and you, CPCR-505 ?

written by Mert SARICA | 1 March 2018

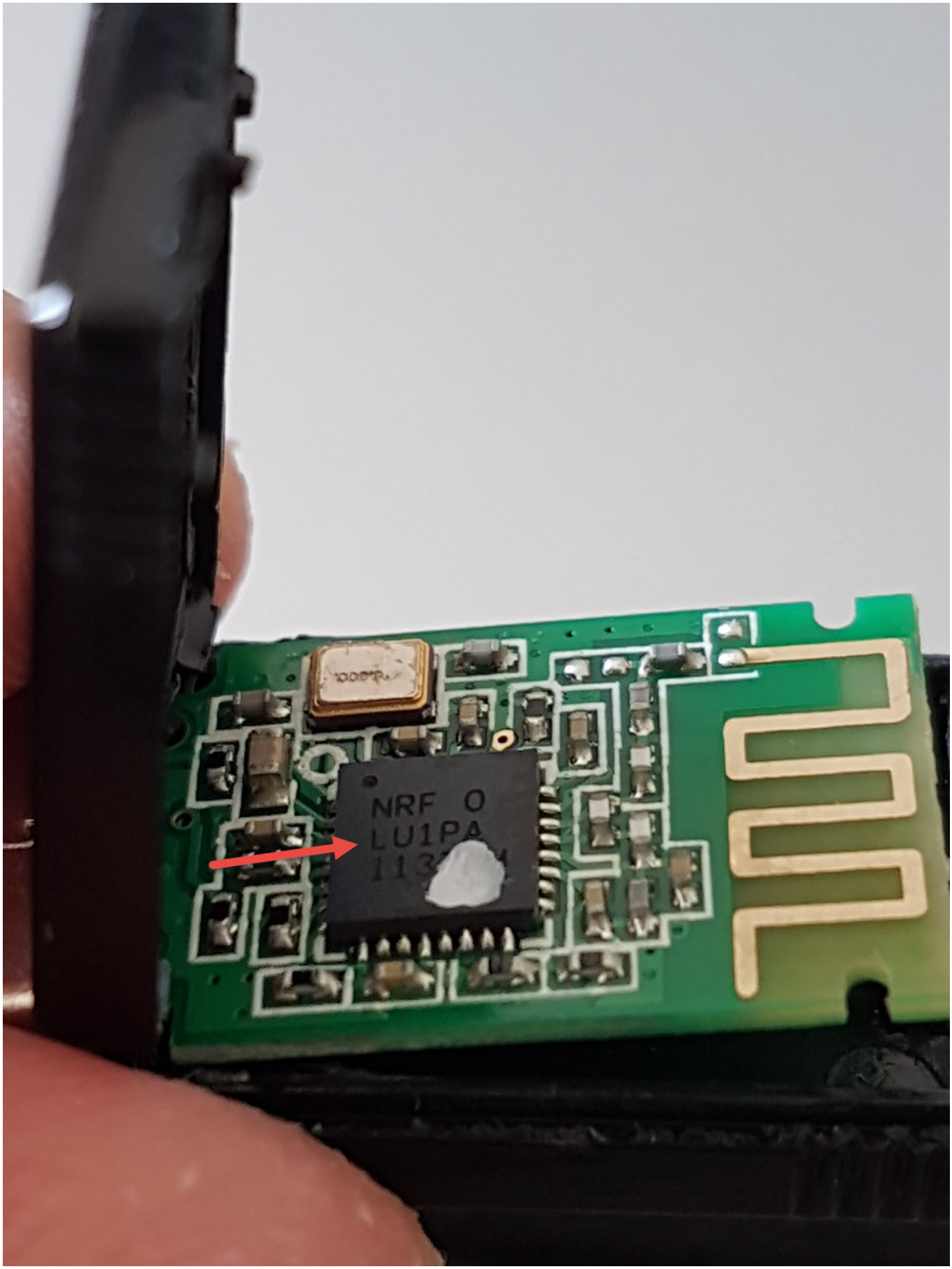
In the beginning of 2012, as I began receiving invitations to give presentations, I decided that my first task would be to purchase a wireless presentation remote for myself. As I searched for one that would not disappoint me in terms of price and performance, I came across the Codegen CPCR-505 presentation remote and immediately purchased it. Although my trusty presentation remote had served me well in the past, allowing me to give many enjoyable presentations, I would not learn until years later that it had the capability to backstab me, like Brutus. This realization would eventually become the topic of this article.



