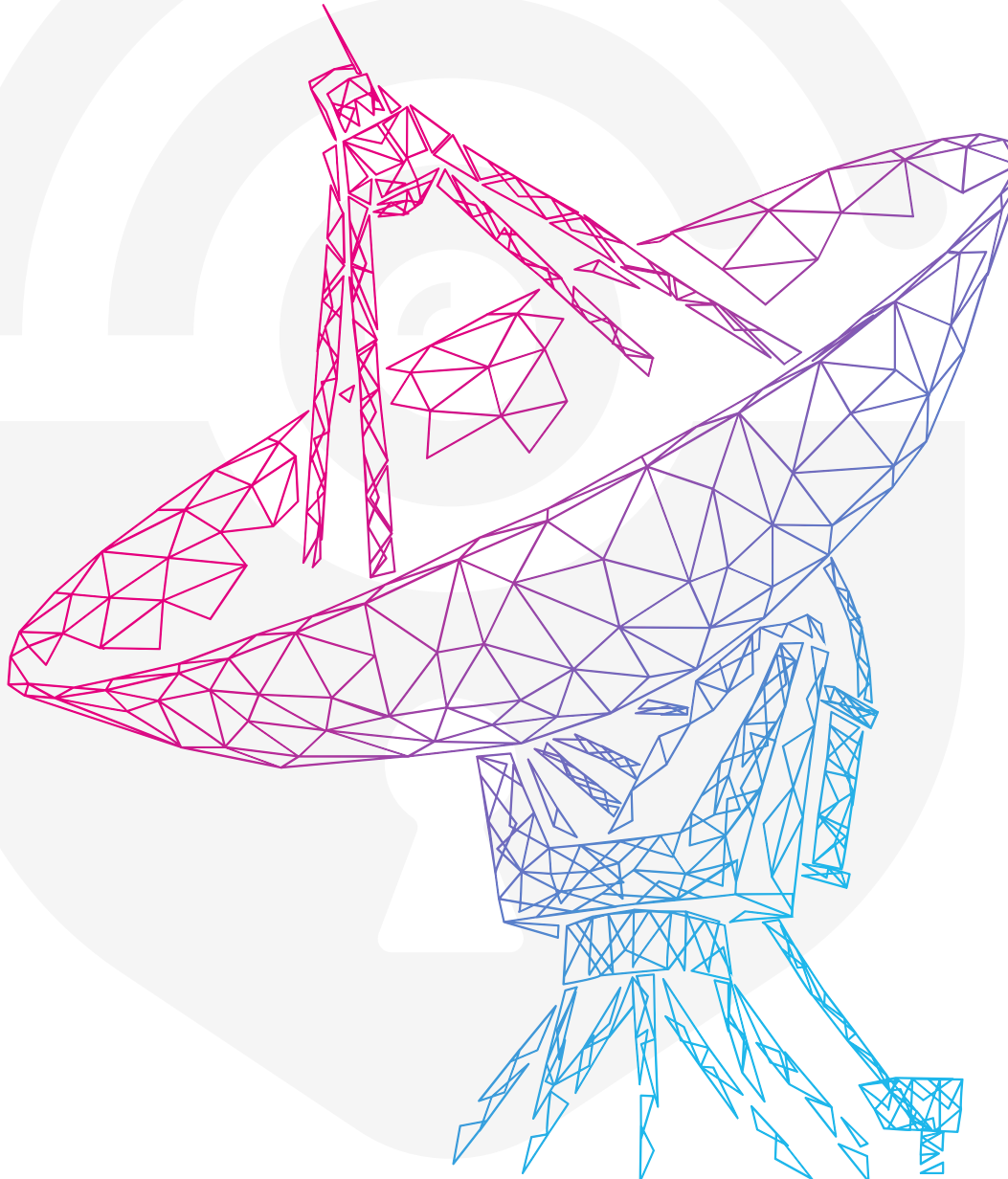
penthertz

RF Signal Hunting

With the help of ML and DL



Founder of Penthertz

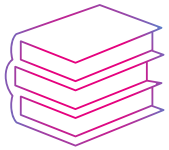
- Sébastien Dudek (@FlUxIuS)
- > 10 years of experience in Software / Hardware security
  - Former Sogeti ESEC R&D as a researcher
  - and Synacktiv as a pentester + vulnerabilities researcher
- Specialized in Wireless communications
- Also, security researcher @Trend Micro
  - Industrial IoT
  - Mobile and other RF/SDR things…

TREND MICRO | research

Main activities

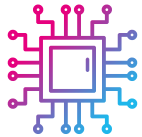
Security assessments

- Wireless communications (RFID, Wi-Fi, Mobile communication, Bluetooth, etc.)
- Embedded devices
- Backend servers
- Red Team



Trainings

- Software-Defined Radio Hacking
- Wi-Fi Red teaming
- RFID Hacking
- Mobile attacks (2G/3G/4G/5G)



Hardware security

- Firmware extraction
- Chip-off
- Secrets extraction
- Libraries analysis
- Vulnerability hunting

Part of our RF lab (wow clean desk)

RF Signal

“

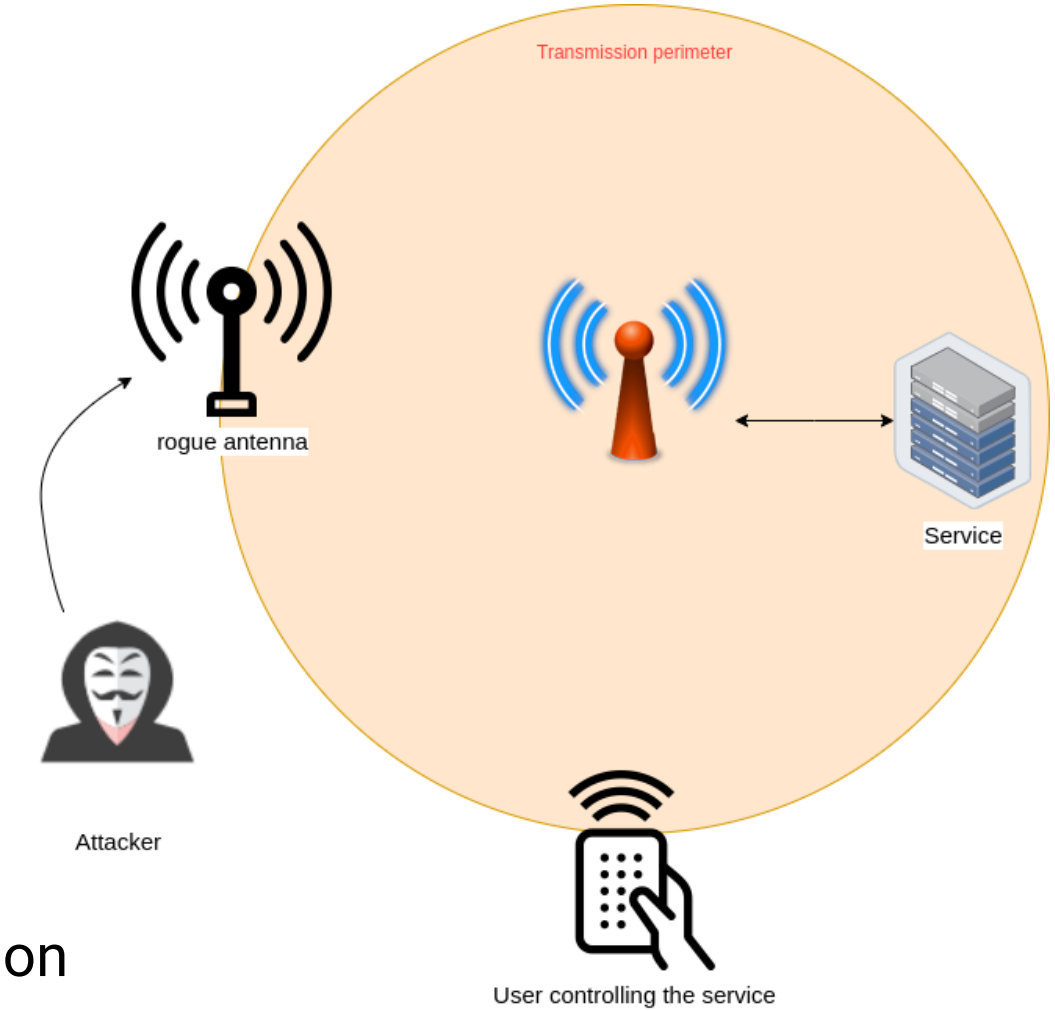
We just have these mysterious electromagnetic waves that we cannot see with the naked eye. But they are there.

Heinrich Hertz

Risks OTA

Common vulnerabilities:

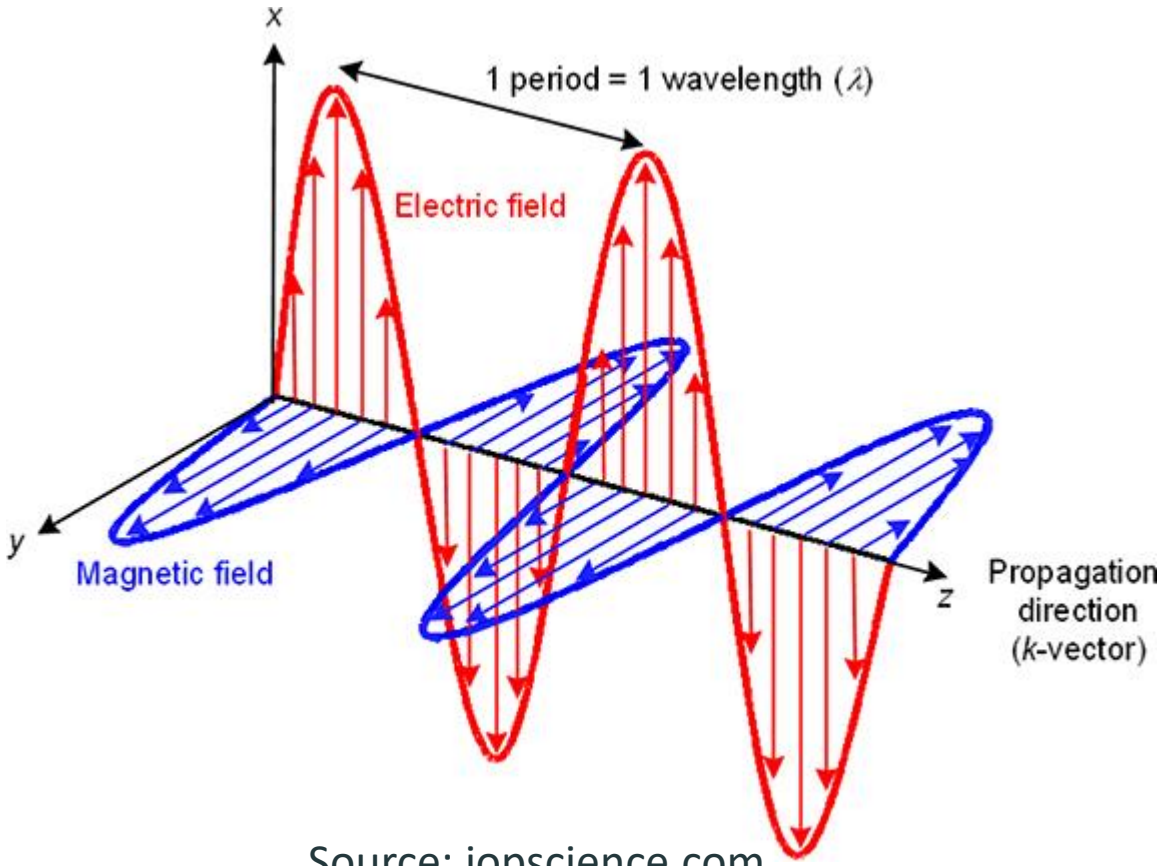
- Eavesdropping
- Replay
- Injection
- Relay
- Jamming
- DoS via (very) high amplitude transmission
- etc.



Radio wave characteristics

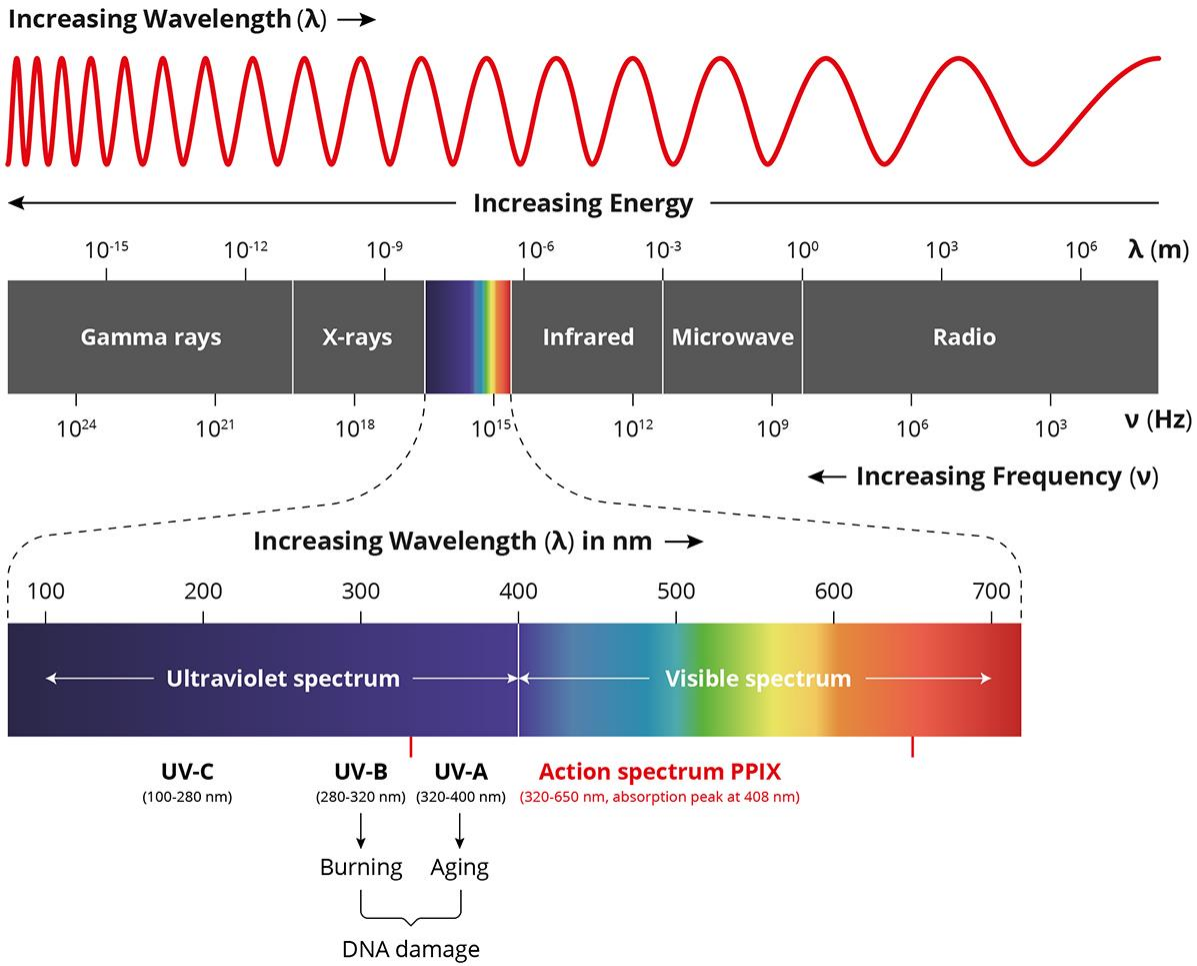
Important:

- λ: wavelength in meter
- c: celerity of light (3.108 $m.s^{-1}$)
- T: the period in seconds ($\frac{\lambda}{c}$)
- f: frequency in Hz ($\frac{1}{T}$)



Source: iopscience.com

Wave regions



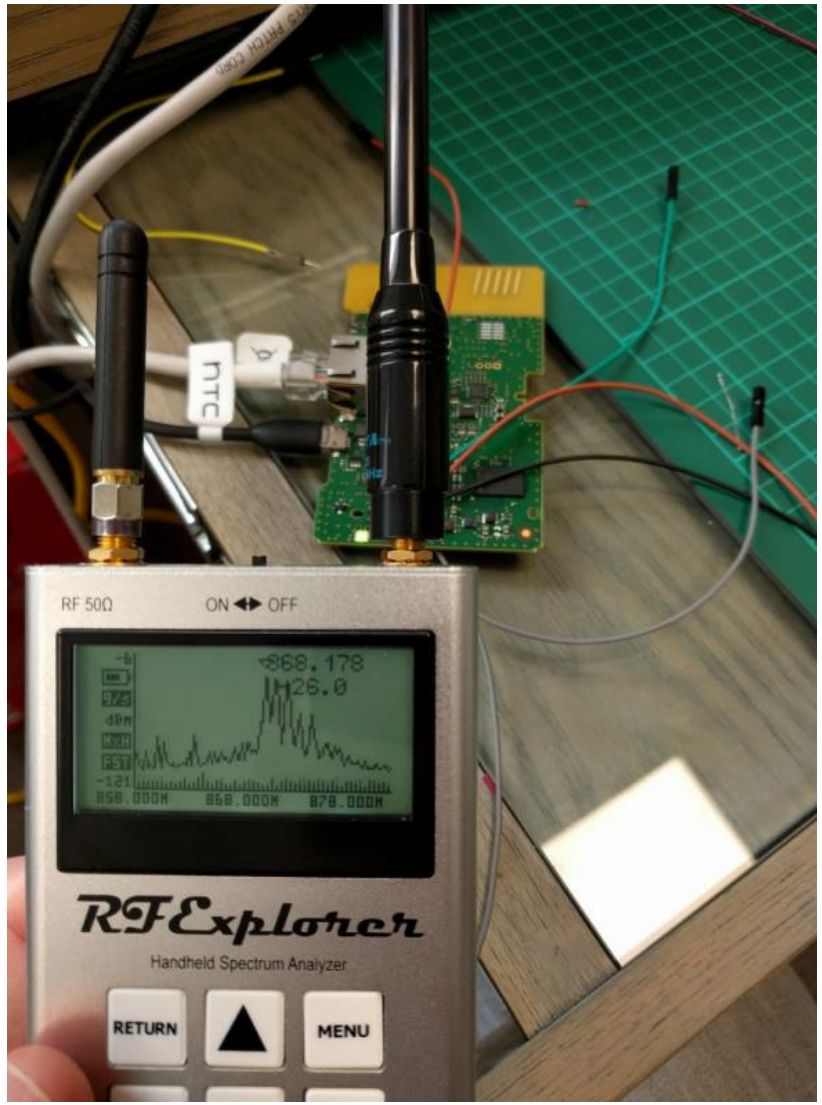
Identification

Spectrum analyzers

- A starting beast:
  - R&S® FSV40
  - 10 Hz to 40 GHz freq band
  - 160 MHz signal analysis bandwidth
  - costs > 43 000€

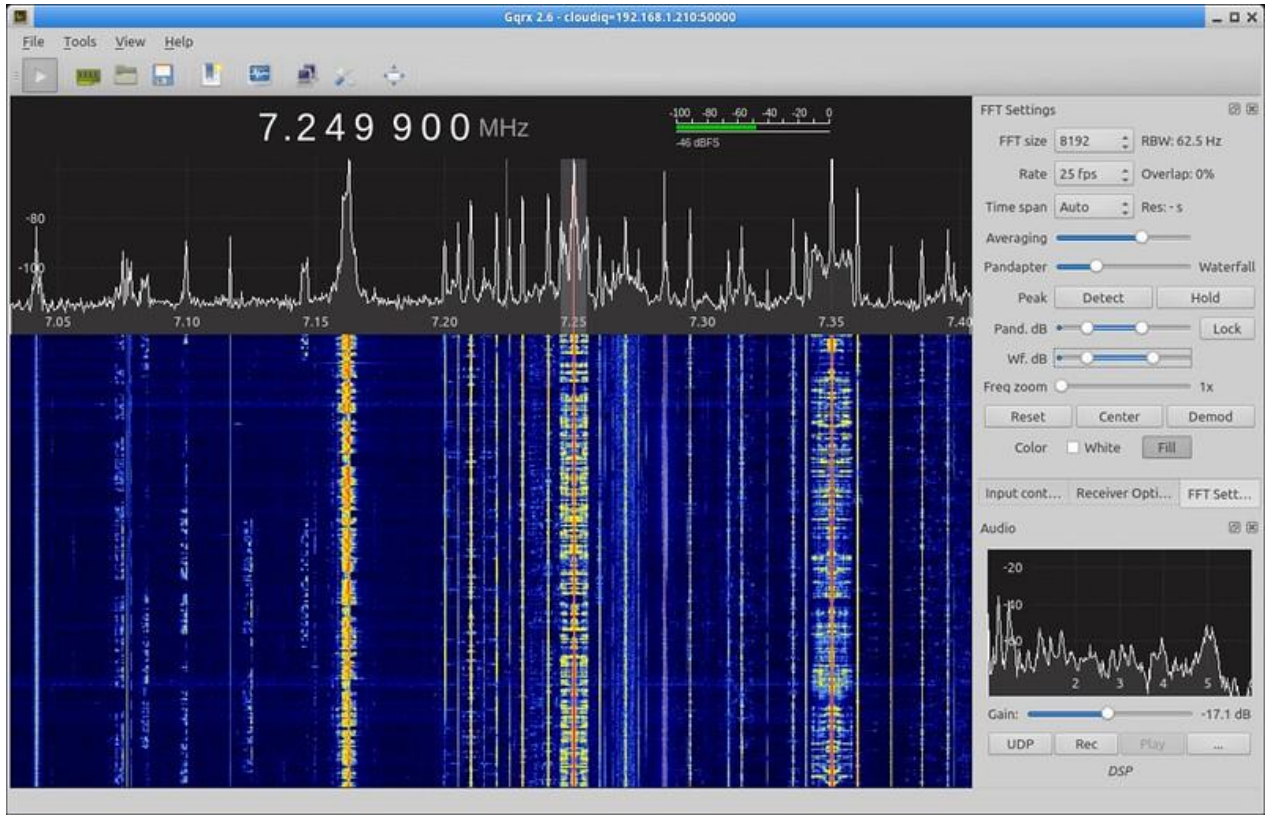
RF Explorer

- A handy gadget for < 300€:
  - Left SMA port (WSUB1G): 240-960MHz
  - Right SMA port (WSUB3G): 15-2700 MHz
  - Resolution bandwidth (RBW): automatic 3Khz to 600Khz



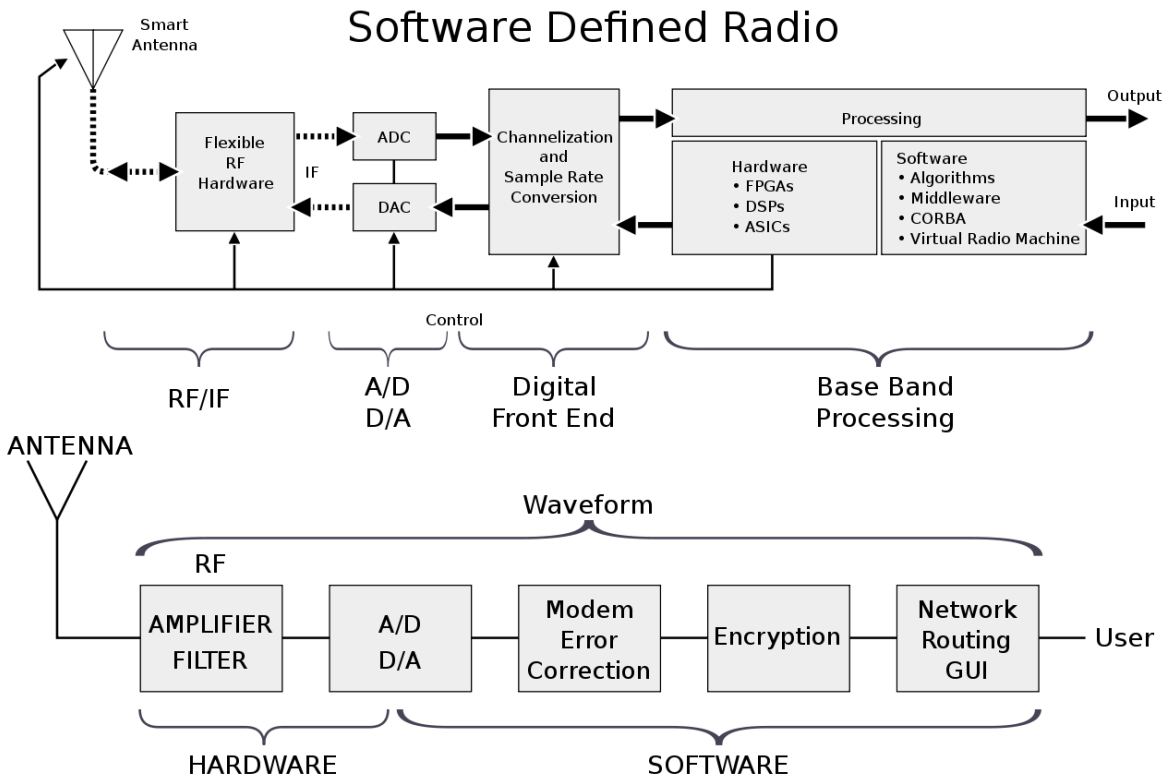
With SDR

- Famous software:
  - GQRX on Linux and Mac OS X
  - HDSDR on Windows
  - SDR# on Windows
  - SDR++ on Linux and Windows



Software-Defined Radio

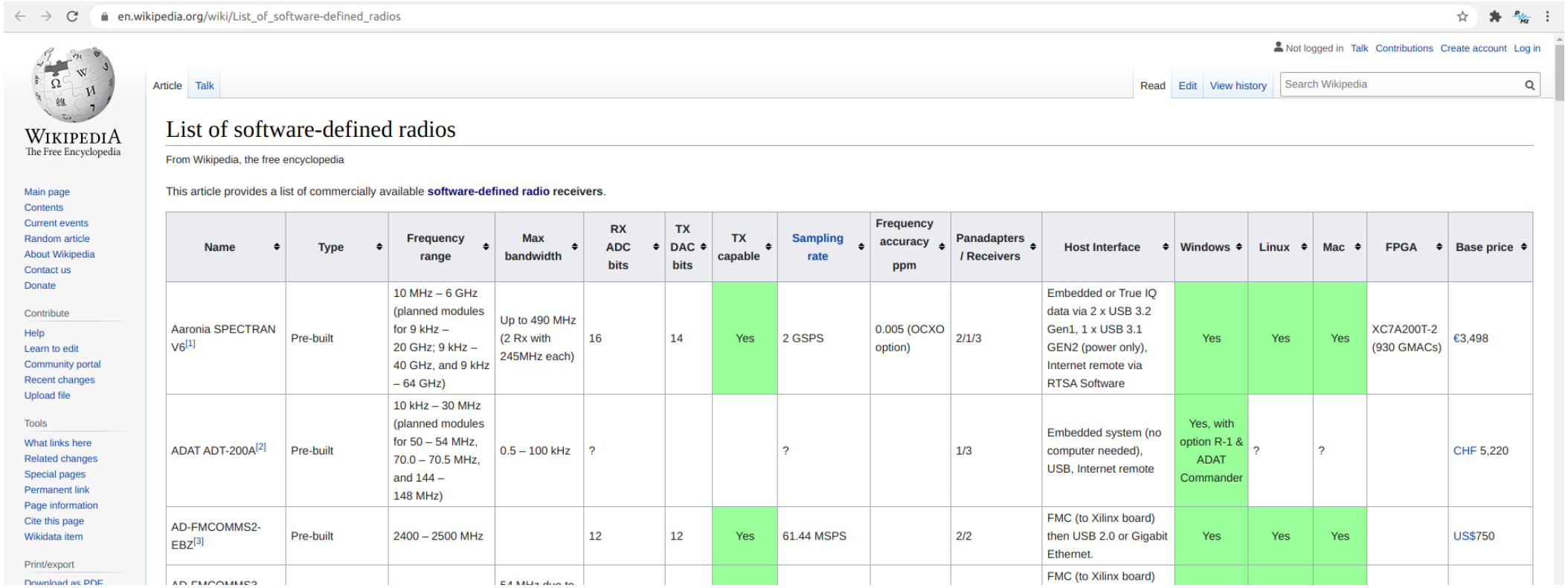
- Before SDR → difficult to get equipment without $$$
- Made radio more accessible
- Hardware:
  - Signal acquisition → IF;
  - ADC/DAC conversion;
  - RF Amp.;
  - and filtering.
- The rest is done in software
  - With your computer



Cheap SDR: RTL-SDR

- RTL-SDR
- Does only RX
- Different tuners/versions:
  - e.g. Elonics E4000 → 52-2200 MHz
- Costs from 15€
- RTL-SDR v3 is about 50€
- Good for a beginning

More than 100 SDR



en.wikipedia.org/wiki/List_of_software-defined_radios

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF

List of software-defined radios

From Wikipedia, the free encyclopedia

This article provides a list of commercially available **software-defined radio receivers**.

