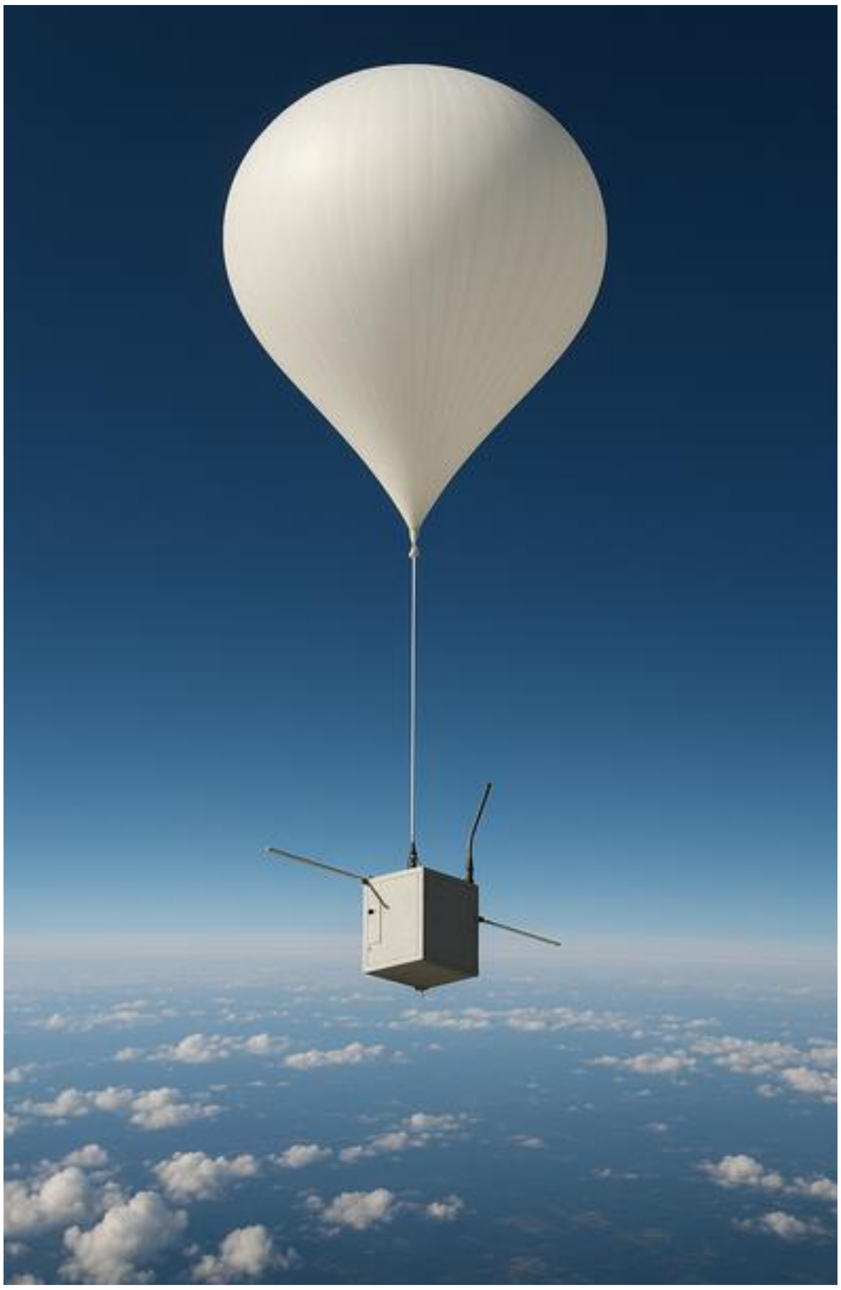
RSGB high-altitude balloon (HAB) Competition

LoRa APRS Tracker Hardware and Software

Setup and User guide

Version 1.8 7.8.2025 Dave Pegler, M0JKS

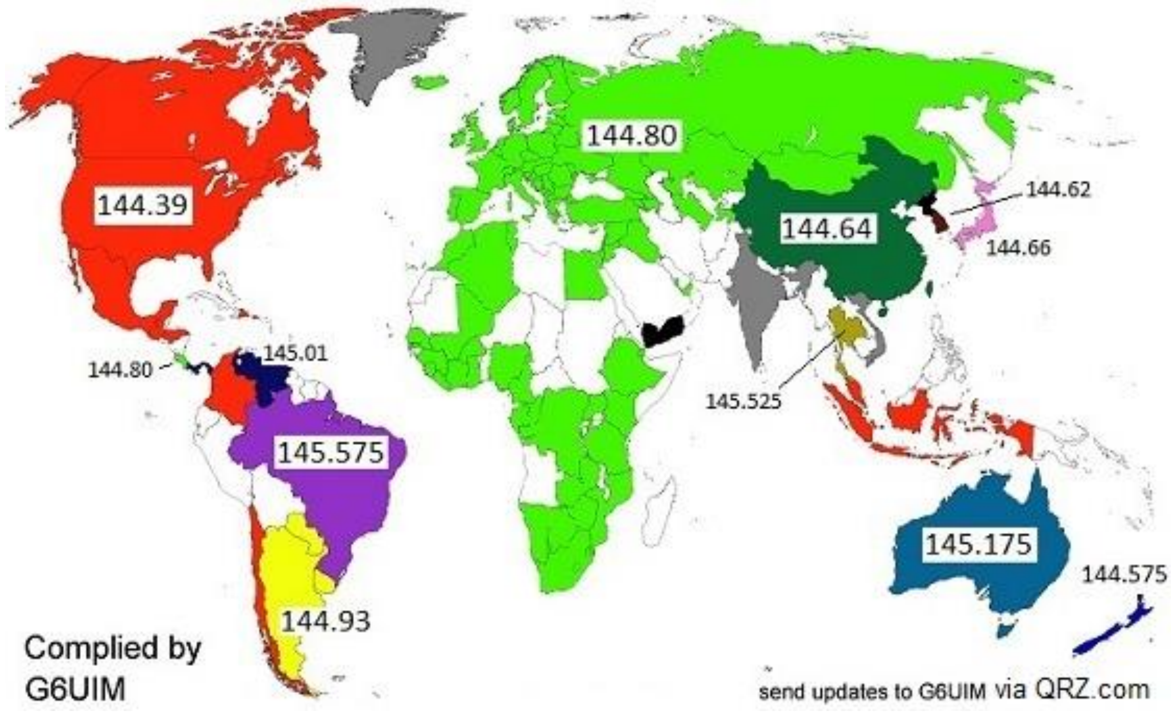


This document describes how to build and configure a LoRa APRS Tracker that will be required by a licensed radio amateur (or radio amateur club) in order to participate in the RSGB high-altitude balloon (HAB) competition being held on Saturday 20 September 2025.

Table of Contents

1 APRS (Automatic Packet Reporting System), what is it?

The Automatic Packet Reporting System (APRS) is an amateur radio based mechanism for real-time communications, which allows small packets of data to be sent and received over RF. More recently, APRS now permits data packets to be routed from RF, through the Internet and then back to RF through devices called iGates (Internet Gateways). It was invented in the 1990s by Bob Bruninga, call sign WB4APR (now sadly SK). It is based on a tweaked version of the ITU-T X25 protocol called AX25, which amongst other things, adds an amateur radio call sign and a 4 bit SSID to the existing X25 protocol. The addition of an amateur radio call sign allows the X25 protocol to be used to send small packets of data to and from APRS equipment belonging to licensed radio amateurs; and the SSID allows the APRS system to differentiate between different APRS equipment belonging to a single radio amateur. Until very recently, APRS was predominately used around the world on 2m using Audio Frequency-Shift Keying (AFSK). Below is a map compiled by G6UIM which shows what frequencies are allocated around the world for AFSK based APRS on 2m, and where.



2 LoRa APRS, how is that different to AFSK APRS?

Audio Frequency Shift-Keying (AFSK) based APRS uses two frequencies in the audio spectrum, 1200Hz and 2400Hz, to represent binary data at a speed of 1200 bits per second (or as only one bit is sent at a time, 1200 baud). For example, the letter "A" in ASCII (a standard for representing alpha numeric characters in computers from the 1960s but still used today) is 65 in decimal (base 10), 41 (written as 0x41 or 8'h41) in hexadecimal (base 16), and 01000001 in binary (base 2). To send the letter "A" using AFSK, a device call a MoDeM (Modulator/Demodulator) takes the binary version of a alpha numeric character (for example 01000001 for "A"), processing it one bit at a time and sending an audio tone of 1200Hz if it encounters a logic "1" and an audio tone of 2400Hz if it encounters a logic "0". The audio tone is then FM modulated to the carrier frequency of 144.8MHz

(EU), 144.39MHz (US) and then transmitted. Depending on how long each tone is transmitted for, dictates the bit rate. A baud/bit rate of 1200 bit/s means each tone is transmitted for a period of just under 1mS (883uS to be precise). This is deliberate to ensure that the receiver can distinguish between a tone at 1200Hz (a logic "1") and a tone 2400Hz (a logic "0"), especially if the Signal to Noise Ratio (SNR) of the received signal is poor.

Until quite recently, APRS really only used AFSK, and usually on 2m. However, a new low power, long range proprietary technology called LoRa (**Lo**ng **Ra**nge) was introduced in the last few years by the French (now US) company Semtech. As the name suggests it is designed for Long Range communications but using Low Power. To achieve this, LoRa uses a digital modulation technique called Spread Spectrum (SS), and a particular variant of SS called Chirp Spread Spectrum (CSS). If this all sounds too good to be true, it's because it is. There is no free lunch with digital communications, and CSS is no different. To achieve long range, low power communications, SS modulation techniques deliberately spread a low bandwidth baseband signal over a much larger modulated bandwidth. This sounds nuts, right? As radio amateurs we are taught that bandwidth is a precious resource and that we must be careful to use it wisely and with consideration for other users. However, Spread Spectrum, which was designed for military communications, and was actually invented by a Hollywood actress called Hedy Lamarr back in the 1940s, has a very different approach to spectrum usage and efficiency. Up at high frequencies (e.g. beyond UHF and mm-wave) there is lots of bandwidth available (especially to the military), and so when operating at these high frequencies, the prevention of interference, either deliberate (i.e. jamming) or due to external factors is far more important than the conservation of bandwidth.

At this point we don't really need to go into much more details about SS and CSS, because it's quite complicated, and as usual involves a lot of complicated maths. To understand how LoRa works it's sufficient to know that when a transmitting station deliberately spreads a baseband signal over a much larger modulated bandwidth, when a receiving station then does the reverse (called despreading) the wanted signal is de-spread back to baseband, but the opposite happens to any interference/noise that is picked up in the communications channel. The act of despreading the wanted signal has the effect of spreading any interference or noise present in the channel over a much larger bandwidth. This attenuates the interference/noise and so increases the Signal to Noise Ration (SNR) of the recovered signal. Thus with CSS (and with other SS techniques also) we get something called processing gain, which is basically the ratio of the baseband signal to the spread signal. In LoRa this is called the Spreading Factor or SF. The higher the SF, the higher the processing gain, but lower the overall data rate. For something with a low data rate such as APRS, but which needs to be Long Range and Low Power; LoRa is thus a great fit for APRS.

3 RSGB high-altitude balloon (HAB) competition

To help radio amateurs begin experimenting with LoRa, APRS, software and coding, the RSGB are running a high-altitude balloon (HAB) competition on Saturday 20 September 2025 to coincide with National Coding Week.

rsgb.org/lora-balloon

A High-altitude Balloon will be launched by RSGB Director Ben Lloyd, GW4BML from Welshpool in Powys, with a cross-frequency LoRa APRS Digipeater payload on-board. The HAB will be airborne for around two hours and will reach a maximum height of 90,000 feet. The uplink frequency of the cross-frequency Digipeater on board the HAB will be 439.850MHz and the downlink frequency will be 433.850MHz. Any LoRa APRS equipment used to participate in the HAB competition must be configured to transmit on 439.850MHz; with 125KHz bandwidth, Spreading Factor (SF) 12, and

Coding Rate (CR) 5. The output power level must be set to less than or equal to +23dBm (200mW) EIRP.

The uplink frequency (439.850MHz, 125KHz B/W) has been allocated by the RSGB ETCC for use by licensed UK radio amateurs during the competition only. It is different to the current UK LoRa APRS frequency of 439.9125MHz to ensure that the HAB cross-frequency Digipeater only receives and decodes APRS packets from radio equipment participating in, and for the duration of, the competition. The downlink frequency of 433.850MHz will be used exclusively by the cross-frequency HAB Digipeater to forward any received LoRa APRS packets on the uplink to APRS-IS (APRS Internet Service) through a number of special iGates. These iGates will be placed on a number of mountain tops across the UK to ensure widespread coverage.

As licensed radio amateurs we are only allowed to operate in the 70cm band (430-440MHz) as secondary users. The primary users of the 70cm band in the UK. are the military; and because of this we are not permitted to operate airborne (i.e. transmit for an aeroplane, hot-air balloon, glider or, a high-altitude balloon). However, the unlicensed LDP433 (Low Power Devices 433MHz) band, which overlaps with the 70cm amateur radio band from 433.050MHz to 434.790MHz, has no such restriction - provided the transmit power level is less than or equal to +10dBm ERP (see Ofcom IR2030/1/10). After careful consultation with the RSGB ETCC, the frequency of 433.850MHz in the unlicensed LPD433 band has been selected for the HAB downlink frequency to allow the RSGB HAB competition to go ahead.

4 Recommended LoRa APRS Single Board Computer (SBC)

The RSGB LoRa APRS HAB competition is being run to encourage anyone with an amateur radio licence (individuals and clubs) to build, program, configure and deploy their own HAB tracker using a LoRa Single Board Computer (SBC). There are many different types and models of LoRa SBC available from online retailers (e.g. Amazon, AliExpress, Pi Hut etc.). All use the same SX1262 LoRa modem from Semtec but they are sold in three different flavours – EU433, EU868 and US915. These are so they can be used by unlicensed individuals in the LPD (Low Power Devices) or ISM (Instrument Scientific and Medical) bands – albeit with power levels of only around +10dBm (10mW). See the following link for more details:

https://en.wikipedia.org/wiki/LPD433

The SX1262 LoRa chipset at the heart of all these SBCs is a SDR (Software Defined Radio) with a direct conversion transceiver (aka zero-IF or homodyne) which is capable of operating from 150MHz up to 960MHz. Each different flavour of SBC available (EU433, EU868 and US915) uses the same SX1262 Semtech modem (150MHz → 960MHz) but with some basic front-end lumped LC filtering. Luckily for radio amateurs the LPD433 band (ITU region 1 ISM band of 433.050MHz to 434.790MHz) conveniently overlaps with the 70cm band (430MHz to 440MHz) for which radio amateurs have a secondary usage (the military being primary). Thus, these cheap LoRa SBCs, designed for the LPD433 band can be used by licensed radio amateurs in the 70cm band in-line with the guidance in the RSGB band plan.

