Paz Hameiri

TEMPEST radio station

About myself

- System engineer
- M.Sc. in Electro-Optical Engineering
- Six years of experience with telecommunication systems design and circuits
- Wrote “The Message Sticker” when I was a teenager

```
; UnPaz - A program to bypass Paz Hameiri's various stickers -
;         TXT to EXE, COM to EXE.
;
; Written by Inbar Raz, 2:401/100.1
;
; Version 1.0

UnPaz   Segment

        assume  cs:UnPaz, ds:UnPaz, es:UnPaz

        org     00100h

SEGSTART        EQU     $

Start:  Jmp     Begin
```

About TEMPEST

- TEMPEST is a U.S. NSA specification and a NATO certification
- The acronym refers to information leakage from a system through unintentional radio signals, audio signals, electrical signals, etc
- In 1985, Wim van Eck published the first unclassified analysis of the security risks of information leakage from computer monitors
- Government researchers were already aware of the danger

TEMPEST radio station ?!?!

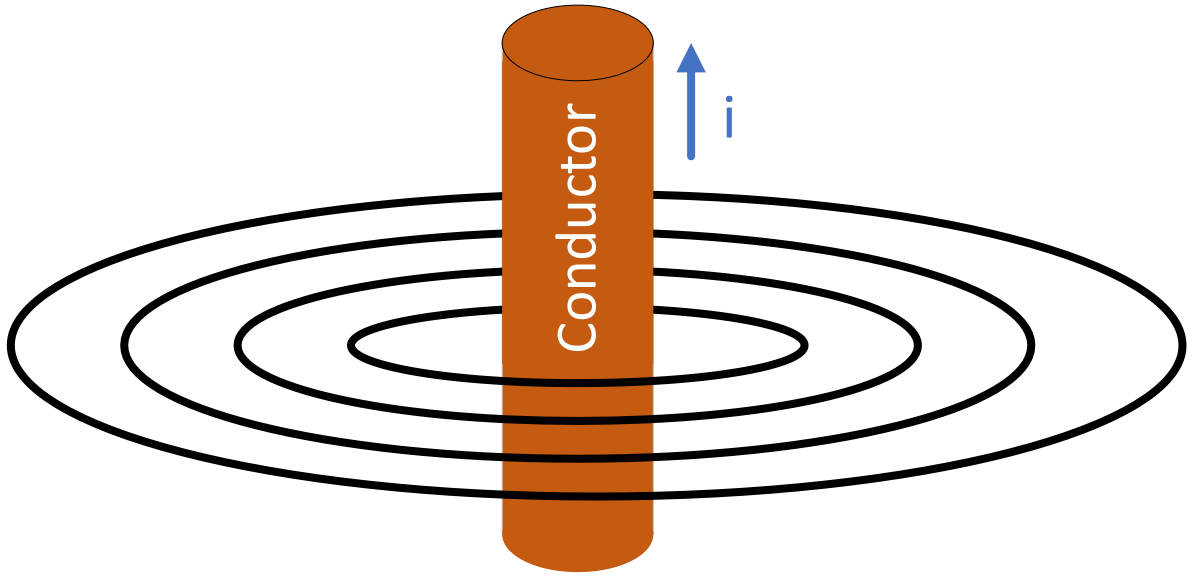
- I read “TEMPEST@Home - Finding Radio Frequency Side Channels” by Davidov & Oldenburg
- I bought an SDR receiver and studied the electromagnetic emissions generated by my laptop
- I wondered:
  - How far can I transmit data using these emissions?
  - Is it possible to transmit audio in real-time?
  - How hard can it be?

TEMPEST radio goals

- Tunable frequency:
  - Receive signals from a specific computer when several computers in the area are active
  - Select a bandwidth with as little interference as possible, to improve the signal to noise ratio
- Maximum bit rate, to maximize audio quality
- Innocent looking software, to avoid detection
- Maximum distance

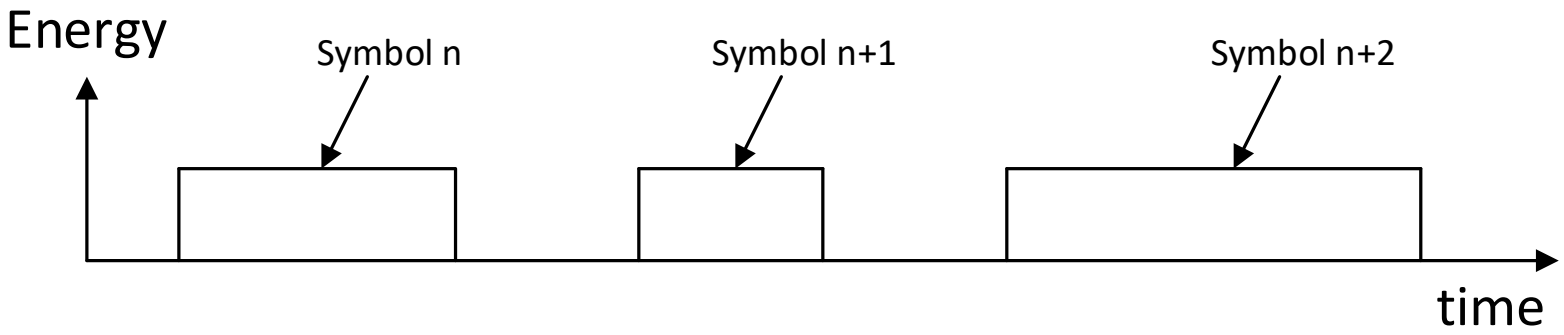
Radio waves crash course

- Transmission: Electromagnetic radiation is propagated from a conducting object, conducting time-varying electric current
- Reception: Electromagnetic radiation around a conducting object generate time-varying electric current



On-Off Keying crash course

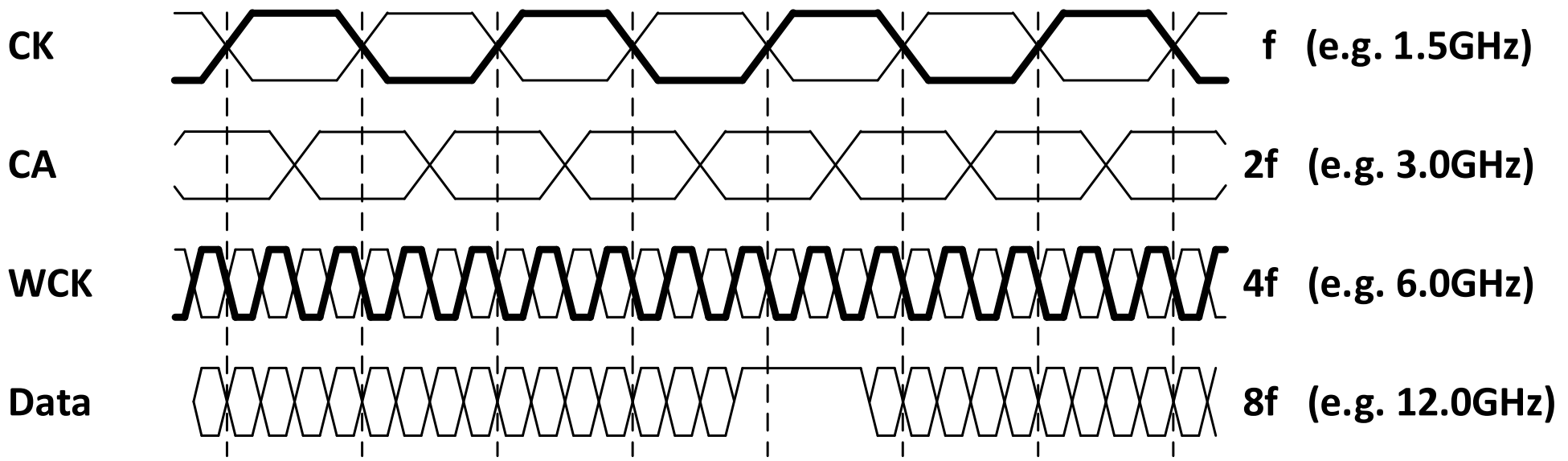
- On-Off Keying (OOK) data modulation represents digital data as the presence or absence of a transmitted wave
- Example: Morse code



