time-relay is happening, since the statistics are sent from the remote station as part of the time-relay packets. And the "today" and "ever" aircraft counts (and packet count) are reset if the device "sleeps", thus "ever" may actually be "today", or only part of today. Check the "uptime" statistic.

The web interface can be set up to be accessible remotely. Choose a "webport" in the configuration file, and tell your router to direct http requests on that port to the OGNbase device. (You ISP may or may not block such requests.)

OLED display

The small display that is part of the T3S3 and Paxcounter hardware, and can optionally be soldered to the T-Beam hardware, shows some information about OGNbase operation. It cycles between 3 or 4 pages. And sometimes clears the display and shows just one line for a short time. It shows the number of radio packets received, the wifi network connected to and the IP address, the OGNbase operational mode, the GNSS satellite reception status, current time, station location, and so forth. The OLED is turned off after 10 minutes of operation. Note: on the T-Beam now now has the choice of attaching the OLED display to pins 21,22 or to pins 2,13, the software automatically detects it.

The OLED display turns off after about 15 minutes - the time can be set via the "oled" "disable" setting in the config file. To turn it back on, can reboot the device, or, on the T-Beam, click the middle button.

On the T3S3 and paxcounter boards there is also a bright green LED (besides the blue one which is always lit if external power is connected). In single-station mode OGNbase will turn on the green LED once NTP (or GNSS) time is available and a connection to the OGN servers is established. In

two-station mode, in the base station, the green LED will turn on once the two stations' time is synchronized, and a connection to the OGN servers is established. In other words, a glance at the green LED will confirm that the station operation is normal. In the remote station in a 2-station system the green LED lights up when time-sync is achieved, and stays on while WiFi is still active. In normal operation a remote station will be out of range of any wifi network, and will turn off its own wifi network - and the green LED - after 10 minutes, to save power.

Serial output

The USB port of the device can be connected to a computer, and by running a suitable terminal program (e.g., "Putty" or "Termite") on the computer one can see additional messages from OGNbase. That is mostly useful for debugging. The messages OGNbase sends to the OGN APRS server are echoed there too.

Updating the OGNbase firmware

There are 3 ways to "flash" a new (or old) version of the OGNbase firmware, replacing the existing version on the device. Remember to use the correct OGNbase .bin file for the hardware, there are a T3S3 version, a T-Beam version, and a "TTGO" version for the Paxcounter.

It may be necessary to replace the configuration and/or index.html files at the same time as updating the firmware, since new versions of the firmware may expect different or additional settings. The best approach is to upload the new files first, and then update the firmware, all within the same session in the web interface, without rebooting until after the firmware update. One can optionally clear the stored files via the "clear" button in the status page first. But if you reboot with the files cleared, you will need to connect to the wifi network generated by the device in order to upload the new files, a bit annoying. The worst case is if the old files are there, it is possible that the new version of OGNbase firmware will not be able to fully start up the web interface, and one must then re-flash the firmware via the USB port. OGNbase tries to avoid that situation by checking this line in the index.html file: meta name="OGNbase-Version" content="MB104" - if the version is wrong it ignores the index.html file and loads the upload-files page. Similarly, OGNbase checks for the same version number near the top of the config.json file. (If that version number is wrong, it will still try and connect to the wifi network specified in it, to preserve remote access.) That is why it is best to upload the files first before updating the firmware.

"On The Air" (OTA) firmware update is done via WiFi, whether within the network the device connects to as directed by the existing confifuration, or via the hotspot it generates. In a web browser, click the "update" button in the status page and then select the .bin file to upload, similar to the uploading of configuration files. Once you click "update" it takes about a minute, do not disturb it. After that, you may need to press the reset button on the device (or power it off and on again).

On the T3S3 or Paxcounter hardware it is also possible to update the firmware (and/or configuration files) via an SD card. Format the SD card as FAT or FAT32, and put the files on it as seen here: https://github.com/moshe-braner/Open-Glider-Network-Groundstation/tree/main/ognbase/sdcard - all in the root folder, except for the "ognbase.bin" firmware file which should be in the "firmware" folder. Insert the SD card into the SD card slot on the Paxcounter and power it on. The OLED display will show what it is doing. Upon successful firmware update the ognbase.bin file is deleted from the SD card.