While giving presentations using my wireless presentation remote, or while watching someone else give a presentation, I would occasionally find myself wondering: “Could someone remotely hack the presentation remote and sabotage the presentation?” While some may say “Mert, it seems you have been watching

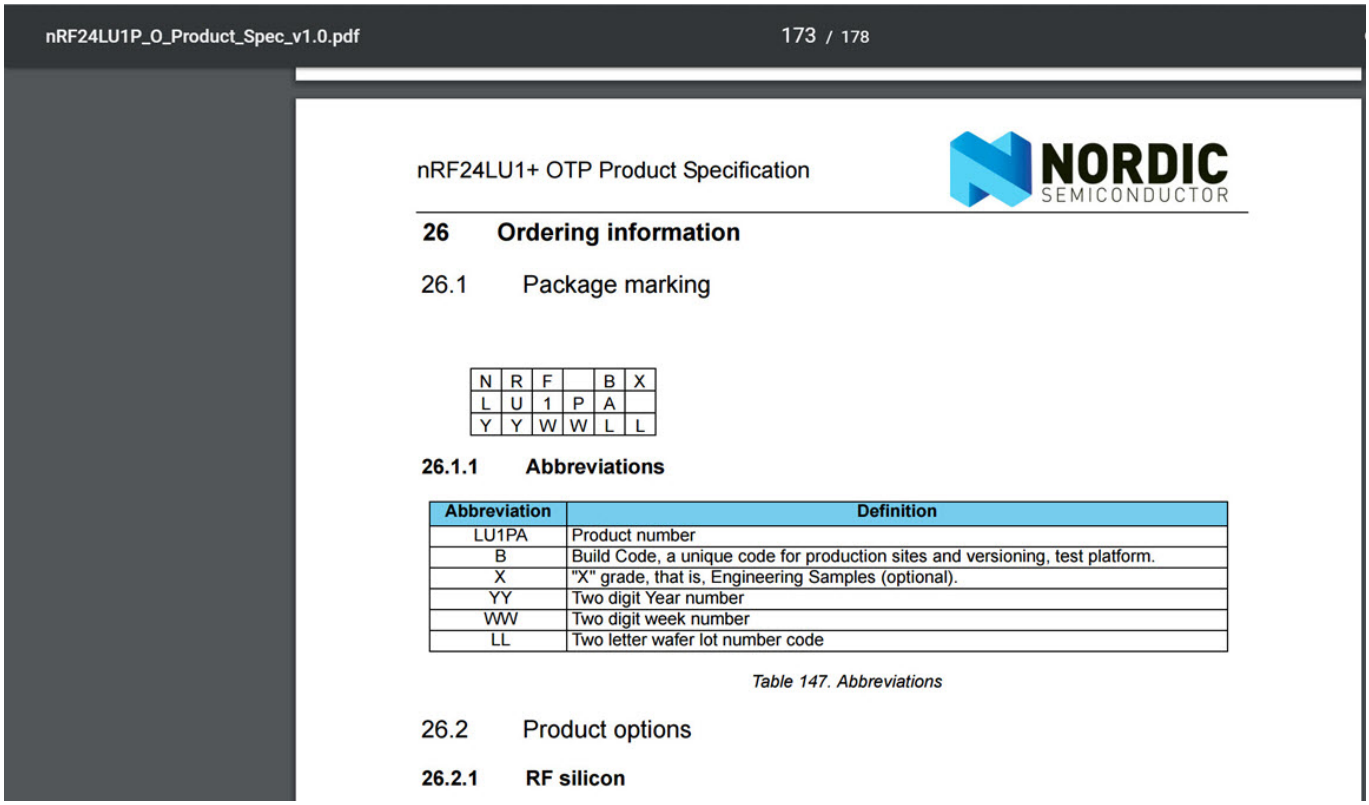
too much Mr Robot," I recently decided to investigate this possibility by closely examining my trusty presentation remote.

Upon inspecting the website of the remote's manufacturer, Codegen, I found that it clearly stated that the remote operates on the 2.4 GHz frequency band. As I had already researched this frequency band in my previous blog post on "Spy Mouse" and knew that wireless keyboards and mice also operate on this frequency, I decided to open up the USB receiver to gather more information.