Data transmission using GDDR 1

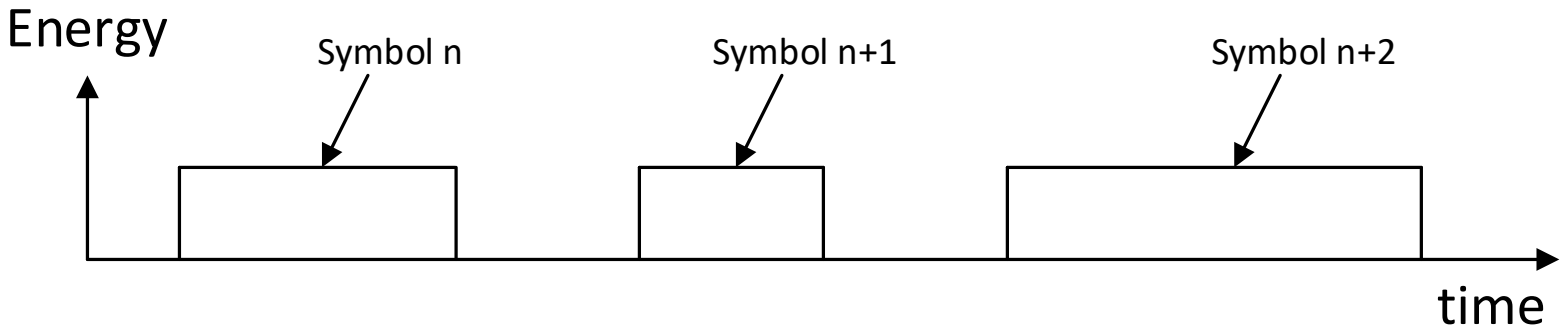
- The GPU performs memory read and write operations by operating the control and data lines

GDDR6 timing diagram:



Data transmission using GDDR 2

- Electromagnetic radiation is emitted when the control and data lines are active
- Data transmission is made by doing memory transfers:
  - A single symbol is transmitted during a single memory transfer
  - The number of bytes transferred at each memory transfer is predefined for each symbol



Data transmission using GDDR 3

Symbol duration ⇔ Symbol byte count

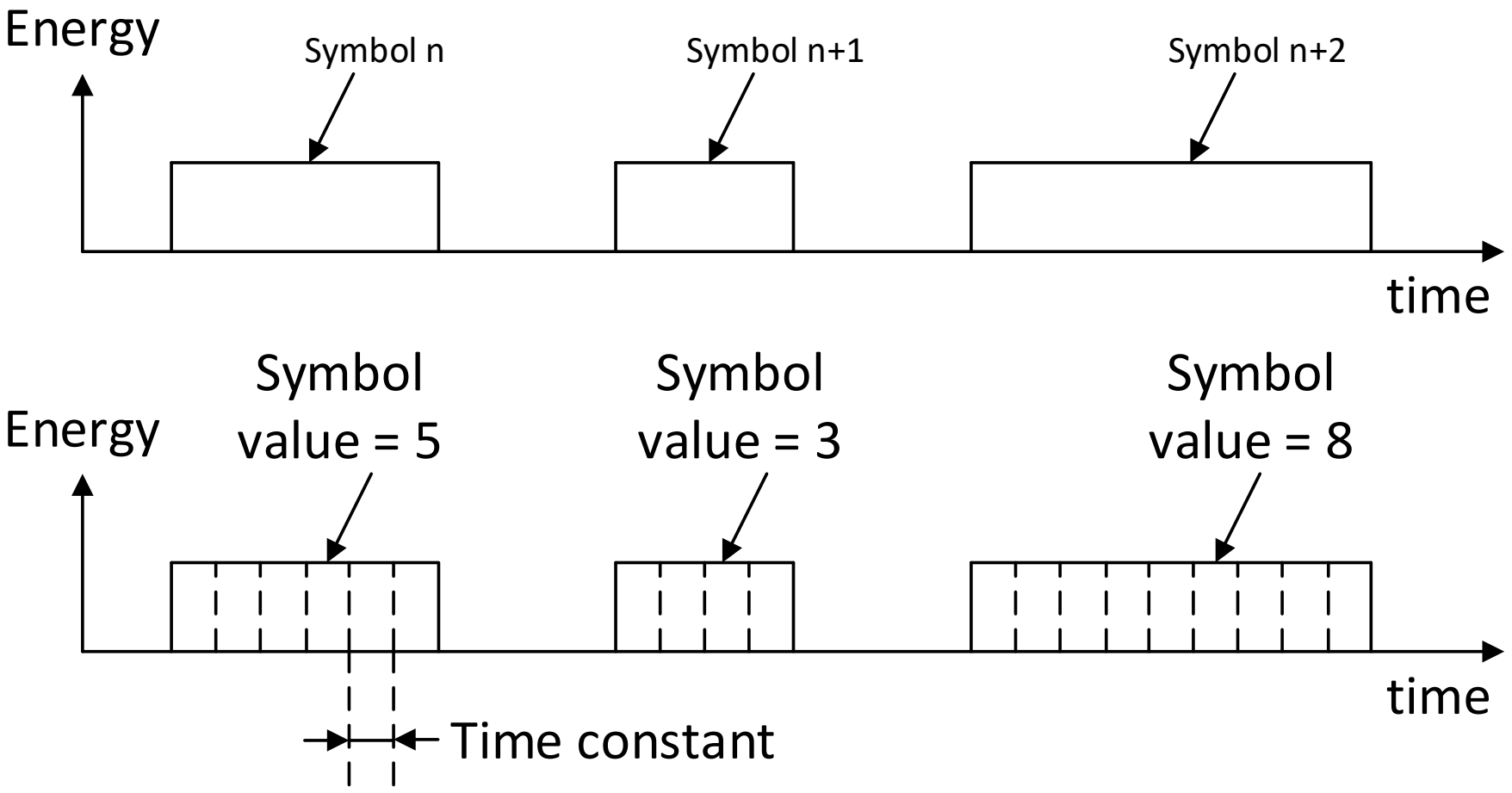
- Symbol duration [time] =

  (Symbol value + 1) * Time constant
- Symbol transfer size [bytes] =

  (Symbol value + 1) * Bytes constant

- Bytes constant = Amount of bytes required to perform a single memory transfer during a time period defined by the time constant