Finally, firmware update via the USB port is always possible, see the section above about "Installing firmware for the first time".

Relay operation

There are several operational modes for OGNbase. The relevant settings are in the "ognrelay" section of the config.json:

```
"enable":1        makes this device the remote station in a relay
"basestation":1   makes this device the base station in a relay (not single-station mode)
"relaytime":1     turns on the relaying of the current time from the remote to the base
"reversetime":1   turns on the relaying of the current time from the base to the remote
"gnsstime":1      says that GNSS in this device should be used to get the current time
```

The "relaykey" is an arbitrary number, that is used to validate relay messages, to ensure the data came from the intended remote station, and was not corrupted. It must be set the same for the remote and base station, and is a secret only known to those stations. It is never broadcast.

Here are examples of the "ognrelay" section of the config.json for the different modes:

Single station without GNSS (gets time from NTP servers on the internet):

```
"ognrelay":{
                "enable":0,
                "basestation":0,
                "relaytime":0,
                "reversetime":0,
                "gnsstime":0,
                "relaykey":54321
},
```

Single station with GNSS as the source of the current time:

```
"ognrelay":{
                "enable":0,
                "basestation":0,
                "relaytime":0,
                "reversetime":0,
                "gnsstime":1,
                "relaykey":54321
},
```

Remote relay station without GNSS, i.e., without a source for the current time - only works if the base station sends the (NTP) time to the remote station:

```
"ognrelay":{
                "enable":1,
                "basestation":0,
                "relaytime":0,
                "reversetime":1,
                "gnsstime":0,
```

```
        "relaykey":54321
  },
```

Base station without without GNSS, gets time from NTP, and sends it to the remote station too:

```
  "ognrelay":{
        "enable":0,
        "basestation":1,
        "relaytime":0,
        "reversetime":1,
        "gnsstime":0,
        "relaykey":54321
  },
```

Remote station with GNSS and time-relay - it sends the (GNSS) time to the base station:

```
  "ognrelay":{
        "enable":1,
        "basestation":0,
        "relaytime":1,
        "reversetime":0,
        "gnsstime":1,
        "relaykey":54321
  },
```

Base station with time-relay from the remote station - does not use GNSS directly, nor NTP:

```
  "ognrelay":{
        "enable":0,
        "basestation":1,
        "relaytime":1,
        "reversetime":0,
        "gnsstime":0,
        "relaykey":54321
  },
```

Remember to set the "relaykey" to the same unique number in both stations of your relay system - do not use the example number. Choosing some "secret" number, like a combination lock, provides security against signals from other systems. Also make sure the same version of the software is installed on both stations. While most software versions will still communicate correctly with a somewhat older version on the other station, that is not guaranteed. In particular, Version MB144 made significant changes in how the stations communicate, and will not work with pre-MB144 versions on the other station.

DETAILS OF OPERATION

Basic ground station

OGNbase receives FLARM signals as they arrive, and keeps a list of aircraft already heard from and waiting to be reported. It only reports to OGN once every 5 seconds. Since aircraft usually

report more often than that, and to avoid sending too many reports to OGN at once, once an aircraft is reported, it is not reported again for some length of time - as many seconds as there are aircraft are being tracked (up to 30).

The following aircraft are NOT reported to OGN:
* those who have the no-track flag set
* those who have been last heard from more than 60 seconds ago ("expired")
* those who are farther away from the station than the "range" setting
* FLARM packets that look like GNSS noise or corrupted transmission - location shifted too far too fast

Why GNSS may (or may not) be necessary

In the regions (USA, Canada and Australia) where there are many channels in the band, FLARM performs "frequency hopping" between the channels, based on the exact time to a fraction of a second. In Europe there are only two channels but choosing between them is also based on the exact time.