When I opened the receiver, I came across the nRF24LU1PA 2.4 GHz receiver-transmitter chip. When I looked at the document for this chip on the website of the chip's manufacturer, Nordic Semiconductor, the first things that

caught my attention were the support for 125 RF channels, AES encryption support, and frequency hopping.



26 Ordering information

26.1 Package marking

N	R	F		B	X
L	U	1	P	A	
Y	Y	W	W	L	L

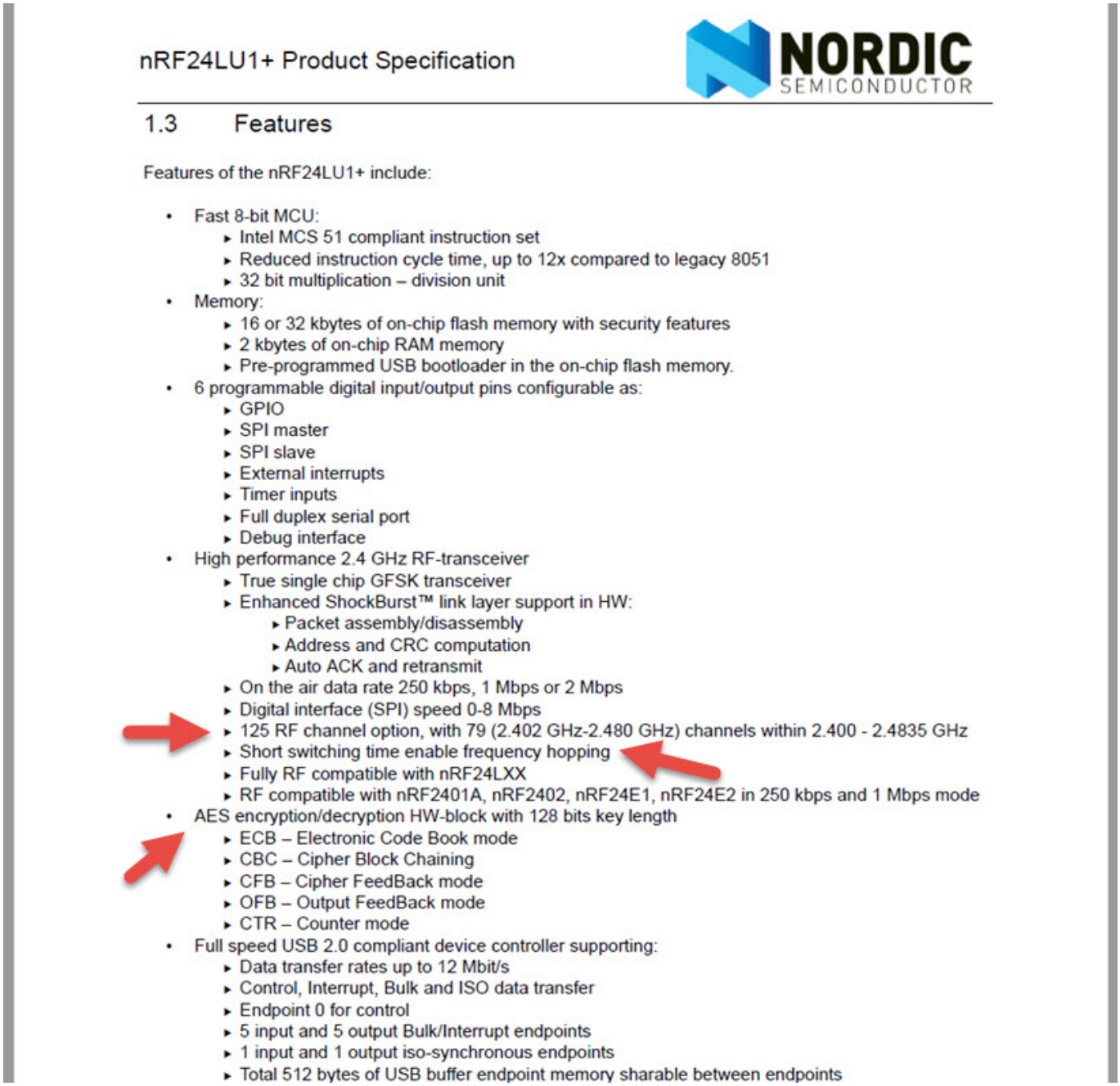
26.1.1 Abbreviations

Abbreviation	Definition
LU1PA	Product number
B	Build Code, a unique code for production sites and versioning, test platform.
X	"X" grade, that is, Engineering Samples (optional).
YY	Two digit Year number
WW	Two digit week number
LL	Two letter wafer lot number code