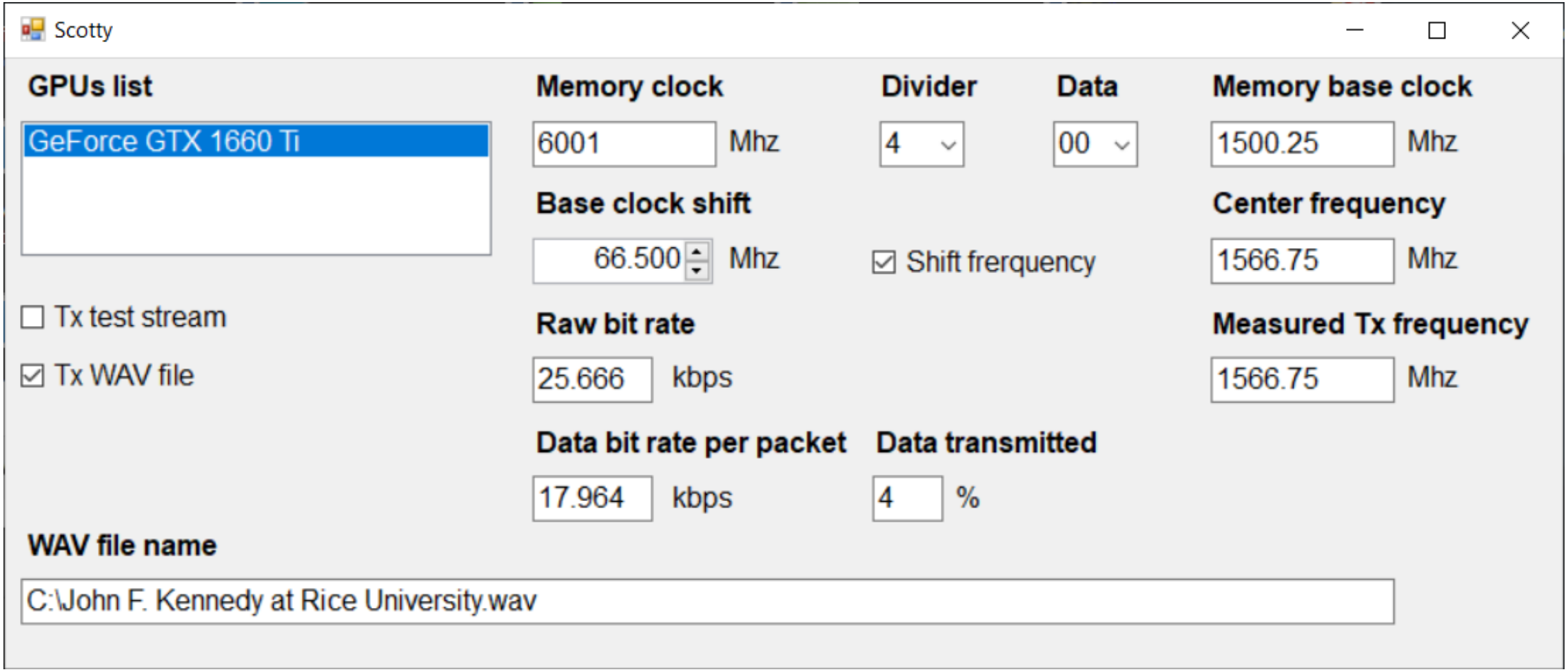
Data transmission using GDDR 4



Why using the GDDR SDRAM ?

- Tunable frequency
- Time deterministic data transfers
- Mostly idle when the GPU is not in use

Meet Scotty


Scotty
GPUs list
GeForce GTX 1660 Ti
Memory clock
6001
Mhz
Divider
4
Data
00
Memory base clock
1500.25
Mhz
Base clock shift
66.500
Mhz
Shift frerquency
Center frequency
1566.75
Mhz
Tx test stream
Tx WAV file
Raw bit rate
25.666
kbps
Measured Tx frequency
1566.75
Mhz
Data bit rate per packet
17.964
kbps
Data transmitted
4
%
WAV file name
C:\John F. Kennedy at Rice University.wav

Scotty's tasks

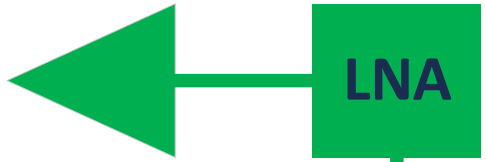
- Measuring the time required to perform large GPU memory transfers
- Calculating the bytes constant for a predefined time constant
- Setting GDDR memory clock frequency
- Loading a WAV file
- Transmitting 8000 audio PCM samples every second

During a one second interval

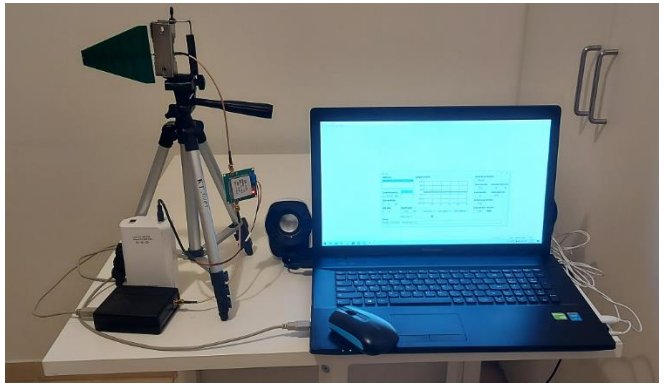
- Encoding 8000 audio PCM samples
- Bundling data into packets according to a protocol:
  - Header bytes
  - Reed-Solomon forward error correction parity bytes
  - Audio packets counter byte
  - G.726 encoded audio bytes
  - Audio data checksum bytes
- Transmitting each packet, symbol by symbol
- When all 8000 samples have been transmitted, the software stops and waits for the one-second interval to elapse