OGN uses a standard frequency hopping scheme for OGN trackers, which is similar to FLARM's, but at any given moment is on a different frequency from FLARM. OGNbase uses the OGN frequencies for relayed traffic data.

The frequency hopping used by FLARM is based on the current time to the second. In addition to that, two frequency "slots" are used, in 400-ms time windows within the second. Thus the moment when the second starts needs to be known to within 100 milliseconds or better. The timing of radio packet transmission within these time windows is randomized to minimize interference.

One way to get the exact time is via a GNSS (GPS) module. OGNbase allows the remote relay station to send the time data to the base station, so that the latter does not need GNSS reception - which may be difficult to arrange in the same indoor location where internet access is available.

Alternatively, more recently, the problem of how to get sub-second timing from NTP servers on the ESP32 has been solved. It is even possible to send the NTP time from the base station to the remote relay station, instead of it using GNSS. Thus the simpler and cheaper T3S3 board can now be used in both the base and remote stations of OGNbase.

Relay operation

The remote station in a relay operation receives FLARM signals as they arrive, and keeps a list of aircraft already heard from and waiting to be relayed to the base station. The relaying is via radio messages in the same band as FLARM, and is subject to the same limitations on band usage. Therefore it only relays the position of one or two aircraft each second. Since many aircraft may be within range, once an aircraft is reported, it is not reported again for some length of time - as many seconds as there are aircraft are being tracked (up to 30).

When the remote station is ready to relay the position of one aircraft, it chooses the one that has been waiting the longest since it was last reported. Since at most only 30 aircraft are tracked, each will be relayed at least once every 30 seconds.

To avoid competing with FLARM for the radio spectrum, OGNbase tries to relay messages on a different frequency than FLARM transmits on. That is of course not possible in regions where only one channel is available. In regions with 2 channels (Europe), it uses the "other" channel, the one not used by FLARM at this moment. That's if the exact time is known, since it is used to choose the channel. Otherwise (without exact time) it only listens to FLARMs on channel 0 (where it should be able to hear half of the FLARM packets), and only relays on channel 1. In regions with more than 2 channels (US, AU) the exact time is required, and it selects the relay channel using the OGNTP protocol, i.e., it uses the same frequency that OGN tracker devices would use, which is always different from that used by FLARM. Since such trackers are uncommon (except in certain international contests) that should not be a problem. And in any case an OGNbase remote station only transmits as many radio packets, in total, as a *single* FLARM or OGN tracker device. Each aircraft's position is relayed much less often than it is transmitted by FLARMs.

Ideally the report to OGN will include the exact original time when the aircraft sent this position. FLARM packets do not include a timestamp, they are supposed to be sent in "real time". The packets relayed by OGN may be delayed by several seconds, or up to 30 seconds if there are many other aircraft reporting in the area. Therefore OGNbase inserts into the relayed packet the time the packet was originally received by the remote station. The base station uses that timestamp for reporting to OGN. This is only possible when the remote station has the exact time data. If it does not, the remote station only sends one bit, signaling whether the packet was relayed more than 12 seconds after it was received. The base station then computes an approximate timestamp: the time it received the relayed packet minus 4 seconds, or minus 16 seconds if the packet was marked as delayed.

The relayed radio packet is converted to the old (pre-2024) FLARM protocol, and 4 bits in the first (un-encrypted) 32 bit word are changed from 0010 (or 0000) in the new (or old) protocol FLARM taffic reports to 1101 (0xD) or 1011 (0xB) to mark it as a relayed packet, with and without a timestamp, respectively. The base station in such a relay operation ignores all radio packets that do not have that mark.