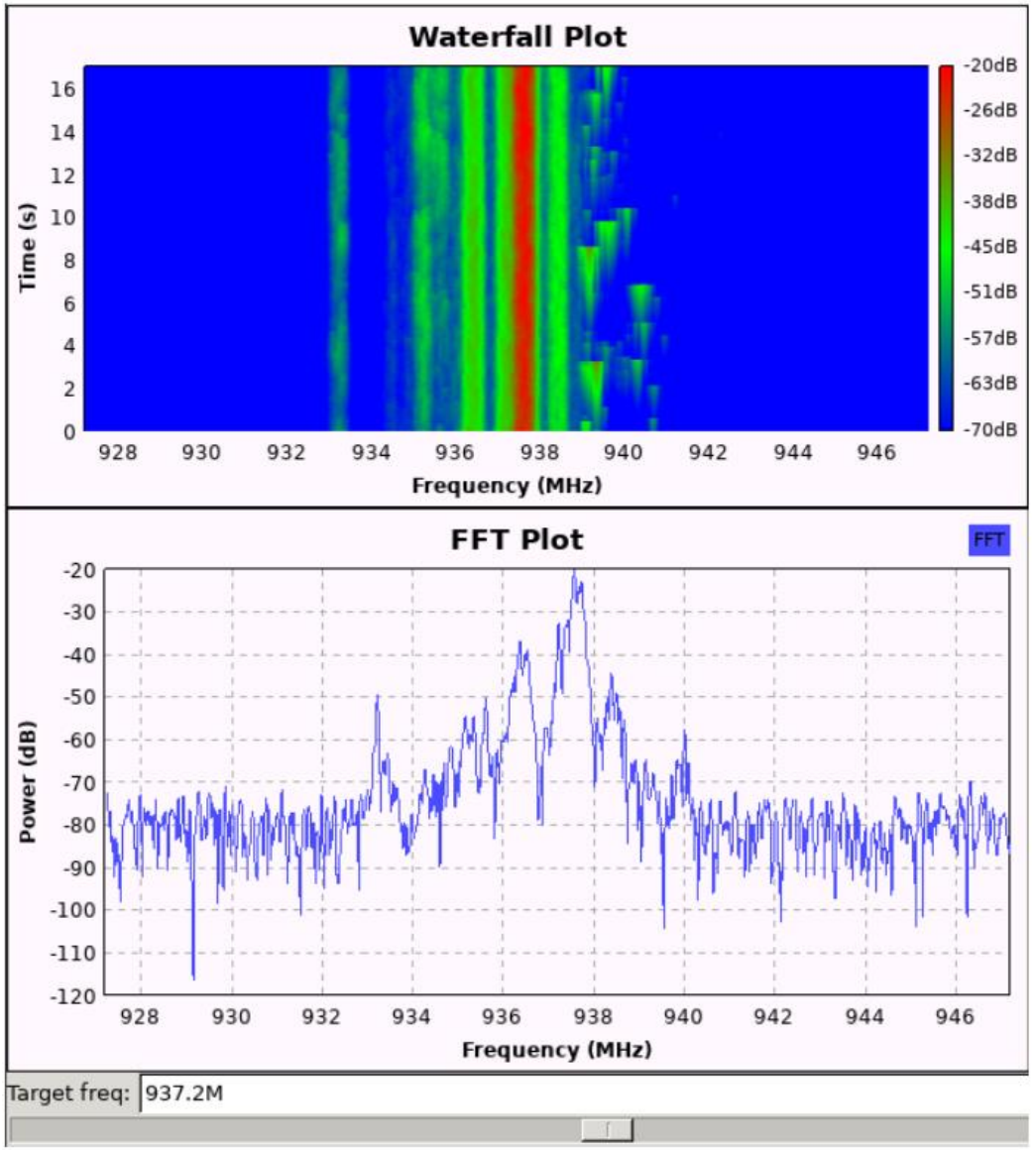
Name	Type	Frequency range	Max bandwidth	RX ADC bits	TX DAC bits	TX capable	Sampling rate	Frequency accuracy ppm	Panadapters / Receivers	Host Interface	Windows	Linux	Mac	FPGA	Base price
Aaronia SPECTRAN V6[1]	Pre-built	10 MHz – 6 GHz (planned modules for 9 kHz – 20 GHz; 9 kHz – 40 GHz, and 9 kHz – 64 GHz)	Up to 490 MHz (2 Rx with 245MHz each)	16	14	Yes	2 GSPS	0.005 (OCXO option)	2/1/3	Embedded or True IQ data via 2 x USB 3.2 Gen1, 1 x USB 3.1 GEN2 (power only), Internet remote via RTSA Software	Yes	Yes	Yes	XC7A200T-2 (930 GMACs)	€3,498
ADAT ADT-200A[2]	Pre-built	10 kHz – 30 MHz (planned modules for 50 – 54 MHz, 70.0 – 70.5 MHz, and 144 – 148 MHz)	0.5 – 100 kHz	?			?		1/3	Embedded system (no computer needed), USB, Internet remote	Yes, with option R-1 & ADAT Commander	?	?		CHF 5,220
AD-FMCOMMS2-EBZ[3]	Pre-built	2400 – 2500 MHz		12	12	Yes	61.44 MSPS		2/2	FMC (to Xilinx board) then USB 2.0 or Gigabit Ethernet.	Yes	Yes	Yes		US$750
AD-FMCOMMS3			54 MHz due to							FMC (to Xilinx board)					

A lot of characteristics to look at: TX/RX, full/half-duplex, max. samp. rate, clock stability, interface, freq range, software support, etc.

Issues

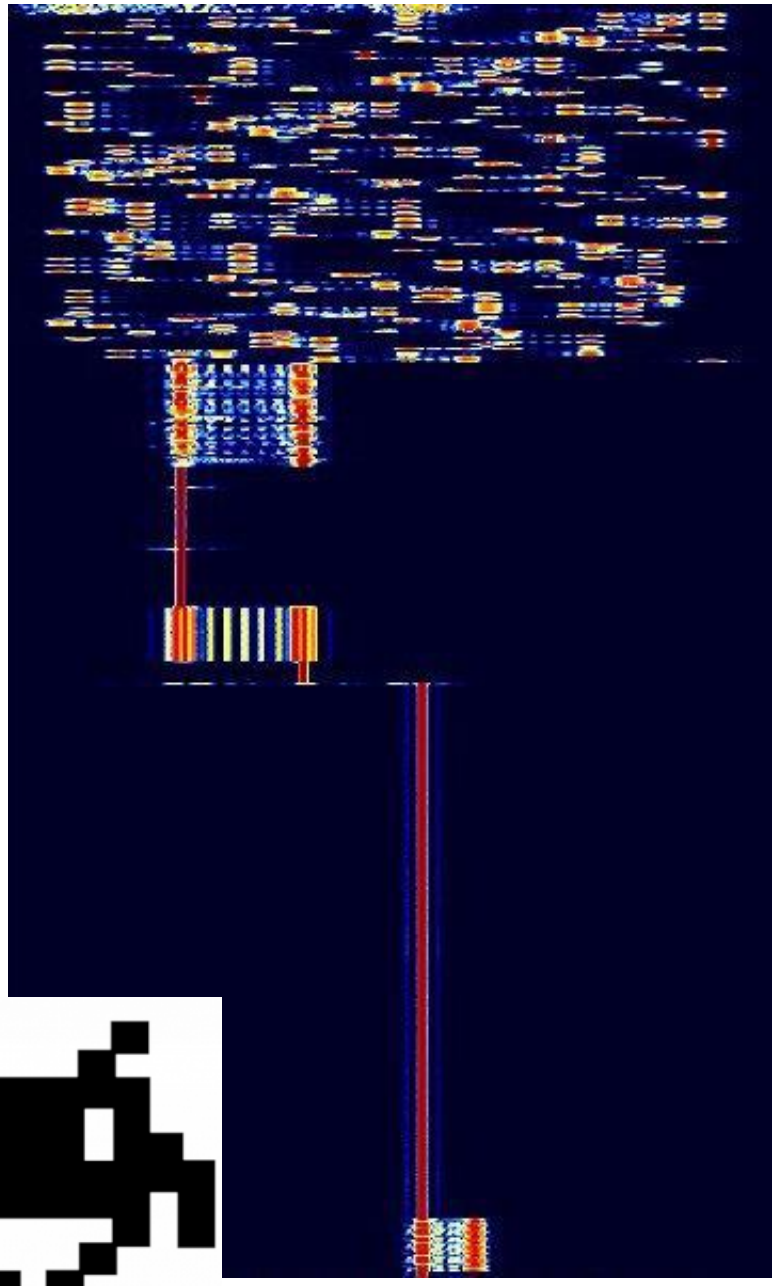
- How to identify the right technology?
  - Identify the frequency range
  - Used bandwidth
  - Duty cycle
  - Pattern
  - Other spectral properties
- But with what tools?

E.g.: FFT


Waterfall Plot
Time (s)
16
14
12
10
8
6
4
2
0
928
930
932
934
936
938
940
942
944
946
Frequency (MHz)
-20dB
-26dB
-32dB
-38dB
-45dB
-51dB
-57dB
-63dB
-70dB
FFT Plot
FFT
Power (dB)
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
928
930
932
934
936
938
940
942
944
946
Frequency (MHz)
Target freq:
937.2M

And then?

- We need a database:
  - Own collection
  - Public resources: like sigidwiki.com

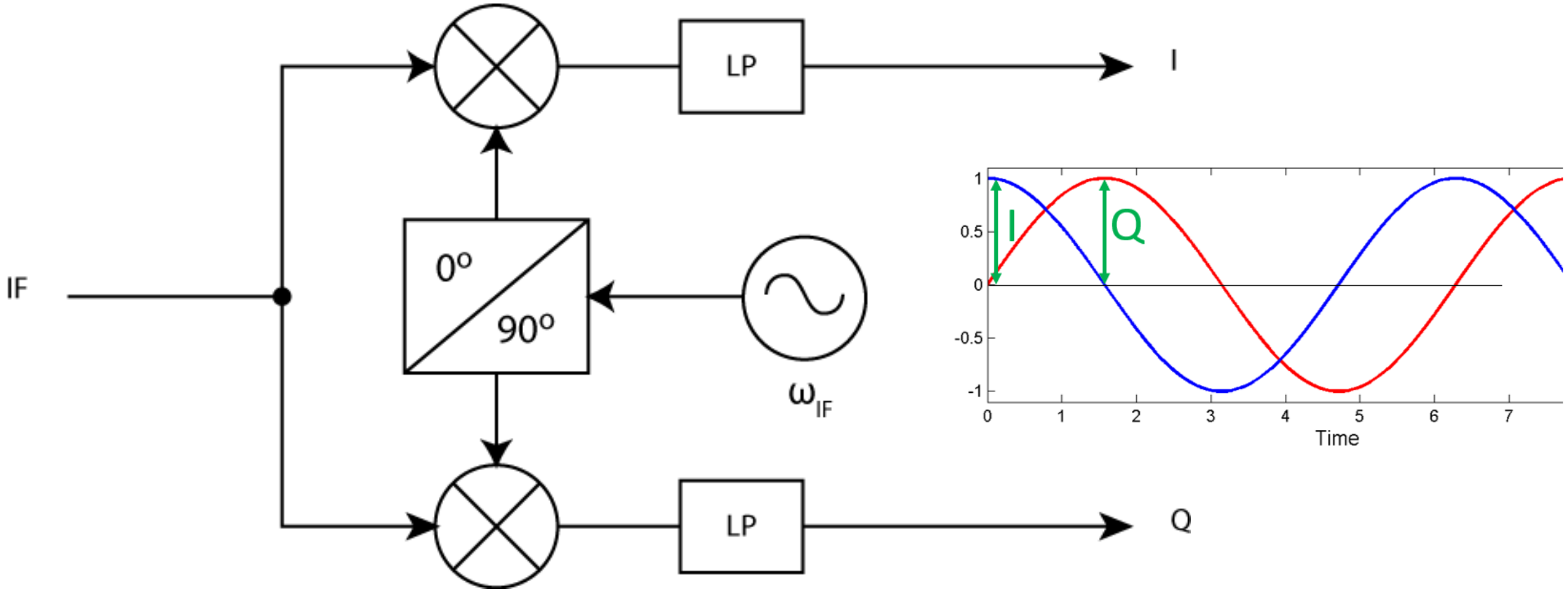


Don’t have time!

- Making my database
- Fetching samples from Sigwiki and others
- Matching collections
- All of that manually

But what if are lazy?