For the RSGB HAB competition we recommend a cheap LoRa (EU433) SBC called the "Heltec LoRa Wireless Tracker". This board is actually a LoRa development PCB designed by Chinese company Heltec Automation, which consists of a ESP32-S3FN8 microprocessor, Semtech SX1262 LoRa modem, 0.96" OLED display, WiFi, Bluetooth and a UC6580 dual-frequency multi-constellation GNSS SoC which supports GPS, GLONASS, BDS, Galileo, NAVIC and QZSS.

For more information about the "Heltec Wireless Tracker" board please check out the following URL or scan the QR code with your phone.

https://heltec.org/project/wireless-tracker/

A full list of equipment, including the "Heltec Wireless Tracker" board, and where to source it from is listed in Section 17. It is recommended to buy the "Heltec Wireless Tracker" board from a reputable source, preferably directly from Heltec Automation or from their shop on Aliexpress, as some of the cheap "clones" available on the market are sub-standard. You must also remember to buy the "**EU433**" version and not the "EU868" version (which is for Meshtastic)

5 Software, Coding and Configuration

The RSGB HAB competition is being run by the RSGB on 20 September 2025 to coincide with National Coding Week. This is a UK wide programme to encourage the young, and those of us not so young, to learn about coding and programming; with the aim of making these skills accessible and engaging. The HAB competition is one way that the RSGB is encouraging radio amateurs to learn new technical skills, particularly coding and electronics during National Coding Week. It is being targeted at both Foundation and Intermediate licence holders who are (thanks to the recent licence changes) now at liberty to build their own equipment.

Although the "Heltec Wireless Tracker" board that we recommend for the HAB competition is an amazing bit of equipment, it's hardware and pretty useless without software to tell it what you want it to do. It's basically a tiny little computer, with LoRa modem, UHF radio, multi-constellation GNSS receiver (which supports GPS, GLONASS and Galileo) on a single-board PCB. It even has WiFi connectivity and Bluetooth should you need to connect to it from your Phone – which we will!
To begin doing anything useful with this little board, we need some software. Don't worry you won't be writing any code (for anyone new to all this malarkey, to "code" means to write software). the software has already been written (coded) for you. The purpose of the HAB competition is to show how easy it is to build, upload and configure the software on this little SBC to become a fully fledged LoRa APRS tracker; and use it to bounce radio signals off a Digipeater floating around at 90,000 feet on a high-altitude balloon somewhere over the UK.

To begin you need to download some software which software engineers call an Integrated Development Environment (or IDE). It's just an application which you run on a Windows or Linux PC, which takes the software written for the little ESP32-S3FN8 microprocessor on the Heltec Wireless Tracker board, and converts it into instructions (called machine code) that the microprocessor understands. Humans are rubbish at reading machine code, so to make "coding" easier it's written in what is called a high-level language. You may have heard of some, for example Cobol, Basic, C, C++, Lisp, Python, Ada, Pascal, Java, C# or Rust – there are a lot of them. These high-level languages use many English words like "If, While, For, Begin, End" to make coding easier for Humans to write. When the software "coding" is done, it needs to converted into the instructions (called machine code) that the microprocessor can actually follow. This conversion process is called "compiling" in some languages, and interpreting in others. In this project, the high-level software we are going to download, and use is written in a language called C++.

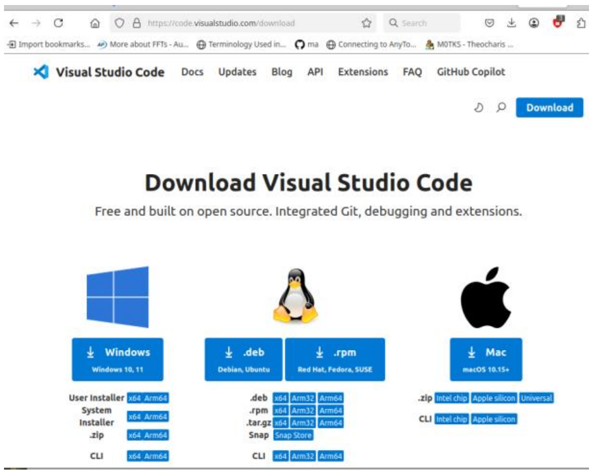
6 Installing Visual Studio Code

For the RSGB HAB project you need to download a Windows application called Microsoft Visual Studio Code (VSC) – this will be the IDE.

To download (Visual Studio Code) VSC click on the following link:

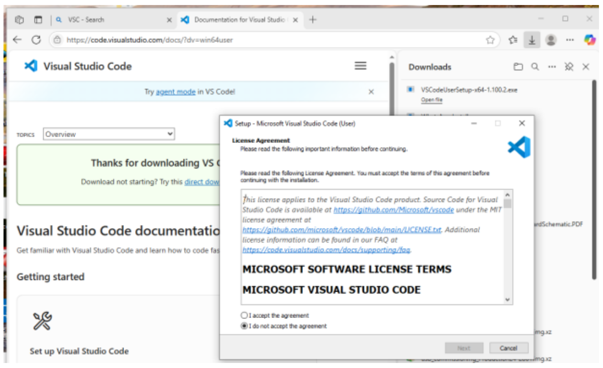
https://code.visualstudio.com/download

When you do so, you should see the following Microsoft web page appear:



To install VSC click on the Windows (Windows 10,11) blue box on the left-hand side. You should see a self-extracting executable called "VSCodeUserSetup-x64-1.100.2.exe" (or something similar) being download to your "Downloads" directory. When the file has downloaded, using the mouse left-hand button double-click on "VSCodeUserSetup-x64-1.100.2.exe" (or whatever it is called) to begin the installation process.

A "Setup" pop-up box should then appear, asking you to agree to some licensing terms.


Setup - Microsoft Visual Studio Code (User)

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

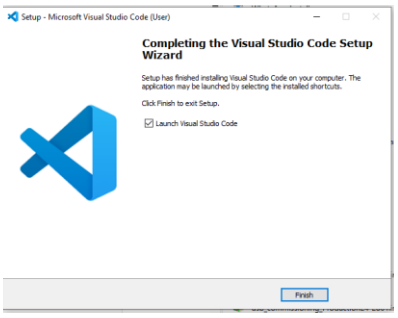
This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO CODE

I accept the agreement [✓ I do not accept the agreement]

Once you have read through the licensing information, if you are happy to proceed select "I accept this agreement" and then click "Next". You will then be asked where you would like VSC installed. Unless you want to install VSC in a particular location, just click "Next". Then when asked whether you would like VSC placing in your shortcut's toolbar. Again click "Next" so that it is. Continue clicking "Next" on all the pop-up boxes until you get to the "Ready to Install" pop-up box. Here click on "Install" to the installation process.


Setup - Microsoft Visual Studio Code (User)

Completing the Visual Studio Code Setup Wizard

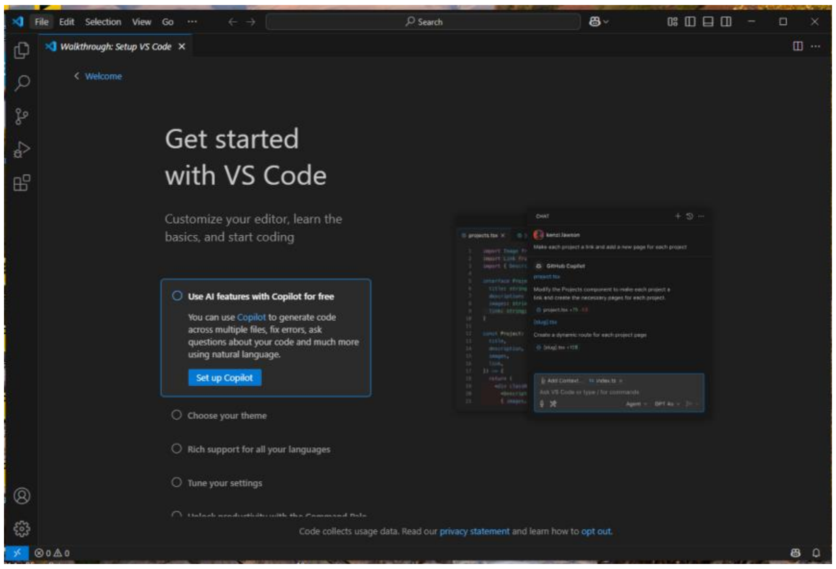
Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

[✓] Launch Visual Studio Code

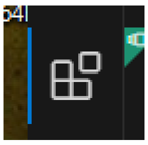
When the installation process of complete, you will see the pop-up box to the right appear. Click on Finish and at this point VSC should automatically start up.

When Visual Studio Code is up and running, it should look similar to the screen shot below:


File Edit Selection View Go
Walkthrough: Setup VS Code
Welcome
Get started with VS Code
Customize your editor, learn the basics, and start coding
Use AI features with Copilot for free
You can use Copilot to generate code across multiple files, fix errors, ask questions about your code and much more using natural language.
Set up Copilot
Choose your theme
Rich support for all your languages
Tune your settings
Code collects usage data. Read our privacy statement and learn how to opt out.

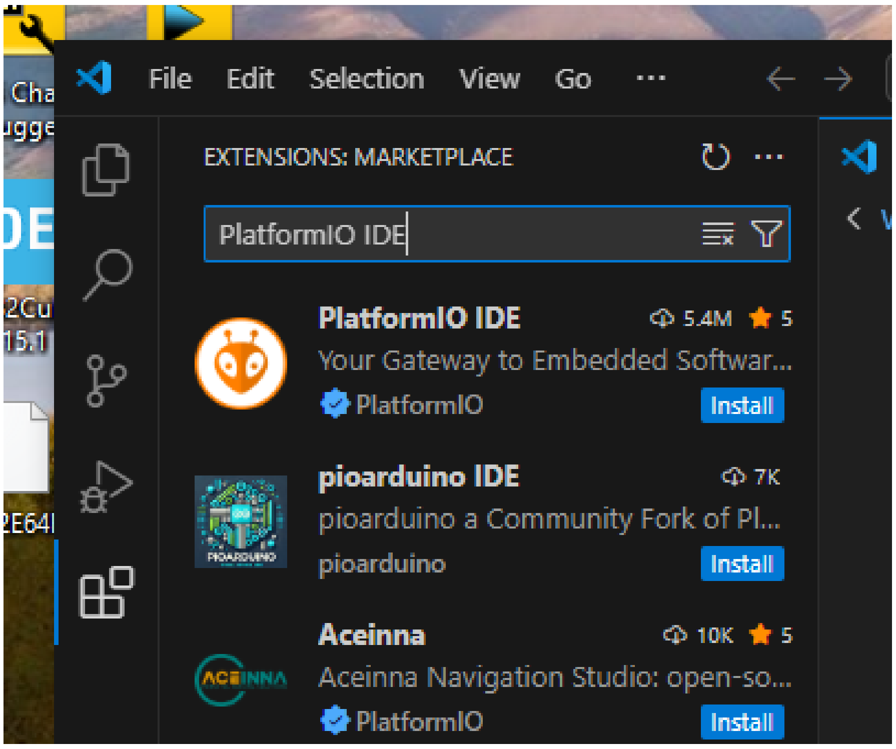
7 Installing VSC Extension "PlatformIO IDE"

The next step in the installation process is to install a VSC extension or plug-in called "**PlatformIO IDE**". This extension is an extra bit of software that runs within VSC that allows it to build code for the ESP32-S3FN8 microprocessor that resides on the Heltec Wireless Tracker board.

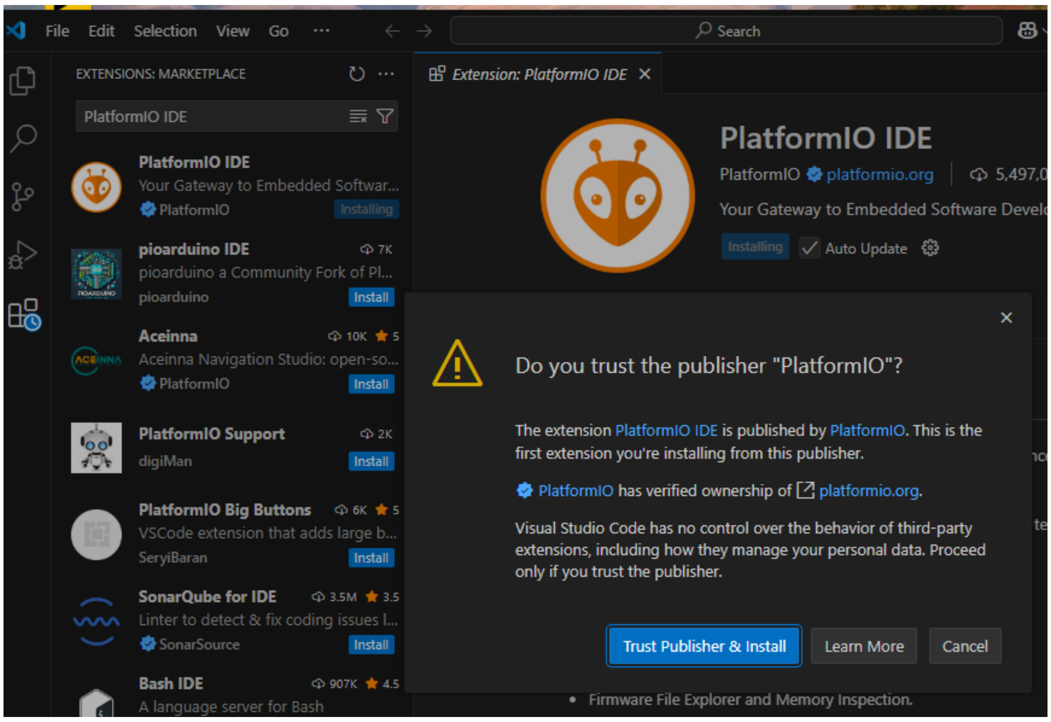


To begin this step, click the "Extension" icon on the left-hand side of VSC. It looks like 4 boxes on top of each other, with one box trying to breaking free.

When you click on the "Extensions" icon, you should be presented with a search box in the top left-hand corner of VSC. Under "EXTENSIONS:MARKETPLACE" there should be a "Search Extensions in MarketPlace" search box. In here type "PlatformIO IDE" as shown in the screenshot blelow:



As you type "PlatformIO IDE" in the search box and orange coloured Wasp looking icon should appear as shown in the Screenshot above. Click the blue "Install" button to the right and this should begin the installation process. You will be asked another series of questions like those shown in the screenshot below:


Do you trust the publisher "PlatformIO"?

The extension PlatformIO IDE is published by PlatformIO. This is the first extension you're installing from this publisher.

PlatformIO has verified ownership of platformio.org.

Visual Studio Code has no control over the behavior of third-party extensions, including how they manage your personal data. Proceed only if you trust the publisher.

