# FAUXY Lady

**Category**: “Can’t Stop the Signal, Mal”

A university needs your help collecting their cubesat’s telemetry. We’ve captured a .wav recording of the satellite signal and the university has published the telemetry definition. The recorded signal has the following characteristics:

* BPSK modulated
* Differentially encoded
* 44.1k samples per second

Can you reconstruct the telemetry packet?

## Description

The challenge presents to us with a waveform file signal.wav and a description that says that the signal is BPSK modulated with a sample rate of 44.1Khz and differentially encoded. We were also given the packet specification of this protocol in a pdf file.

## Solution

Our approach started with the analisys of the signal inside GNURadio Companion, we set the sample rate to 44.1Khz, imported the file with a Wav file source block and converted it into complex type using a Float to Complex block. We then used a BPSK Demodulator block guessing the baudrate while also watching the output with a Time Sink block. As soon as we set baudrate to 1200 and Differential to True we got a nice square wave out. We then exported the bitstream and further processed it Python.

### Data processing

The challenge description mentioned differential encoding, so the first step we applied was a differential decoder:

for b in bits:  
 out.append(b ^ prev)  
 prev = b

Printing out the resulting bits, we noticed that after the first section where there was some noise, splitting the text in 8 bit chunks only ever produced two bitstrings:

* 00000000
* 10000001

We guessed that each of these represented a single bit in the message, and we also knew from the challenge description that the packets we were looking for started with a magic number of 0x1ACFFC1D. We looked for a bit mapping that contained the magic, and decoded the bitstream, mapping 00000000 to 1 and 10000001 to 0.

Finally, we extracted the three packets contained in the bitstream and printed their contents, revealing the three parts of the flag.

## Solution scripts

### Bit mapping

# out.out comes from gnuradio  
with open('out.out', 'rb') as fin:  
 data = fin.read()  
  
bits = []  
for ch in data:  
 for b in bin(ch)[2:].rjust(8, '0'):  
 bits.append(int(b))  
  
out = []  
prev = 0  
  
for b in bits:  
 out.append(b ^ prev)  
 prev = b  
  
out = ''.join(map(str, out))  
msg = ''  
for i in range(0, len(out), 8):  
 if out[i:i+8] == '00000000': msg += '1'  
 else: msg += '0'  
  
print(msg)

### Final decoding

data = [  
 '000110101100111111111100000111010000000001100100000000000110010001111110001100010100000101000010001011010100001101000100010001010011001001000110010001110100100001001001001011010011000100000011111100000110011001101100011000010110011101111011011101110110100001101001011100110110101101100101011110010011100100110001001101000011011000110001001110000110110101101001011010110110010100110100001110100100011101001101011110100101011000110101001110000111100101000101011011010100110101110001010011110110011101010000010110100101010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011010011000000101111110111111111',  
 '000110101100111111111100000111010000000001100100000000000110010001111110001100010100000101000010001011010100001101000100010001010011001001000110010001110100100001001001001011010011000100000011111100000111011101010010011010100111010101001101011011100110110100110010010100000110100000101101010001110110101100110110010000010100111001000001011011010101100001001010010100010110101001000001001100000100010101101000011011010011000101000101010010000111000001001001001110010100110101001000001100110101010001000111011110100011010100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100011101010101101111110111111111',  
 '000110101100111111111100000111010000000001100100000000000110010001111110001100010100000101000010001011010100001101000100010001010011001001000110010001110100100001001001001011010011000100000011111100000100001100110100010010010100001101011000011110000110101101110101001110000100010101001010011010000110101000111001011101010100110001010000010110010011010001101100010000110111000001000100010100000111011101110011011011010100001101101000011100110101100101111101000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101111011001111101111110111111111',  
]  
  
for packet in data:  
 text = []  
 for i in range(0, len(packet), 8):  
 v = int(packet[i:i+8], 2)  
 text.append(v)  
 print(bytes(text))  
  
# flag{whiskey914618mike4:GMzV58yEmMqOgPZTwRjuMnm2Ph-Gk6ANAmXJQjA0Ehm1EHpI9MH3TGz5C4ICXxku8EJhj9uLPY4lCpDPwsmChsY}