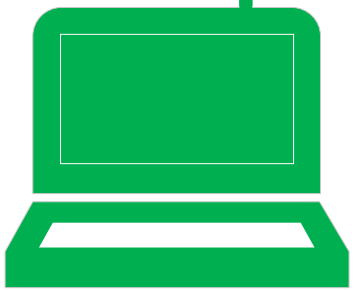
Radio setup

Scotty



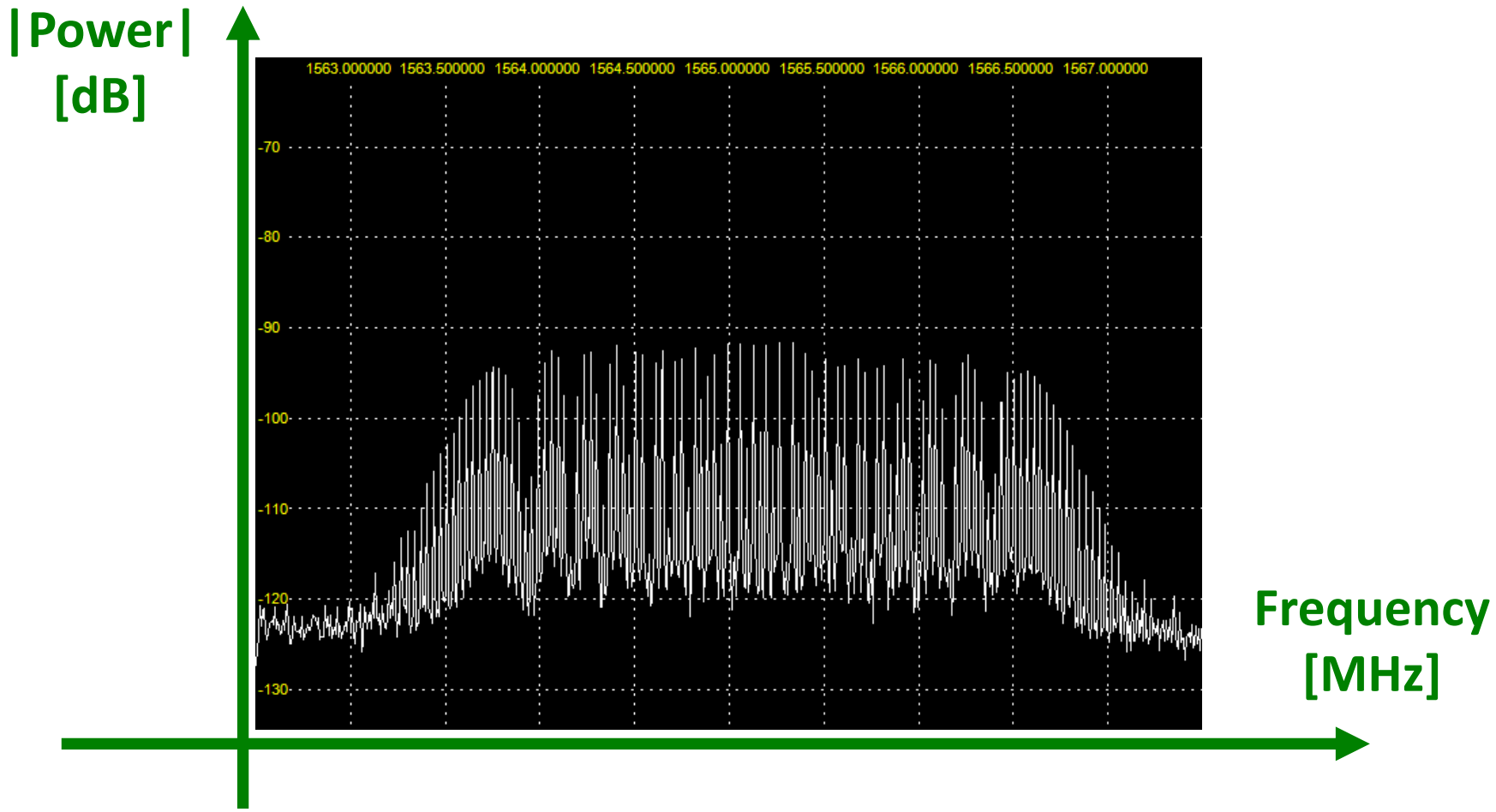
Antenna





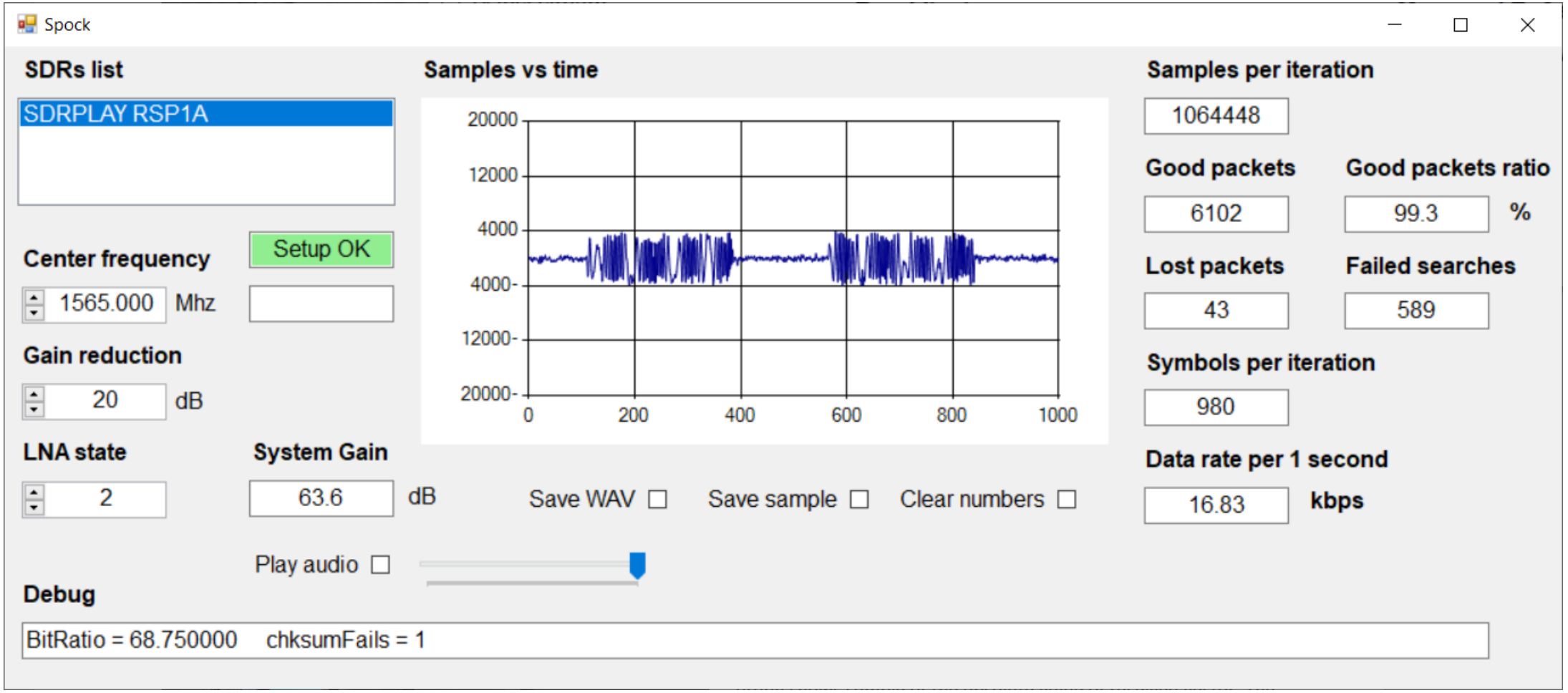
Spock

Target signal: GDDR6 CK



50 feet away from the source computer

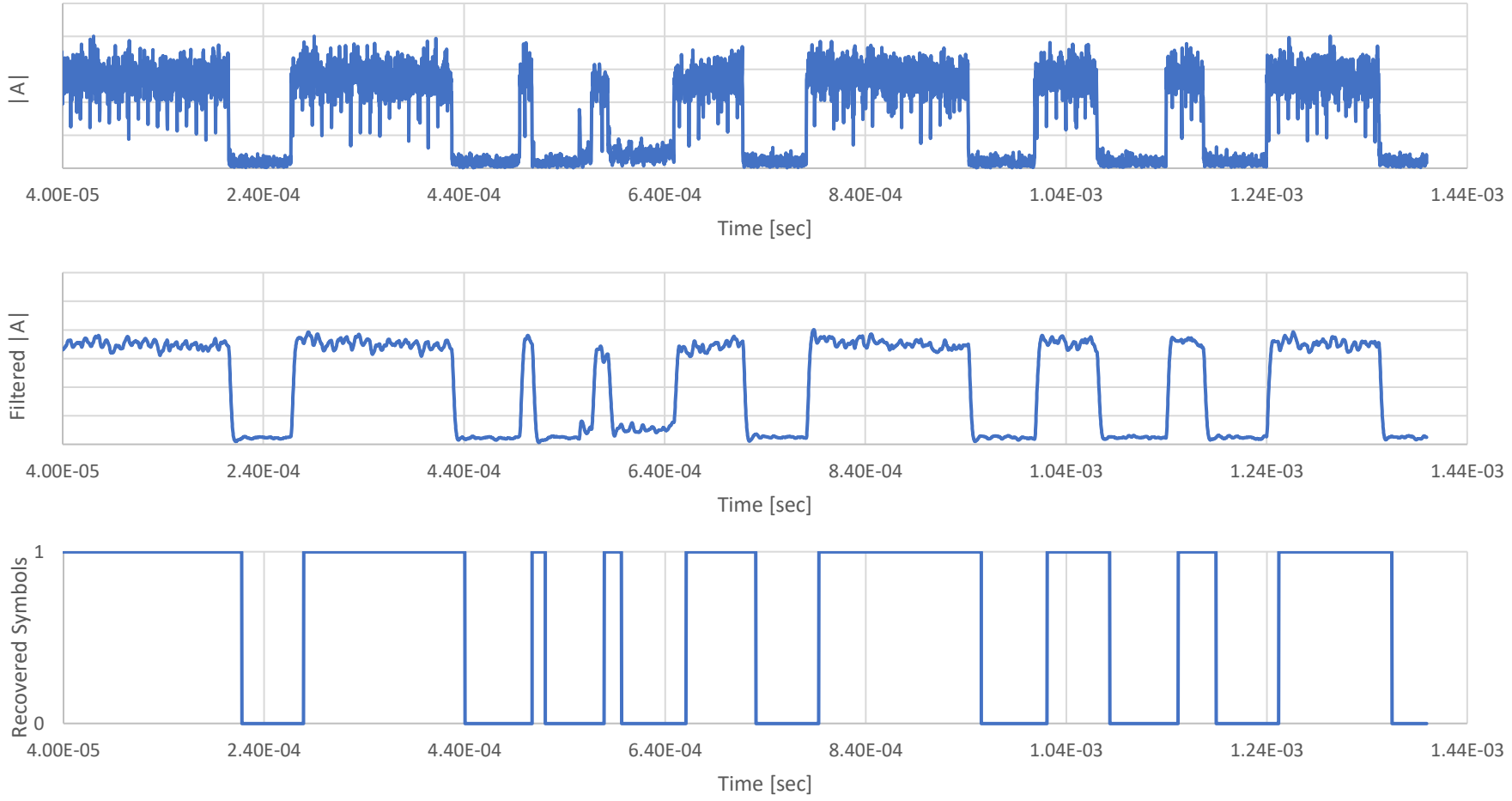
Meet Spock


Spock
SDRs list
SDRPLAY RSP1A
Center frequency
Setup OK
1565.000
Mhz
Gain reduction
20
dB
LNA state
System Gain
2
63.6
dB
Play audio
Debug
BitRatio = 68.750000 chksumFails = 1
Samples vs time
20000
12000
4000
4000-
12000-
20000-
0
200
400
600
800
1000
Save WAV
Save sample
Clear numbers
Samples per iteration
1064448
Good packets
Good packets ratio
6102
99.3
%
Lost packets
Failed searches
43
589
Symbols per iteration
980
Data rate per 1 second
16.83
kbps

Spock's tasks 1

- Setting up the SDR receiver
- Receiving cyclic batches of samples from the SDR receiver
- Calculating the absolute amplitude of the samples
- Filtering the data with a low pass filter