Simply click on "Trust Publisher & Install" to begin the installation. This is the final step installing the IDE (Visual Studio Code) required to build the software for the RSGB LoRa APRS High-altitude Balloon (HAB) Competition

8 Downloading the "LoRa APRS Tracker" Software

As we described earlier, to make the ESP32-S3FN8 microprocessor on the Heltec Wireless Tracker board do what we want, we need some software to run on it. The software we use to do this was written by a licensed radio amateur Richardo Guzman, CA2RXU, using a high-level programming (coding) language called C++. Once this software is downloaded, it then needs to be converted (compiled) into a format that the microprocessor understands. The IDE we just installed (Visual Studio Code or VSC) does this for us.

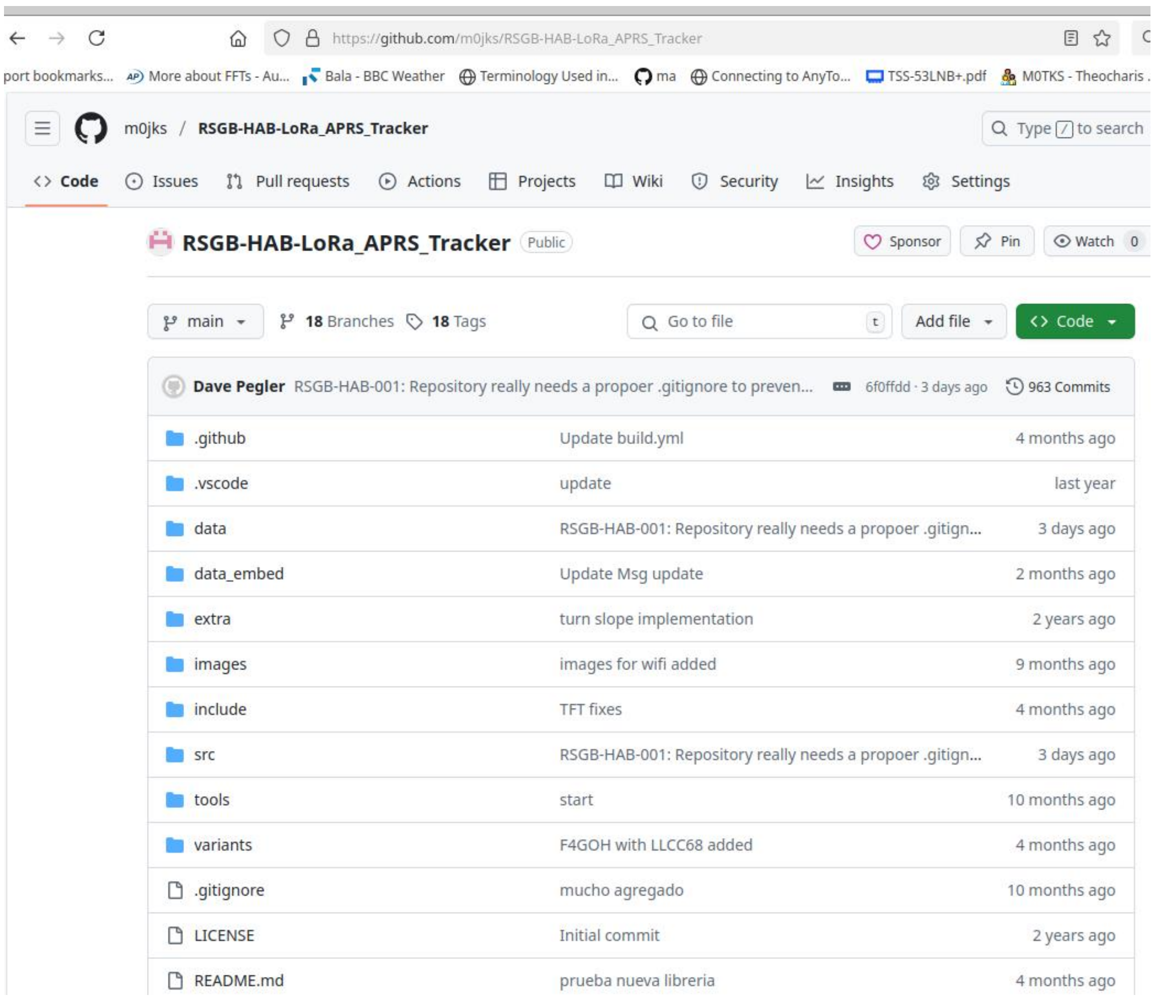
In Richardo's original version of the software, it is possible to switch between three different operating frequencies using the left-hand button on the Heltec Wireless Tracker board. These frequencies are 433.775MHz (EU), 434.855MHz (Poland) and 439.9125MHz (UK). As detailed in section 3 above, for the duration of RSGB HAB competition, a new frequency of 439.850MHz has been allocated by the RSGB ETCC. It is different to the current UK LoRa APRS frequency of 439.9125MHz to ensure that the HAB cross-frequency Digipeater only receives and decodes APRS packets from radio equipment participating in, and for the duration of, the competition. Thus, to ensure only the correct frequency is used by participants of the RSGB HAB competition a new version of the software repository has been created (cloned) on github.com, and this has been tweaked to add 439.850MHz (HAB) as the only frequency option.

The next stage of the process is to download the RSGB HAB version of the “LoRa APRS Tracker” software from github.com; and then use VSC to convert (compile) it to a format we can then upload to the ESP32 microprocessor on the Heltec Wireless Tracker board.

To download this version of the software, please click on the following link to navigate to the RSGB HAB LoRa APRS Tracker github page:

RSGB HAB LoRa APRS Tracker Software

When you click on the above link you should see something similar to the screenshot below.

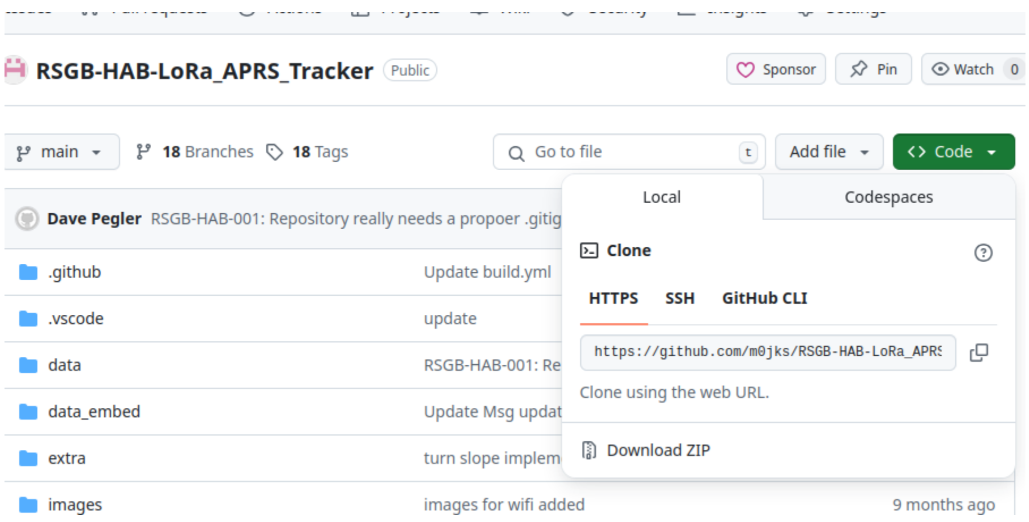

https://github.com/m0jks/RSGB-HAB-LoRa_APRS_Tracker
m0jks / RSGB-HAB-LoRa_APRS_Tracker
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
RSGB-HAB-LoRa_APRS_Tracker Public
Sponsor Pin Watch 0
main 18 Branches 18 Tags Go to file Add file Code

Dave Pegler RSGB-HAB-001: Repository really needs a propoer .gitignore to preven... 6f0ffdd · 3 days ago 963 Commits

.github	Update build.yml	4 months ago
.vscode	update	last year
data	RSGB-HAB-001: Repository really needs a propoer .gitign...	3 days ago
data_embed	Update Msg update	2 months ago
extra	turn slope implementation	2 years ago
images	images for wifi added	9 months ago
include	TFT fixes	4 months ago
src	RSGB-HAB-001: Repository really needs a propoer .gitign...	3 days ago
tools	start	10 months ago
variants	F4GOH with LLCC68 added	4 months ago
.gitignore	mucho agregado	10 months ago
LICENSE	Initial commit	2 years ago
README.md	prueba nueva libreria	4 months ago

Github (github.com) is a internet wide software storage repository that uses a distributed software version control system called “git” that was written by Linus Torvald – the guy who wrote the Linux operating system. It is free to use and is a very powerful software repository and version control tool. Here, we don’t care about any of that, we are just using it to get the latest stable version of the RSGB HAB “LoRa APRS tracker” software.

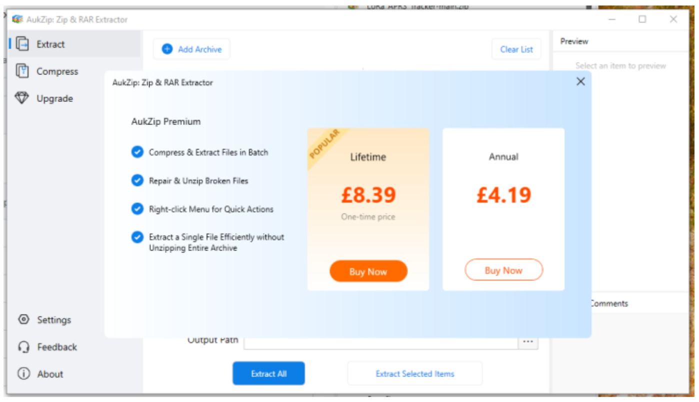
To down the latest version of the "RSGB-HAB-LoRa APRS tracker" software, click on the Green "<> Code" icon, and then select "Download ZIP" as shown below.



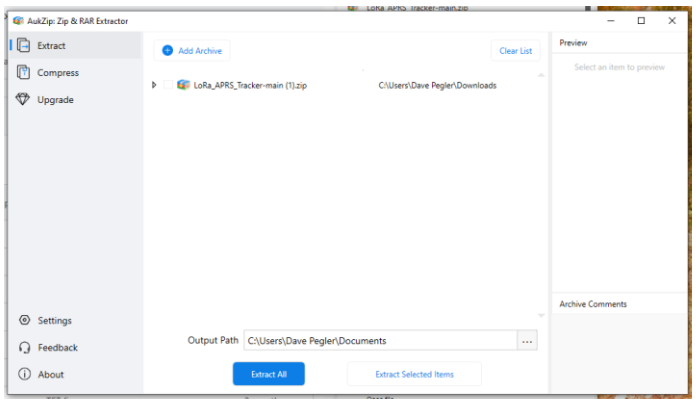
This will then start downloading a zip file to your Downloads directory called:

"RSGB-HAB-LoRa_APRS_Tracker-main.zip"

Once this zip file has finished downloading, you will need to extract it so we can use it with VSC. To do this double click on the file you just downloaded, and you will see the following pop-up window appear. It initially tries to flog you some extra zip extraction software that is already installed in Windows. To get rid of the annoying advertisement, click the "X" at the top right-hand corner



and you should be left with the following pop-up window:

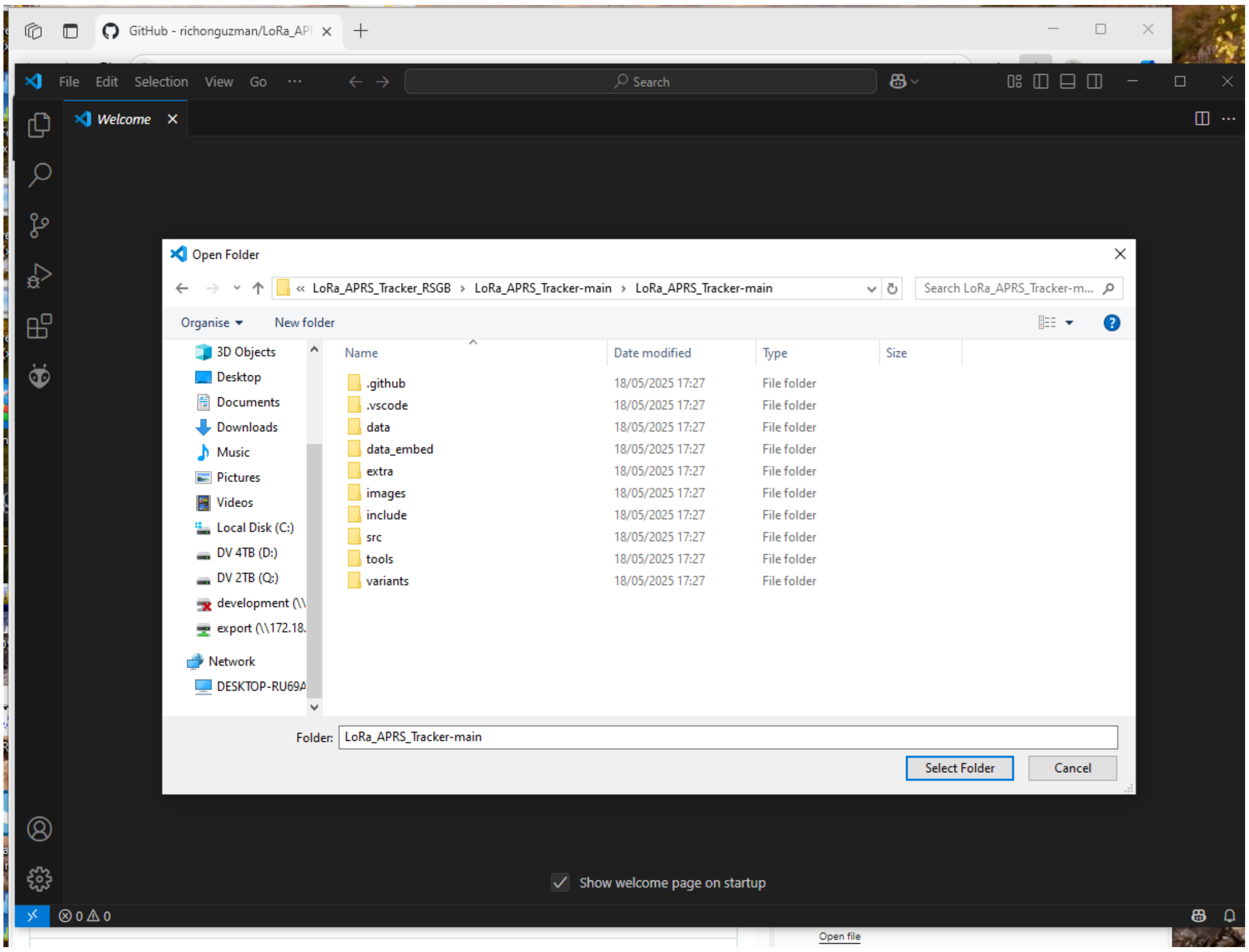

AukZip: Zip & RAR Extractor
Extract
Compress
Upgrade
Add Archive
Clear List
Preview
Select an item to preview
LoRa_APRS_Tracker-main (1).zip
C:\Users\Dave Pegler\Downloads
Archive Comments
Settings
Feedback
About
Output Path
C:\Users\Dave Pegler\Documents
Extract All
Extract Selected Items

In “Output Path” change the location where you want the software to be extracted to; and remember the location for later! Finally Click on “Extract All” and when all the archive extraction is complete, close down the extraction utility.

