Python Radio 6: Field Strength Meter

Simon Quellen Field

Simon Quellen Field

Follow

4 min read

·

Aug 23, 2024

Listen

Share

More

How strong is my signal?

Press enter or click to view image in full size

Photo by author

A helpful tool for debugging radio projects is a simple field strength meter. It is just an antenna, a diode, and a meter:

Press enter or click to view image in full size

Photo by author

In this case, the antenna is a loop of wire attached to the meter at one end and connected to a diode at the other end. The diode is then attached to the other terminal of the meter. The cathode of the diode (the side with the band) is on the side connected to the meter so that when a radio signal arrives, the meter swings in the positive direction.

Press enter or click to view image in full size

Photo by author

We use a germanium 1N34A diode because it is more sensitive than a silicon diode. Even so, our little 10-milliwatt radio transmitter must be very close to the wire loop to move the needle.

But our little computer has an analog input pin that can read voltage levels and convert them into numbers. So we can replace the analog meter with our computer and print the numbers out on the screen.

We remove the wire loop and its diode from the meter and connect the diode to the A0 pin of the computer. We connect the other end of the wire to the ground pin.

Our program is once again very simple:

From machine import ADC

Pwr = ADC(0)

While True:

Avg = 0

For count in range(20):

Avg += pwr.read()

Avg /= 20

Print(avg)

We import the ADC (Analog to Digital Converter) class from the module machine.

Our little computer only has one analog input, so we will always use ADC zero. Later, on beefier machines, we can use more. Then we loop forever, averaging 20 readings and printing them. Feel free to average 100 or 1000, I picked the number 20 out of a hat.

Now as we move our transmitter closer to the receiver, we can watch the numbers get bigger.

Instead of numbers, let’s make a bar graph out of LED lights to indicate the signal strength. We can get strips of Reg/Green/Blue LED lights called NeoPixels by the yard, and we can cut them into any length we wish. I chose to use 16 of them for this project, but you can use three or a hundred if you wish.

NeoPixels are very easy to use. They have only three things to connect, no matter how many lights you use. The connections are power (the +5 volts from the computer), ground, and data. We will use pin D4 as the data pin (GPIO 2).

MicroPython has a convenient NeoPixel module built in. Our code now looks like this:

From machine import Pin, freq, ADC

From time import sleep, sleep\_ms

From neopixel import NeoPixel

PIN\_D4 = 2

Def main():

Freq(160000000) # 160 MHz

Pwr = ADC(0)

Np = NeoPixel(Pin(PIN\_D4), 16)

While True:

Avg = 0

For count in range(20):

Avg += pwr.read()

Avg /= 20

How\_many\_lights = avg \* 16 / 1024

For i in range(16):

If i < how\_many\_lights:

Np[i] = (0, 2, 0)

Else:

Np[i] = (0, 0, 0)

Np.write()

Main()

Just for fun, I doubled the clock speed to 160 megahertz to show you it can be done. It uses the battery up twice as fast at this rate, and our program doesn’t actually benefit from the extra speed, so feel free to omit that.

The program is basically the same since the only addition is the NeoPixel code.

We import the NeoPixel class, and then create an instance of it called np, telling it to use pin D4, and that there will be a string of 16 lights.

We convert the average reading (which would be between 0 and 1023) into a number between 0 and 15 and call that how\_many\_lights.

Then we loop through the 16 lights, assigning them either (0, 2, 0) [a dim green] or (0, 0, 0) [completely dark], based on how many lights we want in our bar graph. I chose green, but you can play with any colors you wish. The lights can be very bright, which is why I chose the number 2, which is less blinding. Using the number 1 might even be better, and would allow longer battery life when running on batteries.

The end result looks like this:

Press enter or click to view image in full size

Photo by author

The schematic for the circuit looks like this:

Press enter or click to view image in full size

Image by author