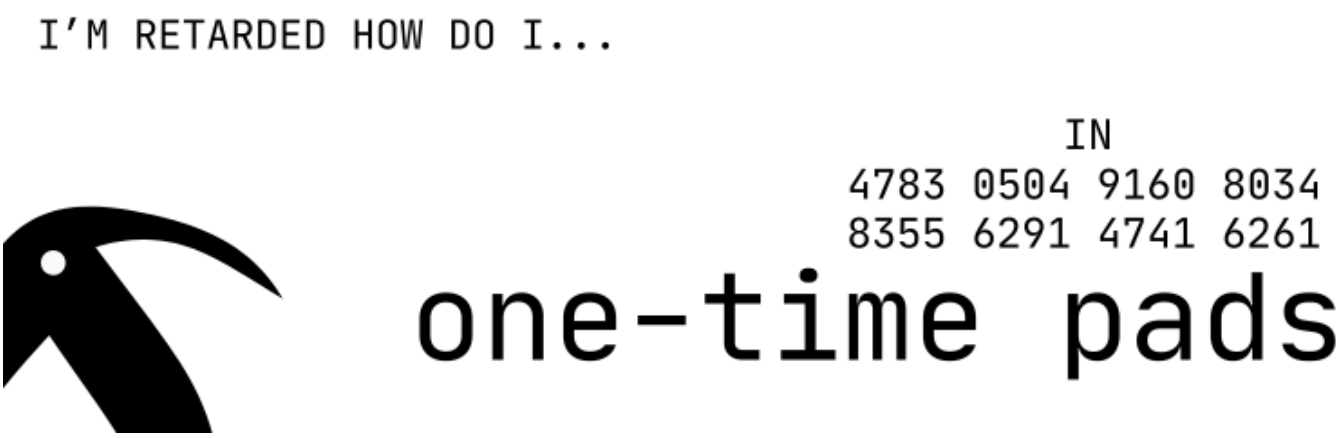


By the end of this guide you will know how to:

- Use a simple but effective form of encryption that is unbreakable when used correctly
- How to plan a contact
- Understand the basics of HF propagation
- Understand digital modes like JS8call
- Combine them to reach someone almost anywhere on earth

Table of Contents

---

1. Intro

This is a guide for using the one-time pad method as a means of secure, low tech communication.

The one-time pad (OTP) is a technique for unbreakable encryption ***when it is used correctly***.

The plaintext (message we want to send) can't be reversed from the ciphertext (our encrypted message) without the key being known.

There is no form of cryptanalysis or method to reverse the ciphertext to the plaintext, even with infinite time and computational power. It's the only form of encryption that has perfect secrecy.

One-time pads can be created with something as simple as a pen, paper, and set of 10 sided die. Only basic mental math of addition and subtraction is required for encrypting and decrypting messages using the OTP.

This guide will also provide an example of the full lifecycle of a OTP encrypted message, covering both transmission and reception on HF using the JS8call software.

2. The OTP

The one-time pad technique is an evolution of the Vernam cipher. If you've ever listened to a recording of a shortwave number station where a string of numbers or letters is read out loud, chances are they're using a one time pad. Some stations like Poland's E11 still use this technique to this day.

A one-time pad is composed of multiple keys of randomly generated digits. You can think of the keys in the one-time pad as "noise" we combine with our message.

The basic process of both encryption and decryption is converting the characters of your message into digits using a checkerboard (map of characters to digits), and combining them with the randomly generated key digits either subtracting or adding, with modulo 10.

"Modulo 10" for our purposes just means looking at the last digit. So for addition if we added 9 + 7 together, 16 becomes 6. We look at the last digit and that's our result.

And for decryption, subtraction of the key digits from the encrypted digits with modulo 10. Then converting the resulting numbers back into characters with our checkerboard to get our message.

The process is very simple and only uses mental math. The next three sections will also go into more detail of this process.

Below is an example of a one-time pad. Each new line represents a new key in the OTP:

Code:

```
                    IN 001

 43652    26456    29667    06821    32751    29755
```

```
37218    51925    69763    56023    54157    63547
60422    84583    78052    13125    99741    59842
```

The one time pad can be written on anything, like a piece of paper, index card, or notebook.

Both the sender and receiver must have an identical copy of the same one-time pad. It's best practice to label the sender's copy OUT, and the recipients copy IN, followed by the sequence if there are multiple one-time pads created.

The length of the keys in the one-time pad can be as long or as short as needed. There can be as many keys as required for however many messages are intending to be sent. It's best however to keep messages simple and set a maximum limit of 75 characters used per one-time pad (which includes all key material)

The main disadvantage to the OTP is the logistics of ensuring that both parties can securely maintain and exchange a copy of the key. OTP communications require coordination of when the message will be sent, and when one time pads will be exchanged.

The one-time pad is single use
It must be destroyed after it's used.

Both the sender and receiver must do this. The integrity, and perfect secrecy of the one-time pad communications hinges entirely on this factor. Destroying it physically also prevents accidental re-use of the same one time pad.

3. Generating the OTP

To generate a one-time pad, we need a good source of randomness.

Computers are bad at generating random numbers, and awful at security. While it may appear "random", numbers generated by computers are deterministic, generated by an algorithm, and not truly random.

As a general rule of thumb, the plaintext (message you want to send) and key itself should NEVER touch a computer.
Do not use online number generators. Do not use online "one time pad generators".
Do not use software applications, mobile software, /dev/random, etc.

Nothing that isn't our final encrypted message should ever be put on a computer, phone, or electronic device.

A set of 5 D10 die, like the one pictured below, serves as a good enough source of unpredictability for generating our random digits.

To generate the keys for our one time pad, toss 5 at a time and write the digits down.

Read them in the order they landed from left to right. Do not pick the order. **Write them as they landed.**

Repeat this for as many groups as we want to have. The one time pad should use keys that are all of the same group length, for as long or longer than the messages we want to send.

Once we're finished we'll have something like this:

Code:

```
06356 13440 97398 07987 56769 59181
```

We rolled our set of 5 ten sided die 6 times to get 6 groups of random numbers. Each group needs to be the same length.

We can repeat this process for as many keys as we would like and add them as a new line.

The first group in our one time pad is always the key ID. The key ID is a unique identifier that allows the recipient to know which key from their one-time pad to use for decryption. This number isn't used in the decryption or encryption process itself, but tells the recipient which key to use for each message that we send.

Code:

```
[KEY ID] 13440 97398 07987 56769 59181
```

It's important that the key ID is generated using the same random process as the rest of the key. Do not pick a "memorable" number, do not use the date or increment sequentially.

After we've generated our random digit groups, we label our copy of the pad OUT, and the recipients copy IN. This is to prevent accidental re-use of the same key.

This is what a completed example of our one-time pad might look like, with three keys:

Code:

```
              OUT 001

 06356  13440  97398  07987  56769   59181
 74829  30516  82647  19053  47281   65934
 28195  64073  91826  50374  18629   73045
```

It may be tedious, but do not copy the key electronically.

Do not use a phone to take a picture of it and print it out.
Do not take a photo of the one time pad and sent it to someone.
Once again the key or our unencrypted message should never touch an electronic device.

Once we have both an identical IN and OUT pair, our one time pad is now complete and we are ready to use it.

4. The checkerboard

The checkerboard is what enables the conversion of characters into digits that we combine with the keys of our one time pad.

Checkerboards are ideal because they allow for fractonation and compression. That means in simple terms, the most common letters in the English language (like E, T, A, O, I, N) get a single digit, while less common letters and numbers get two digits. This keeps our encrypted messages shorter than they would be otherwise. They are not used in the encryption process itself, but for encoding and decoding letters and numbers.

Both the sender and receiver must use the same checkerboard for the message to be decodable.

The most well known checkerboard is the straddling checkerboard and can be remembered by the mnemonic "AT ONE SIR"

	0	1	2	3	4	5	6	7	8	9
	A	T		O	N	E		S	I	R
2	B	C	D	F	G	H	J	K	L	M
6	P	Q	U	V	W	X	Y	Z	--	//

In this checkerboard:
A => 0
T => 1
O => 3
B => 20
V = > 63

and so on. You may notice that the checkerboard has no numbers. This is where the figure shift character // character (69) comes in. When we see 69 (//), we shift from letters to literal numbers. And shift back after we see another //.

Spaces should be omitted where the message is readable without them.

An example of the conversion of FARMERCHUCK using the checkerboard is:

F	A	R	M	E	R	C	H	U	C	K
23	0	9	29	5	9	21	25	62	21	27

```
23 0 9 29 5 9 21 25 62 21 27
```

and another example. Let's say we were planning a meeting at Farmer Chuck's at 18:45z. We need to use the figure shift to write the time.

When we use the figure shift to write numbers, we write each number twice.
So `1845` becomes `11 88 44 55`

F	A	R	M	E	R	C	H	U	C	K	S	A	T	//	1	8	4	5	//	z
23	0	9	29	5	9	21	25	62	21	27	7	0	1	69	11	88	44	55	69	67

```
23 0 9 29 5 9 21 25 62 21 27 7 0 1 69 11 88 44 55 69 67
```

For decoding, each number starting with **2** or **6** in the AT ONE SIR table means we use two digits for decoding the letter.

There are also additional checkerboards we can use, like the CT-37 table. The CT-37 table uses the mnemonic ESTONIA.

E 0	S 1	T 2	O 3	N 4	I 5	A 6	CT - 37		
B 70	C 71	D 72	F 73	G 74	H 75	J 76	K 77	L 78	M 79
P 80	Q 81	R 82	U 83	V 84	W 85	X 86	Y 87	Z 88	FIG 89
SPACE 90	(.) 91	(,) 92	(') 93	(?) 94	(/) 95	(+) 96	(-) 97	(=) 98	CODE 99

(Image from: https://www.ciphermachinesandcryptology.com/en/table.htm | archive)

The CT-37 table lets us make use of a **codebook**. It also provides mapping for punctuation. In this table, much like the // figure shift in the previous checkerboard, we use **89** to shift from letters to numbers, and 99 to use the codebook.

A codebook is a shorthand for a group of phrases that are used frequently. Instead of spelling out a long phrase character by character, we assign it a numeric code. This compresses the message further and reduces the amount of key material we burn through.

Both sender and receiver must have an identical copy of the codebook, just like the one-time pad and checkerboard. The codebook itself should also be kept secret, but can be kept after the one-time pad is destroyed.

A small codebook might look like this:

```
CODEBOOK 001
00    ARRIVED SAFELY
01    NEED RESUPPLY
02    MEETING CONFIRMED
03    MEETING CANCELLED
04    LOCATION COMPROMISED
05    STANDING BY
06    ABORT
07    ACKNOWLEDGE
08    WILL TRY AGAIN
09    NO CONTACT
```

The codebook allows us to write "NEED RESUPPLY AT CHUCKS" as:

CODE	01 NEED RESUPPLY	CODE	A	T	C	H	U	C	K	S
99	01	99	6	2	71	75	83	71	77	1

saving 15 full digits.

