Python Radio 30: Catch the Fox

A Game Like Cat and Mouse

Simon Quellen Field

Simon Quellen Field

Follow

9 min read

·

Dec 1, 2024

Listen

Share

More

Press enter or click to view image in full size

A fox with a transmitter.

MidJourney

Radio fox hunting. It’s a fun game that came about through necessity.

It is unfortunately often that we have to find a source of radio interference, rescue a boat at sea, or find an enemy transmitter on a battlefield. Radio amateurs practice doing this with an entertaining game called a fox hunt.

Someone hides a small battery-powered transmitter, and others use techniques and tools to find it. It’s a bit like geocaching. At Easter time, it’s fun to hide a transmitter inside a plastic Easter egg and play the game with kids.

We will build a few transmitters and receivers, as well as show how to use amateur radio transceivers and directional antennas in the hunt.

In keeping with our unspoken theme in this series, we will add the constraint of being cheap, as well as small, battery-powered, and programmable in Micropython.

Our computer

One of the least expensive computers that can run Micropython is the ESP8266–01S. I get them from AliExpress literally by the handful — they cost $11.68 for ten of them, and they arrive in about ten days.

Press enter or click to view image in full size

A handful of ESP8266–01S’s

A handful of ESP8266–01S’s

They are 32-bit computers, running at 160 megahertz, with a megabit of flash memory (128k bytes), 32k bytes of instruction RAM, 80k bytes of data RAM, 802.11 b/g/n Wi-Fi, I2C, SPI, UART, and a 10-bit ADC.

That’s a lot of computer for $1.17.

I can’t help but compare them to the original IBM PC, which was much slower, only 8-bit, had no flash memory, had less memory, no UART, no Wi-Fi, networking, I2C, ADC, or SPI. And the IBM PC was 1,500 times more expensive.

The original Apple Macintosh came out a few years later with a 16-bit computer, but it also lacked the specs of the ESP8266–01S and was much slower and had no networking.

Since we want long battery life, we chose the ESP8266–01S over the Wemos D1 Mini we have used before. The D1 has a USB adapter on the board, which uses power all the time. It is also larger than the 01S, and we want to hide our fox in little pill bottles.

Programming

To program the 01S, we will need a separate USB to UART adapter. Search for “ESP8266–01 programmer”. You can find them on AliExpress for less than a dollar. I got a couple from Amazon for ten times that, but they arrived the next day.

Press enter or click to view image in full size

ESP8266–01S programmer

ESP8266–01S programmer

The 01S plugs into it like this:

Press enter or click to view image in full size

ESP8266–01S programmer with computer

ESP8266–01S programmer with computer

Be sure to get one with the programming switch on it, or you will have to solder one on yourself. Some have a slide switch like the one above, others have a pushbutton.

To program the 01S, set the switch to PROG (or hold down the push button) and plug it into a USB slot. Once it is plugged in, it is programming mode and you can release the button or set the slide to UART.

I use the following setup.cmd program on Windows to program the 01S:

Set comport=%1

Esptool –port %comport% erase\_flash

Pause

Esptool –port %comport% --baud 460800 write\_flash –flash\_size=detect 0 esp8266\_mpy\_1M.bin –verify

Micropython

There are three versions of the ESP8266 Micropython. One is for the boards that have 4 megabits of flash, one for those with 512 kilobits of flash, and one for those (like our 01S) that have a megabit of flash.

Download that last one, and rename it esp8266\_mpy\_1M.bin, and then run “setup com3”. Your com port will probably be some other number.

The Python code

Def beep(f):

From machine import PWM, Pin

From time import sleep\_ms

Print(“F is”, f)

Speaker = Pin(0, Pin.OUT)

PWM(speaker, freq=f, duty=512)

Sleep\_ms(100)

PWM(speaker, freq=f, duty=0)

Def main():

From network import WLAN, STA\_IF, AP\_IF, AUTH\_OPEN

From time import sleep\_ms

Sta = WLAN( STA\_IF )

Sta.active( True )

Ap = WLAN( AP\_IF )

Ap.active( True )

Mac = ap.config(‘mac’)

Mac\_str = “”

For b in mac:

Ch = hex(b)

Mac\_str = mac\_str + ch[2:] + ‘:’

Mac\_str = mac\_str[:-1]

Who\_am\_i = “Fox “ + mac\_str

Print(“I am”, who\_am\_i)

Ap.config( essid=who\_am\_i, authmode=AUTH\_OPEN )

Nets = sta.scan()

For net in nets:

Print(“net:”, net)

Ssid = net[0].decode(“utf-8”)

If ssid.startswith(“Fox “):

Print(“Found a fox!”, ssid)

Sta.connect(ssid)

While not sta.isconnected():

Machine.idle() # save power while waiting

Print(“Connected to”, ssid)

Break

While True:

Rssi = sta.status(‘rssi’)

Print(“RSSI:”, rssi)

If rssi < 0:

Beep((90 - -rssi) \* 10)

Sleep\_ms(300)

Main()

To load main.py onto the 01S, we use ampy:

Ampy -p com3 put main.py

The beep() function uses pulse-width-modulation to send a tone to pin 0. We will connect a speaker to that pin.

The main() function sets up two Wi-Fi connections. One is an access point. We copy its MAC address into a string and add it to “Fox ” to create its SSID.

The second Wi-Fi connection is a station. We scan for access points looking for any that start with “Fox ”. When we find a fox, we connect to it.

In this way, the same code works for both the fox and the fox hunter (the hound).

The hound looks at the RSSI (receive signal strength indicator) and calls the beep() function to emit a tone that rises in pitch as we get closer to the transmitter.