Table 147. Abbreviations

26.2 Product options

26.2.1 RF silicon



nRF24LU1+ Product Specification

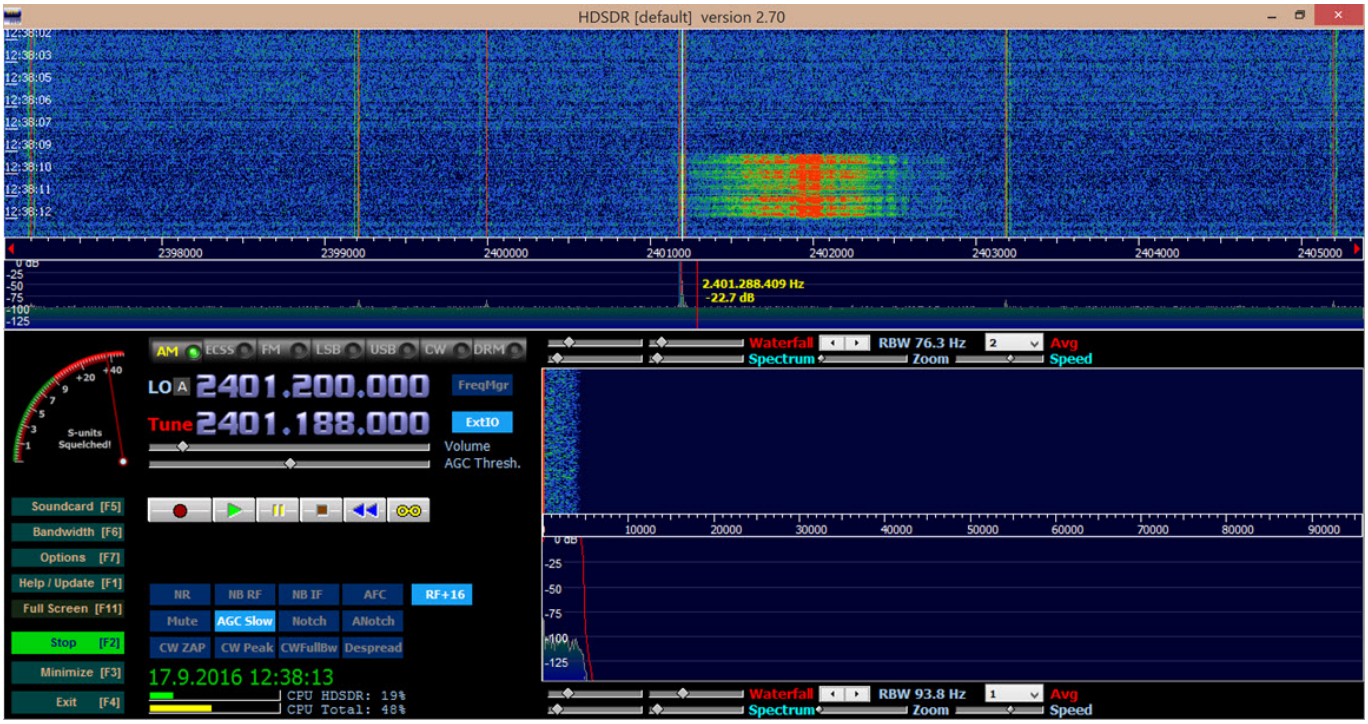