As with 2 and 6 in the AT ONE SIR table, the ESTONIA CT-37 table marks two digit sequences with **7 or 8.** Numbers after the figure shift are also copied twice.
There are also a few other checkerboard variants. But for sake of simplicity, I recommend choosing either the AT ONE SIR or ESTONIA CT-37 checkerboard.

5. Encrypting a message

Let's encrypt the message `FARMERCHUCKSAT1845z` using key 001 from our pad earlier. For the encoding and decoding segments, we will use the simpler AT ONE SIR checkerboard.

Our first step is to convert the plaintext to digits using the checkerboard:

F	A	R	M	E	R	C	H	U	C	K	S	A	T	//	1	8	4	5	//	z
23	0	9	29	5	9	21	25	62	21	27	7	0	1	69	11	88	44	55	69	67

When we are doing our conversion, write each digit in groups of 5 like our pad:

```
23092 95921 25622 12770 16911 88445 56967
```

With the message converted using the checkerboard, it's now time to combine it with our keys from our one-time pad to get our encrypted message.

Our pad from earlier is

Code:

```
             OUT 001
06356     13440   97398    07987    56769   59181
74829     30516   82647    19053    47281   65934
28195     64073   91826    50374    18629   73045
```

Put our plaintext digits with below the key of the one time pad, **ignoring the key IDs** (the first 5 digit group)

Our message here needs to use two key groups. I've made it purposely obtuse just to show how to handle messages that exceed one key's worth of material. Normally you should size your keys to comfortably contain a full message.

The process of combining the key is extremely simple. Take the first digit of the plaintext and add it to the key using modulo 10 addition. Modulo 10 addition just means that we only look at the last number.

So 1 + 2 is 3, but 9 + 9 (18) is 8. And then we work our way from left to right writing down each number until we're complete.

```
06356 13440 97398 07987 56769 59181 74829 30516 82647 | key
[KEY] 23092 95921 25622 12770 16911 [KEY] 88445 56967 | plaintext
-----------------------------------------------------------------+
06356 36432 82219 22509 68439 65092 74829 18951 38504
```

Our final encrypted message is:

```
06356 36432 82219 22509 68439 65092 74829  18951 38504
```

6. Decrypting a message

To decrypt, we combine the key with the ciphertext, but this time using subtraction mod 10 instead of addition. Write the received encrypted message **above** the key for decryption.

Modulo 10 subtraction just means if the ciphertext digit is smaller than the key digit, we borrow 10.

So 3 - 1 = 2, but 0 -7 becomes 10 - 7 = 3. We just pretend the ciphertext digit is 10 higher when we need to.

```
06356 36432 82219 22509 68439 65092 74829 18951 38504 | ciphertext
06356 13440 97398 07987 56769 59181 74829 30516 82647 | key
-----------------------------------------------------------------
[KEY] 23092 95921 25622 12770 16911 [KEY] 88445 56967
```

This gives us back our plaintext: `

`23092 95921 25622 12770 16911 88445 56967`

We can then use our AT ONE SIR checkerboard to convert it back

which gives us our decrypted message:

```
FARMERCHUCKS AT(figure shift) 1845 (figure shift)z
```

7. Tips & Planning Communications

To reiterate above before we continue:

The one-time pad is single use.
It must be physically destroyed after it's used, even if there is material left over.

Never use a computer to store or generate the one-time pad.
Keep messages short and concise. Omit spacing where it's not needed and the recipient can infer them. A 75 character limit should be the upper bound for messages per each pad.

The disadvantage to the one-time pad method is the requirement the planning overhead, and ensuring that both parties can maintain a secure copy of the one-time pad.

All key material, codebooks, and schedules must be exchanged **in person**.

One-time pad communications can be bidirectional, with both parties having a IN pad, and OUT pad. That means two parties would be able to communicate with eachother 1 to 1 if both had a IN and OUT pair.

While possible, it's not recommended to give multiple recipients the same copy of an IN pad.

There are two patterns of communication that can be used for one-time pad messaging: **asynchronous** and **synchronous**

Synchronous means that the sender and receiver coordinate an exact window of time to broadcast and receive. This is pretty much how every classic shortwave spy number station operates. It usually requires a fixed schedule that is known to both parties beforehand.

Asynchronous could mean something as simple as leaving a sticky note with your encrypted message, or sending it over an email. It doesn't involve a fixed window of time.

The example in the next section will look at a synchronous one-way communication using HF radio.

HF radio is a vital tool when we want to send messages across long distances where our recipient is *beyond line of sight*. [todo]

8. Transmission

This section will cover an example of transmission of a one-time pad encrypted message using the open source software **JS8call**

JS8call takes data, such as text, and modulates it into sound. That sound is then transmitted over the air from a radio connected to a computer running the software.

Voice (sometimes called phone or SSB) takes more bandwidth and a much stronger signal to be heard, while JS8call works with even poor propagation conditions and below the noise floor. The only disadvantage is that both parties need to have a computer running the software able to decode JS8 messages.

To begin we, first need to understand our setup and threat model.

Any radio signal by nature of physics can be located.

For sustained communications using this method it's ideal to operate from a portable setup that can be deployed and torn down.

This can be anywhere from the top of a mountain on a trail, side of the road, or a public park.

The importance of portable operations can't be understated. If you are at the stage where one-time pad communication is necessary it's likely that doing so comes at a great risk.

Understanding the basics of HF radio will give you a leading edge, especially the basics of portable HF antennas. A vast amount of knowledge is available online for that topic, and a future guide will cover this as well.

Having a radio license for your jurisdiction with HF privileges gives you plausible deniability for both owning and using HF equipment. JS8call is frequently used in amateur radio communications as a part of the hobby.

The first step to HF communications is **propagation** planning.

Propagation planning

HF radio enables our signals to travel long distances because it uses the ionosphere. The ionosphere is a layer of particles in the upper atmosphere that physically carries our signal across. This is referred to as propagation.

We need to know the general location of where our recipient is before communications can take place.

This is imperative, as we need to plan around HF propagation conditions that allow us to reach them. The time of day we broadcast at and frequencies we use will be determined by this and are the two variables we are confined by.

Segments of the HF radio frequencies are referred to by **bands**. An example is 40 meters (around 7 MHz) or 20 meters (around 14 MHz). Each band has its own unique propagation characteristics.

The "meters" refers to the wavelength of the signal. Wavelength and frequency are just two ways of describing the same wave, and they move in opposite directions. The longer the wavelength, the lower the frequency. So 40 meters is a longer, lower frequency wave than 20 meters.

Each band propagates differently, and how a band behaves is tied closely to the time of day. The sun charges the ionosphere, so the range of usable frequencies rises during daylight and falls after dark.

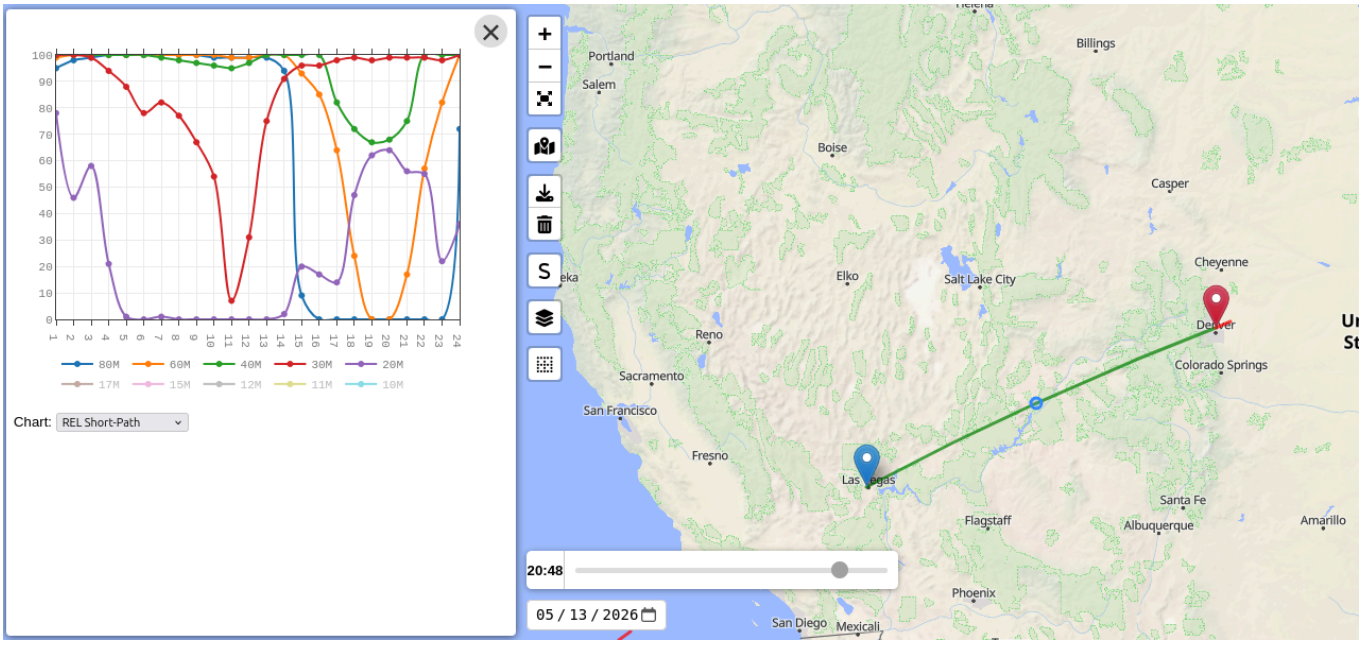
As a general rule of thumb, the lower bands (longer wavelengths like 80 and 40 meters) tend to perform best at night, while the higher bands (shorter wavelengths like 20 meters and 17 meters) tend to open up during the day.

This is why our two variables, time of day and frequency can't be planned separately. The band we choose also depends on how far away our recipient is.

While there is no tool to predict propagation with 100% accuracy, we can plan based on established patterns and use prediction tools.

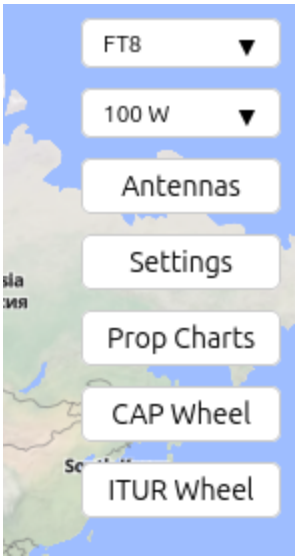
If they are regional at a distance of 0 - 500 kilometers [300 miles], it's ideal to use NVIS propagation and plan around the fof2 for the time of day for your latitude. NVIS (Near Vertical Incidence Skywave) is best suited for regional communications, while traditional propagation methods are better for longer distances outside of that.