SDR and IQ

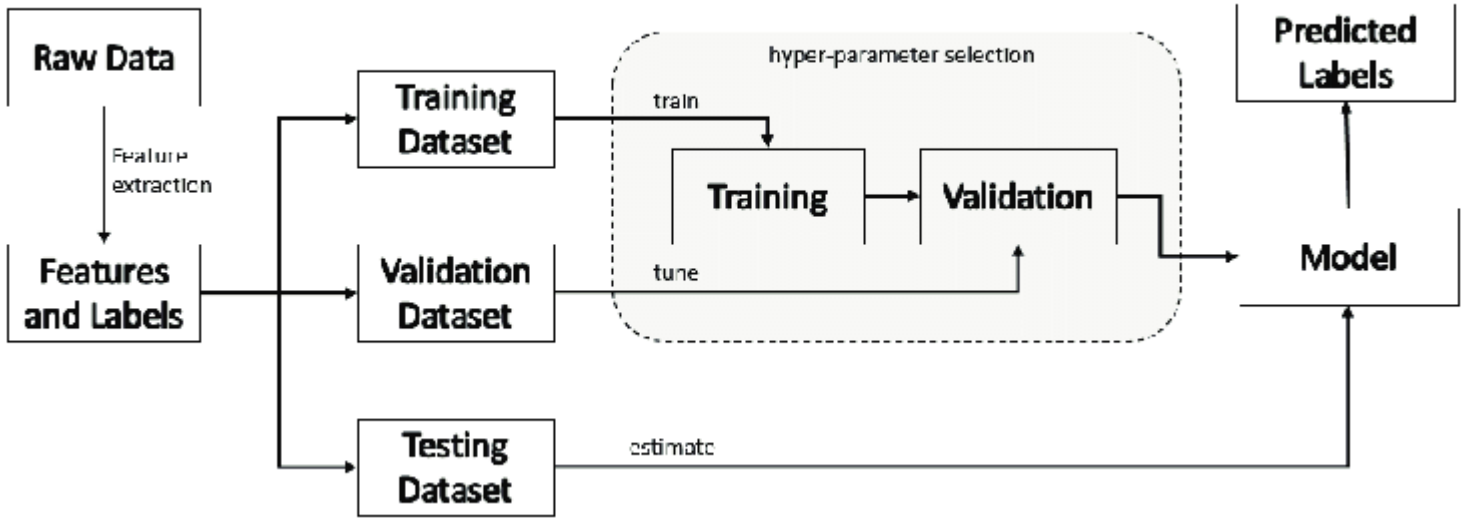


ML & DL

Me beginning in machine learning

Machine learning

- Science → learn from data
- Used to simplify complex rules applications: e.g. “spam filter”



Source: [A Machine Learning Approach to Road Surface Anomaly Assessment Using Smartphone Sensors]

Machine learning types

- Can be:
  - Supervised: feed the algo with desired solution → labels
    - Linear Regression
    - k-Nearest Neighbors
    - Logistic Regression
    - Super Vector Machines
    - Decision Trees and Random Forest
    - Neural Network

Machine learning types (2)

- Unsupervised:
  - Clustering (K-Means, DBSCAN, Hierarchical Cluster Analysis (HCA))
  - Anomaly detection and novelty detection (On-Class SVM, Isolation Forest)
  - Visualization (Kernel PCA, etc.)
  - Association rule learning (Apriori, Eclat)

Deep learning

- An application to Machine Learning
- Artificial neural network as a learning model
- Different kinds of layers

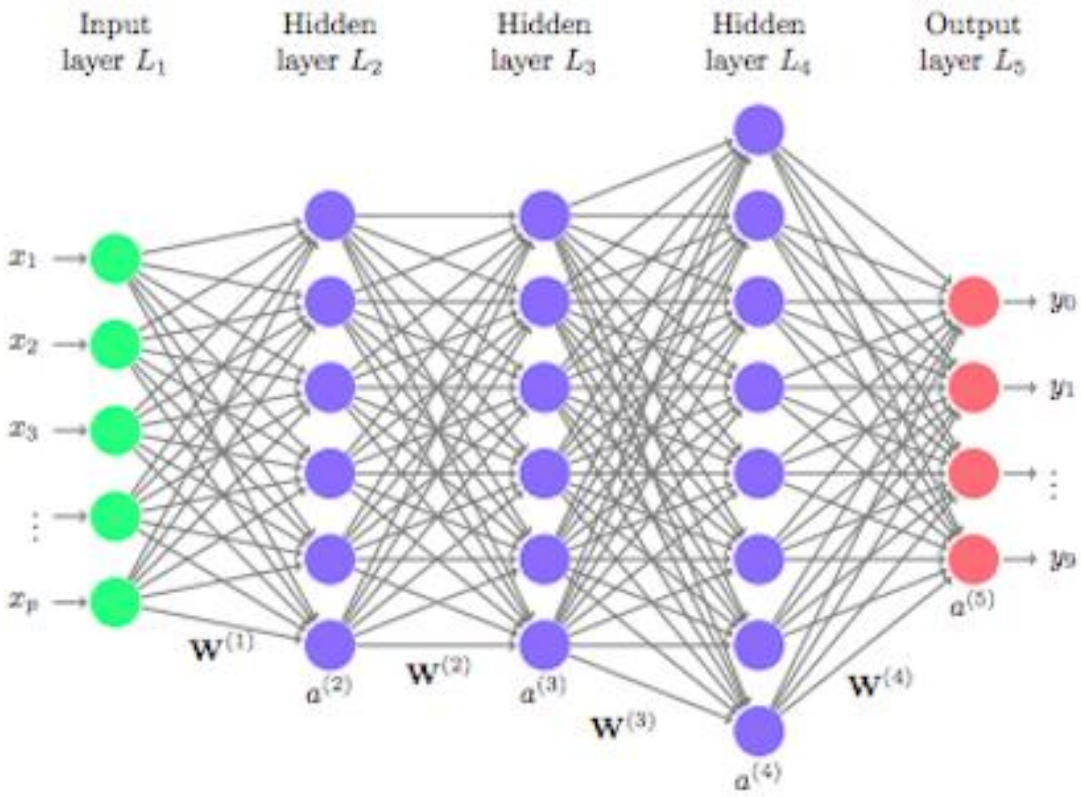


Image: UC Business Analytics R Programming Guide

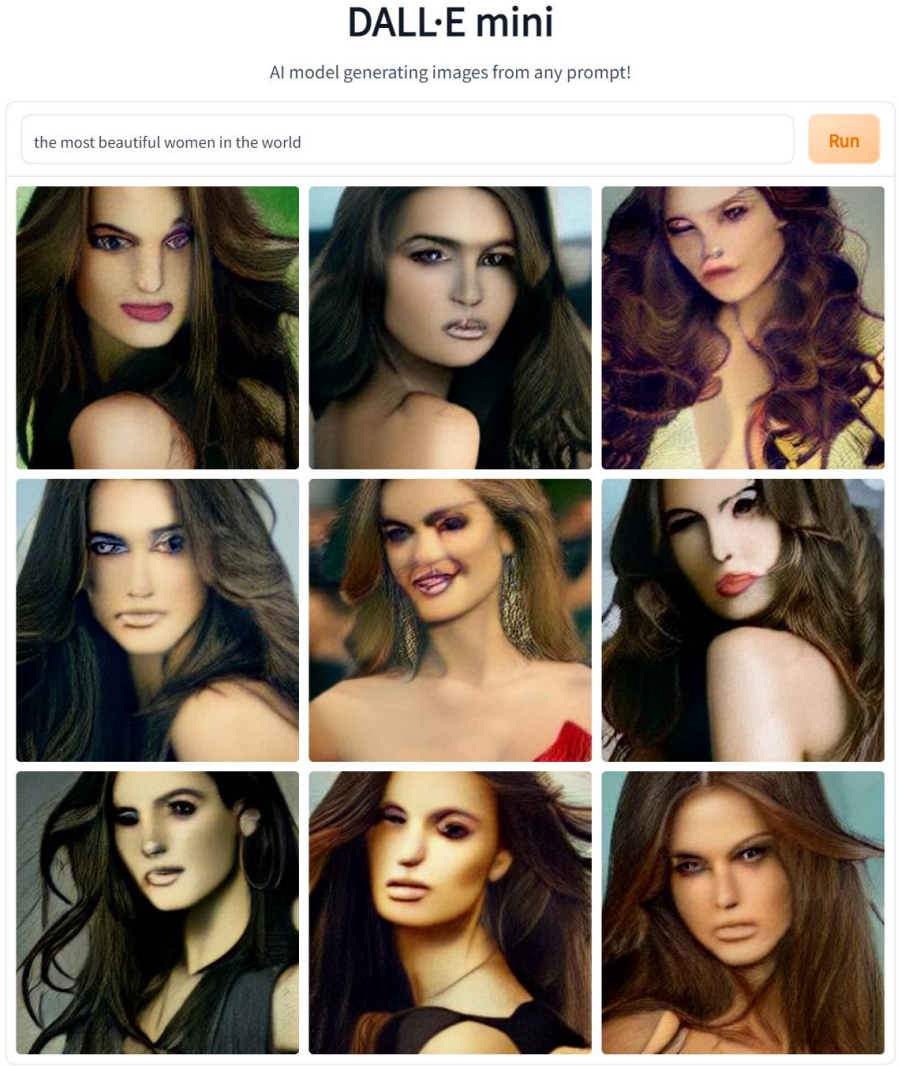
Good resources

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition
- Coursera Machine learning and Deep Learning courses
- https://towardsdatascience.com/
- Etc.

Challenges

- Bad data
- Bad algorithm(s) in use
- Insufficient quantity of training data
- Noisy data
- Nonrepresentative set
- Overfitting
- Underfitting
- Etc.

A lot of chance to mess



This model is excellent!

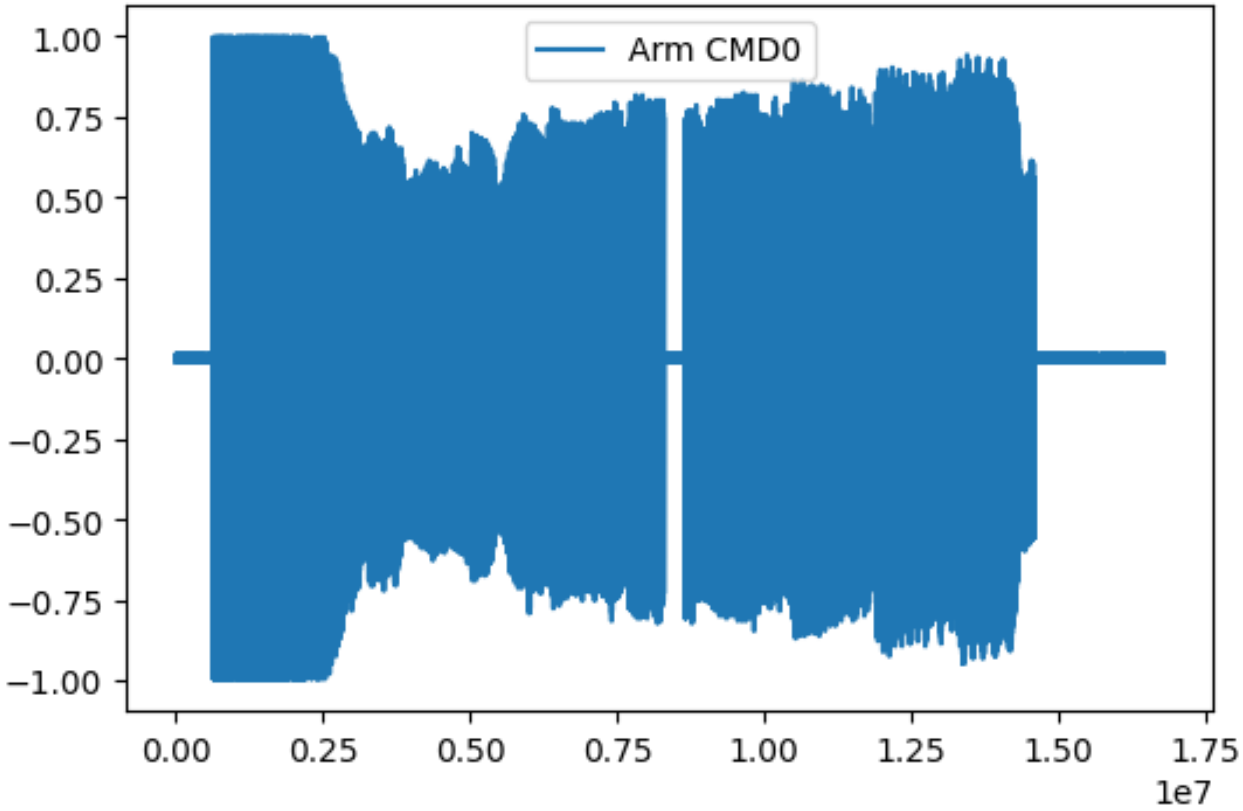
Model is as bad/good as your training set

- Need a lot of samples
- But filtered/clean samples
- Make a good balance between training and test sets
  - Warning! Test set always go at the end to test the model
  - Warning2: Careful with data and test splits
- Test different models → Ensemble learning is a win!

Machine learning for signal classification

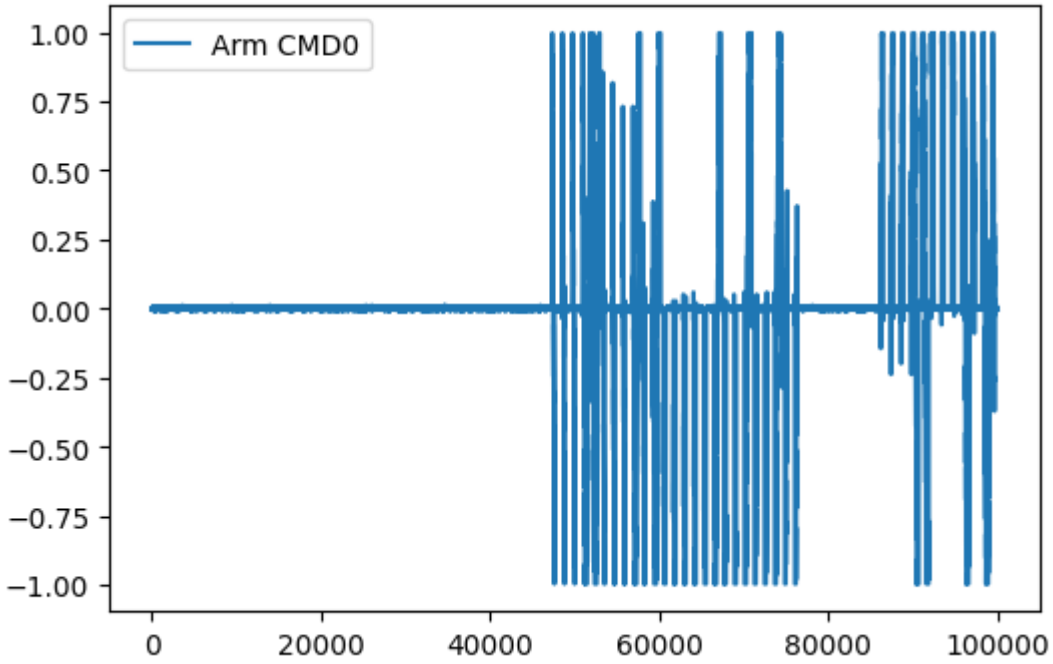
First capture we identified at an ISM frequency

- Doesn’t tell a lot
- Need to zoom in to see the properties
- There are maybe multiple packets



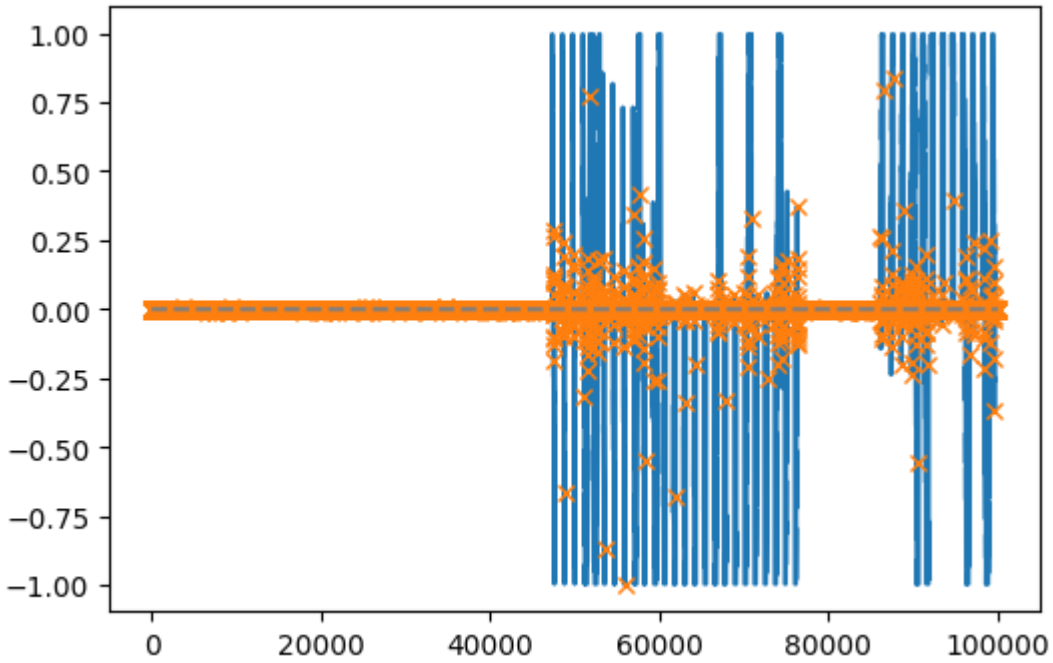
First capture zoomed

- Better → observe changes in amplitude → ASK/OOK
- Need to extract some properties (we can also apply some sound techniques):
  - Zero Crossing Rate
  - Mel-Frequency Cepstral Coefficients (even if applied to sound)
  - Roll-Off Point (even if applied to sound)
  - Number of peaks
  - Spectral Centroid
  - etc.