NORDIC SEMICONDUCTOR

1.3 Features

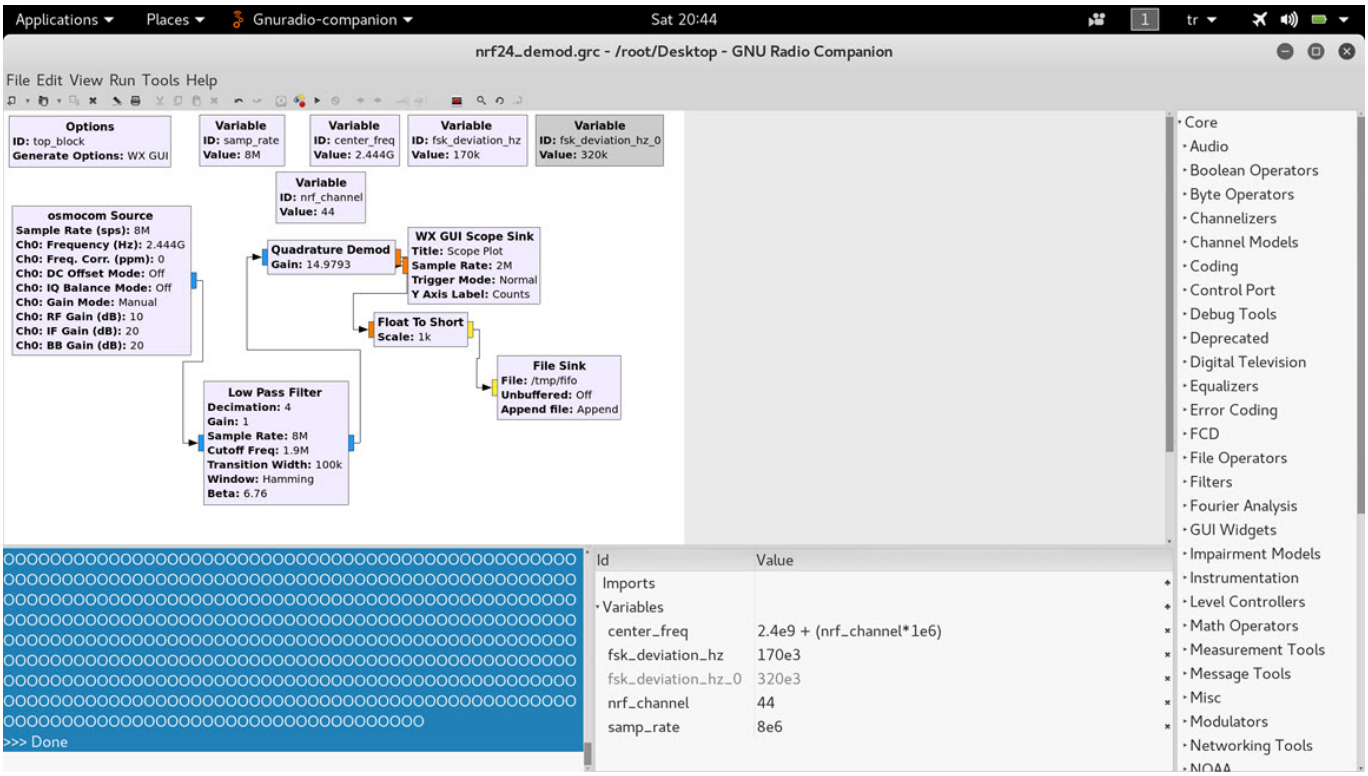
Features of the nRF24LU1+ include:

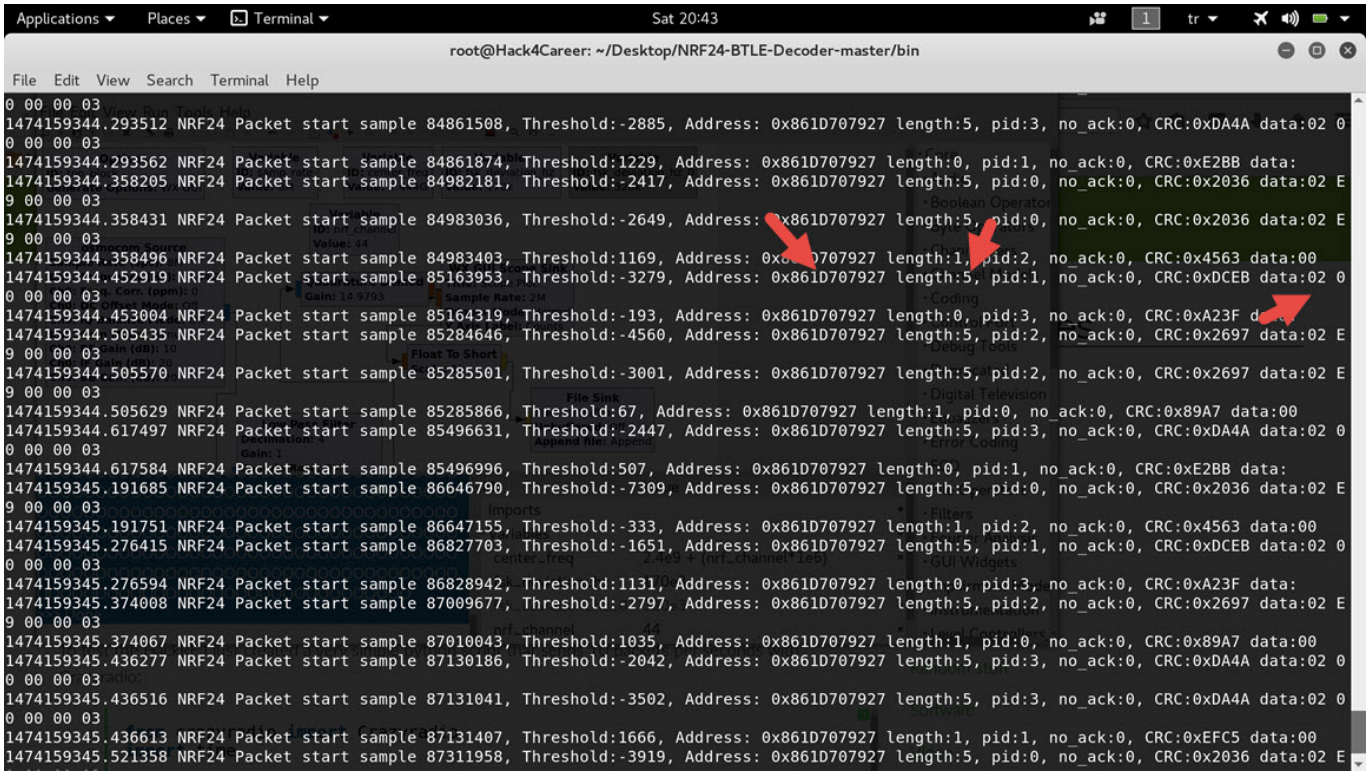
- Fast 8-bit MCU:
  - Intel MCS 51 compliant instruction set
  - Reduced instruction cycle time, up to 12x compared to legacy 8051
  - 32 bit multiplication – division unit
- Memory:
  - 16 or 32 kbytes of on-chip flash memory with security features
  - 2 kbytes of on-chip RAM memory
  - Pre-programmed USB bootloader in the on-chip flash memory.
- 6 programmable digital input/output pins configurable as:
  - GPIO
  - SPI master
  - SPI slave
  - External interrupts
  - Timer inputs
  - Full duplex serial port
  - Debug interface
- High performance 2.4 GHz RF-transceiver
  - True single chip GFSK transceiver
  - Enhanced ShockBurst™ link layer support in HW:
    - Packet assembly/disassembly
    - Address and CRC computation
    - Auto ACK and retransmit
  - On the air data rate 250 kbps, 1 Mbps or 2 Mbps
  - Digital interface (SPI) speed 0-8 Mbps
  - 125 RF channel option, with 79 (2.402 GHz-2.480 GHz) channels within 2.400 - 2.4835 GHz
  - Short switching time enable frequency hopping
  - Fully RF compatible with nRF24LXX
  - RF compatible with nRF2401A, nRF2402, nRF24E1, nRF24E2 in 250 kbps and 1 Mbps mode
- AES encryption/decryption HW-block with 128 bits key length
  - ECB – Electronic Code Book mode
  - CBC – Cipher Block Chaining
  - CFB – Cipher FeedBack mode
  - OFB – Output FeedBack mode
  - CTR – Counter mode
- Full speed USB 2.0 compliant device controller supporting:
  - Data transfer rates up to 12 Mbit/s
  - Control, Interrupt, Bulk and ISO data transfer
  - Endpoint 0 for control
  - 5 input and 5 output Bulk/Interrupt endpoints
  - 1 input and 1 output iso-synchronous endpoints
  - Total 512 bytes of USB buffer endpoint memory sharable between endpoints

In 2015, while attending the Black Hat Cybersecurity Conference, I purchased a HackRF One device that I had also used in my RF World and Security blog post. I started monitoring the communication between the USB receiver that I had connected to my computer with the presentation remote control using the HDSDR program.