9 Importing LoRa Tracker software into Visual Studio Code (VSC)

Now that Visual Studio Code (VSC) is installed, and you have successfully downloaded the RSGB-HAB-LoRa_APRS_Tracker software, it is time to import this into VSC so you can begin to compile it.

Using VSC click on “File” (top left-hand corner) and then “Open Folder”. A “Open Folder “window should then appear and you will be invited to navigate to the location where you downloaded and extracted the RSGB-HAB-LoRa_APRS_Tracker software. The folder should contain files similar to those shown in the screenshot below.



Name	Date modified	Type	Size
.github	18/05/2025 17:27	File folder	
.vscode	18/05/2025 17:27	File folder	
data	18/05/2025 17:27	File folder	
data_embed	18/05/2025 17:27	File folder	
extra	18/05/2025 17:27	File folder	
images	18/05/2025 17:27	File folder	
include	18/05/2025 17:27	File folder	
src	18/05/2025 17:27	File folder	
tools	18/05/2025 17:27	File folder	
variants	18/05/2025 17:27	File folder	

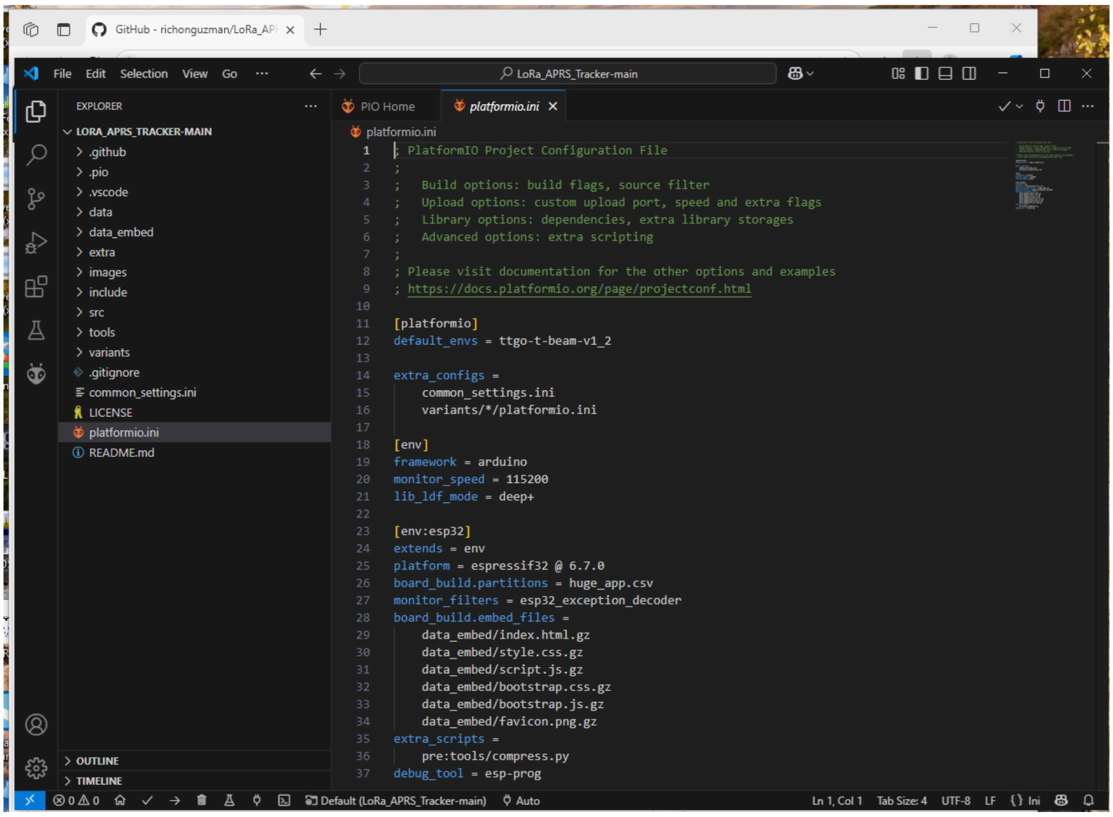
Folder: LoRa_APRS_Tracker-main

Click “Select Folder” and then wait for VSC to do its thing. It will at some stage ask you whether you Trust the authors. Click “Yes, I trust the authors” to tell VSC to begin the import process. It will then begin by opening up a file called “platformio.ini”, and then in the bottom left-hand corner will show the progress of it configuring PlatformIO.

Please wait patiently for this process to finish before moving on to the next stage.

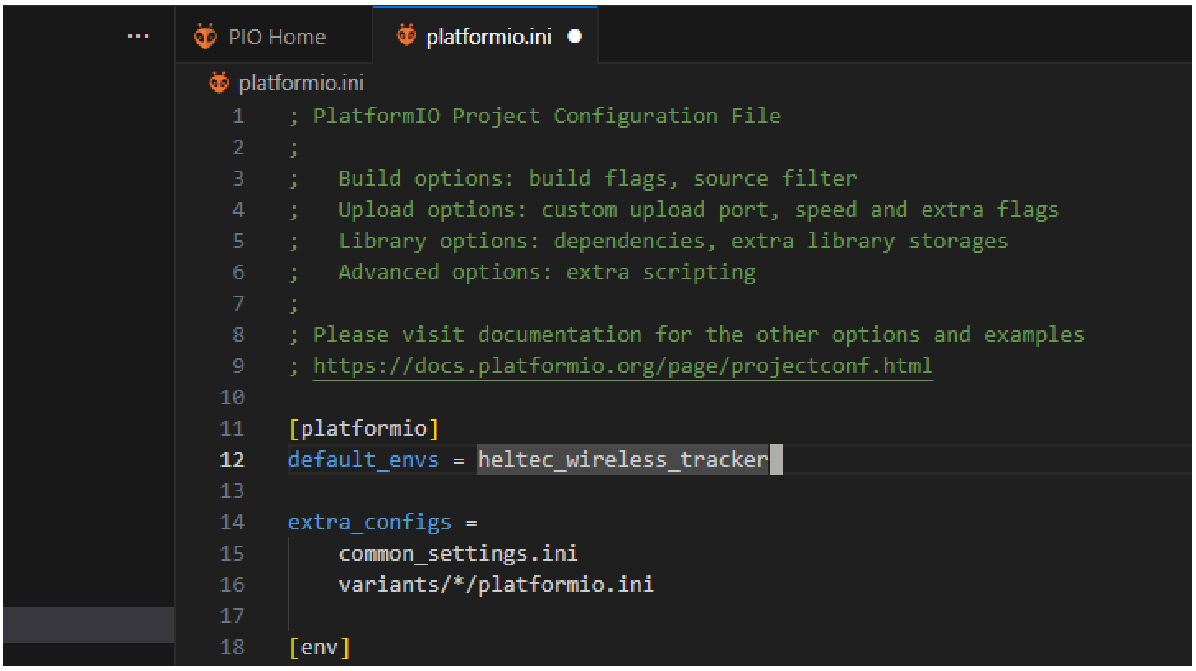
10 Building the Software for the Heltec Wireless Tracker board

When VSC and PlatformIO have finished importing the "RSGB-HAB_LoRa_APRS_Tracker" software, you will be presented with the "platformio.ini" project file as shown below.



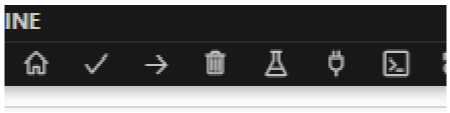
```
1   ; PlatformIO Project Configuration File
2   ;
3   ;   Build options: build flags, source filter
4   ;   Upload options: custom upload port, speed and extra flags
5   ;   Library options: dependencies, extra library storages
6   ;   Advanced options: extra scripting
7   ;
8   ; Please visit documentation for the other options and examples
9   ; https://docs.platformio.org/page/projectconf.html
10
11  [platformio]
12  default_envs = ttgo-t-beam-v1_2
13
14  extra_configs =
15      common_settings.ini
16      variants/*/platformio.ini
17
18  [env]
19  framework = arduino
20  monitor_speed = 115200
21  lib_ldf_mode = deep+
22
23  [env:esp32]
24  extends = env
25  platform = espressif32 @ 6.7.0
26  board_build.partitions = huge_app.csv
27  monitor_filters = esp32_exception_decoder
28  board_build.embed_files =
29      data_embed/index.html.gz
30      data_embed/style.css.gz
31      data_embed/script.js.gz
32      data_embed/bootstrap.css.gz
33      data_embed/bootstrap.js.gz
34      data_embed/favicon.png.gz
35  extra_scripts =
36      pre:tools/compress.py
37  debug_tool = esp-prog
```

If you take a closer look at this file, you will see that by default the software is configured for a "TTGO Tbeam V1.2" board – see lines 11 and 12 in the screenshot above.
In the "platformio.ini" configuration file, the option "**default_envs**" (see line 12) tells VSC which LoRa Single Board Computer (SBC) you want it to compile the LoRa_APRS_Tracker software for. Clearly "ttgo-t-beam-v1_2" is the wrong board here, so we need to use VSC to edit this file and change the "**default_envs**" options to "heltec_wireless_tracker" like so:

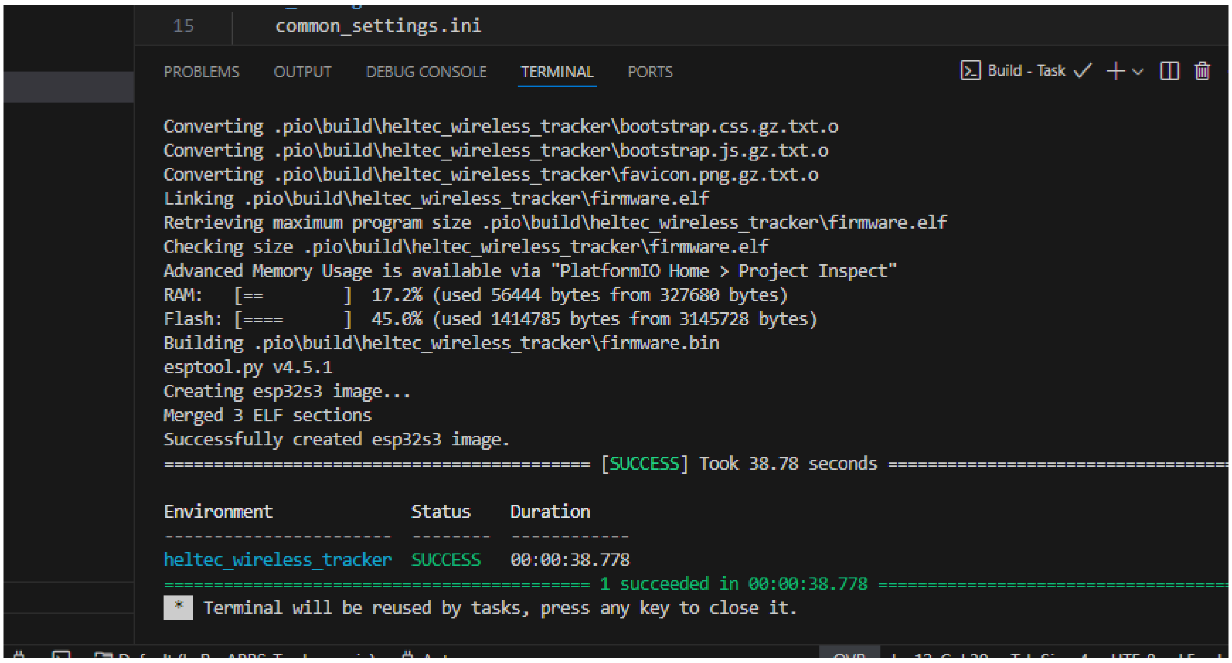


```
1   ; PlatformIO Project Configuration File
2   ;
3   ;   Build options: build flags, source filter
4   ;   Upload options: custom upload port, speed and extra flags
5   ;   Library options: dependencies, extra library storages
6   ;   Advanced options: extra scripting
7   ;
8   ; Please visit documentation for the other options and examples
9   ; https://docs.platformio.org/page/projectconf.html
10
11  [platformio]
12  default_envs = heltec_wireless_tracker
13
14  extra_configs =
15      common_settings.ini
16      variants/*/platformio.ini
17
18  [env]
```

The next step is to instruct VSC to compile the "RSGB-HAB-LoRa_APRS_Tracker" software for the Heltec Wireless Tracker board. To do this, click on the White tick that is located on the bottom tool bar. This is positioned between the House symbol and the Right Arrow as shown in the screenshot below:



After a short period, if everything goes well, you should see "Successfully created esp32s3 image" displayed, and the word SUCCESS (highlighted in Green) displayed several times in the "TERMINAL" output window as shown in the screenshot below:



```
Converting .pio\build\heltec_wireless_tracker\bootstrap.css.gz.txt.o
Converting .pio\build\heltec_wireless_tracker\bootstrap.js.gz.txt.o
Converting .pio\build\heltec_wireless_tracker\favicon.png.gz.txt.o
Linking .pio\build\heltec_wireless_tracker\firmware.elf
Retrieving maximum program size .pio\build\heltec_wireless_tracker\firmware.elf
Checking size .pio\build\heltec_wireless_tracker\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [==        ]  17.2% (used 56444 bytes from 327680 bytes)
Flash: [====      ]  45.0% (used 1414785 bytes from 3145728 bytes)
Building .pio\build\heltec_wireless_tracker\firmware.bin
esptool.py v4.5.1
Creating esp32s3 image...
Merged 3 ELF sections
Successfully created esp32s3 image.
========================================== [SUCCESS] Took 38.78 seconds ==================================

Environment             Status    Duration
----------------------  --------  ------------
heltec_wireless_tracker  SUCCESS   00:00:38.778
========================================== 1 succeeded in 00:00:38.778 ==================================
 *  Terminal will be reused by tasks, press any key to close it.
```

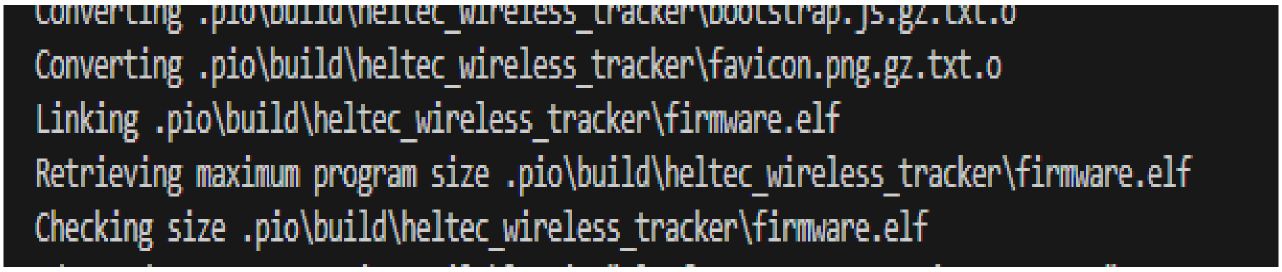
This indicates that the compilation process completed successfully.