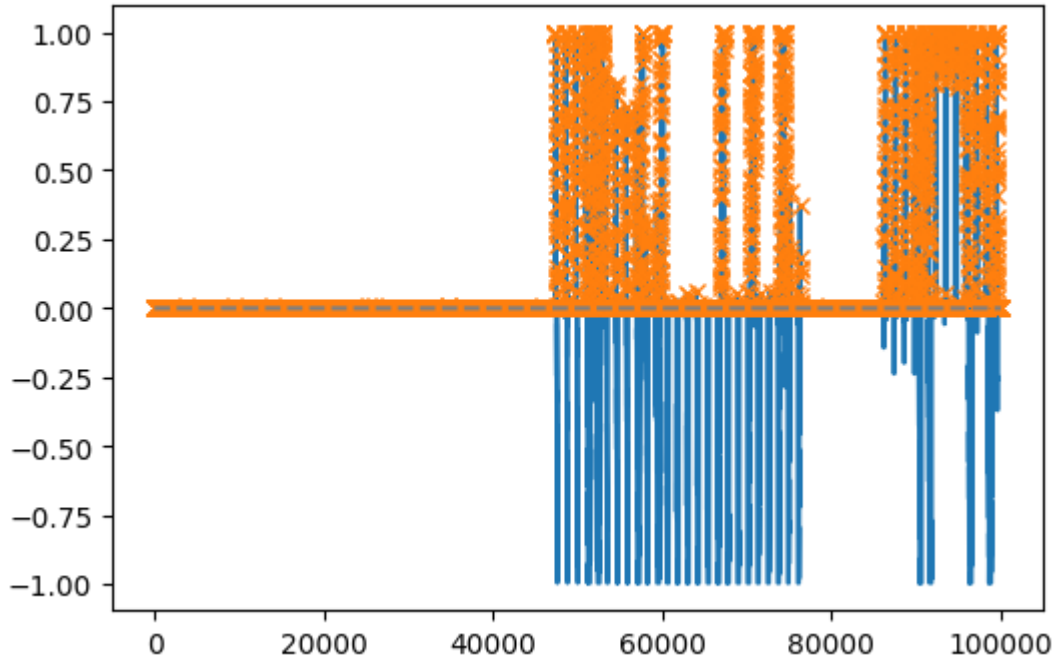


ZCR : Zero Crossing Rate

$$ZCR = \frac{1}{2N}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])|$$



Number of peeks

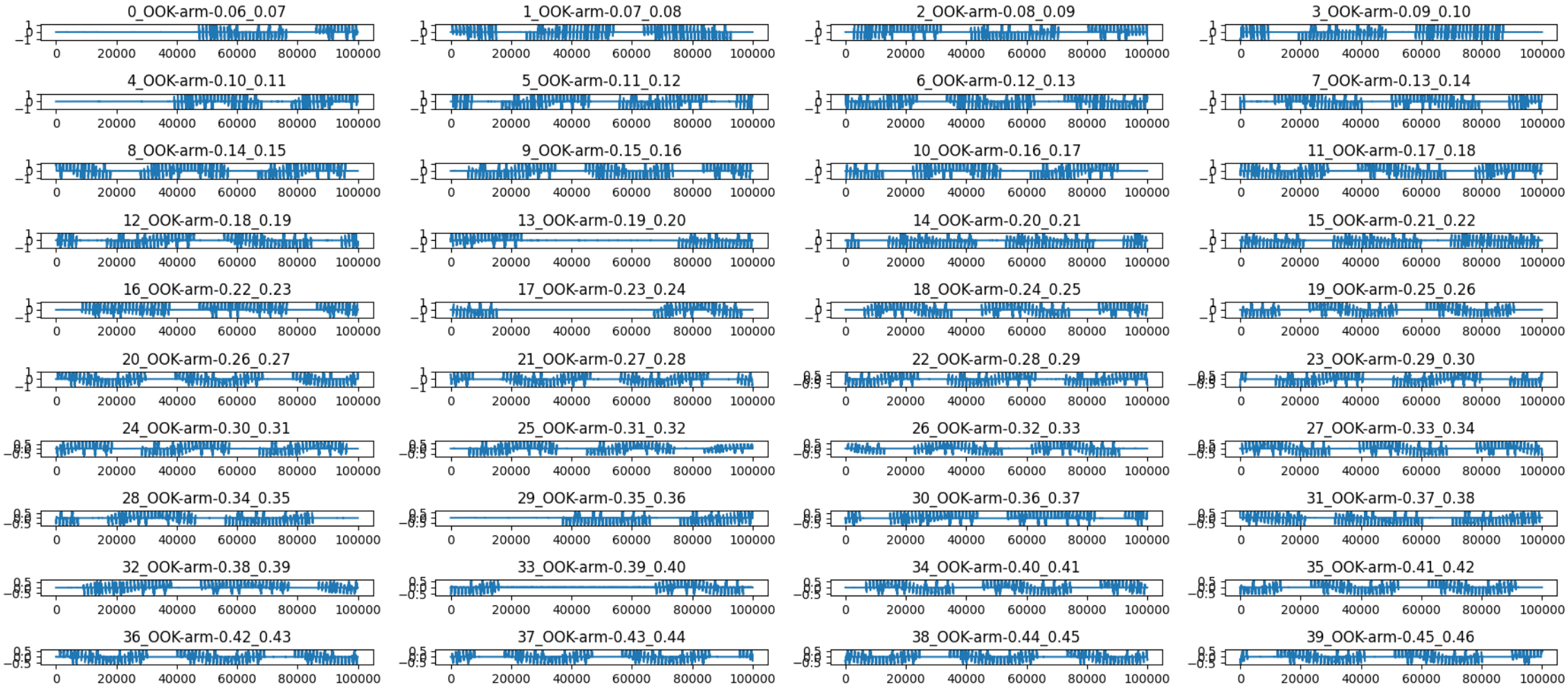


Our first list of features

```python
import pandas as pd

x = armbutton_01_slice.real

column_names = ['zcr', 'spectral_c', 'peaks' , 'label']
df = pd.DataFrame(columns = column_names)

def get_features(csignal, label):
    peaks, _ = find_peaks(csignal, height=0.99)
    return (len(np.where(np.diff(np.sign(csignal)))[0]), spectral_centroid(csignal), len(np.diff(peaks)), label)

df.loc[0] = get_features(x, "OOK")
df.head()
```

	zcr	spectral_c	peaks	label
0	35720	102043.176225	183	OOK

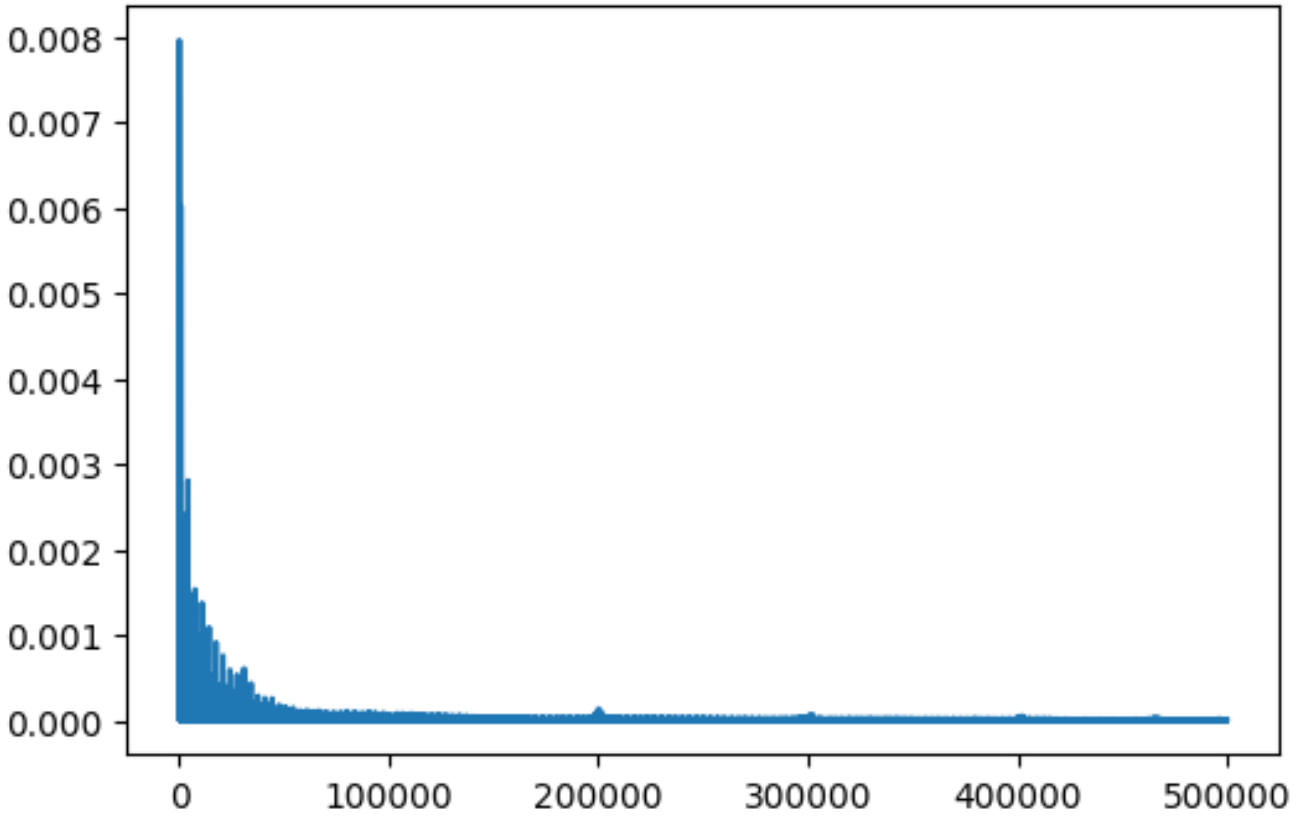
Making slices



All features for OOK

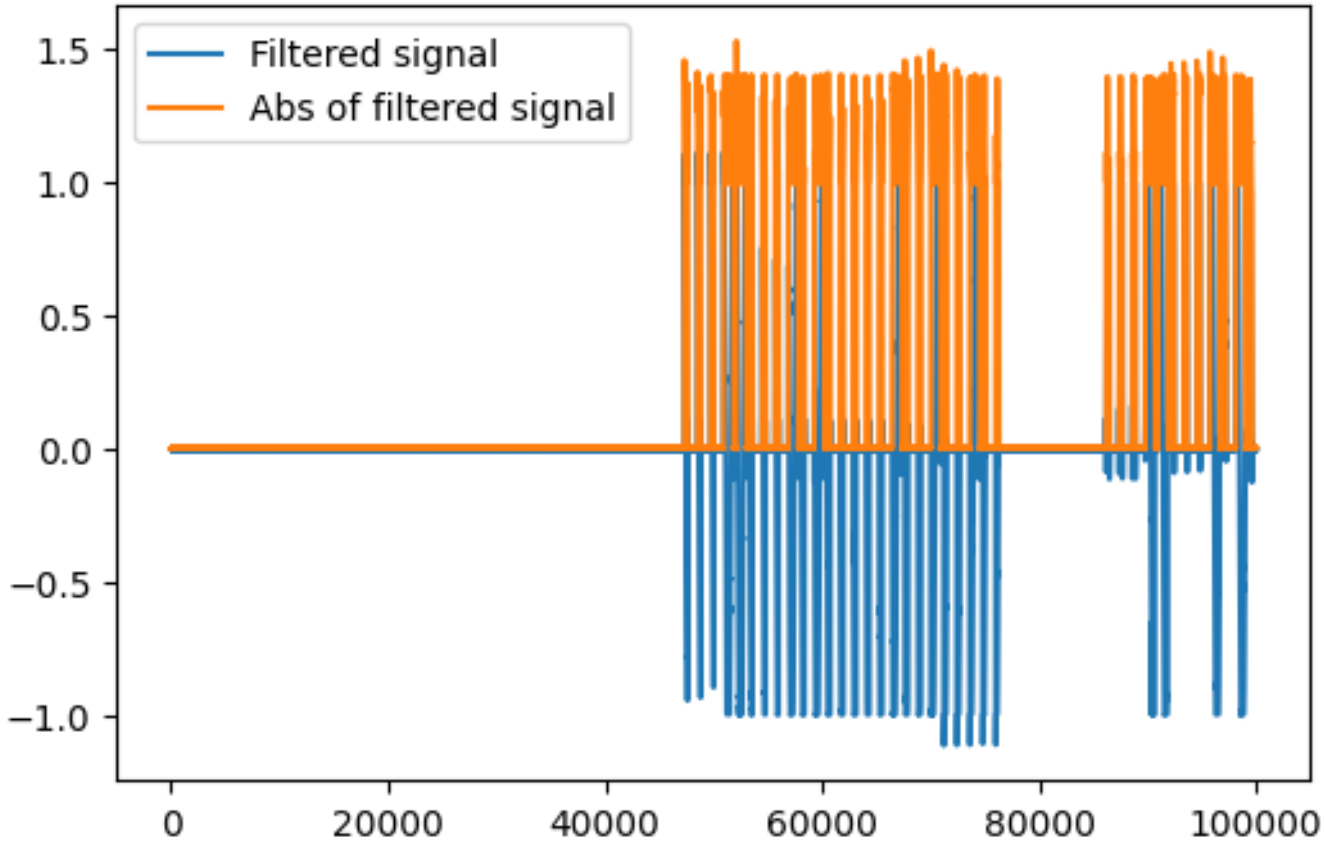
	zcr	spectral_c	peaks	label
0	35720	102043.176225	183	OOK
1	31570	112136.896603	286	OOK
2	30196	104925.260302	372	OOK
3	31894	107870.609573	303	OOK
4	35140	107022.365464	237	OOK
5	31257	106172.448807	328	OOK
6	29946	108299.517874	301	OOK
7	31224	106972.528967	328	OOK
8	30409	106918.584870	383	OOK
9	31027	104921.485225	328	OOK
10	31304	102660.881477	296	OOK
11	30050	104113.696990	333	OOK
12	31473	101537.998925	258	OOK
13	34870	103812.785526	248	OOK