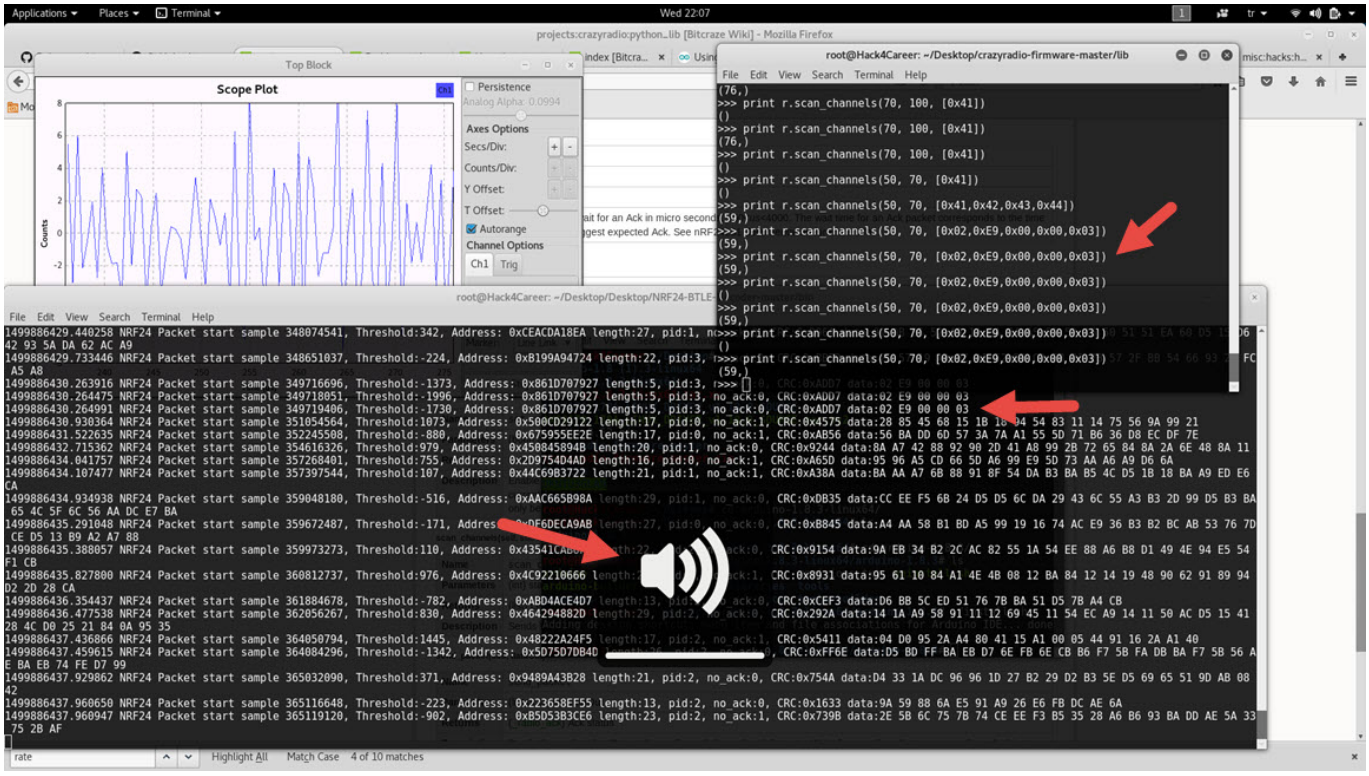
While thinking about how I could catch and decode the data packets due to the frequency hopping of the presentation remote, I came across an article on Bitcraze's Wiki page that dealt exactly with the topic I was looking for. By applying the specifications mentioned in the article letter by letter, I was able to easily obtain various data packets (address, size, data) resulting from the various buttons (volume increase, decrease, etc.) pressed on the presentation remote, from the mouse movements between the remote and the receiver.





The next step was deciding which device I would use to send the data packets to the receiver of the presentation remote. At the moment, I could do this with the Arduino Uno R3 and NRF24L01+ 2.4GHz receiver transmitter module that I had in my possession. Bill Gates once said in an interview “I always hire the laziest people because they are the ones who will find the shortest way to do a job.” and that’s what came in my mind, I was thinking about working with Arduino, module and cables then I remembered the CrazyRadio PA USB device that I used in my blog post titled “Spy Mouse“. With the Python library, it would be very easy for me to send any data packet to the desired receiver with just 5 lines of code, regardless of the operating system. Using the scan_channels() function, I sent a data packet containing the volume increase command to channels between 50-70 (by sending data packets to channels between 50-70, I increased the chance of the data packet reaching the receiver due to frequency hopping) and succeeded. ;)

```
import crazyradio
r = crazyradio.Crazyradio()
r.set_data_rate(r.DR_2MPS)
# Alıcı/Verici adresi
r.set_address((0x27, 0x79, 0x70, 0x1D, 0x86))
# Ses yükseltme komutu
print r.scan_channels(50, 70, [0x02,0xE9,0x00,0x00,0x03])
```