- Calculating amplitude thresholds to recover the symbols from the filtered data
- Recovering the symbols using the calculated amplitude thresholds and a minimum time threshold (to filter short-term noise).
- Saving the length of each symbol in a buffer

Samples to symbols



Spock's tasks 2

- Finding the header symbols
- Recovering the data packet from the symbols
- Using forward error correction decoding to correct errors
- Verifying packet validity
- Decoding the audio using a G.726 decoder
- Storing the PCM samples in a buffer
- Filling zeros for missing packets
- Playing the audio

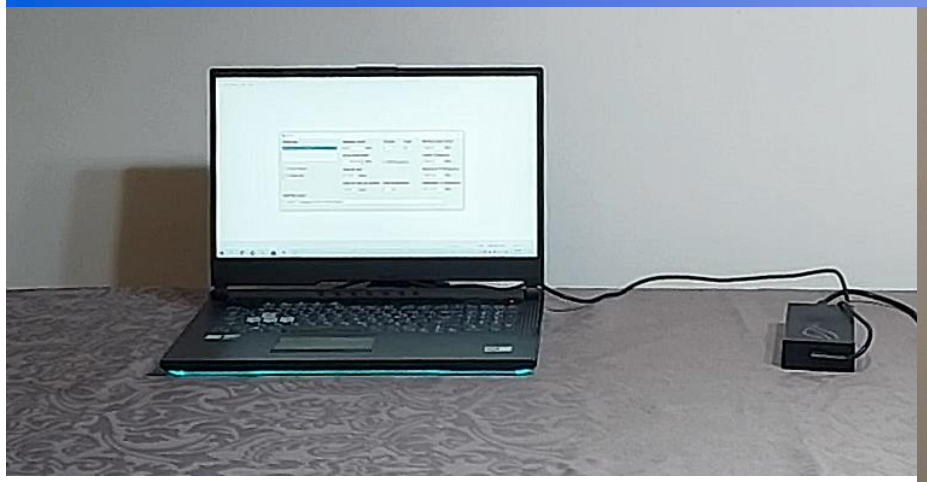
Tests setup 1

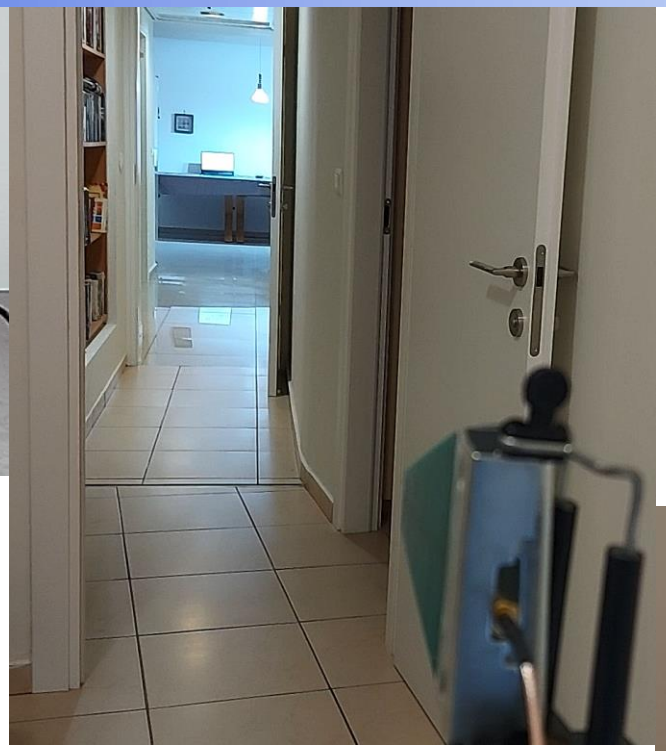
- Time constant = 14 µsec
- Data packet structure:
  - 4 header bytes
  - 20 Reed-Solomon forward error correction parity bytes
  - 1 audio packets counter byte
  - 63 G.726 encoded audio bytes, 2 bits per PCM sample
  - 2 audio data checksum bytes
- 4 bits per symbol