The receiver

The hardware part of the receiver is pretty simple. We build a little wiring harness for the computer, battery, and speaker:

Press enter or click to view image in full size

Wiring harness

Wiring harness

When everything is connected, it looks like this:

Press enter or click to view image in full size

Wired up

Wired up

I prefer rechargeable batteries to AAs, but two or three AA cells in series will work just fine instead of the lithium polymer battery I used. AA cells (and my LiPo battery) produce 1800 milliampere-hours of power. This will power the radios all day.

The positive wire from the battery connects to the 3V3 pin on the computer. The negative wire connects to ground. The last wire to connect is the one between the speaker and the IO0 pin. The other side of the speaker is connected to ground.

The IO0 pin cannot be connected before the computer is powered up, or the computer will not boot.

The Transmitter

The wiring for the transmitter is even simpler.

Press enter or click to view image in full size

The transmitter

The transmitter

We just connect it to the battery.

The Game

I like to hide the transmitter in a little pill bottle. They are waterproof, so I can toss it into wet weeds or a freshly watered lawn. Plastic Easter eggs are also fun.

When you plug in the IO0 wire on the receiver, it takes a little bit of time to find the signal from the fox and connect to it. After that, it starts beeping. If your body is between the receiver and the transmitter, the pitch will drop, so you know which direction to walk. The players don’t know what you hid the transmitter in, so they may hunt around for a while, even if the pill bottle is in plain sight.

A poor man’s dish

Blocking the signal with your body is a time-tested radio direction finding trick. But we can do better.

We discussed how to make Yagi-Uda directional antennas in an earlier part of this series. We could do that for 2.4 gigahertz, but instead, let’s do something new.

You are probably familiar with parabolic reflectors. They are used in telescopes, in directional microphones, and solar cookers. But making a dish parabolic is a chore.

Instead, we will build a spherical dish. Our radio has an antenna that is larger than the focus of a parabolic dish anyway, and a spherical dish will still focus the energy on our (relatively) large antenna (the gold printed squiggle on the printed circuit board).

I inflated a latex balloon and crumpled some aluminum foil over it. It was spherical enough. At 2.4 gigahertz, the wavelength is about 12 centimeters. This means the crumpled peaks and valleys of the foil are still well under a tenth of a wave, so the radio waves will focus nicely.

On top of the foil I used school glue and paper towels to make a quick and dirty paper mache back to keep the foil in shape. A pencil stuck through the center of the dish holds the radio.

Press enter or click to view image in full size

The front of the dish

The front of the dish

The battery and speaker are hot glued to the back.

Press enter or click to view image in full size

The back of the dish

The back of the dish

The pitch of the receiver changes quite a bit as we aim the dish around — much more than it did when we just blocked the signal with our body. The dish focuses the signal when aimed at it, and blocks the signal when aimed away. It is still light enough to easily hold in one hand, and the paper mache is stiff enough to handle normal use.

Going Further

Hunting Wi-Fi transmitters is fun, and might even be useful if you want to locate the best place to sit in a cafe with free Wi-Fi (you just change the “Fox ” in the code to the SSID you are looking for).

But Wi-Fi only goes so far. To make the game more challenging, we can use a transmitter that can be heard for a mile. Our simple Morse Code transmitter from earlier in the series is just the thing.

It uses the FS1000A transmitter, and optionally a ground plane antenna. We change the main.py code just a little:

From morse import Morse

From machine import Pin

From network import AP\_IF, STA\_IF, WLAN

From esp import deepsleep, sleep\_type, SLEEP\_MODEM

Ap\_if = WLAN(AP\_IF)

St\_if = WLAN(STA\_IF)

Sleep\_type(SLEEP\_MODEM) # Adding this made no difference since we turned the WIFI off

Def main():

If ap\_if.active():

Ap\_if.active(False) # Disable access point

St\_if.active(False) # Disable station interface

Deepsleep(1, 4) # The radio only turns off when we go into deepsleep

# Now the user must ground the reset pin

# After reset, we get here with the WIFI radio turned off to save power.

# Power went from 80 mA down to 18.85 (and 21 mA when transmitting using FS1000A)

GPIO\_0 = 0

GPIO\_2 = 2

Led = Pin(GPIO\_2)

Led.off()

Morse = Morse(GPIO\_0)

Wpm = 5

Morse.speed(int(wpm))

While(True):

Morse.send(“catch me if you can!”)

Main()

Here again we use the 01S to save battery power. To save more power, we turn off the Wi-Fi radio. A peculiarity of this computer is that we can only turn off Wi-Fi by going into deep sleep mode. When we wake up (by grounding the reset pin or toggling power) Wi-Fi remains off (unless we turn it back on in Micropython or reflash the device).

At 19 milliamps of power draw (21 when transmitting) the 1800 mAh battery (LiPo or AA) will last about a hundred hours.

We can use the little Morse receiver from the earlier project as the receiver, but it has a serious drawback: the signal does not change as you get closer. If it hears the Morse code at all, it beeps the speaker. Neither the volume nor the pitch change.

We can play with the receiver’s antenna, blocking it or shortening it until the signal goes away. But this is a challenge.

Thankfully, any UHF FM radio receiver or transceiver can pick up the signal and give us an RSSI reading, usually as a number of bars on the screen. The BAOFENG transceiver is only $25, and works fine for this task.

We can use our Yagi-Uda antenna, or we can use a more robust commercial antenna with its much lower wind load than a big sheet of foam core board.

Press enter or click to view image in full size

A Baofeng radio with a Yagi-Uda directional antenna

A Baofeng radio with a Yagi-Uda directional antenna

An antenna like that can be used to talk to amateur radio satellites. A computer program that tracks the satellites will tell you where to aim.

Radio Hackers

Python

Radio