FFT to reveal component



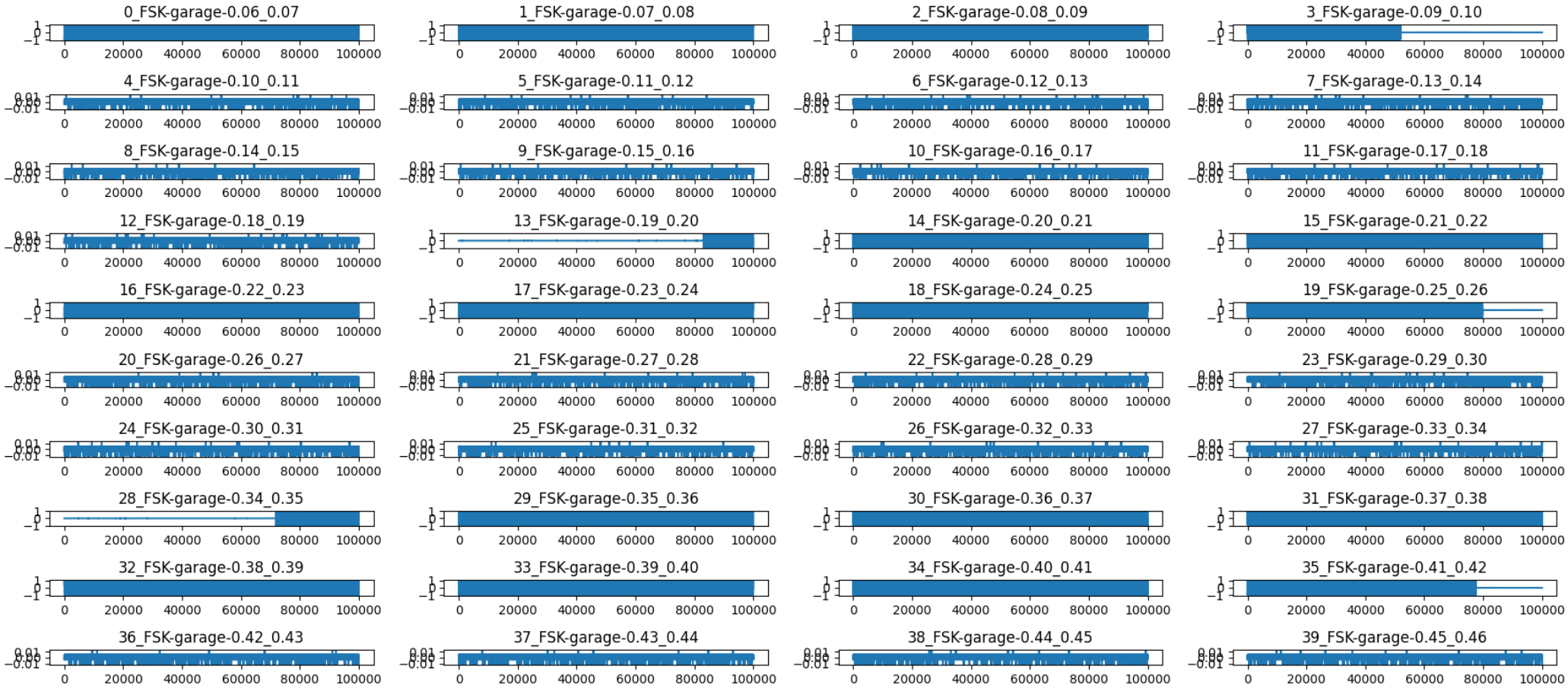
A simple low-pass filter would do the job !

Clean “packet” and it’s abs



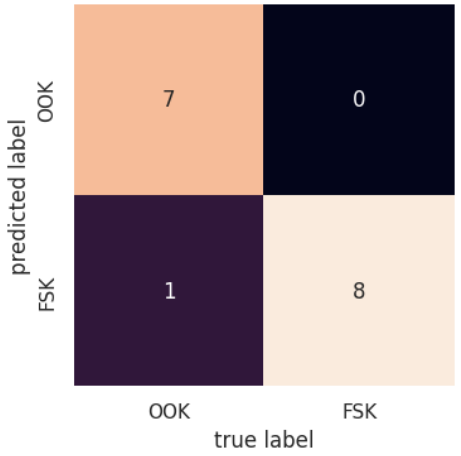
The signal can remain at default edge

Second capture : e.g. FSK



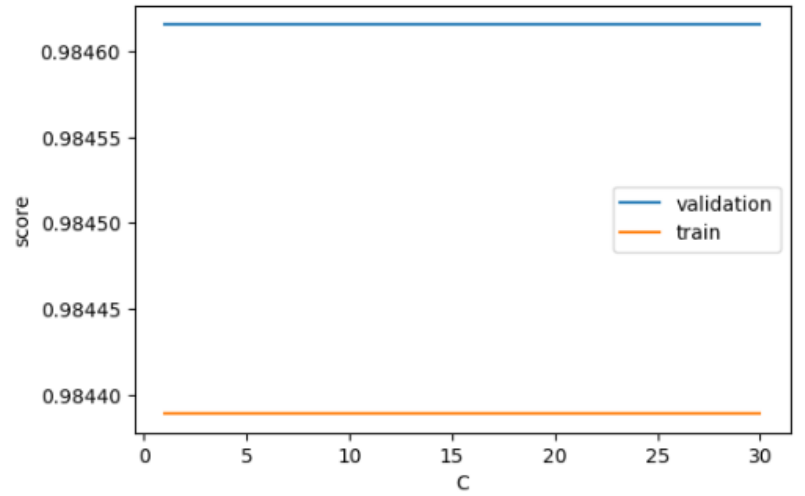
Use of SVM

- Results look great but can fail with more classes
- Need to test with != algorithms
- More data, of course
- But also, more features!




```
from sklearn.svm import SVC
from sklearn.model_selection import validation_curve
model_svm = SVC()
model_svm.fit(X_train, y_train)

print('Train score : ', model_svm.score(X_train,y_train))
print('Test score : ', model_svm.score(X_test,y_test))

k_3 = np.arange(1,31)
tr_score_3, val_score_3 = validation_curve(model_svm, X_train, y_train, param_name='C', param_range=k_3, cv = 5)

#5 splits sets de cross validation, on fait la moyenne des scores obtenus sur chacun des 5 splits

train = model_svm.predict(X_train)
predictions = model_svm.predict(X_test)

plt.plot(k_3, val_score_3.mean(axis = 1), label = 'validation')
plt.plot(k_3, tr_score_3.mean(axis = 1), label = 'train')

plt.ylabel('score')
plt.xlabel('C')
plt.legend()
```

```
Train score :  0.984375
Test score :  0.9375
<matplotlib.legend.Legend at 0x7f8ac3f7a390>
```



Random forest

- Found a got hyperparameters

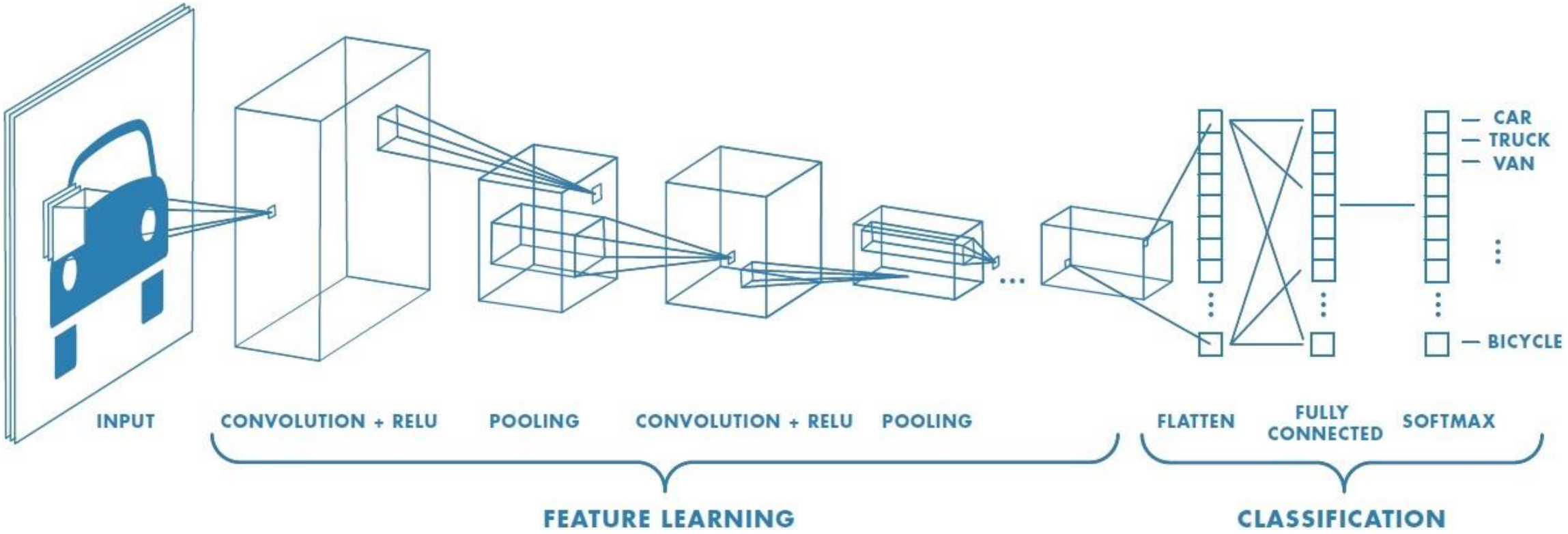
```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
```

	max_depth	min_samples_split	n_estimators	Accuracy
0	20.0	2	500	0.966667
1	20.0	2	4000	0.966667
2	NaN	2	500	0.966667
3	NaN	2	4000	0.966667

- But accuracy to good to be true for large sets

CNN implementation

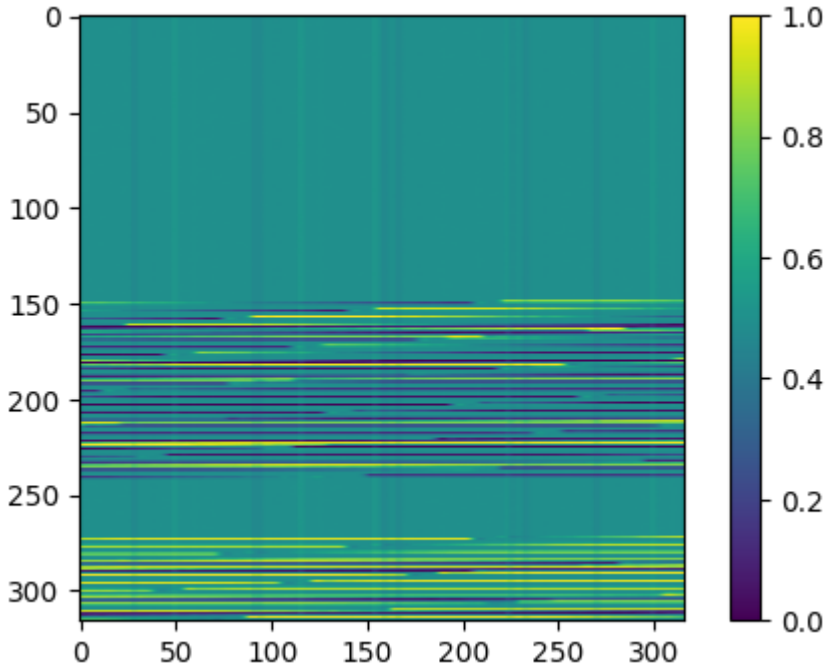
Convolutional Neural Network



Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

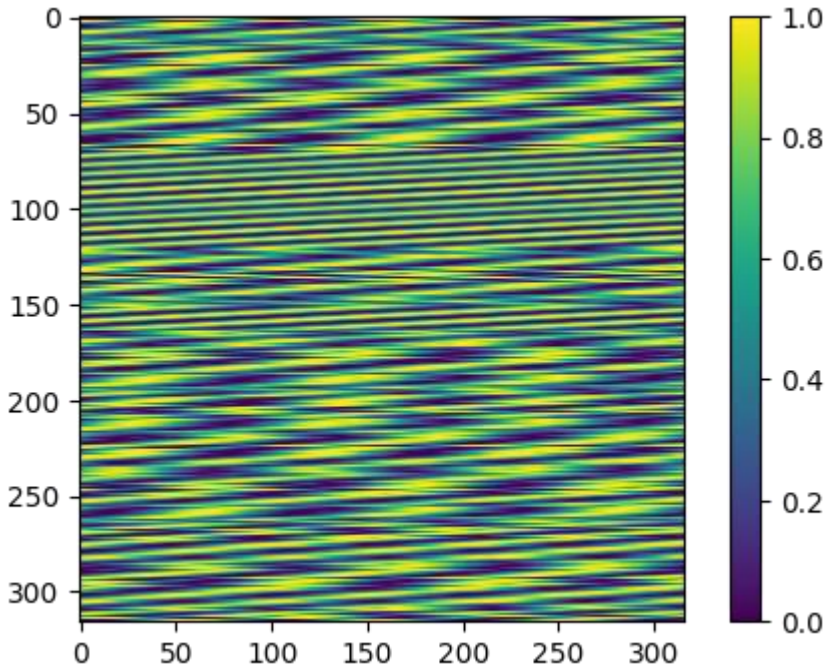
Convolutional Neural Network (2)

- We can miss a lot of features when extracting them from a signal on our own
- It’s possible to transform the signal to image for the OOK signal:



FSK to image

- We can notice a difference in the pattern:

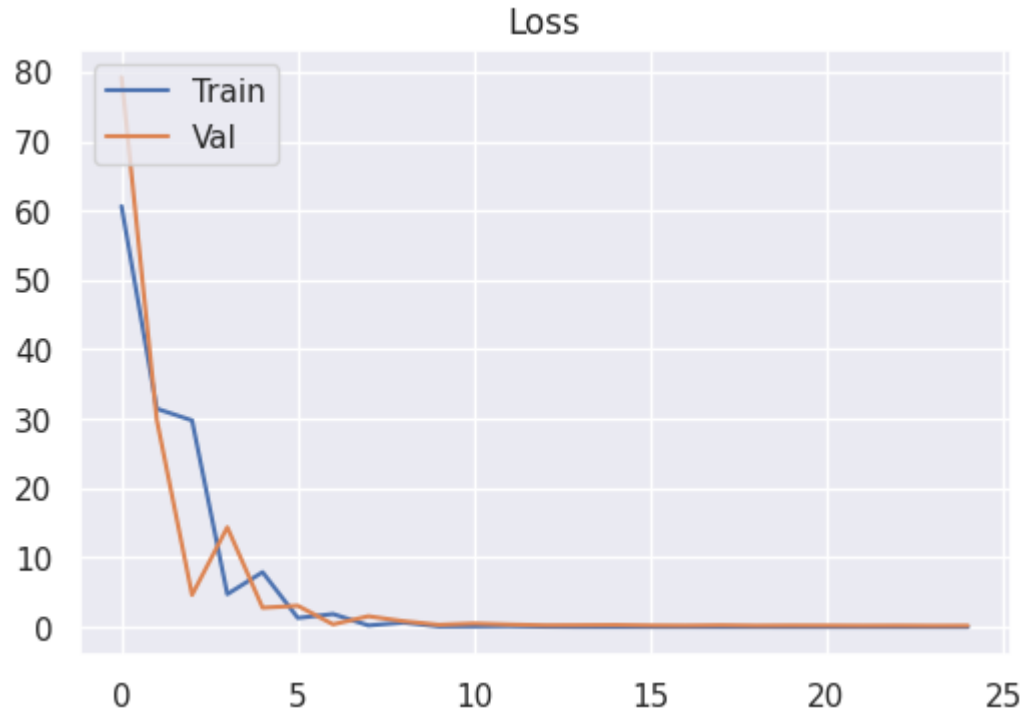


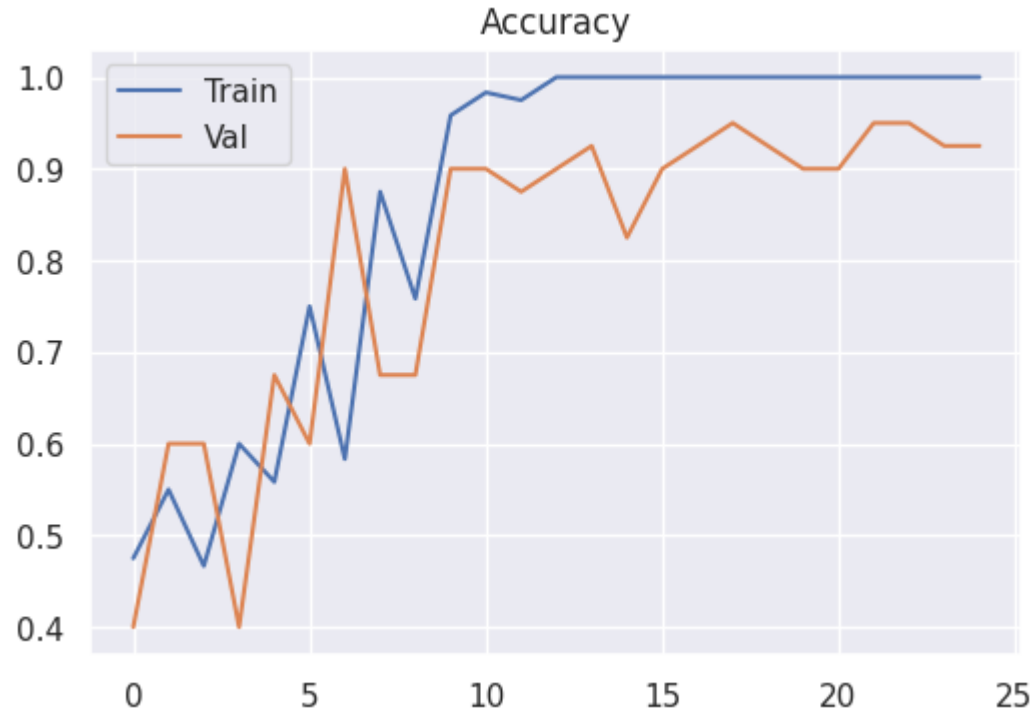
Results

- With 80 samples each
- 25 Epochs
- Following model:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 314, 314, 64)      640

 max_pooling2d_1 (MaxPooling  (None, 157, 157, 64)     0
 2D)