11 Building the esp32s image for the Heltec Wireless Tracker board

As we described earlier in this document, the ESP32-S3FN8 microprocessor on the Heltec Wireless Tracker board does not understand high-level code (written in C+) - so we need to modify it into a format that it does understand. Part of this process is called compiling which we discussed above, but there are also a few further steps required after compiling. We need to gather together bits of pre-compiled code (compiled by someone else) called libraries. These libraries are bits of compiled code that do things that we need, but we really don't need to know too much about how they do it. One of these libraries we need is called "RadioLib", and the RSGB-HAB-LoRa_APRS_Tracker software uses features in this library (called functions) to setup and control the SX1262 LoRa modem on the Heltec Wireless Tracker board. We don't care how it does this, we simply use it by calling the appropriate "functions" it provides.

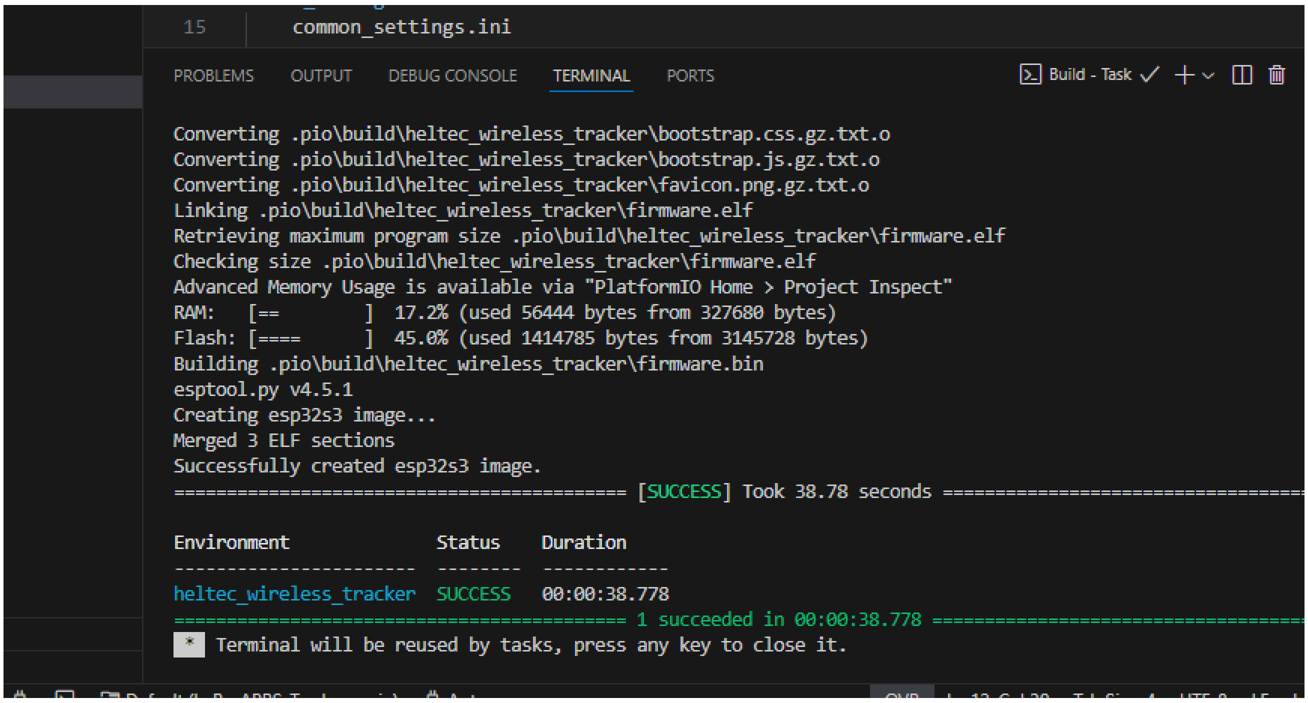
After compiling the RSGB-HAB-LoRa_APRS_Tracker software, the VSC build process must perform a few more steps. The first is called linking, where it gathers together (or links) the RSGB-HAB-LoRa_APRS_Tracker software we compiled earlier, with various bits of pre-compiled software stored in these libraries (e.g. RadioLib). The resultant file is called a Executable Linkable Format

(ELF) binary. If you look closely at the output from the TERMINAL window (on the previous page) you will see the following lines in the Terminal Window:



```
Converting .pio\build\heltec_wireless_tracker\bootstrap.js.gz.txt.o
Converting .pio\build\heltec_wireless_tracker\favicon.png.gz.txt.o
Linking .pio\build\heltec_wireless_tracker\firmware.elf
Retrieving maximum program size .pio\build\heltec_wireless_tracker\firmware.elf
Checking size .pio\build\heltec_wireless_tracker\firmware.elf
```

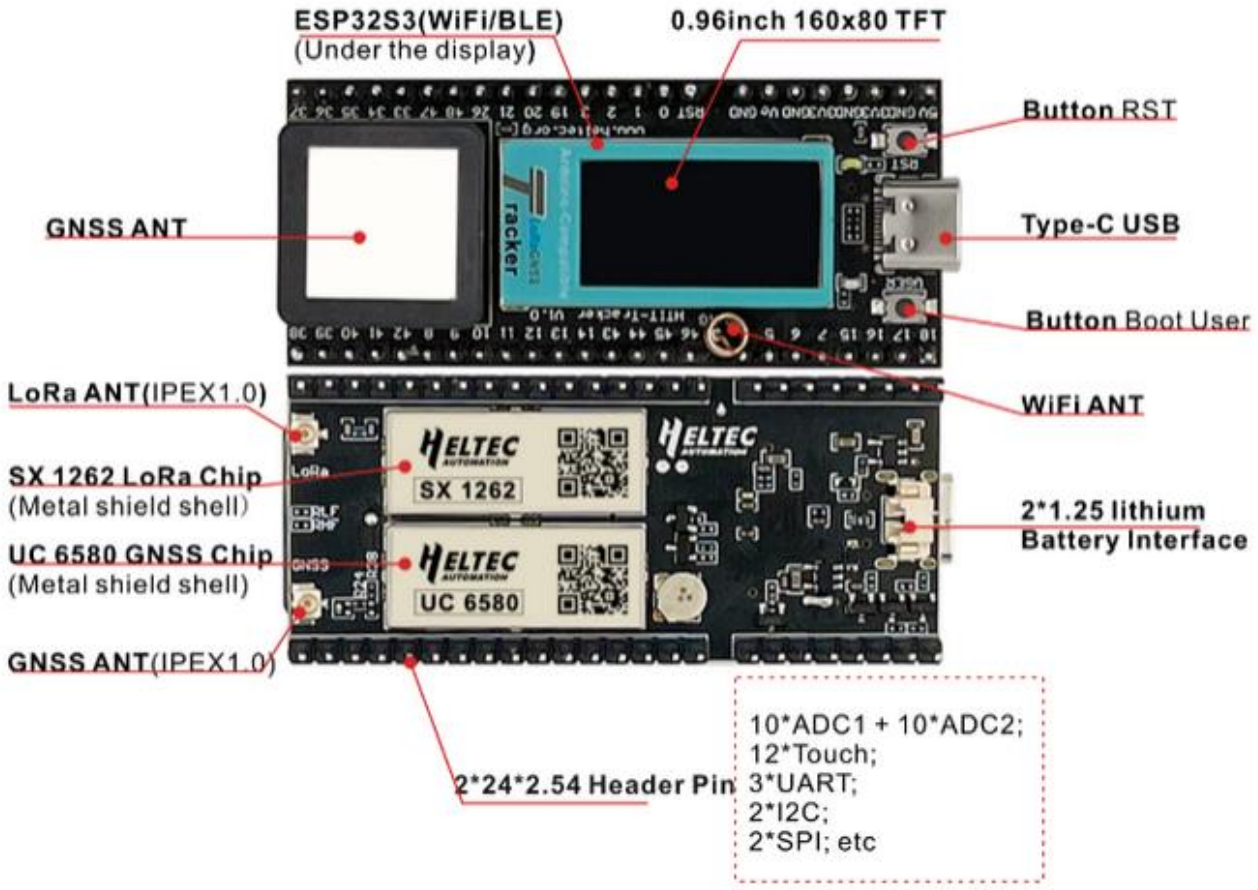
Here VSC is linking together all the different compiled bits of software (called object files) into a single binary blob called “firmware.elf”. This image is still not quite in the correct format to be uploaded to and run on the ESP32-S3FN8 microprocessor; it still needs a little more tweaking.
The very last compilation/linking stage that VSC performs is shown in the screenshot below. It converts “firmware.elf” into “firmware.bin” for the specific ESP32 microprocessor on the Heltec Wireless Tracker board (i.e. the ESP32-S3FN8).



```
Converting .pio\build\heltec_wireless_tracker\bootstrap.css.gz.txt.o
Converting .pio\build\heltec_wireless_tracker\bootstrap.js.gz.txt.o
Converting .pio\build\heltec_wireless_tracker\favicon.png.gz.txt.o
Linking .pio\build\heltec_wireless_tracker\firmware.elf
Retrieving maximum program size .pio\build\heltec_wireless_tracker\firmware.elf
Checking size .pio\build\heltec_wireless_tracker\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [==        ]  17.2% (used 56444 bytes from 327680 bytes)
Flash: [====      ]  45.0% (used 1414785 bytes from 3145728 bytes)
Building .pio\build\heltec_wireless_tracker\firmware.bin
esptool.py v4.5.1
Creating esp32s3 image...
Merged 3 ELF sections
Successfully created esp32s3 image.
========================================== [SUCCESS] Took 38.78 seconds ==================================

Environment              Status    Duration
-----------------------  --------  ------------
heltec_wireless_tracker  SUCCESS   00:00:38.778
========================================== 1 succeeded in 00:00:38.778 ==================================
 *  Terminal will be reused by tasks, press any key to close it.
```

12 Preparing Windows for Heltec Wireless Tracker board upload

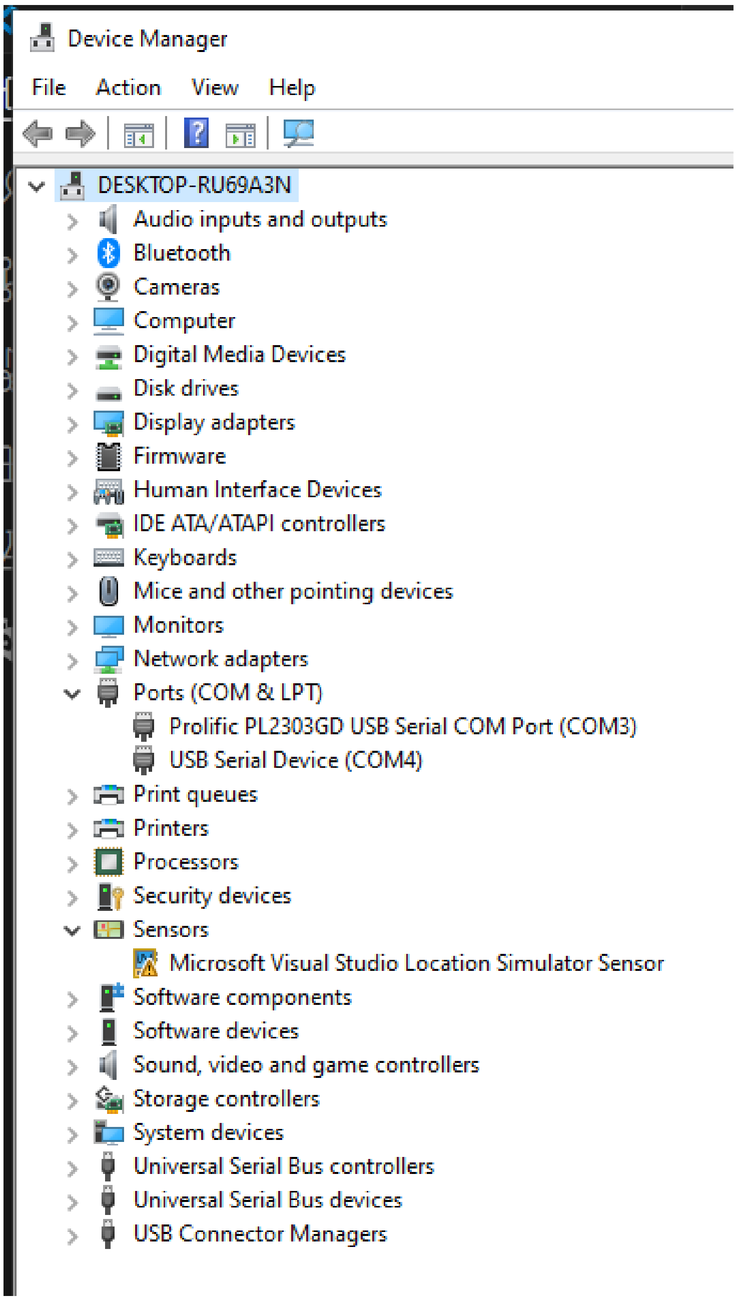
All the hard work is now done, and the fun can begin. It's time to open the Heltec Wireless Tracker plastic case and take out the little SBC. As shown below, on the top side of the SBC PCB is a 9.6" OLED display, USB-C connector, a GNSS antenna and the ESP32-S3FN8 microprocessor (hidden under the OLED display). On the reverse side, you will see the SX1262 LoRa Modem, the UC6580 dual-frequency multi-constellation GNSS SoC, a JST 1.25mm connector (for a connecting to a 3v7 Lithium-Ion battery), some push buttons and two IPEX U.FL antenna connectors marker LoRa and GNSS:



To upload the "firmware.bin" image (create by VSC) to the Heltec Wireless Tracker board, it is necessary to connect your Heltec Wireless Tracker board to your Windows PC via USB . You need a USB-C to USB-A lead, the same type use to charge Android mobile phones. Plug the USB-A end into a spare USB-A slot on your PC, and the USB-C end into the Heltec Wireless Tracker board. The first time you do this Windows will pop-up a message window about installing drivers for a "USB JTAG/Serial".

Once Windows has finished installing drivers, in the "Type here to search" search bar at the bottom left-hand corner, type "Device Manager". This will bring up the Windows "Device Manager" which should look similar to the screenshot on the next page.

Windows “Device Manager” is a useful tool for checking that Windows has correctly detected the USB to serial chipset on the Heltec Wireless Tracker board as you connect it to your PC via USB.