If they are beyond that distance, use VOACAP with *general* (province, state, city) estimates of where the recipient will be. Below is an example of what our propagation chart might look like for a transmission from Las Vegas to Denver



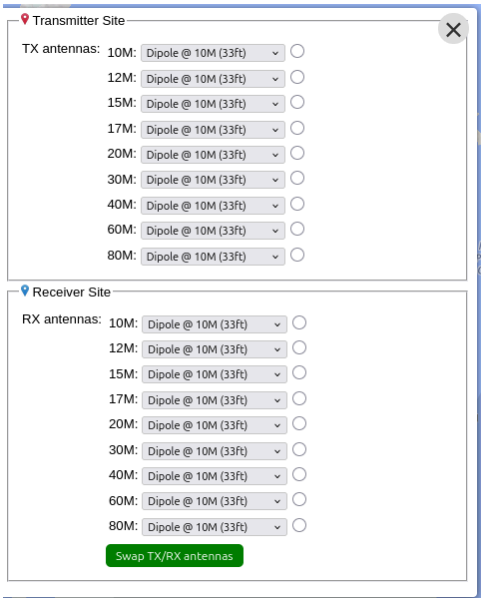
To use VOACAP for propagation prediction, drag the blue RX pin to the *general* location of your recipient, and the red TX pin to the *general* location of where you will be transmitting from. Precision is not necessary here. A neighboring city with a high population will work just as well as your exact grid marker.

For our case with JS8call, set the mode on the right to FT8 and our power to whatever power we will be transmitting with.



Next, go into **Antennas**

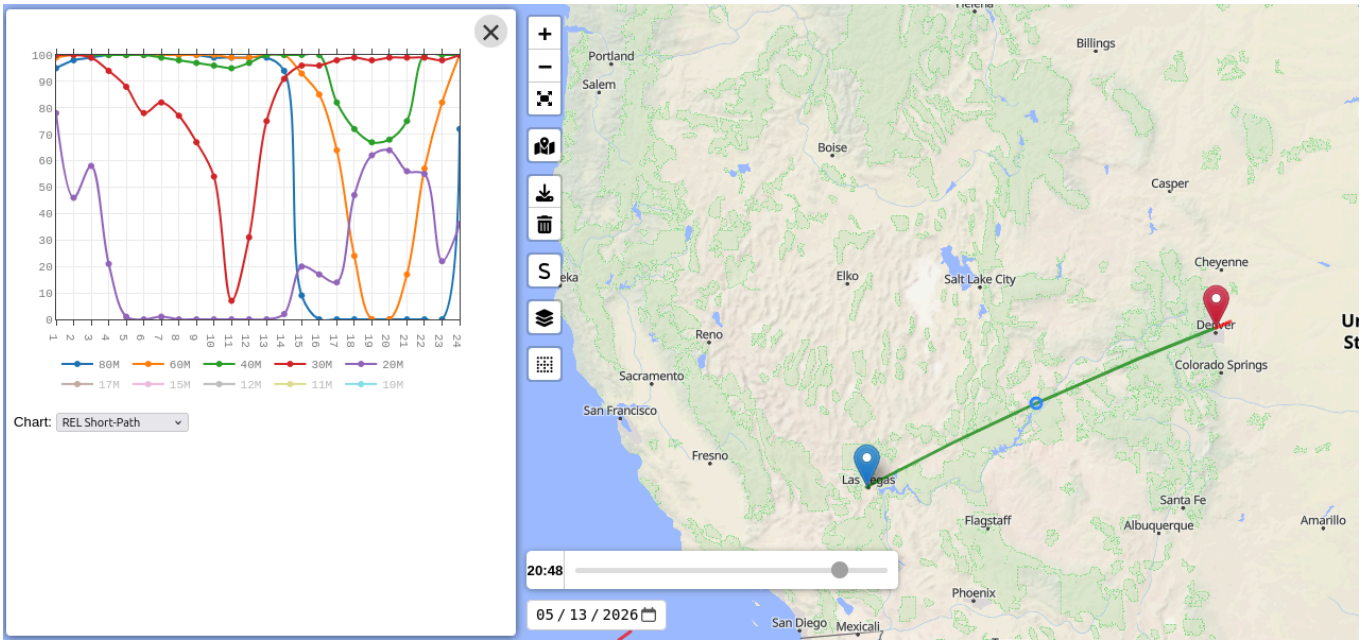
For a rough estimate set all of them to Dipole @ 10 Meters from the dropdown. You can click the circle icon on the right of the dropdown to set all for TX. Then go down to RX and do the same thing.



Once we have VOACAP all setup and your markers are in place, hit "Prop Charts".

The default chart selected will be "REL Short Path". This is the chart we want to look at.

The chart on the left shows the fraction of days in that month, at that hour and on that band, when the received signal will be good enough to work.



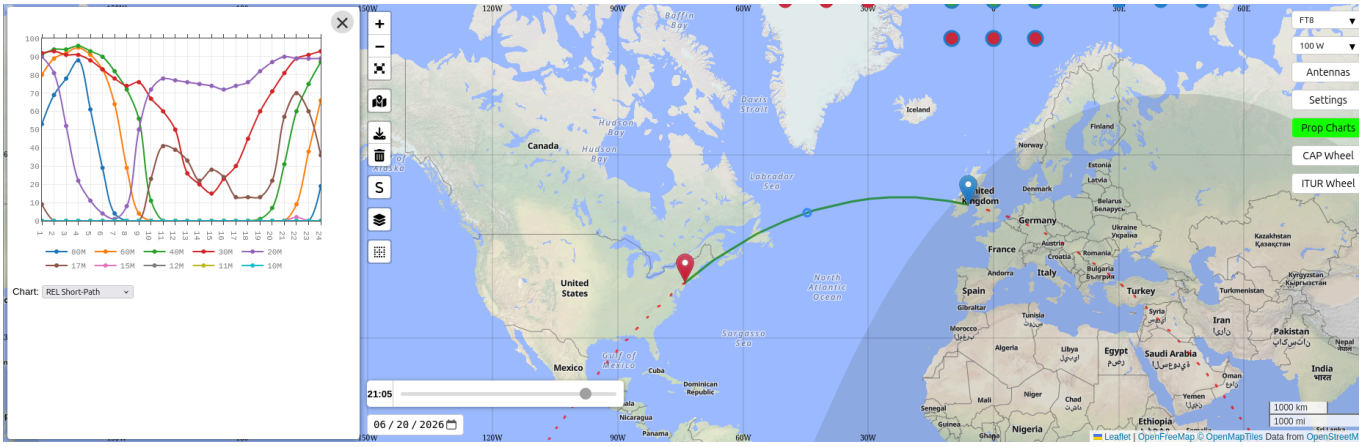
All times are shown in UTC. Each band is shown as a line in a different color. We can see from our chart that 40 meters in green (7.0 MHz - 7.3 MHz), 60 meters (5.2 - 5.4 MHz), and 30 meters (10.100 MHz - 10.150 MHz) seem to be the most reliable.

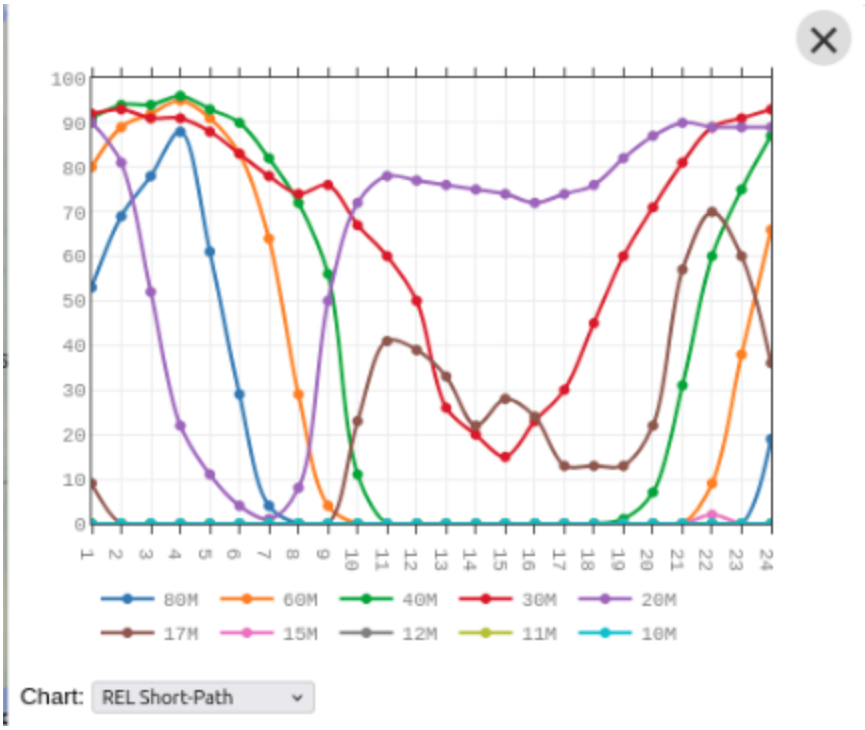
Note that VOACAP is more like an almanac versus realtime prediction. It uses trends and observed data to make a best guess as to what bands will be available when. Solar weather phenomena like solar flares can cause temporary blackouts on the daylight side of the earth which interfere with propagation.

VOACAP can also be used completely offline with software like voacapgui https://github.com/jawatson/pythonprop and HAMCAP by VE3NEA

Before we move on, let's look at one more example of propagation predictions with VOACAP:

Here's the propagation reliability chart for New York, USA to Dublin, Ireland





40 meters and 20 meters are our two best options. 40 meters opens during the evening, with 20 meters open during the day.

Now that we understand the basics of propagation and have a picture of what band conditions look like, we will need to pick our times we transmit based on our usable frequencies.

We want to pick two frequencies , a primary frequency and a secondary frequency.

The secondary to be used when there is interference or obstruction to the first frequency. In our example above we might chose

Our primary frequency could be 7.181 MHz. Our secondary frequency could be 7.209 MHz, which we move to if the primary is busy or being interfered with.

Both sit on 40 meters, so the same propagation carrying our signal applies to either one.

We would hold that schedule during 40 meters reliable window, roughly 21:00 to 24:00 UTC on this path. If we wanted a daytime option as well, we would do the same thing on 20 meters, our other strong band.

We would pick a primary and secondary frequency there too, say 14.237 MHz and 14.241 MHz, during its midday window around 14:00 to 18:00 UTC.

Monitoring the bands beforehand using your own equipment or publicly available SDRs like WebSDR or KiwiSDR will help you avoid interference.

Another way is to skip the amateur radio allocations entirely and **MARS mod** your radio. Factory stock HF rigs often come locked to just the Ham allocations. MARS modding, or

enabling full range of transmit lets you transmit your signal anywhere. It usually just involves removing a simple diode or resistor.