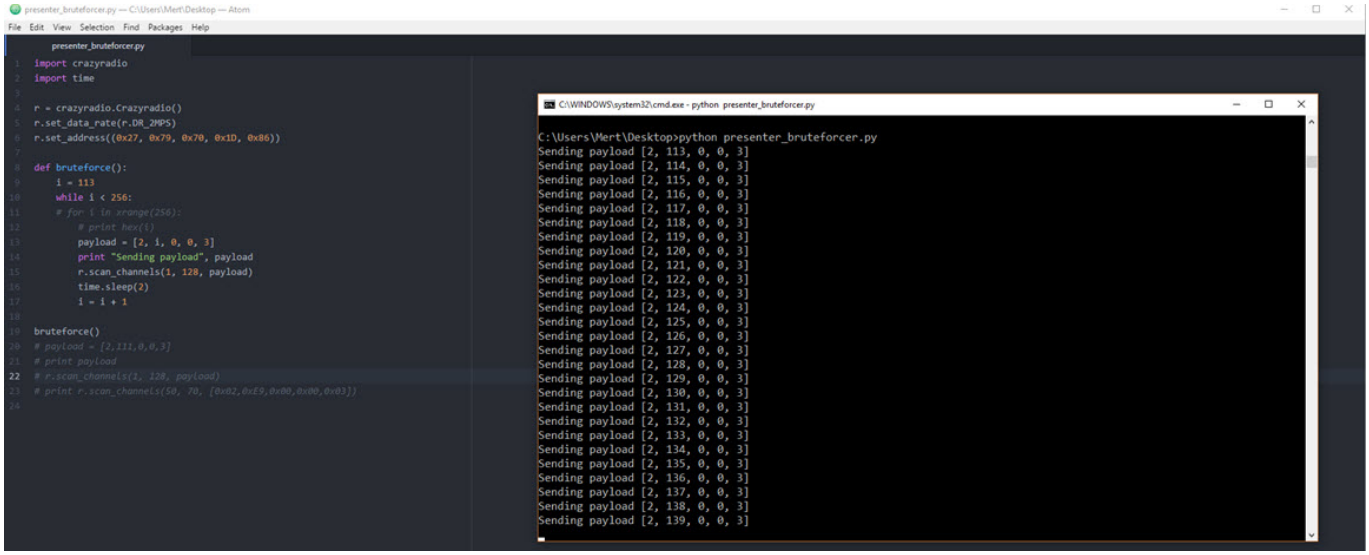
Of course, it is not very possible to sabotage a presentation by increasing or decreasing the volume, so I decided to detect hidden commands that the presentation remote does not normally send but that the receiver of the presentation remote supports, by using a trial and error (brute-force) method. For this, I wrote the following Python code.

```
import crazyradio
import time

r = crazyradio.Crazyradio()
r.set_data_rate(r.DR_2MPS)
r.set_address((0x27, 0x79, 0x70, 0x1D, 0x86))

def bruteforce():
    i = 0
    while i < 256:
        # Gi
        payload = [2, i, 0, 0, 3]
        print "Sending payload", payload
        r.scan_channels(1, 128, payload)
        time.sleep(2)
        i = i + 1
```

bruteforce()



It didn't take long for me to discover the following command, which does not exist on the presentation remote but is supported by the receiver of the presentation remote and reduces the screen brightness. I was able to reduce the screen brightness to a level that would make it very difficult to read, by sending this command to the receiver multiple times with the Crazy Radio PA USB device. This is really enough to sabotage a presentation and more. :)

[2, 111, 0, 0, 3] # Increases screen brightness.
[2, 112, 0, 0, 3] # Decreases screen brightness.

Taking into account that if keyboard key press commands are also sent to the receiver of the presentation remote, as in the Spy Mouse blog post, the situation could become a real threat to system security, I retired my presentation remote after this work and started looking for an encrypted presentation remote.

In similar unencrypted communications (e.g. drones) that take place on the 2.4 GHz (ISM band) frequency band, it is important to consider that similar security vulnerabilities may occur. I remind everyone to keep this in mind and wish everyone safe days until our next meeting.

Hope to see you in the following articles.