Device Manager

File Action View Help

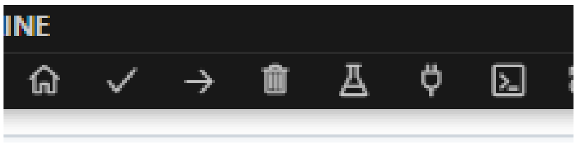
DESKTOP-RU69A3N
- Audio inputs and outputs
- Bluetooth
- Cameras
- Computer
- Digital Media Devices
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Prolific PL2303GD USB Serial COM Port (COM3)
  - USB Serial Device (COM4)
- Print queues
- Printers
- Processors
- Security devices
- Sensors
  - Microsoft Visual Studio Location Simulator Sensor
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers
- Universal Serial Bus devices
- USB Connector Managers

As you plug and unplug the USB cable to your Heltec Wireless Tracker board, look at the list of devices detected by “Device Manager”. Under Ports (COM & LPT) you should observe a Windows “USB Serial (COMX)” device (where X is a number) appear as you plug in and unplug the USB cable. As shown in the Screenshot above, Windows reports the device as COM4 on my Windows 11 PC. On your Windows PC this number might be different so make note of it before moving on to the next stage

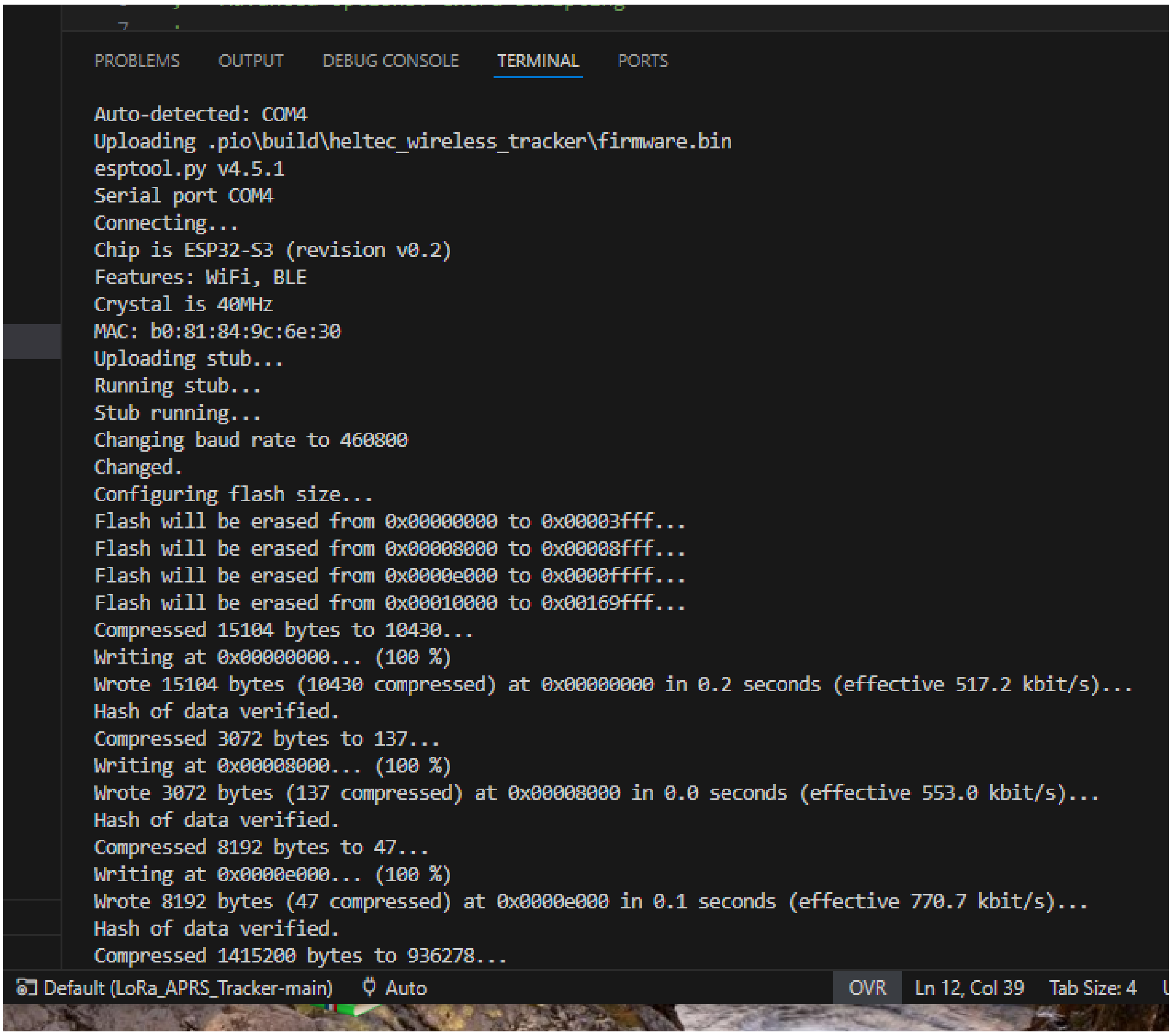
13 Uploading to Heltec Wireless Tracker board

The final step to get the LoRa_APRS_Tracker software running on the Heltec Wireless Tracker board is to upload the firmware.bin image that was created by VSC earlier. You also need to upload a configuration file which tells the RSGB-HAB-LoRa_APRS_Tracker software vital information it needs – such as the APRS frequency, your call sign and APRS symbol to display.

With the Heltec Wireless Tracker board still plugged into the USB port on your Windows PC, you can now return to VSC to perform the upgrade. Earlier we described how to start VSC compiling the software by clicking on the white tick on the bottom-left tool bar (between the house and right arrow symbol)



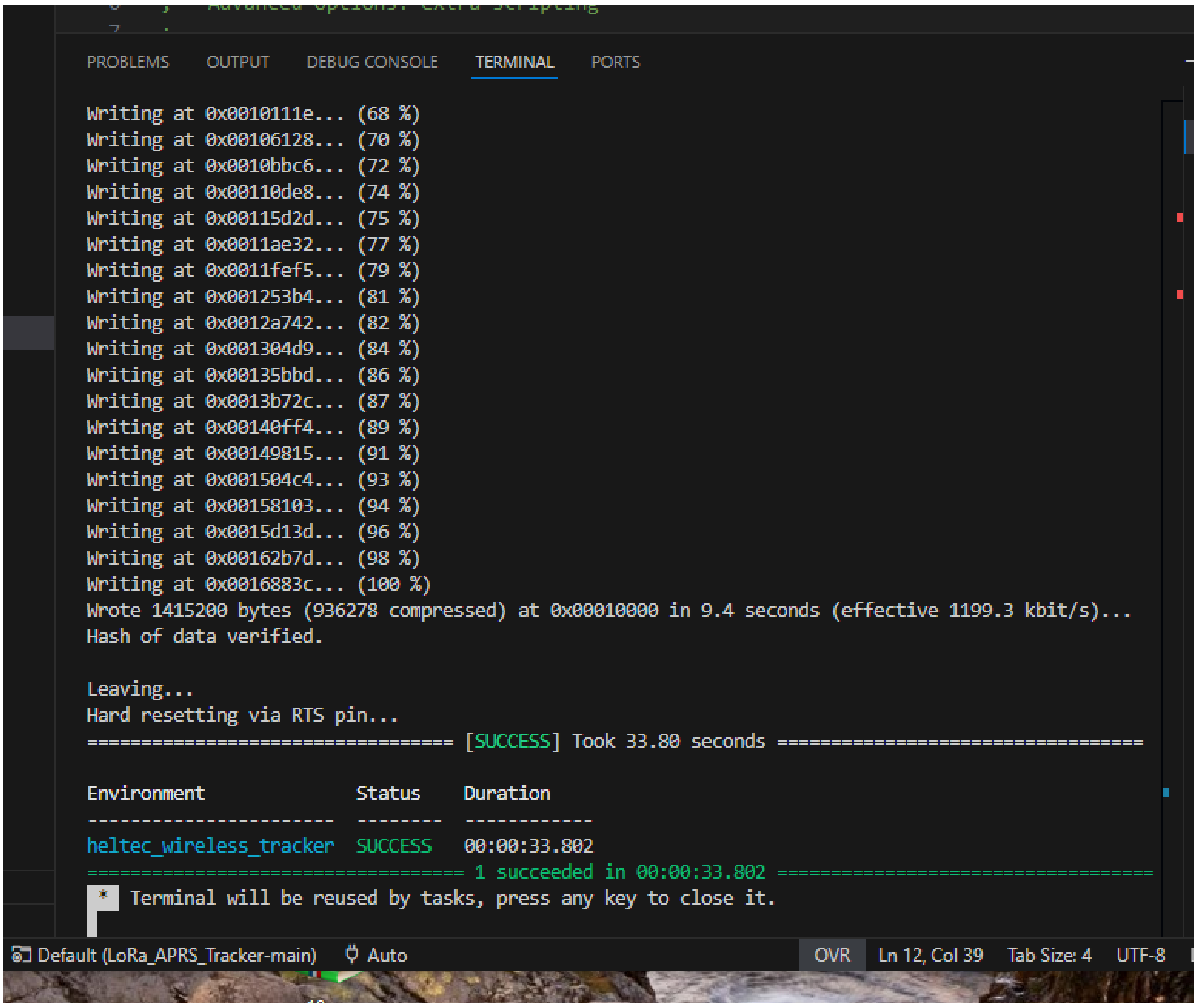
Next to white tick is a white arrow pointing to the right; this is the firmware.bin upload button. When you click on it, VSC will recompile some of the code if required (for example if something has changed), create a fresh version of firmware.bin and then proceed to upload it via the USB serial link.



```
Auto-detected: COM4
Uploading .pio\build\heltec_wireless_tracker\firmware.bin
esptool.py v4.5.1
Serial port COM4
Connecting...
Chip is ESP32-S3 (revision v0.2)
Features: WiFi, BLE
Crystal is 40MHz
MAC: b0:81:84:9c:6e:30
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00003fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x00169fff...
Compressed 15104 bytes to 10430...
Writing at 0x00000000... (100 %)
Wrote 15104 bytes (10430 compressed) at 0x00000000 in 0.2 seconds (effective 517.2 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 137...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (137 compressed) at 0x00008000 in 0.0 seconds (effective 553.0 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 770.7 kbit/s)...
Hash of data verified.
Compressed 1415200 bytes to 936278...
```

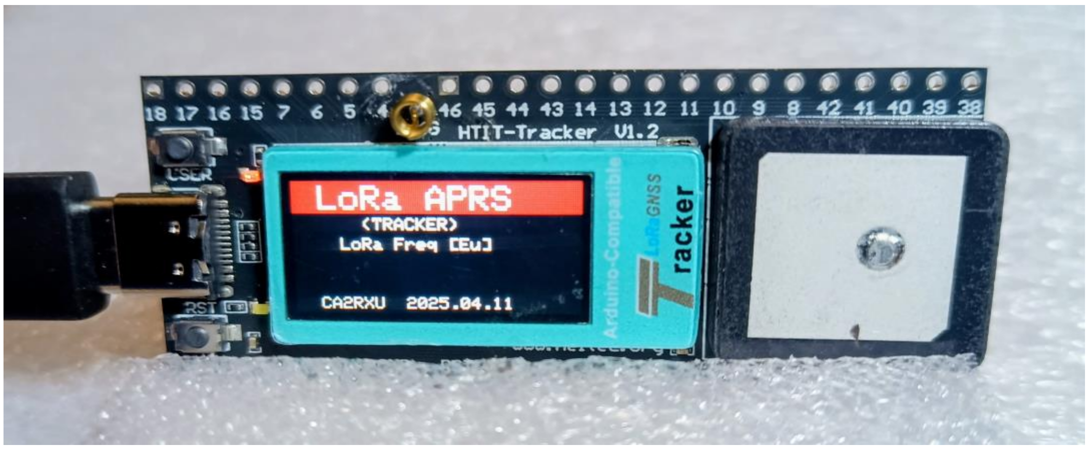
As the upload begins you will see output similar to that in the above screenshot being displayed in the TERMINAL window.

As firmware.bin is uploaded, the amount of information displayed in the TERMINAL window will be quite verbose; so just ignore it. If all goes well you should observer "SUCCESS" in green displayed a number of times in the TERMINAL window. It should look something like this:



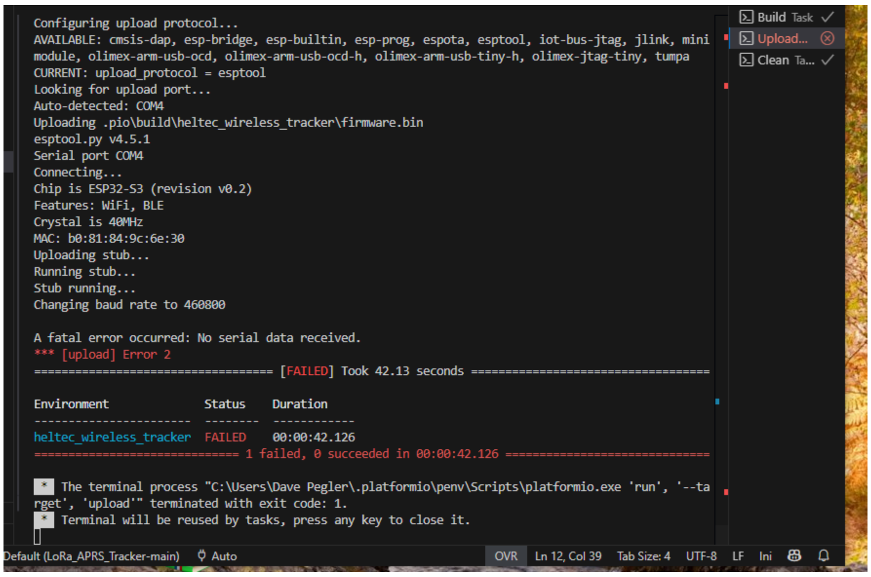
```
Writing at 0x0010111e... (68 %)
Writing at 0x00106128... (70 %)
Writing at 0x0010bbc6... (72 %)
Writing at 0x00110de8... (74 %)
Writing at 0x00115d2d... (75 %)
Writing at 0x0011ae32... (77 %)
Writing at 0x0011fef5... (79 %)
Writing at 0x001253b4... (81 %)
Writing at 0x0012a742... (82 %)
Writing at 0x001304d9... (84 %)
Writing at 0x00135bbd... (86 %)
Writing at 0x0013b72c... (87 %)
Writing at 0x00140ff4... (89 %)
Writing at 0x00149815... (91 %)
Writing at 0x001504c4... (93 %)
Writing at 0x00158103... (94 %)
Writing at 0x0015d13d... (96 %)
Writing at 0x00162b7d... (98 %)
Writing at 0x0016883c... (100 %)
Wrote 1415200 bytes (936278 compressed) at 0x00010000 in 9.4 seconds (effective 1199.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
================================= [SUCCESS] Took 33.80 seconds =================================

Environment               Status    Duration
------------------------  --------  ------------
heltec_wireless_tracker   SUCCESS   00:00:33.802
================================== 1 succeeded in 00:00:33.802 ==================================
 * Terminal will be reused by tasks, press any key to close it.
```

As soon as VSC has finished uploading firmware.bin and performed a hard reset, you should observer the Heltec Wireless Tracker board spring into life.