This lets you pick frequencies that are far outside the amateur radio spectrum which are prone to less interference.

Both frequencies, cadence, and times are agreed upon in person. They are always written in UTC so both stations are working off the same clock.

The last thing to determine is the **cadence** of messages. This is how often we will meet on the air. It might be every Sunday evening, once every two weeks, or the first of every month.

Cadence matters because of how the one-time pad works. Every message we send uses one key, and that key is destroyed afterward.

If we have traffic at most of our scheduled contacts, a weekly cadence works out to around 52 keys a year, while a monthly one uses about 12. The more often we talk, the more key material we burn through and the sooner we will need to meet in person again to hand over a fresh pad.

It is worth deciding ahead of time what happens when there is nothing to send. The receiver still listens at every scheduled window. The simplest option is to send nothing and let the receiver stand by until the next contact.

Our schedule as a written table might look like this:

Day	Every Sunday
Window	Evening
Band	40m
Time (UTC)	21:30 - 22:00
Primary freq	7.181 MHz
Secondary freq	7.209 MHz

Once we have our cadence, specified time (or window of time), selected bands and frequencies, we'll need to make sure we have all of the equipment to transmit.

Equipment

Any HF rig capable of doing 80 through 10 meters and with a soundcard interface to use with JS8call will work. One with an ATU (autotuner) like the Xiegu G90 is a recommended choice for a low cost rig.

Model	Xiegu G90	Yaesu FT-891	Xiegu X6100	Xiegu X6200	(tr)uSDX	QRPLabs QMX	QRP Labs QMX+	Lab599 TX-500
Price point	$650.00	$690.00	$565.00	$799.00	$123.00	$173.00	$185.00	$1,190.00
Max Pwr Out	20W	100W	5W (10W w/ ext. batt.)	5W (8W w/ ext. batt.)	5W	5W	5W	10 W
Nominal Input Voltage	12 V	13.8 V	9 – 15 V	9 – 15 V	6 – 15 V	12 V	12 V	9-15 V
RX current draw	0.77 A	0.99 A	0.33 A	0.65 A	0.08 A	0.08 A	0.08 A	0.1 A
TX current draw (@max power)	5.2 A	14 A	3 A	3.5 A	0.5 A	0.8 A	1.0 A	3 A
Weight	1.63 kg (3.59 lbs)	1.9 kg (4.18 lbs)	880 g (1.94 lbs)	880 g (1.94 lbs)	151 g (0.33 lbs)	301 g (0.66 lbs)	578 g (1.27 lbs)	550 g (1.21 lbs)
Bands	160m – 10m	160m – 6m	160m – 6m	160m – 6m	80, 40, 20, 15, 10m*	80-20, 60-15, or 20-10	160-6	160m-6m
Receiver	SDR	3x Conv. Superhet.	SDR	SDR	Superhet.	SDR	SDR	SDR
Antenna Interface Type	SO-239	SO-239	BNC	BNC	SMA	BNC	BNC	BNC
Display	1.8" color LCD	B/W LCD	4" (10.2cm) color LCD	4" (10.2cm) color LCD	2.4" (6.1cm) OLED	2.4" (6.1cm) mono LCD	2.4" (6.1cm) mono LCD	3.6" (9.2cm) mono LCD
VFO dial	clicky	smooth	clicky	clicky	clicky	clicky	clicky	smooth
Waterproof / Water-resistant?	No	No	No	No	No	No	No	IP54
Battery built-in?	No	No	Yes (3 Ah)	Yes (3.2 Ah)	No	No	No	No
Built-in Auto ATU	Yes	No	Yes	Yes	No	No	No	No
CW decoder?	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
FT8 built-in?	No	No	Yes	Yes	No	No	No	No
PSK31 built-in?	No	No	Yes	Yes	No	No	No	No
RTTY built-in?	No	No	Yes	Yes	No	No	No	No
sound card built-in?	No	No	Yes	Yes	No	Yes	Yes	No
SSB (LSB/USB)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CW	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FM	Yes	Yes	Yes	Yes	Yes	No	No	Yes
D-U (Digital Upper-Sideband)	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Digital Filter	Fixed bandwidth	Yes, Variable Bandwidth	Yes, Variable Bandwidth	Yes, Variable Bandwidth	Fixed bandwidth	Only for CW	Only for CW	Yes, Variable Bandwidth
Digital Noise Reduction / Noise Blanker	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Notch Filter	Yes	Yes	Yes	Yes	No	No	No	Yes
Built-in operating system	No	No	Yes	Yes	No	No	No	No
Bluetooth / WIFI	No	No	Yes	Yes	No	No	No	No
Built-in MIC	No	No	Yes	Yes	Yes	Yes, not yet functional with current firmware (Dec. 2025)	Yes, not yet functional with current firmware (Dec. 2025)	No
Body PTT	No	No	Yes	Yes	Yes	No	No	No
Preset voice messages	No	Yes	Yes	Yes	No	No	No	Yes
CW automatic call	No	Yes	Yes	Yes	Yes	No	No	Yes
Intermediate Frequency or I/Q output	Yes	No	Yes	*RF direct acquisition, no IF and IQ signal	No	No	No	Yes
Spectrum/waterfall display	Yes	No	Yes	Yes	No	No	No	Yes
SWR scanning function	Yes	No	Yes	Yes	No	No	No	Yes
WFM broadcast receiving	No	No	No	Yes	No	No	No	Yes
Aviation frequency band reception	No	No	No	Yes	No	No	No	No

The most important part of any HF setup is the antenna.

An antenna, at its core, is just a wire or other electrical element resonant on the wavelength of the band it's intended to work on.

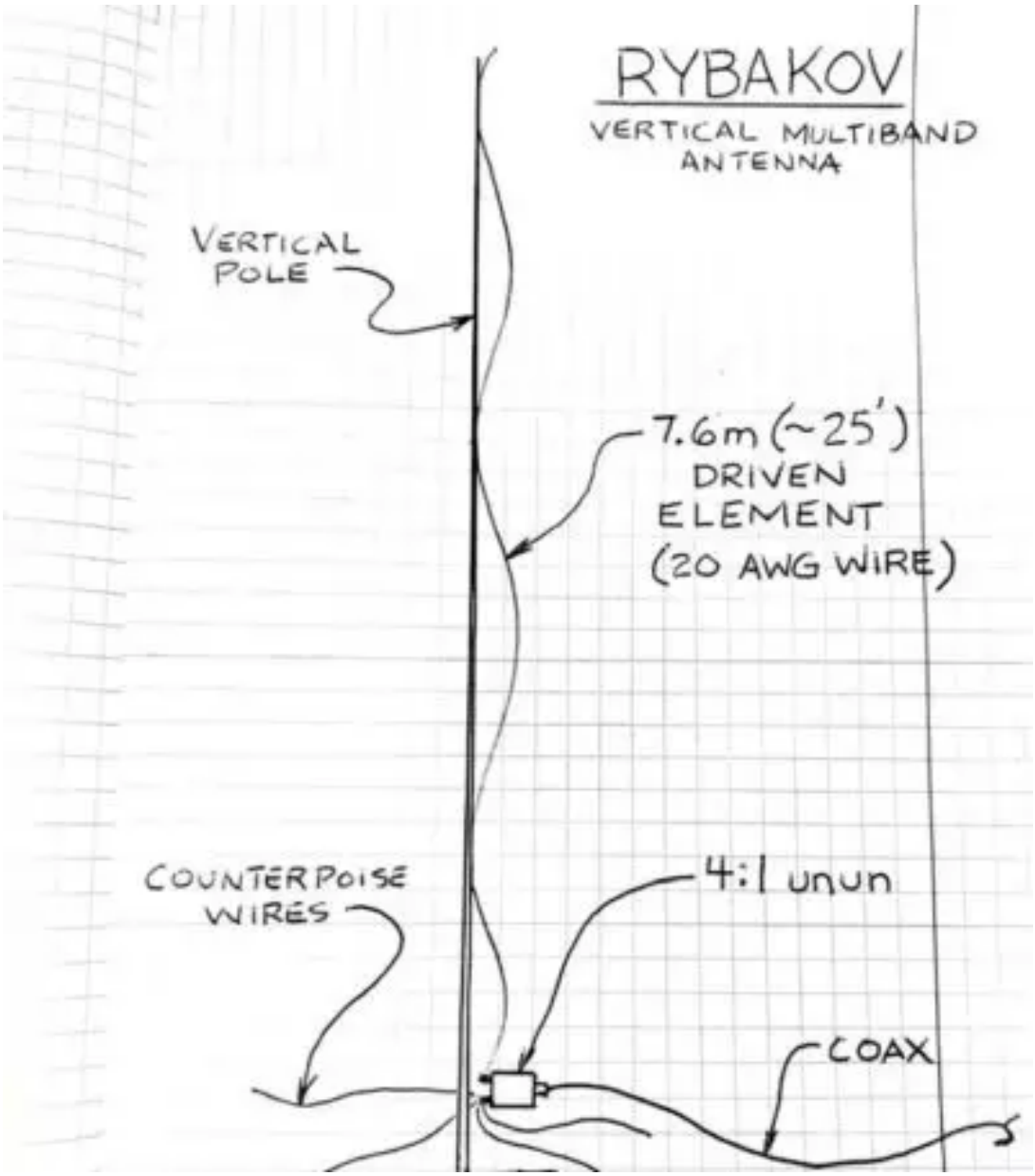
The most basic HF antenna, the half-wave dipole, is just a wire cut to half a wavelength long. Since a band's name in meters is roughly its full wavelength, the dipole for that band ends up about half that length.

That means the lower the band, the bigger the antenna. A 40 meter dipole is around 20 meters (66 feet) of wire, while a 20 meter dipole is only about 10 meters (33 feet). A rough rule for a half-wave dipole is 143 divided by the frequency in MHz to get the length in meters:

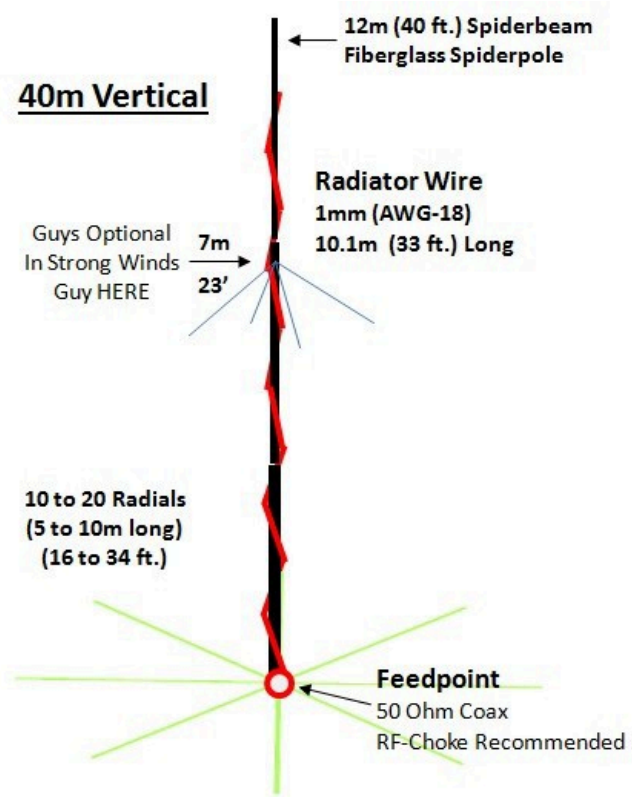
7.1 MHz gives ~20 m, and 14.1 MHz gives 10 m. The 40 meter antenna is roughly twice the size of the 20 meter one.