The choice of different frequencies from the FLARM frequencies, along with a different protocol, and those mark bits, should keep FLARMs from receiving the relayed packets and mistaking them for real traffic. (It is real traffic, but shifted somewhat in time.) Additionally, the encrytion key used by FLARM depends on the current time. If the remote station has the exact time data, OGNbase de-crypts the packet, and later re-encrypts the packet for relay transmission, using a different key, based on the "relay key" from the configuration. If the remote station does not have GNSS time, and is not receiving the NTP time from the base station, then the remote station cannot de-crypt the original FLARM packet. It then relays the packet as-is with the original encryption, but on the non-FLARM frequency, and with some bits set in the first word to mark it as something other than a normal traffic packet. The packet contents will also look like garbage to a FLARM (or standard OGN ground station) that tries to decrypt it, since the first word is used as part of the encryption key in the FLARM protocol. Perhaps in the future OGNbase will use a different message protocol, e.g., OGNTP protocol, or even FANET, for the relayed messages.

Time relay

It is easy to set up the remote station with GNSS reception since it is located somewhere in the open. The base station needs to have access to the internet via WiFi, and that usually means a location inside a building, where GNSS reception is not reliable - but it can use NTP time data.

OGNbase includes the option to send the exact time from the remote station with GNSS to the base station without GNSS. The data sent includes the current time to the second, and the milliseconds that have passed since the beginning of the current second. There is a small delay (about 10 milliseconds) due to the process of sending and receiving the time-data packets, but the receiving device simply adds a small fixed amount to the received time, and that is accurate enough for the purpose.

Alternatively, OGNbase in a 2-station setup can now send the time data in "reverse", from the base station to the remote station. This allows the base station (with internet access) to get the time from NTP, and send it to the remote station, which then does not need GNSS. Both stations can now use simpler hardware such as the T3S3.

When the stations start up, one station does not have the time data yet, and thus cannot use the usual frequency hopping. Therefore, both stations communicate at first on channel 0. Since these transmissions are only once in several seconds, an only until communications are established, it does not crowd that channel.

The time-data packets include a data check based on the sent time hashed with the "relay key". That ensures that the data was received correctly - and from the intended source station.

The station sending time data does not assume that the other station has received the time data until it replies with an acknowledgment packet - and that packet too has a data integrity check.

Once the time-receiving station gets the time from the time-source station, it can keep its own clock running accurately enough for a while without further input. The time-source station tries to send new time data packets every 10 seconds. Some will not be sent or not be received. If it hears no acknowledgment from the time-receiving station for 185 seconds, it returns to the original state of communicating on channel 0. And so does the time-receiving station if it receives no new time data for that long.

Time packets are distinguished from traffic reports by a specific "aircraft ID", and by a few other bits in the first (unencrypted) word of the FLARM-protocol radio packet set to a specific value, different from FLARM (old protocol) traffic packets (where those bits are all zeros) and from relayed traffic packets (where they hold one of two other values, as mentioned above). Time packets are also validated using a hash based on the "relay key" known to both the remote and base stations.

Be patient, it takes about a minute for the time-relay to be established. And if one station is reset but not the other, it will take about 4 minutes for the latter to decide that time sync has been lost, and re-establish it. During that time traffic packets are not relayed. Once time-sync is achieved, the time-receiving station will display the correct UTC time on the OLED display, and the base (if *not* a T-Beam) station will turn on its green LED (if it has also established communications with the OGN servers).

Status messages sent to OGN

You can monitor the operation of your station remotely via the internet. Every few minutes the base station sends a "status" message to the OGN servers. These messages are passed to clients that request them. For example, if you use gliderradar.com, select your station, click on "raw packets", and you can see the recent status (and position) messages from your station. You can bookmark the final URL.

The status messages from OGNbase include the following information: For example:
K2B9>OGNSXR,TCPIP*,qAC,GLIDERN2:>235055h vMB103-ESP32-OGNbase 3.7V 0/min 0/0Acfts[1h] 120_m_r_uptime
Where the fields are:

K2B9	name of the ground station reporting
OGNSXR	identifies the source as OGNbase
GLIDERN2	name of OGN server handling this message
235055h	time stamp
vMB123-ESP32-OGNbase	software version and type of hardware (may say "sx1262" or "2-station")
3.7V	battery voltage (in remote relay station if used)
0/min	traffic packets received in the last minute
0/0Acfts[1h]	number of unique aircraft heard from in the last hour
120_m_r_uptime	the remote relay station has been "up" for (approximately) 120 minutes