If all is well, it should display "LoRa APRS" on the 0.96" OLED, as shown in the image above.

If during the upload of firmware.bin you encountered errors (FAILED displayed in red like those shown below), it's most likely because VSC has failed to force the ESP32-S3FN8 microprocessor to revert back to bootloader mode.

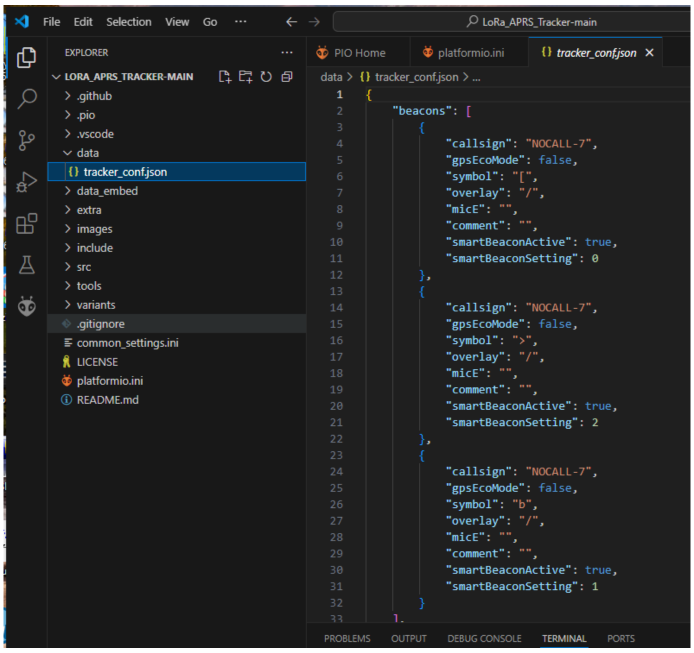


```
Configuring upload protocol...
AVAILABLE: cmsis-dap, esp-bridge, esp-builtin, esp-prog, espota, esptool, iot-bus-jtag, jlink, mini
module, olimex-arm-usb-ocd, olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa
CURRENT: upload_protocol = esptool
Looking for upload port...
Auto-detected: COM4
Uploading .pio\build\heltec_wireless_tracker\firmware.bin
esptool.py v4.5.1
Serial port COM4
Connecting...
Chip is ESP32-S3 (revision v0.2)
Features: WiFi, BLE
Crystal is 40MHz
MAC: b0:81:84:9c:6e:30
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800

A fatal error occurred: No serial data received.
*** [upload] Error 2
================================== [FAILED] Took 42.13 seconds ==================================

Environment               Status    Duration
------------------------  --------  ------------
heltec_wireless_tracker   FAILED    00:00:42.126
=========================== 1 failed, 0 succeeded in 00:00:42.126 ===========================

 *  The terminal process "C:\Users\Dave Pegler\.platformio\penv\Scripts\platformio.exe 'run', '--ta
rget', 'upload'" terminated with exit code: 1.
 *  Terminal will be reused by tasks, press any key to close it.
```

This situation is easily remedied by forcing the ESP32-S3FN8 microprocessor into bootloader mode using the two push buttons either side of the USB-C socket. With the USB socket facing down, place a thumb on each of the push buttons. Press and hold the left-hand button, then press and release the right-hand button. After about 1 second finally release the left-hand button. The ESP32-S3FN8 microprocessor will now be into bootloader mode, and you will hear Windows make all sorts of beeping noises as it detects the serial USB interface once again. With the ESP32-S3FN8 microprocessor now into bootloader mode, click on the white arrow (pointing right) on the bottom tool bar to perform the upload of firmware.bin once again.

14 Uploading data/tracker_conf.json file to Heltec Wireless Tracker board

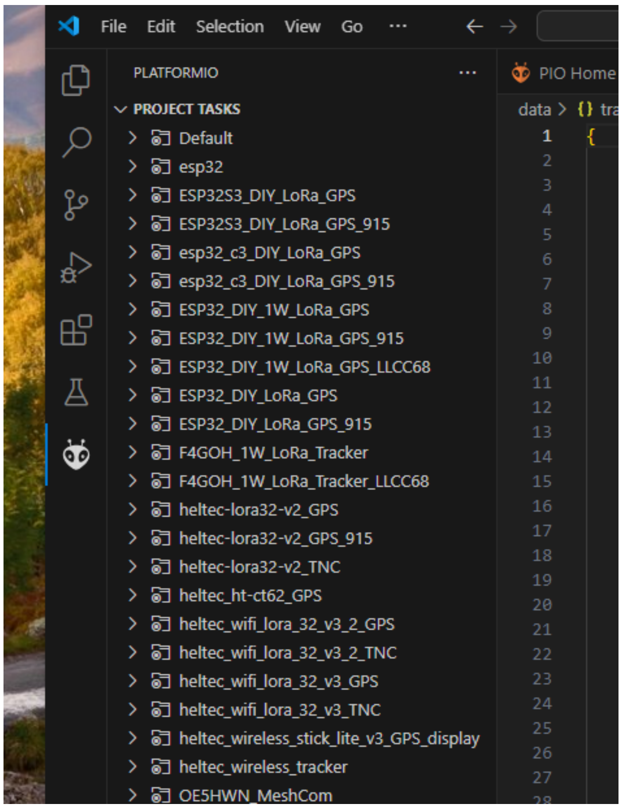
The LoRa_APRS_Tracker software uses a special Json configuration file called "tracker_conf.json" (which can be found in the "data" directory) to store some useful configuration information. When the LoRa APRS Tracker software was being developed, this file had to be edited to add information such as your call sign, APRS SSID, frequency and other preferences. This had to be done before it was uploaded. Nowadays you do not need to do this as the LoRa_APRS_Tracker software is now configured through a very nice web based GUI – which we will discuss a bit later. Even though the web interface is now the preferred way to configure the tracker, it is still possible to edit this file to configure the tracker software. Either way, the configuration file "tracker_conf.json" (clear or edited) needs to be uploaded to the Heltec Wireless Tracker board via the USB serial connection.



```
{
    "beacons": [
        {
            "callsign": "NOCALL-7",
            "gpsEcoMode": false,
            "symbol": "[",
            "overlay": "/",
            "micE": "",
            "comment": "",
            "smartBeaconActive": true,
            "smartBeaconSetting": 0
        },
        {
            "callsign": "NOCALL-7",
            "gpsEcoMode": false,
            "symbol": ">",
            "overlay": "/",
            "micE": "",
            "comment": "",
            "smartBeaconActive": true,
            "smartBeaconSetting": 2
        },
        {
            "callsign": "NOCALL-7",
            "gpsEcoMode": false,
            "symbol": "b",
            "overlay": "/",
            "micE": "",
            "comment": "",
            "smartBeaconActive": true,
            "smartBeaconSetting": 1
        }
    ],
```

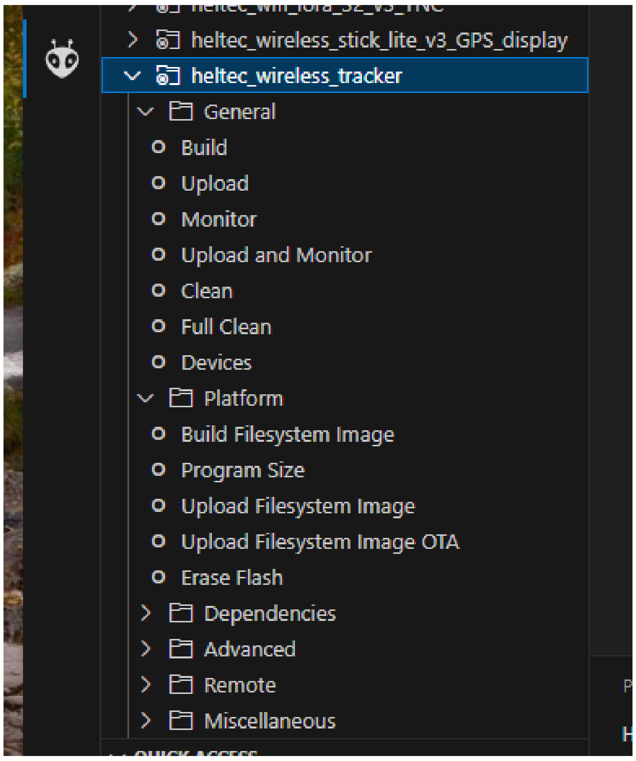
In the left-hand "EXPLORER" window, you will see all the files and directories that are used to build the software. If you navigate to the "data" directory in there you will find the file "tracker_conf.json" . If you click on that file it will be loaded up so you can explore it. Don't change anything at this stage.

To get the Tracker software up and running properly we need to upload this file to the flash memory on the Heltec Wireless Tracker board using a similar, but subtly different mechanism for uploading “firmware.bin”. On the left-hand side of VSC you will observe a white icon which looks like a Wasp or Alien. If you click on this icon a list of different types of “ESP32” and “heltec-lora“ boards should appear as show in the screenshot below:



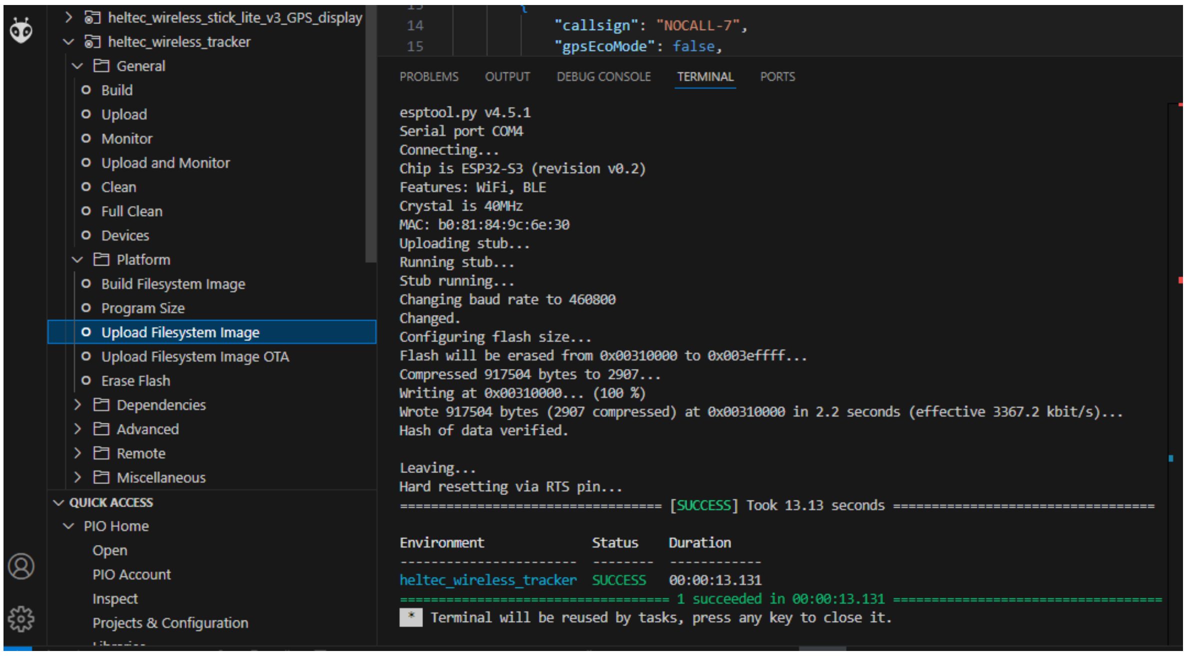
PROJECT TASKS
- Default
- esp32
- ESP32S3_DIY_LoRa_GPS
- ESP32S3_DIY_LoRa_GPS_915
- esp32_c3_DIY_LoRa_GPS
- esp32_c3_DIY_LoRa_GPS_915
- ESP32_DIY_1W_LoRa_GPS
- ESP32_DIY_1W_LoRa_GPS_915
- ESP32_DIY_1W_LoRa_GPS_LLCC68
- ESP32_DIY_LoRa_GPS
- ESP32_DIY_LoRa_GPS_915
- F4GOH_1W_LoRa_Tracker
- F4GOH_1W_LoRa_Tracker_LLCC68
- heltec-lora32-v2_GPS
- heltec-lora32-v2_GPS_915
- heltec-lora32-v2_TNC
- heltec_ht-ct62_GPS
- heltec_wifi_lora_32_v3_2_GPS
- heltec_wifi_lora_32_v3_2_TNC
- heltec_wifi_lora_32_v3_GPS
- heltec_wifi_lora_32_v3_TNC
- heltec_wireless_stick_lite_v3_GPS_display
- heltec_wireless_tracker
- OE5HWN_MeshCom

In the list that appears, there should be a directory called “heltec_wireless_tracker”. Click on that and you should see the following list of tasks appear under this board type:



- heltec_wireless_stick_lite_v3_GPS_display
- heltec_wireless_tracker
  - General
    - Build
    - Upload
    - Monitor
    - Upload and Monitor
    - Clean
    - Full Clean
    - Devices
  - Platform
    - Build Filesystem Image
    - Program Size
    - Upload Filesystem Image
    - Upload Filesystem Image OTA
    - Erase Flash
  - Dependencies
  - Advanced
  - Remote
  - Miscellaneous