As radio operators we can use **compromise antennas**. These are antennas that aren't quite a full half wavelength long, and often include types like verticals. They require less of a footprint to deploy too.

A recommended portable vertical antenna design is the **Rybakov antenna**. It's simple, cheap and easy to deploy with suitable performance for digital modes like JS8call. But it's non-resonant and needs a tuner/counterpoise



A Rybakov lets us operate on bands 40m / 30m, 20m, 17m and 10m. Verticals like random wire antennas are also an option.



Vertical antennas can be rapidly deployed using something like a fishing pole or fibreglass mast. Various manufacturers like DX Engineering and Spiderbeam make masts for this exact purpose

A mast with guy lines or a flag pole holder style mount are two options for mobile deployment of the mast element.

A future guide will go into more detail on HF antennas. But the above section should give you some starting points.

The last piece of equipment we need is a computer and a way to connect it to the radio. JS8call is cross-platform and runs on Windows, macOS, and Linux, so almost any computer will work. It isn't demanding either, so for portable use a small laptop or even a single-board computer like a Raspberry Pi Zero 2 is plenty.

The link between the computer and radio has to do two things: carry audio in both directions, and key the transmitter when it is time to send. How you make that connection depends on the radio. Keying the radio from the computer is called CAT control. In some cases it takes an audio cable and CAT cable for operating the radio.

Some radios, like the Icom IC-7300 and Xiegu X6100, have a built-in USB sound card and rig control. A single USB cable between the radio and computer handles everything: receive audio, transmit audio, and keying. This is about as close to plug and play as it gets.

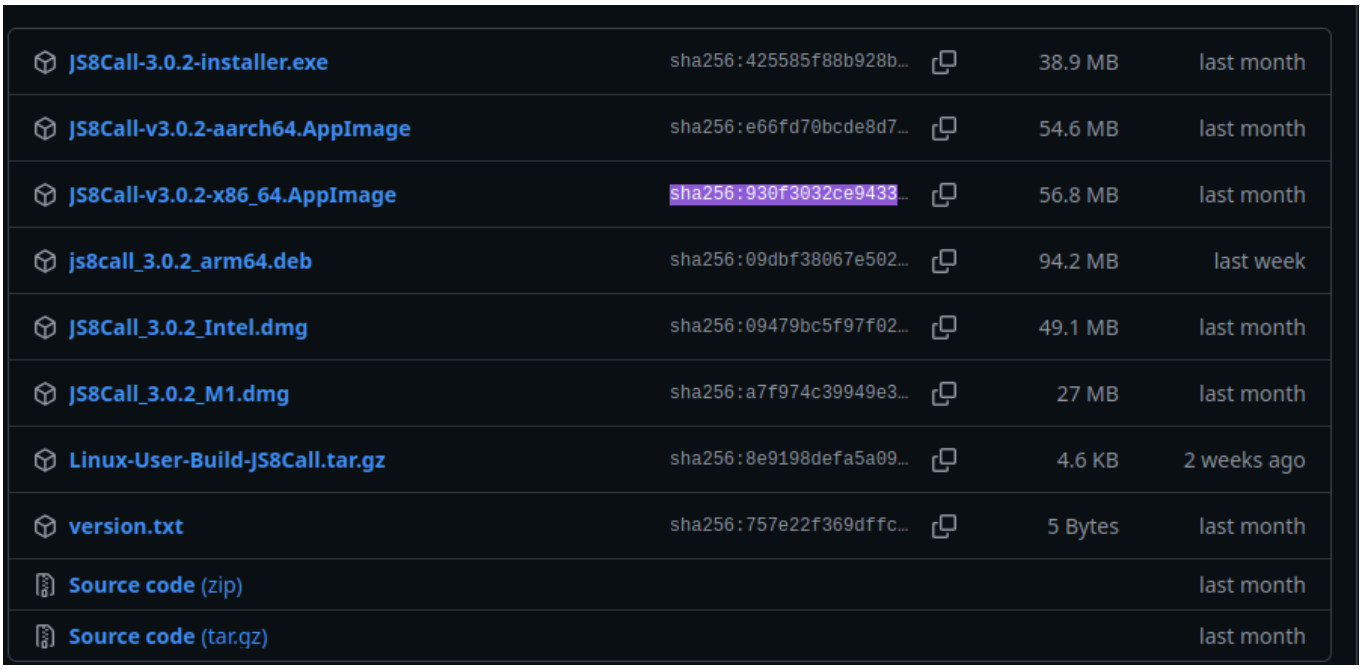
Other radios, like the Xiegu G90 we recommended earlier, do not have a built-in sound card. For these we add a small external interface that sits between the radio and computer and provides the sound card and keying over USB. A search online for "Xiegu G90 data cable" will yield you different options.

With our schedule set, a rig, and an antenna to match our bands, we have everything we need to get on the air. The last step is setting up JS8call itself and sending the message.

JS8call setup and transmit

JS8call can be downloaded from https://github.com/JS8Call-improved/JS8Call-improved/releases or https://js8call.com/downloads.html

On many Linux distributions it's also available as a package through the package manager, though the packaged version is often older than the current release. **It's best to download directly from the releases in the links above.** The latest version as of writing 21/06/26 is ver 3.0.2.



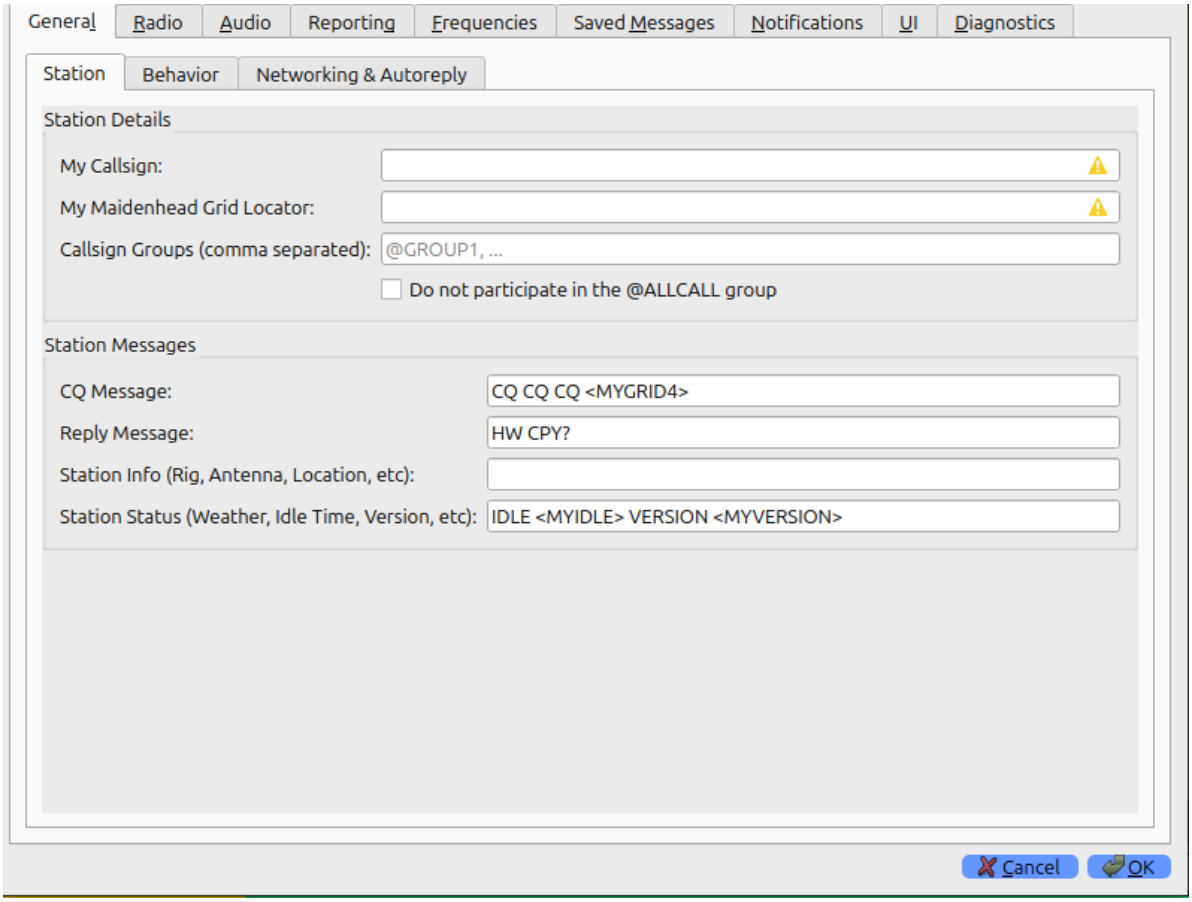
Make sure you also verify the release and it's SHA256 sum to ensure the integrity of the files. This lets us know if the file has been tampered with.

On Linux:
`sha256sum (the file path)`

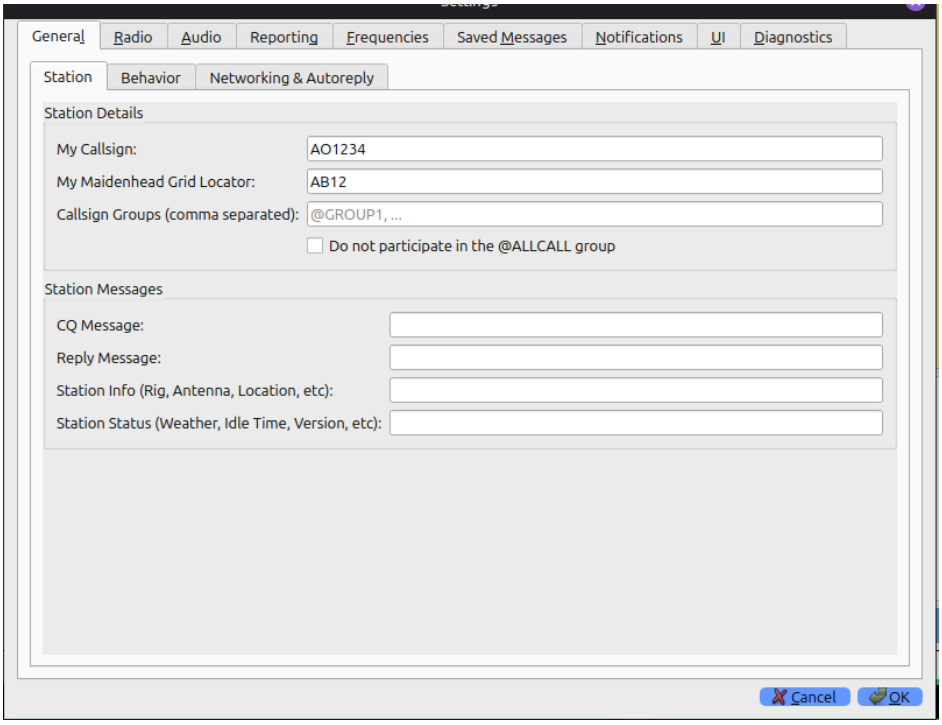
In Windows Powershell:
`CertUtil -hashfile "filename" SHA256`

Once JS8call is downloaded, turn on your radio, connect the audio and CAT cables to your computer, and launch JS8call.