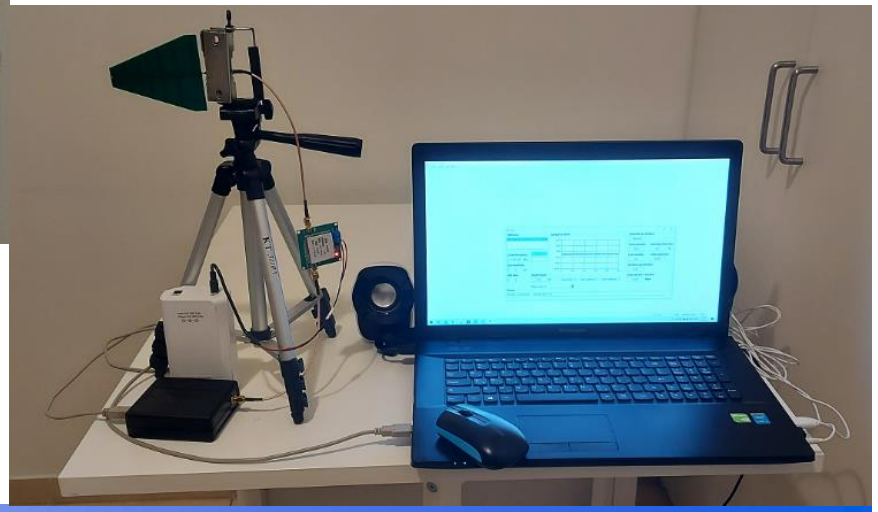
Tests setup 2

Computer	GPU	GDDR	Processor	RAM
Laptop	GTX 1660 Ti	6GB GDDR6	i7-9750H	16GB
Desktop	GTX 1650 Super	4GB GDDR6	i7-6700K	16GB

Tests setup 3

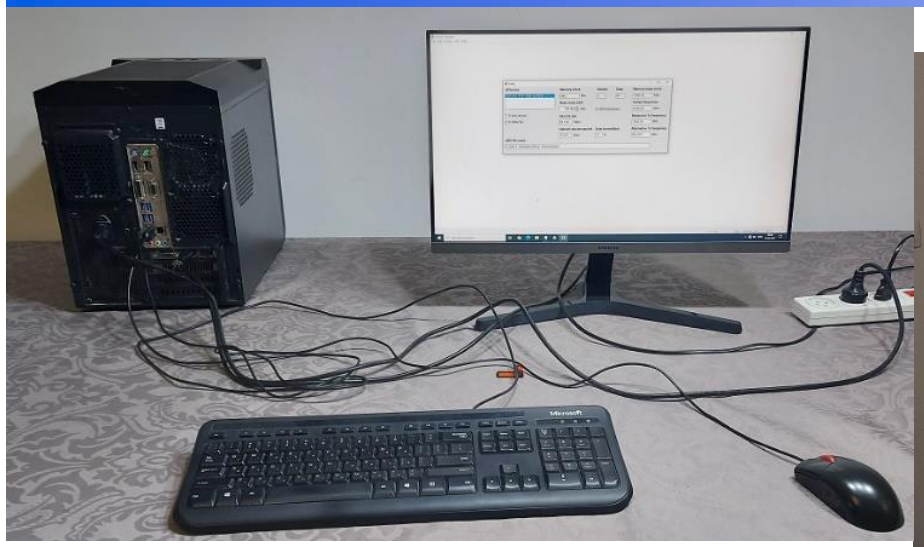


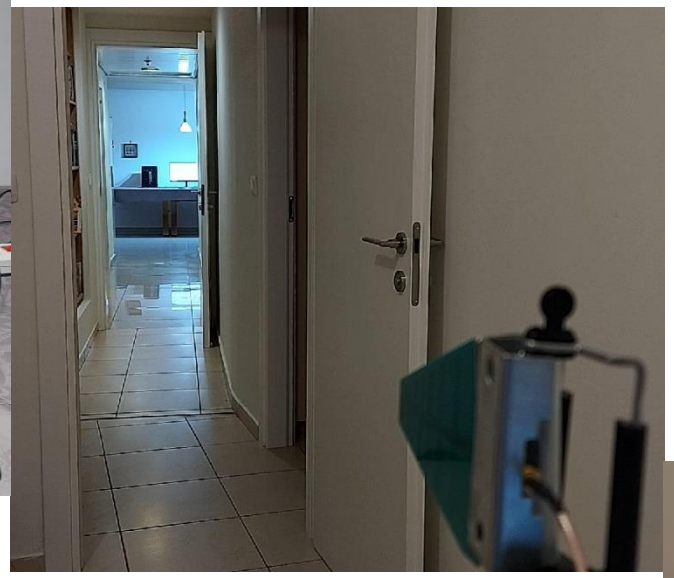


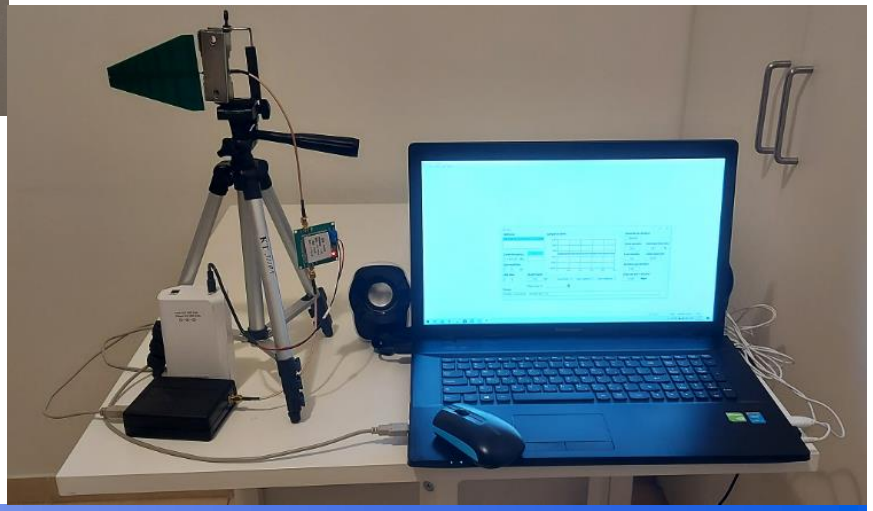


50 feet apart

Tests setup 4







50 feet apart

Tests video clip 1

Tests results 1

Computer	Average bit rate [kbit/s]	Valid packets received with the monitor turned on [%]	Valid packets received with the monitor turned off [%]
Laptop	26	> 99	Irrelevant
Desktop	23	89.5	> 99