Data fields may also include:

6_m_r_uptime	the remote relay station has been "up" for 6 minutes
36_d_r_uptime	the remote relay station has been "up" for 36 days
331_m_uptime	the base station has been "up" for 331 minutes
8sat	number of GNSS satellites received (in remote relay station using GNSS)
time_synched	the base station and remote station are exchanging clock time data
time_not_synched	the base station and remote station lost time synch, will try again
1325_m_sleep	the base station will soon go to sleep for 1325 minutes
1325_m_r_sleep	the remote relay station will soon go to sleep for 1325 minutes

Radio packets relayed by SoftRF

SoftRF (from version MB09x) optionally relays radio packets from landed (or far or low) aircraft. To avoid recursive relaying, it marks the packet by changing the address (ID) type, to values not otherwise used by FLARM. OGNbase restores the original address types when it reports the relayed traffic to OGN, but optionally changes the reporting station name to "relayed" so as not to affect the station range stats.

Noise measurement

Version MB139 introduced measurement of the background radio noise (whether random or unwanted signals), for the sx1276 and sx1262 radio types. This serves two purposes: The SNR reported to OGN used to be the RSSI minus a constant. Now it is the RSSI of the signal minus the RSSI of the background noise sampled several times in the preceding 30 milliseconds or so. E.g., if you add an LNA between the antenna and the device OGNbase runs on, and it makes the signal stronger, but also makes the noise floor higher, the reported SNR is not biased. The sampled noise level can also be collected over time for comparison between different antenna locations, LNA vs. no-LNA, etc. A sample of the RSSI (dBm) is read every 7 ms. Up to 4096 samples are collected in each channel - with 65 channels in North America, it takes a while to reach that maximum. To turn on the data collection, add this line to the "radio" section of config.json:

```
  "noise":2
```

or

```
  "noise":1
```

The value 2 means: collect noise samples at all times, but only if the receiver was already set up previously, and no packet has arrived meanwhile. The value 1 means: collect noise samples only between 240 and 390 ms after PPS - that is, during a period when valid radio packets cannot arrive (after time slot 1 ended, and before time slot 0 starts). Use the value 0 (or omit that line) to not

collect noise data. To extract the data, access the URL "base_url/noise" within the web interface (where "base_url" is the URL you use to access the status page). Then you can "save page as". When the /noise page is accessed, the data is also reset to all zeros and new data is collected afterwards. The data is plain text, comma separated, with the fields: channel number (0-64), count of samples within that channel, and the sum of the RSSI readings in that channel. Divide the sum by the count to get the average reading, e.g., with 451 samples and a sum of -49429 that is -109.6 dBm.

From version MB143 OGNbase also sends background noise sampling data collected in the remote station to the base station where it can be extracted via the /remote_noise page in the web interface. This data transfer only happens when there is traffic data being relayed from the remote to the base station. And the data is more sparse, since only one sample is sent per relayed taffic packet. Best to extract it at the end of a busy day at the airport. For longer term average can collect multiple such samples and combine them.

Compiling OGNbase

Until recently I compiled it under Ubuntu 20.04 using the Arduino IDE verion 1.8.16, and the ESP32 board support version 2.0.3. Bugs in the ESP32-S3 support in 2.0.3 forced me to shift to the ESP32 board support version 2.0.17, using the Arduino IDE verion 1.8.19 running on Windows 10. The file build_opt.h must include either -DT3S3 or -DTTGO or -DTBEAM. Disable PSRAM when compiling for TTGO. For the T3S3 use these settings:
Board: ESP32S3 dev module
USB method: hardware CDC
USB CDC on boot: enabled
MSC and DFU: disabled
CPU: 80 MHz
Flash: QIO 80 MHz
Flash size: 4MB
Partition: minimal SPIFFS with OTA
PSRAM: QSPI