The first step we need to do is configure our station settings. Under "**General**", strip everything.

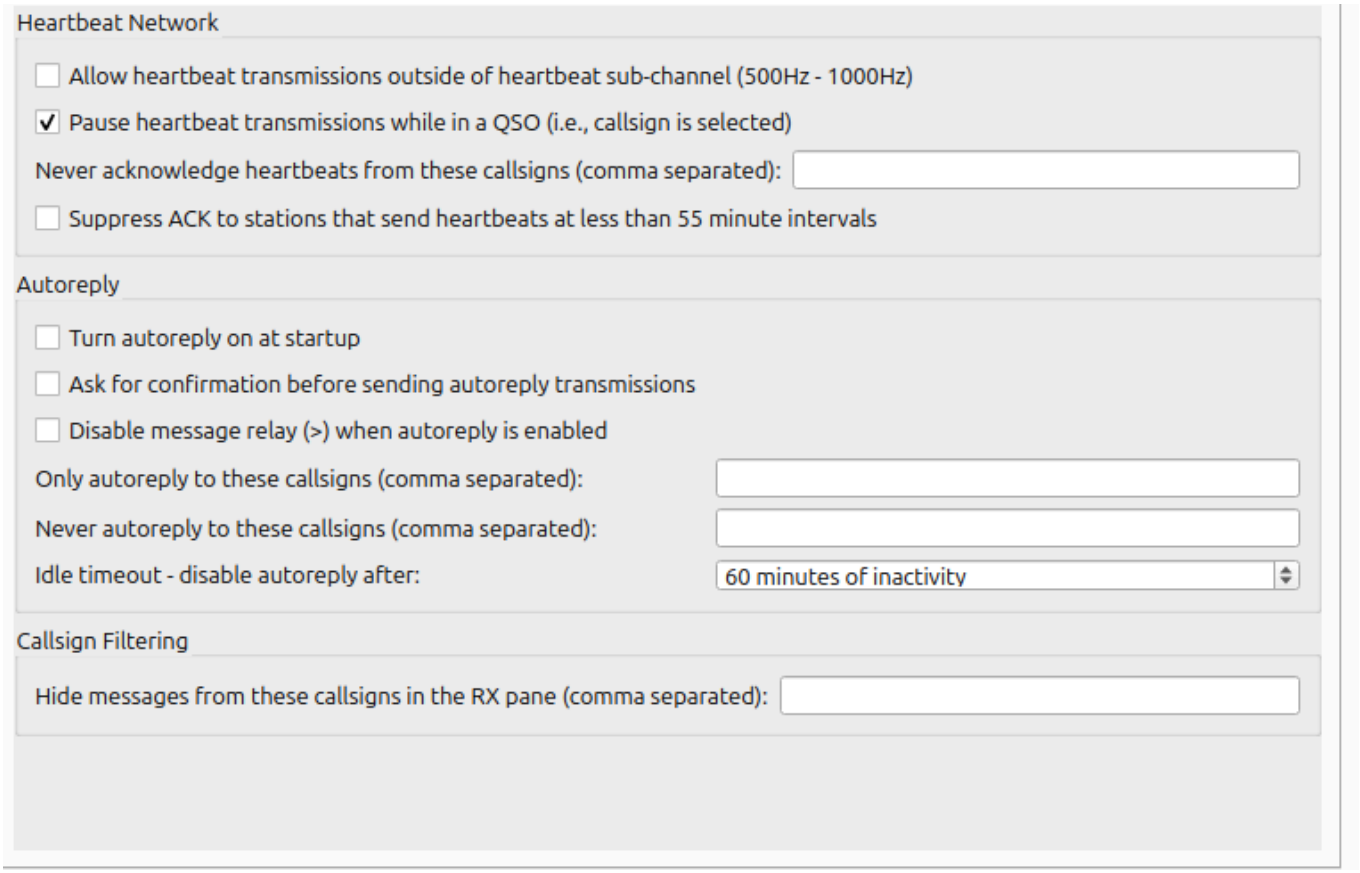


Remove all station messages. "My Callsign" and "My Maidenhead" only need something that matches the pattern. Below is an example of what our config should look like:

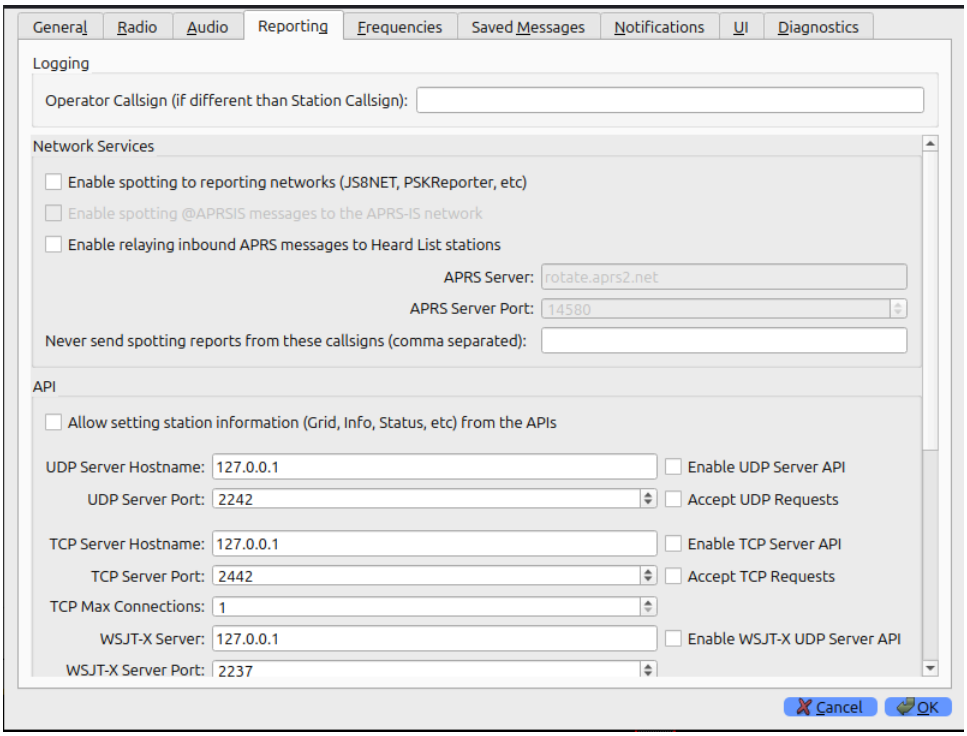


Next, go to "**Networking & Autoreply**"

Turn off all of the autoreply and heartbeat settings:

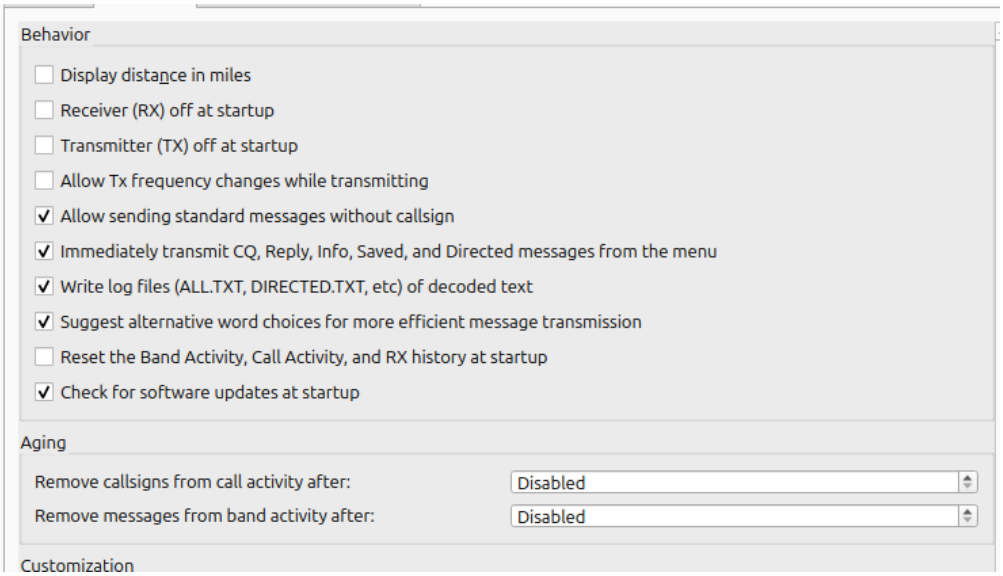


One last thing we do before we configure our rig and audio is to go to reporting. Turn OFF all reporting settings.



(I recommend that your setup is isolated from any network, including cellular and Internet entirely)

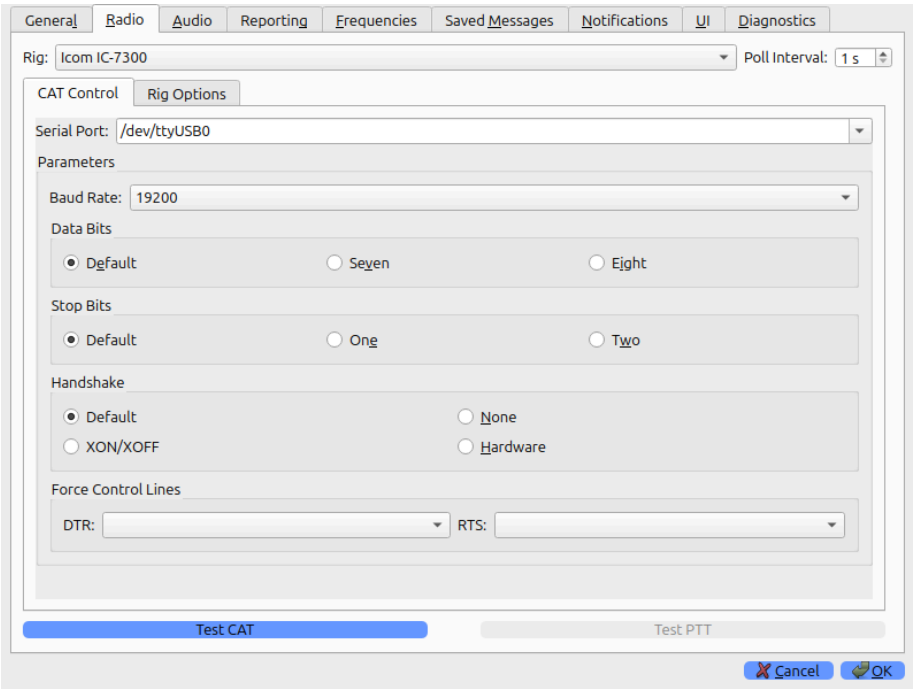
Then go to "**Behavior**"



Check allow sending standard messages without a callsign

Disable "Remove messages from band activity after...." by clicking all the way down or backspacing the default.