Improve audio quality

- Tests showed that the desktop computer emitted signals which Scotty did not generate
- The computer stops transmitting these signals once the monitor is turned off by the Windows power plan
- When the monitor is off higher bit rate can be achieved
- Maximum audio quality setup:
  - Time constant = 8 µsec
  - 4 Reed-Solomon forward error correction parity bytes
  - G.726 encoder: 3 bits per PCM sample

Tests video clip 2

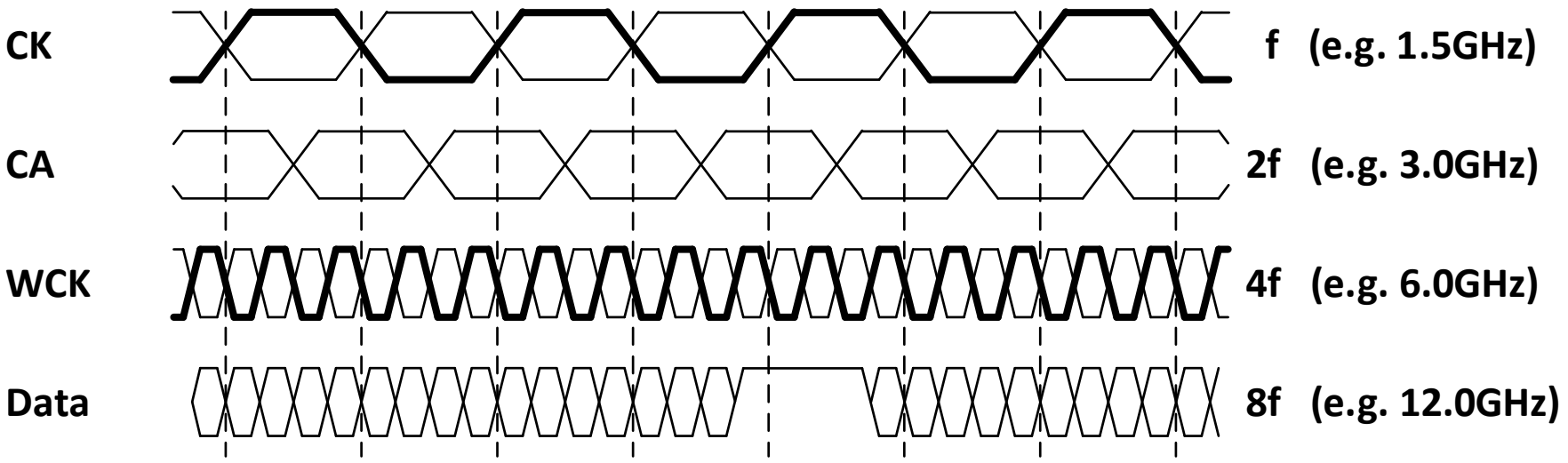
Tests results 2

Computer	Average bit rate [kbit/s]	Valid packets received with the monitor turned on [%]	Valid packets received with the monitor turned off [%]
Laptop	33	> 99	Irrelevant
Desktop	30	Low	> 99

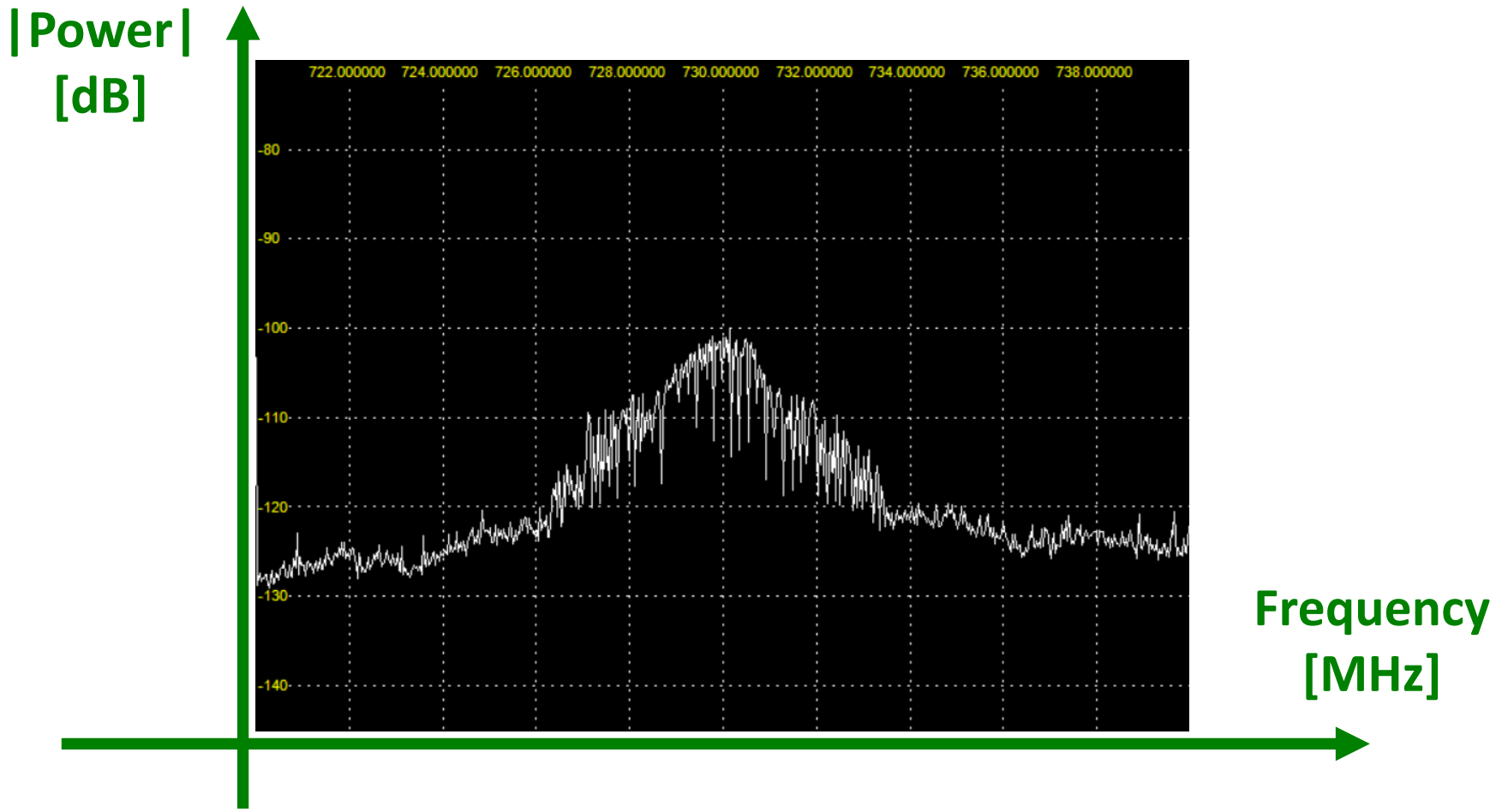
Multiple emissions per operation

- During every memory operation, electromagnetic waves are emitted at multiple frequencies