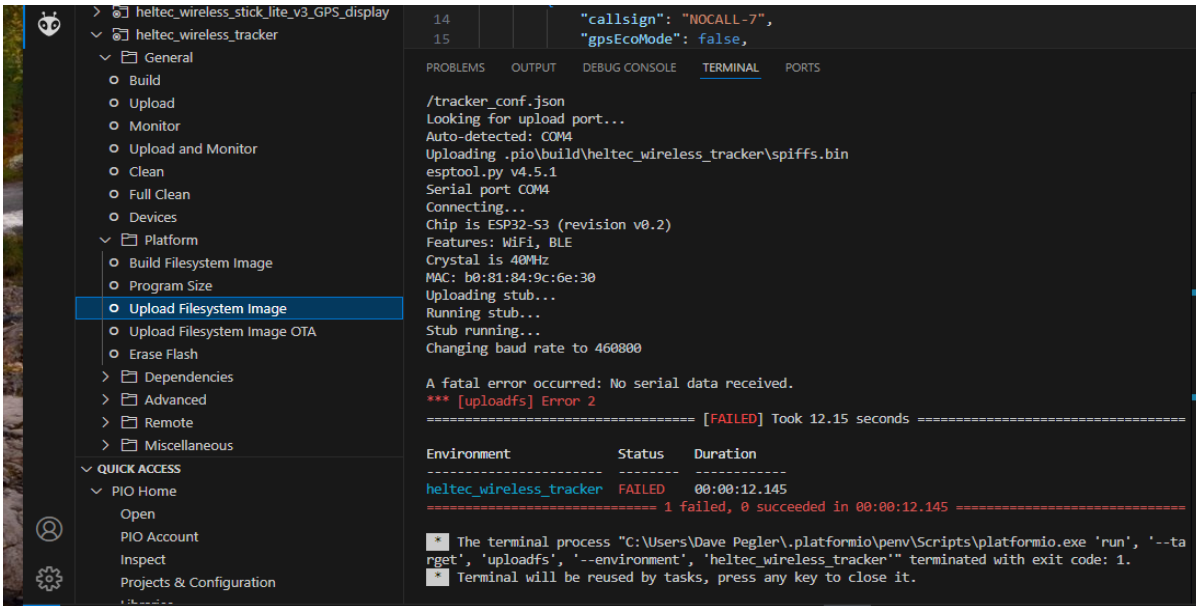
With the Heltec Wireless Tracker board still connected to your Windows PC (via USB), select "Upload Filesystem Image" and then observer the debug output in the TERMINAL window.
You should observe VSC converting "tracker_conf.json" to a format for uploading (a file called spiffs.bin) and then actually uploading it. If all goes well you should see "SUCCESS" displayed in green multiple times in the TERMINAL window, as shown below:



```
14          "callsign": "NOCALL-7",
15          "gpsEcoMode": false,
```

```
esptool.py v4.5.1
Serial port COM4
Connecting...
Chip is ESP32-S3 (revision v0.2)
Features: WiFi, BLE
Crystal is 40MHz
MAC: b0:81:84:9c:6e:30
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00310000 to 0x003effff...
Compressed 917504 bytes to 2907...
Writing at 0x00310000... (100 %)
Wrote 917504 bytes (2907 compressed) at 0x00310000 in 2.2 seconds (effective 3367.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
================================ [SUCCESS] Took 13.13 seconds ================================

Environment             Status    Duration
----------------------  --------  ------------
heltec_wireless_tracker SUCCESS   00:00:13.131
================================ 1 succeeded in 00:00:13.131 ================================
 *  Terminal will be reused by tasks, press any key to close it.
```

Sadly, in my experience this process can be a little unreliable, and you may observe errors ("FAILED" in red), like those shown below:



```
14          "callsign": "NOCALL-7",
15          "gpsEcoMode": false,
```

```
/tracker_conf.json
Looking for upload port...
Auto-detected: COM4
Uploading .pio\build\heltec_wireless_tracker\spiffs.bin
esptool.py v4.5.1
Serial port COM4
Connecting...
Chip is ESP32-S3 (revision v0.2)
Features: WiFi, BLE
Crystal is 40MHz
MAC: b0:81:84:9c:6e:30
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800

A fatal error occurred: No serial data received.
*** [uploadfs] Error 2
=================================== [FAILED] Took 12.15 seconds ===================================

Environment             Status    Duration
----------------------  --------  ------------
heltec_wireless_tracker FAILED    00:00:12.145
============================= 1 failed, 0 succeeded in 00:00:12.145 =============================

 *  The terminal process "C:\Users\Dave Pegler\.platformio\penv\Scripts\platformio.exe 'run', '--ta
rget', 'uploadfs', '--environment', 'heltec_wireless_tracker'" terminated with exit code: 1.
 *  Terminal will be reused by tasks, press any key to close it.
```

To remedy this, you will need to force the ESP32-S3FN8 microprocessor into bootloader mode once again. As we described above, this is easily done using the two push button either side of the USB-C socket. With the USB socket facing down, place a thumb on each of the buttons. Press and hold the left-hand button, then press and release the right-hand button. After about 1 second finally release the left-hand button. Windows will make all sorts of beeping noises as it detects the serial USB interface once again. With the ESP32-S3FN8 microprocessor now into bootloader model (note that in bootloader mode, the OLED screen goes blank) try again to upload the configuration file.

15 First boot LoRa APRS Tracker Configuration

After you have successfully upload “firwware.bin” and “tracker_conf.json” to the Heltec Wireless Tracker board, it’s now time to configure it. In previous releases this had to be done be editing “tracker_conf.json” prior to uploading via USB. In the latest releases this is now done through a Web page running on the Wireless Tracker which is accessible from a smartphone over WiFi. In the software this is referred as “Config WiFi AP mode”

To configure the software running on your Heltec Wireless Tracker, using the “Config WiFi AP mode” use your phone to connect to the WiFi Access Point running on the tracker called:

LoRaTracker-AP

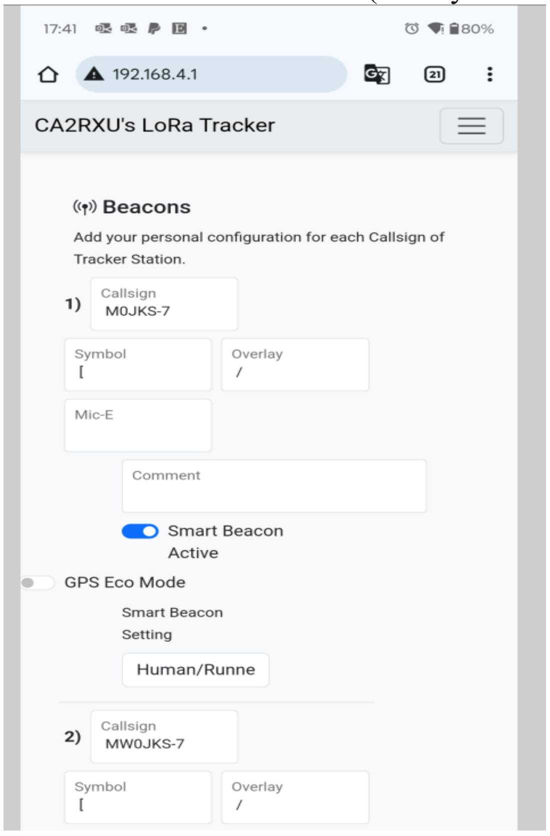
When prompted, enter the following WiFi passcode:

“1234567890”

When the Wi-Fi is connected, open up a web browser on your phone and enter the following URL:

http://192.168.4.1

My Motorola (Android) phone tends to get confused at this point as it is way too clever for it’s own good. It works out that it cannot access the internet via the Wi-Fi AP “**LoRaTracker-AP**” on the tracker, so automatically switches to a WiFi AP it knows is connected to the internet (i.e. my home WiFi router). If you encounter this issue, you may need to tell you phone to forget about your home WiFi connection (so it cannot automatically connect), or configure the tracker out of the range of your home WiFi.


17:41 80%
192.168.4.1
CA2RXU's LoRa Tracker
Beacons
Add your personal configuration for each Callsign of Tracker Station.
1) Callsign: M0JKS-7
Symbol: [
Overlay: /
Mic-E
Comment
Smart Beacon Active
GPS Eco Mode
Smart Beacon Setting: Human/Runne
2) Callsign: MW0JKS-7
Symbol: [
Overlay: /

If all goes correctly, you should be presented with a web page on IP address 192.168.4.1. It should be similar to that shown in the screenshot to the right. You will then be able to change the call signs you want to use and any other options you see fit.

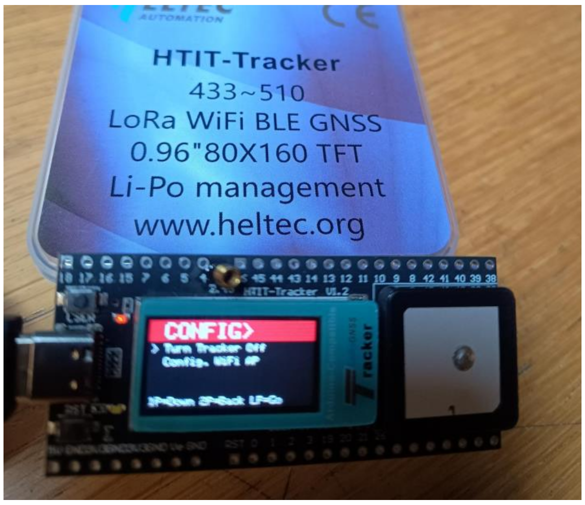
Once you are happy with the configuration options you have changed, click on the three parallel bars at the top right-hand corner (next to “CA2RXU’s LoRa Tracker”) and you should be presented with a big green “Save” button.

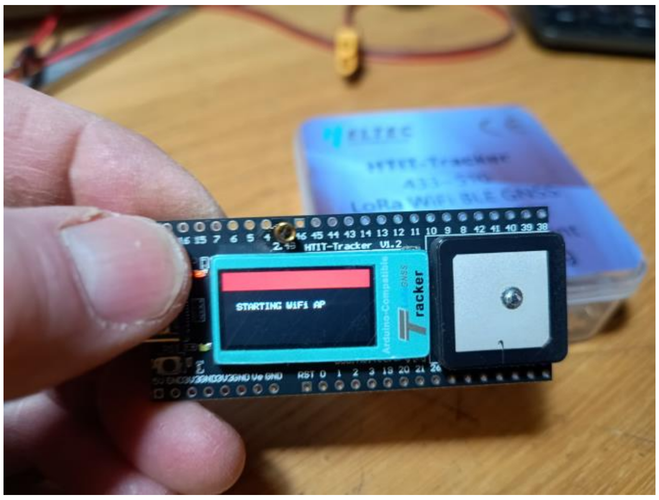
When you click on the green “Save” button, this will save your changes to the internal flash memory and force the Heltec Wireless Tracker board to reboot. When it comes back up it should now display the updated call signs, or whatever else you changed.

16 Tweaking the SOTA LoRa APRS Configuration

The first boot configuration mode described above only runs on the Heltec Wireless Tracker board if it has never been configured. If you have successfully configured the Heltec Wireless Tracker but then want to change call sign or SSID, the easiest way to do this is to force the tracker back into "Config WiFi AP mode ". You can then easily connect to it from a web browser on your smartphone via WiFi to reconfigure it.

To do this press the left-hand button on the Heltec Wireless Tracker board three times in succession. The display will switch to "CONFIG" mode as shown in the picture below (left). Next short press the left-hand button so that the ">" character moves down next to "Config WiFi AP". Then long press





the left-hand button for about 2 seconds and you should see "STARTING WIFI AP" shown on the display. This is show in the picture above (right). At this point the Heltec Wireless Tracker board should reboot and come back up in "Config WiFi AP mode". Please refer to section 15 on how to reconfigure.

17 Recommended RSGB HAB LoRa APRS Tracker Hardware

For the RSGB HAB competition we recommend a cheap LoRa (EU433) SBC called the "Heltec LoRa Wireless Tracker". This board is actually a LoRa development PCB designed by Chinese company Heltec Automation, which consists of a ESP32-S3FN8 microprocessor, Semtech SX1262 LoRa modem, 0.96" OLED display, WiFi, Bluetooth and a UC6580 dual-frequency multi-constellation GNSS SoC which supports GPS, GLONASS, BDS, Galileo, NAVIC and QZSS.

The board can be purchased directly from Heltec Automation.

Heltec Automation Direct

Remember to select the 470-510MHz (EU433) version

or from their shop on AliExpress.

Heltec Automation AliExpress Store

Again, remember to select the 470-510MHz (EU433) version

You can also buy it from the SOTA shop

SOTA

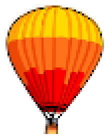
18 Tracking RSGB HAB LoRa APRS Digipeater

The RSGB HAB LoRa APRS competition will begin when Director Ben Lloyd, GW4BML launches the high-altitude balloon from Welshpool in Powys on the morning of Saturday 20 September at 10 o'clock. It will contain a cross-frequency LoRa APRS Digipeater payload on-board and is expected to be airborne for around two to three hours; reaching a maximum height of 90,000 feet.

The LoRa APRS cross-frequency Digipeater on board the HAB will be configured to beacon its location every minute or so on a special downlink frequency, which will be received by a number of strategically located ground stations (e.g. LTE/4G iGates). These iGates, situated on mountain tops, will then relay the beacon packets from the HAB to the APRS Internet Service (APRS-IS). This is so participants can track its location using any one of the APRS tracking websites now available:

LoRa APRS Live Map aprs.fi

For the duration of the flight, the HAB Digipeater will use the special call sign **GB1HAB** and this will appear on the various APRS mapping sites as the APRS hot air balloon symbol "/O"



19 Opening up the RSGB HAB Digipeater

To open up the LoRa APRS Digipeater on the RSGB HAB, a LoRa Tracker must be configured to transmit on frequency 439.850MHz, 125KHz bandwidth, Spreading Factor (SF) 12, and Coding Rate (CR) 5. As outlined in section 8, these are the default settings in the RSGB HAB Tracker software. By default, the software is also configured to send out a APRS location packet (or ping) whenever there is a significant change in location of the Tracker (i.e. walking, cycling, running or driving). Of course, this will only occur if the UC6580 dual-frequency multi-constellation GNSS receiver on the Heltec Wireless Tracker board can see a sufficient number of satellites to accurately determine it's location. It must therefore be used outside and will normally only transmit a change of location packet when actually moving. To override this, as long as the 0.96" OLED display is showing the Latitude and Longtitude of the Tracker (i.e. it has satellite lock) pressing the left-hand button on the Heltec Wireless Tracker board can be use to override this and force a location packet to be transmitted over RF. When the RSGB HAB is airborne, one can use this mechanism to try and open up the LoRa APRS Digipeater, and use the various websites outlined in section 18 above to observe if your Trackers was successful in opening up the HAB Digipeater, or not.

20 Enclosures for the Heltec Wireless Tracker board

As one might expect, on its own the Heltec Wireless Tracker board is not really designed to be used outside. It should therefore be housed in a rugged enclosure with a battery to power it. If you have a 3D printer, there are numerous designs on thingiverse.com and printables.com for the Heltec Wireless Tracker board; many of which also provide housing for a Li-Ion battery (e.g. 18650).

Below are a couple of examples:

Heltec Wireless Tracker Case #1

Heltec Wireless Tracker Case #2

Heltec Wireless Tracker Case #3

Heltec Wireless Tracker Case #4

If you do not have access to a 3D printer, the following case designed specifically for the Heltec Wireless Tracker board (including 2500mAH Li-Ion battery) is available from the SOTA shop:

SOTA LoRa APRS Tracker Case For Heltec Wireless Tracker board

There are also many case designs for the Heltec Wireless Tracker board, some with, some without batteries on both eBay and Etsy:

The ROOK for Heltec Wireless Tracker

Heltec Wireless Tracker Case and 2000mAH battery

Tracker case & Battery fits Heltec Wireless Tracker

Heltec LoRa Wireless Tracker Case

Heltec Wireless Tracker case and Battery