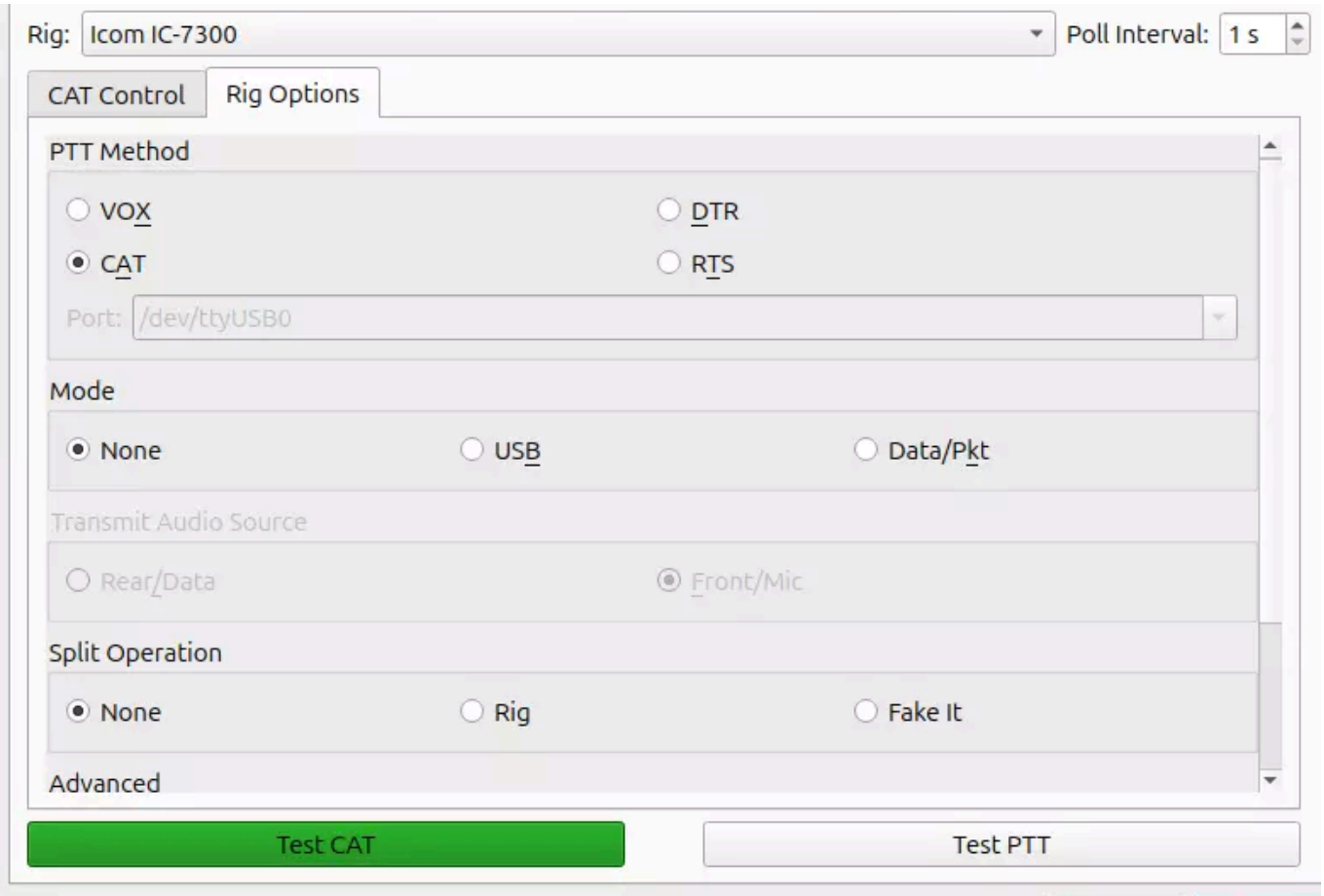
We can now configure our radio. Your radio settings will be rig specific. Select from the dropdown your rig. Then set the:



- serial port
- baud rate
- and data bits if needed

An online search for your rig will tell you all of what needs to go here.

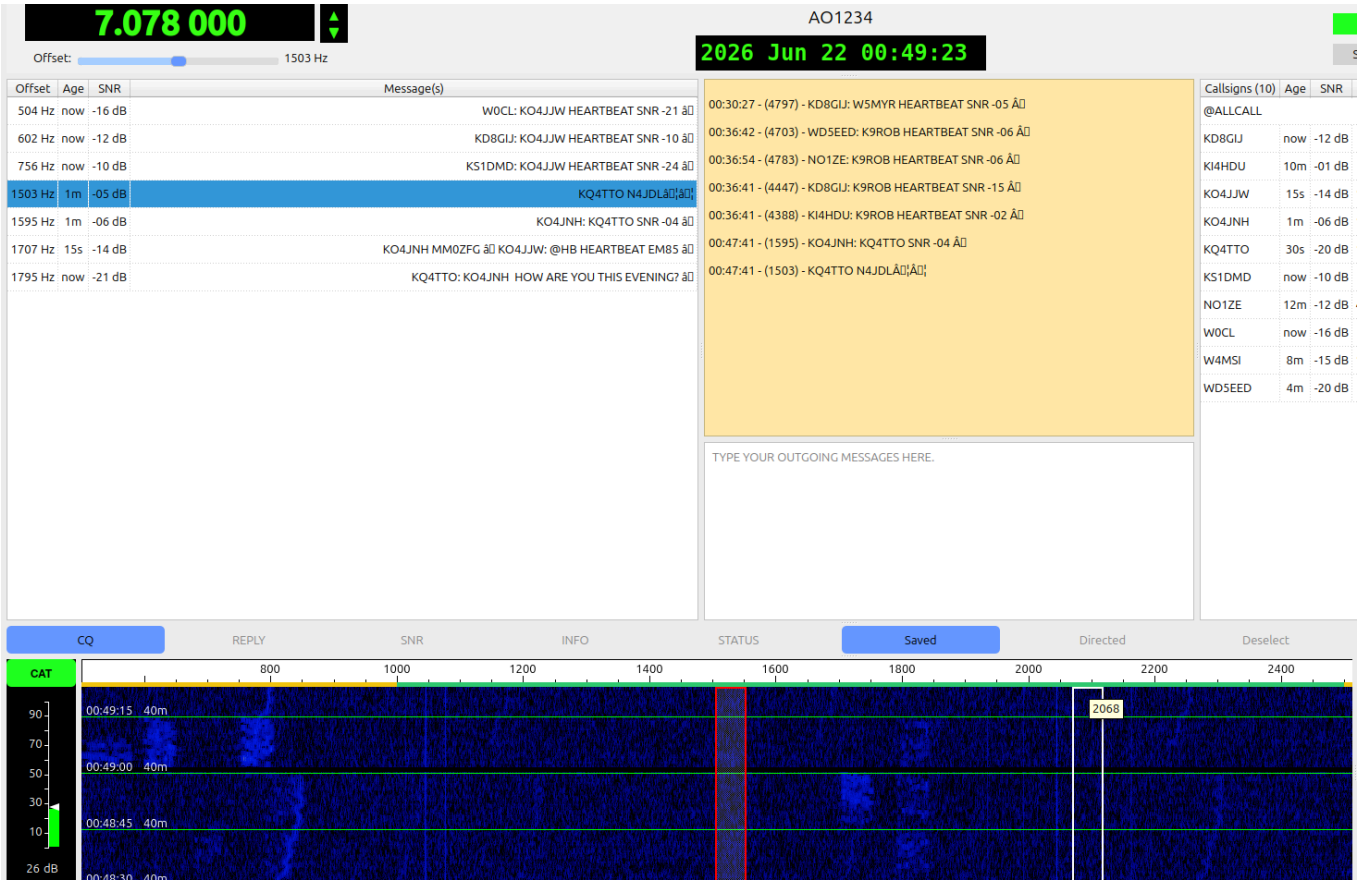
Next, go to **Rig options**.
Select CAT as your PTT method. Almost all HF rigs will use CAT for the push to talk



When you are finished, hit "Test CAT". Then hit "Test PTT". If it's greyed out make sure to press "Test CAT" first. You should see your radio key up. Once you are done, hit "Test PTT" again and it should go from red to grey, unkeying the radio.

The next thing to configure is audio. You'll want to set your input and output device to whichever audio device your soundcard or radio appear as. We can also set a notification soundcard output. Select your computer speakers or headphones for this, as it plays a notification sound when we receive a message.

Finally, a quick sanity check. Depending on what bands you are using you can find JS8call activity at them by searching your "band + JS8call frequencies" . For general activity, 40 meters at 7.078 MHz and 20 meters at 14.078 MHz are usually the busiest. Set your rig to USB and after a few minutes you will begin to see some messages being decoded.



We can now send a test transmission.

Text goes in the outgoing message box, the field labeled "TYPE YOUR OUTGOING MESSAGES HERE."

Before sending, pick a clear spot to transmit on. Look at the waterfall and find an empty vertical lane with no other signals in it, then click that spot to move your offset there. This keeps us from transmitting on top of another station. Anywhere between 500 and 2500 Hz is fine.

We want to make sure our radio is set to **USB** (or USB-Data if it's an option) and is operating with a filter of **2400 Hz**.

With a message typed and a clear offset chosen, set your power. Hit the **Tune** button in the top right.