GDDR6 timing diagram:

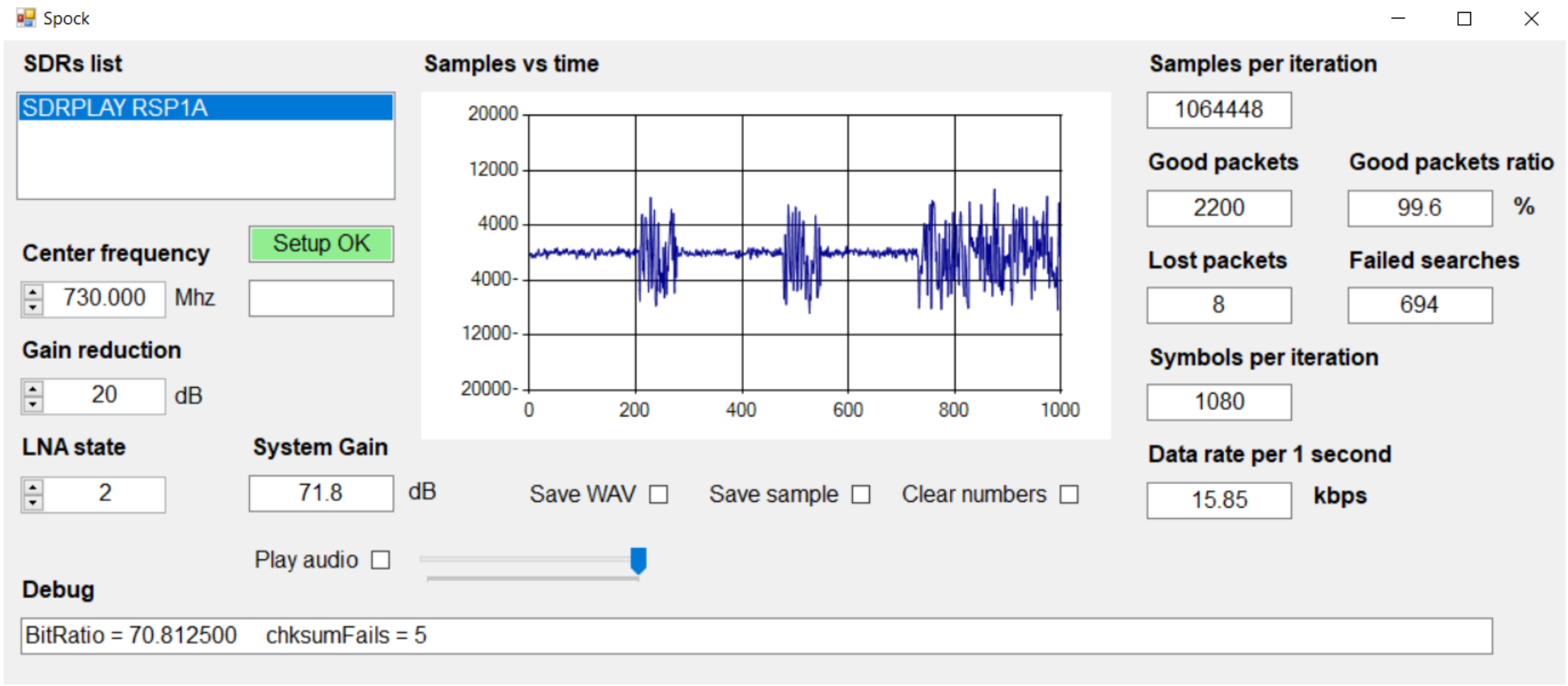


Target signal: GDDR6 CK / 2



Laptop, close range, without the LNA, CK = 1461.25MHz

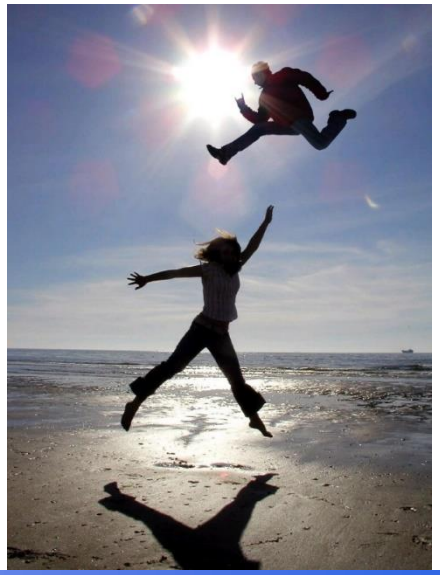
Spock at CK/2



Laptop, close range, without the LNA, CK = 1461.25MHz

Fun conclusions

- It works
- The apartment is too small for the range tests
- I've made a jingle:



Alarming conclusions

- Timed memory transfers are easy to produce
- The method can be used to silently leak data as well
- The method can be used to leak audio and data out of air-gapped computers
- Especially during non-working hours:
  - No supervision
  - The monitor can be turned off to achieve maximum bit rate
  - The attacker can select the time of the transmissions

Examples of usage

- This data extraction method is not supervised by anti-virus software, firewalls, port monitoring software, etc.
- The technique might be used for:
  - Extracting confidential plans and designs from internal networks
  - Extracting confidential files from executives in the company one works for
  - Extracting data from colleges who work on confidential projects

Thank you

- Source code:
  - https://github.com/TEMPESTRadioStation/Scotty
  - https://github.com/TEMPESTRadioStation/Spock

Thank you

- References:

  - Davidov, M., Oldenburg, B., “TEMPEST@Home - Finding Radio Frequency Side Channels” 2020. https://duo.com/labs/research/finding-radio-sidechannels
  - Eck W. “Electromagnetic radiation from video display units: an eavesdropping risk?” Computers and Security, 4, no. 4: 269-286, 1985.
  - Kuhn, M. G., and Anderson, R. J. Soft. “Tempest: Hidden Data Transmission Using Electromagnetic Emanations.” In Information Hiding (1998), ed. D. Aucsmith, vol. 1525 of Lecture Notes in Computer Science, (Springer): 124–142.
  - Thiele, E., “Tempest for Eliza.” 2001. http://www.erikyyy.de/tempest/.
  - Kania B., “VGASIG: FM radio transmitter using VGA graphics card.” 2009. http://bk.gnarf.org/creativity/vgasig/vgasig.pdf.
  - Guri M., Kedma G., Kachlon A., Elovici Y. “AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies.” In Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on IEEE, 2014: 58-67.
  - 2pkaqwtuqm2q7djg,"OVERCLOCKING TOOLS FOR NVIDIA GPUS SUCK, I MADE MY OWN". 2015. https://1vwjbxf1wko0yhnr.wordpress.com/2015/08/10/overclocking-tools-for-nvidia-gpus-suck-i-made-my-own/
  - nvapioc project: https://github.com/Demion/nvapioc
  - SDRplay API Specification v3, https://www.sdrplay.com/docs/SDRplay_API_Specification_v3.pdf
  - Simon Rockliff's Reed-Solomon encoding-decoding code at http://www.eccpage.com/rs.c