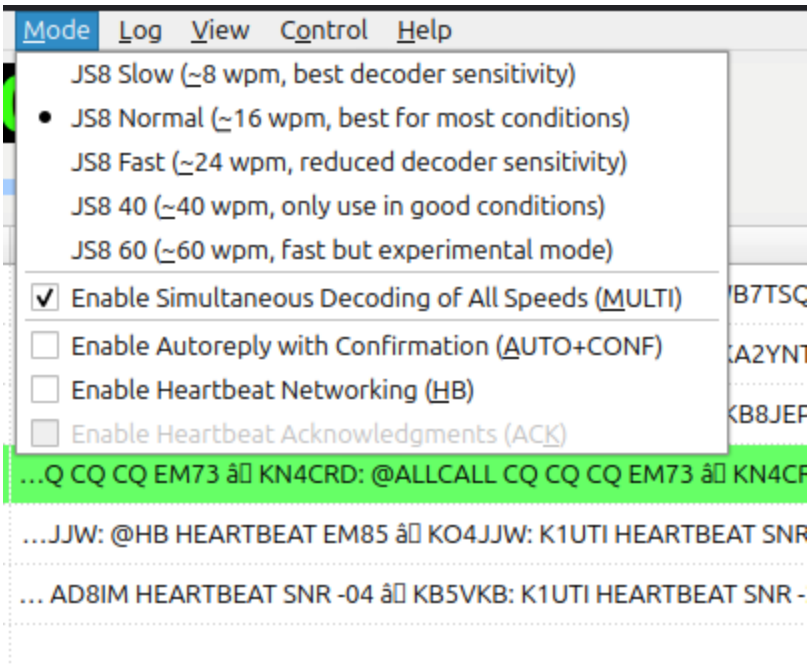
The radio will key up and sends a steady tone. Adjust your power or audio drive so the ALC reads at or near zero, then hit Tune again to stop. Overdriving here is the most common mistake, and it splatters your signal across the band instead of making it louder.

Verify everything on your rig. SWR, power output, and that the ALC is barely moving.

Confirm the radio is actually keying when JS8call transmits and that the power out matches what you set.

Now hit **Send**. For our test transmission we can put anything we want in the box.

JS8call queues the message and transmits it at the start of the next cycle. You will see the radio key up through CAT, your text appear in the message window, and your signal show on the waterfall. When it finishes, the radio unkeys on its own.



JS8call has several different modes ranging from Slow to JS8 60 (in js8call-improved). Each speed is measured in words per minute.

Slow is great for when there is solar weather anomalies or extremely poor propagation conditions and works down to -28 dB below the noise floor. **Normal** is what we want to use under most circumstances, and if propagation permits, JS8 Fast can be used too

The best way to verify propagation and reception is to do a dry run with our intended recipient. It validates both parties setups, but communication of reception reports must be done out of band.

If this isn't possible there are other methods:

We can use JS8calls native heartbeat feature to get reports from stations on the main JS8call frequencies for the band we are operating. These are stations operating automatically which will respond to our heartbeat with our SNR

We can use online SDRs like publicly available KiwiSDRs and WebSDRs to audit our reception.

Note that doing so creates a digital footprint the same as it would when you access any other website. KiwiSDRs in particular have the ability to log user activity and frequencies alongside arrival / departure. Use online SDRs at your own discretion.

Once we've done a function check of our rig and software, we now have a station ready to go on air.

If your HF rig has CAT control, you can add your schedule frequencies to JS8call's frequency list, in the Config under Frequencies. When a scheduled contact comes around, you can then select the right frequency straight from the dropdown instead of tuning the radio by hand.

When it is time to send our one-time pad message, we follow this procedure. We will use the encrypted message from the earlier sections as our example, so feel free to go back for a refresher.

Our OTP ciphertext is:
`06356 36432 82219 22509 68439 65092 74829 18951 38504`
(our "Farmer Chucks at 1845z" message)

Recall that this message was longer than a single key, so it used two keys from our pad. The key ID is always the first group of a key's material, which here gives us **06356** for the first key and **74829** for the second.

To mark these key IDs so the receiver knows where to switch keys, we write each one twice. Everything else stays as it is.

Then we send the whole message a second time. This is a one-way transmission with no acknowledgment, so sending it twice gives the receiver a second chance at any groups lost to a fade or a burst of noise.

Resending costs nothing extra in key material, since it is the same ciphertext.

Our message, ready to send in JS8call, looks like this:

```
06356 06356 36432 82219 22509 68439 65092 74829 74829 18951 38504
06356 06356 36432 82219 22509 68439 65092 74829 74829 18951 38504
```

``

When the message is ready, hit Send. **M**ake sure no callsign or group is selected so it goes out as plain free text, and double check what you have entered in the message box.

Once you hit send, our message is now transmitting over the air.

9. Reception

Reception is a similar procedure to transmission. But our receiving party doesn't need a full HF rig, nor a full antenna setup.

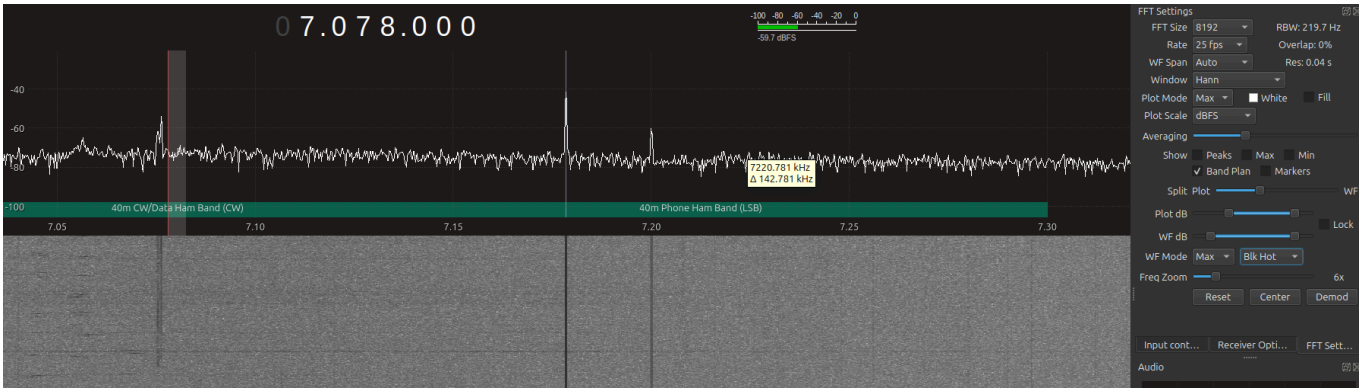
There are a variety of options for receiving the JS8call OTP message over HF.

A 40 USD **RTL-SDR** and active magnetic loop receive antenna like the **MLA-30** is one of the best combinations for a receiving station.

An RTL-SDR is a USB powered receiver that can pickup signals on the HF spectrum.

(Left: RTL-SDR, right: MLA-30 magnetic loop receive antenna)

It plugs directly into a computer over USB and you use software like gqrx or SDR++ to control it.



The receive antenna does not need to be large or efficient. We are only listening, not transmitting, so the strict size and resonance rules from the transmit side do not apply.

An active magnetic loop like the MLA-30 is a great match here. It is compact, quiet on local noise, and works in limited space.

A loop on a patio, a balcony, or even indoors is usually enough to pull in signals from across the country or the world.

To decode we use JS8call much like we did for transmitting with two differences. All of the same options should be applied in the settings with a few exceptions:

First there is no radio to key, so we set the rig to None.

Instead of a sound card / cable connected to a radio, our audio comes from the SDR software.

We route the audio output of gqrx or SDR++ into JS8call's input into an audio loopback. It's best to search online for your OS specific way to set this up.

The SDR demodulates the signal to audio, the loopback carries that audio across, and JS8call decodes it into text.

A spare laptop or a Raspberry Pi running the SDR software and JS8call can sit on the schedule frequency and log all of the decodes unattended, although it's recommended during the window to monitor the frequency if it needs to move to the alternate backup.

Even a simple shortwave receiver works as long as it covers the HF bands and can demodulate SSB. A portable set like the Tecsun PL-330 can be tuned to the schedule frequency and plugged into the computer's audio input with an aux cable.

At the scheduled time, we listen on our primary frequency. If it is jammed or dead, we move to the secondary frequency, the same way the sender will. When the sender transmits, the decoded groups appear in JS8call's left panel.

An important thing to note for the recipients JS8call config is to disable message expiry under Settings -> General -> Behavior -> "Remove messages from band activity after ..." in addition to under "**Mode**" ensuring that "Enable simultaneous decodes" is checked and enabled.

Copy the groups down onto paper exactly as they arrive. Remember that the sender transmits the whole message twice.

Once we have a clean copy of the full message, we move away from the computer to decode. Only the OTP ciphertext ever touches the computer.

Recall that the doubled groups mark the key IDs. In our example the receiver copies down:

```
06356 06356 36432 82219 22509 68439 65092 74829 74829 18951 38504
```

The two doubled groups, 06356 and 74829, are the key IDs.

We collapse each doubled pair back to a single group and use those IDs to pull the matching keys from our IN pad, key 001

From here it is exactly the decryption from Section 6. Write the OTP ciphertext above the key, ignore the key ID groups, and subtract mod 10 down the line. That gives back the plaintext digits, which we convert to letters with our checkerboard of choice.

For our example this returns FARMERCHUCKS AT (figure shift) 1845 (figure shift) z, our meeting at Farmer Chuck's at 18:45z.

The final step is to destroy the key.

Once the message is read, the keys that carried it are spent. Both stations physically destroy the keys they used. This is not optional. The perfect secrecy of the one-time pad holds only as long as each key is used once and then gone.

The full lifecycle of the OTP message is now complete, and the stations should prepare for the next message.

10. Closing Remarks

If you've made it this far, you now know all the basics of the one-time pad technique, HF propagation, how to establish and make contact, and the JS8call digital mode software.

This is not a comprehensive overview of all subjects, but a brief introduction designed to cover as much as possible. I encourage you to go out and research each individual topic if you have any questions.

The only way to really develop these skills is by practice. Participation in the amateur radio hobby side of things is a great way to develop and hone skills, especially if they ever one day become a necessity.

The information included above on JS8call does not have to be used for just one-time pads either. You can use the same methods for sending out something like a text bulletin.

JS8call is also just one of many options for digital modes and software. Software like fldigi offers even more modes to encode our data with and send over the air. Modes like Olivia and Contestia are even better suited for one-time pad transmissions. If you want a challenge, go experiment with these.

Attached alongside this guide is the JS8call manual for version 3.0.2, and "THE COMPLETE GUIDE TO SECURE COMMUNICATIONS WITH THE ONE TIME PAD CIPHER" by D. Rijmenants which goes even more in-depth into the one-time pad technique, with a few things not covered by this guide.

This will be the first document in a series. I'd like to next make a guide on portable HF antennas and mobile operations.

If you're reading this guide from far away lands in it's .md or PDF format, drop by and say hi on the Kiwi Farms Ham Radio thread
on the clearnet @ https://kiwifarms.st/threads/ham-radio-off-grid-communication.174222 or on TOR @ https://kiwifarmsaaf4t2h7gc3dfc5ojhmqruw2nit3uejrpiagrxeuxiyxcyd.onion/threads/ham-radio-off-grid-communication.174222

, 55

Document revision: 22/06/26