 flatten_1 (Flatten)         (None, 1577536)           0

 dropout_1 (Dropout)         (None, 1577536)           0

 dense_1 (Dense)             (None, 64)                100962368

 dense_2 (Dense)             (None, 2)                 130

=================================================================
```

Results (2)


Loss
Train
Val
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25


Accuracy
Train
Val
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25

Details in the Jupyter notebook

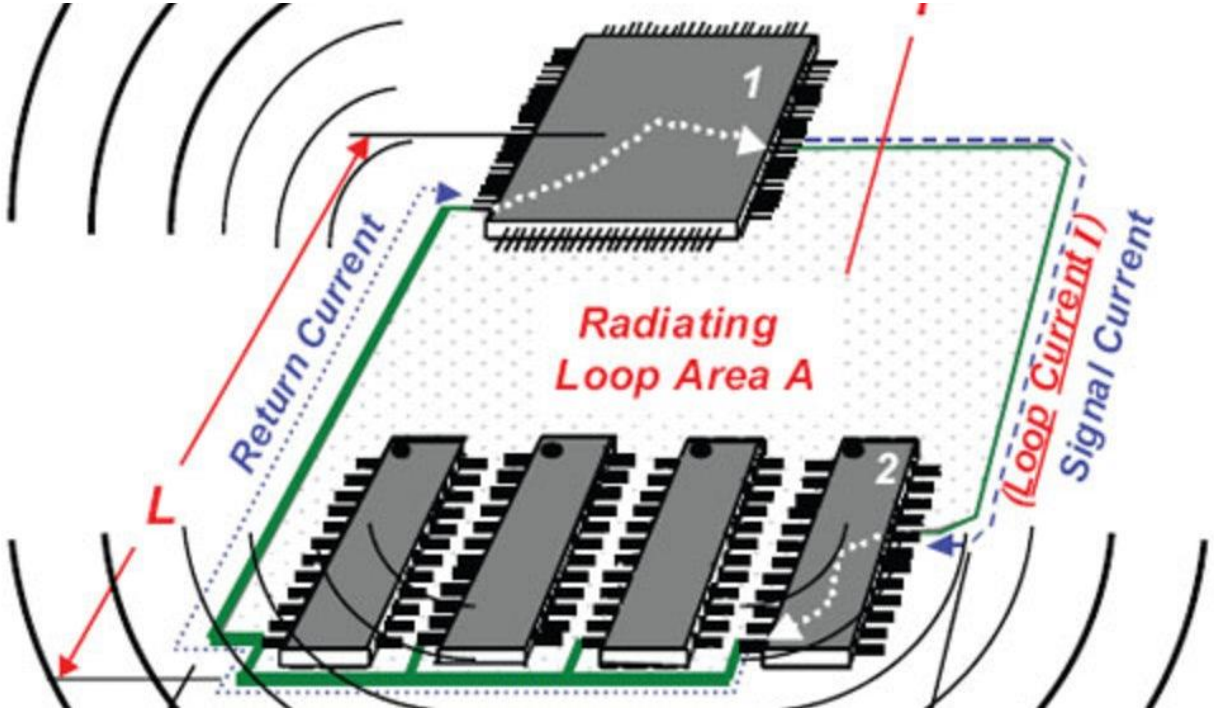
Other applications: EM attacks

EM side channels

- Compromise electromagnetic emanations → radiative coupling
- We try to infer what was computed
- Like power side channels → HW emits EM radiation → computation
  - Amplitude → proportional to consumed power
  - Some computations require +/- power
    - In CMOS → power is related to data being processed
- EM radiation → leaks certain operations:
  - Operation of encryption
  - PIN/key pressed
  - Memory RW

Differential-mode radiation

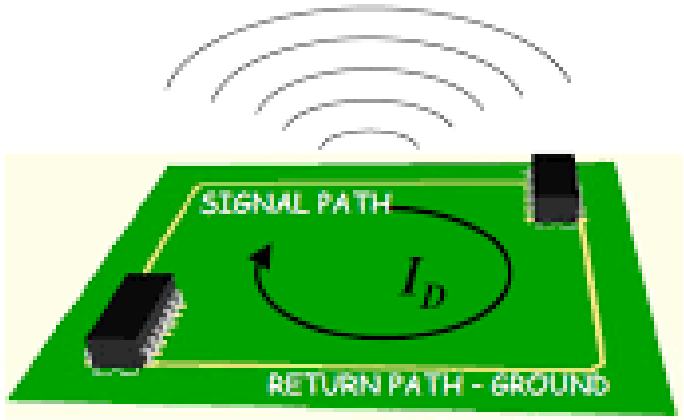
- Loops formed by components (circuit traces, ribbon cables, etc.)
- Act like an antenna → radiate
- Can be avoided with some shielding



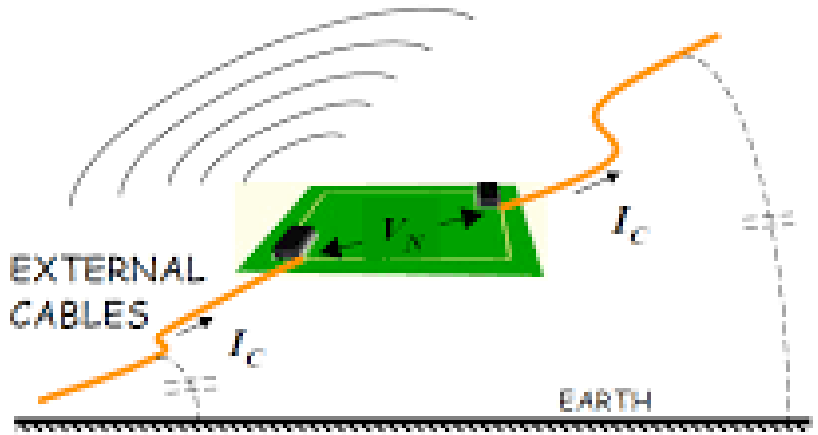
Source : electronicdesign.com

Common-mode radiation

- Undesired internal voltage drops in the circuit
  - Generally, in the ground loop
- External cable act as an antenna
- Difficult to control and control



DIFFERENTIAL MODE RADIATION

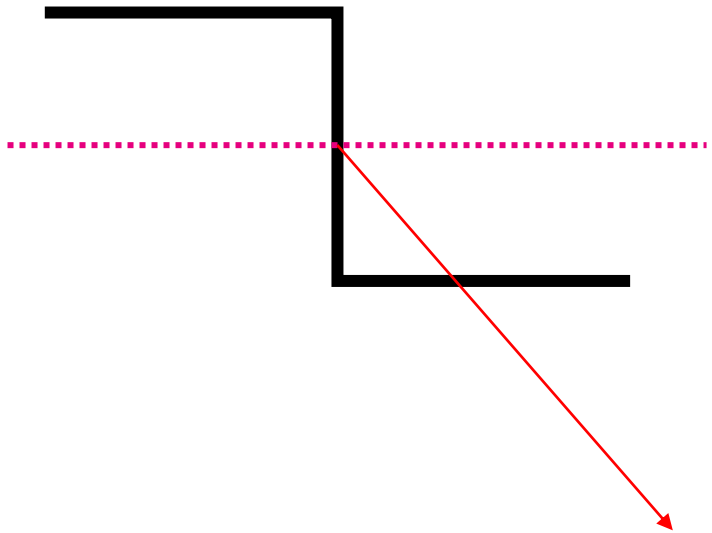


COMMON MODE RADIATION

Source : ursi.org

Direct emanations

- Provided straight by the wire transmitting sensitive data
- Faster is the transition → intense radiation



The activity of D Flip Flop (DFF) → commuting at HF ~1ns

EM wave emitted at max frequency → duration rise/fall time

Unintentional/Indirect Emanations

- Components are at short distance from each other
- Interaction with active electronic components → induce new type of radiations
  - Modulations, and inter-modulations (frequency, phase, and amplitude)
  - And carrier signal (clock and its harmonics)
- Can be observed at greater distance than direct emanations → observing harmonics in different distances
- Which frequency?
  - Use of periodic activity T in second, $f = 1/T$
  - Other harmonics → *2f, 3f, 4f*, etc.
  - Can be modulated often in AM with another operating clock “fc” → fc ± f, fc ± 2f, etc.

Side-channels applications

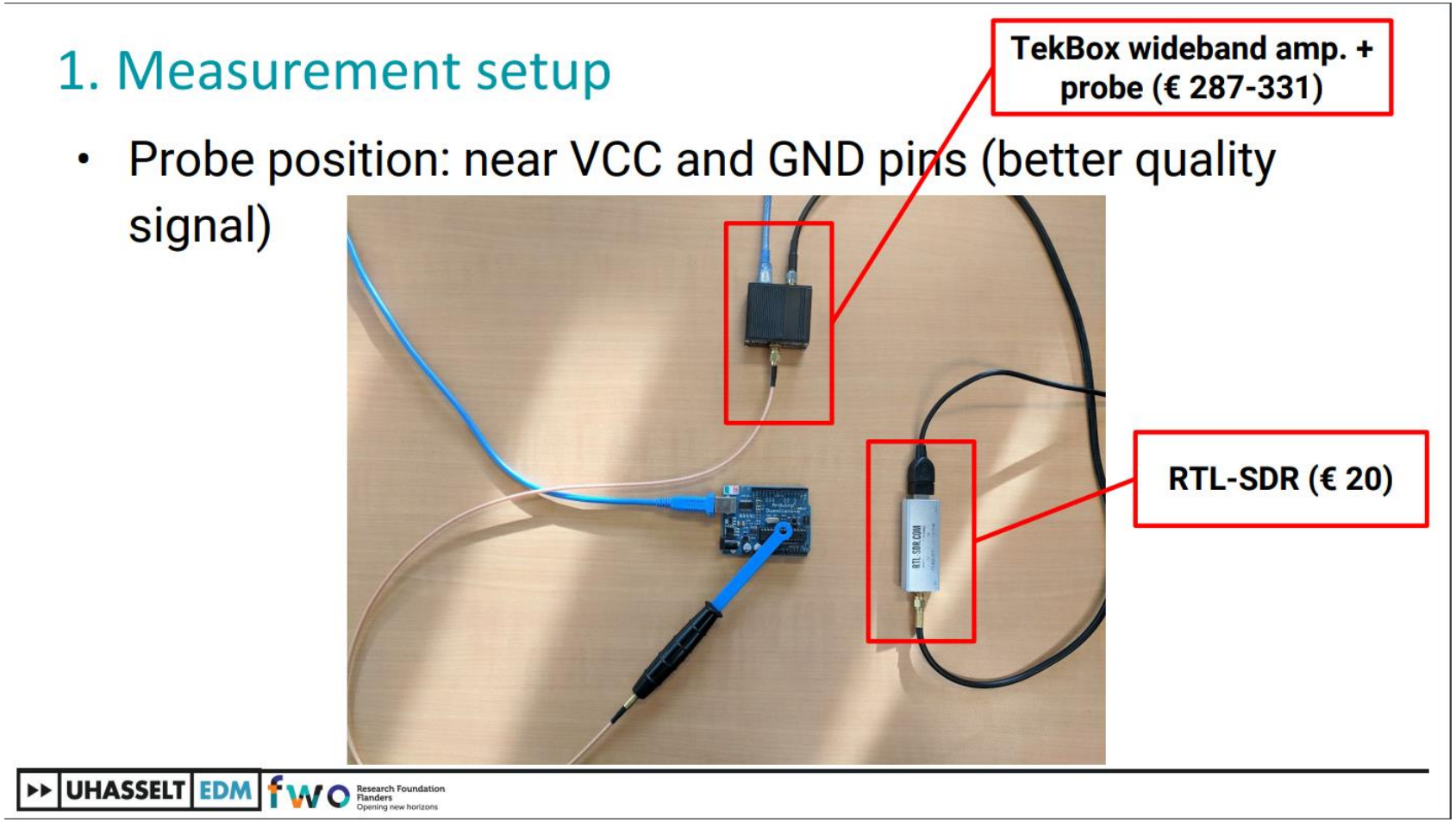
Side-channel attack on Ledger blue

By Thomas Roth, Josh Datko, Dmitry Nedospasov

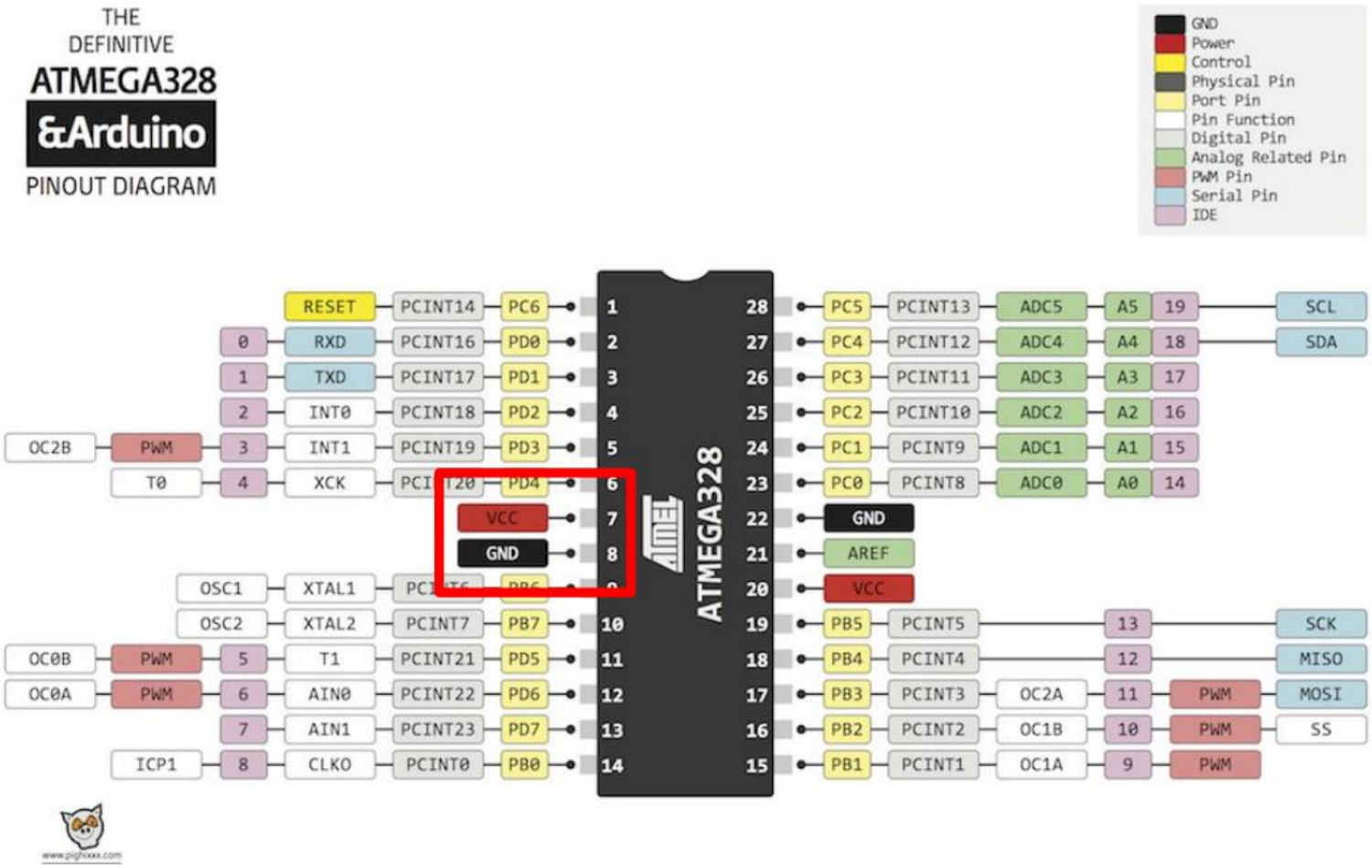
Leaking secrets of a device

AES CEMA attack on Arduino Duemilanove

- Performing Low-cost Electromagnetic Side-channel Attacks using RTL-SDR and Neural Networks by Pieter Robyns



Pins to target for better signal



Leakage Models

- Two common models:
  - Hamming Weight Model → linear relation between the number of bits “1” and power

$$HW(D) = \sum_{j=0}^{n-1} d_j$$ (D: input data, j bit, n: number of bits)

  - Hamming Distance Model → make use of dynamic leakage

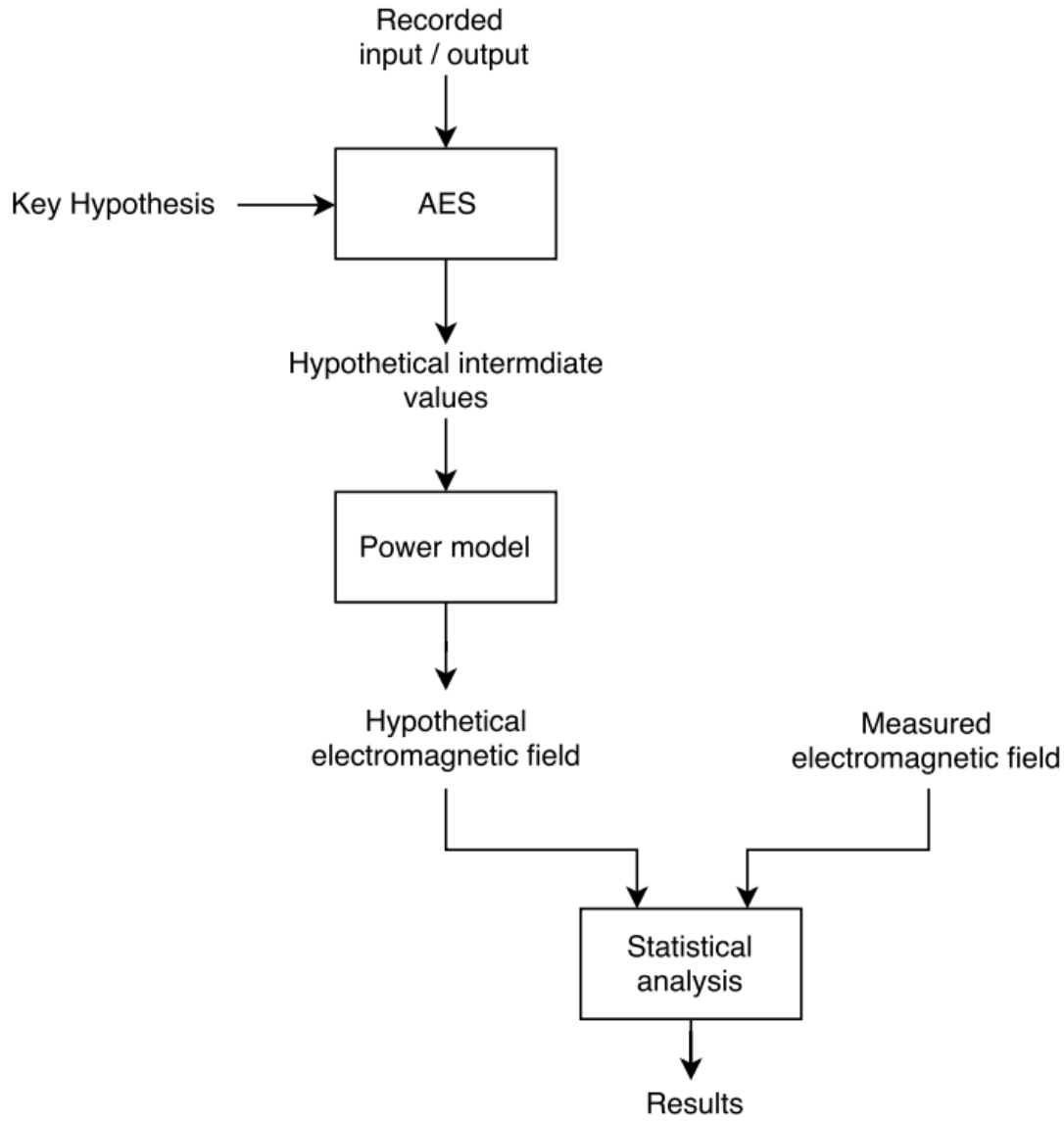
$$HD(D, D') = HW(D \oplus D')$$

(D and D’ → two input bytes/words)

Side-channel Attacks

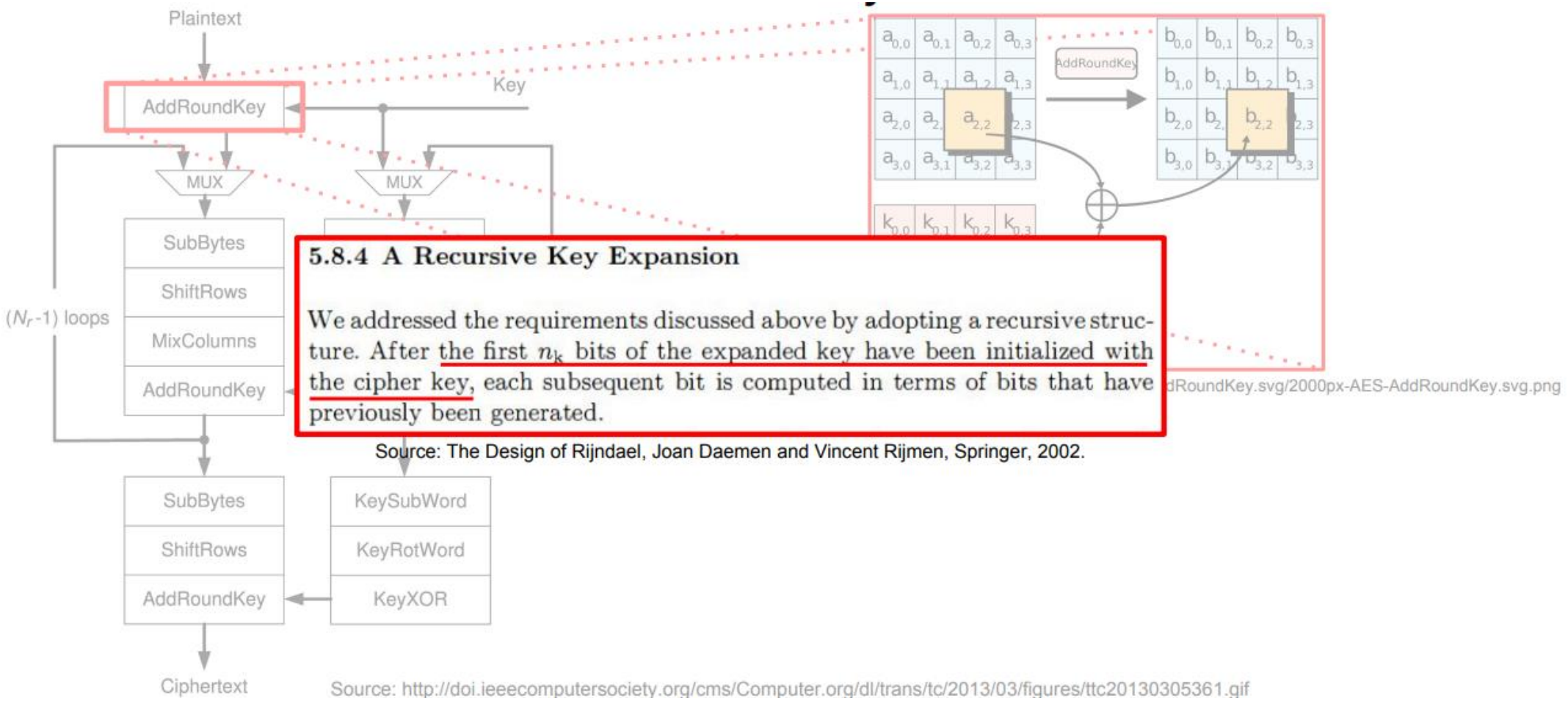
- Main attacks:
  - SEMA: Simple Electromagnetic Analysis → comes from SPA (visual for RSA e.g., and template)
    - Template is often used in AES → high frequency and noise
    - But need a similar device to characterize them and search space grow a lot when doing modeling leakage
  - DEMA & CEMA : Differential and Correlation Electromagnetic Analysis
  - TVLA: Test Vector Assessment → consisting of 1800 Welch's t-tests [18] knowing the plaintexts and the secret key.

DEMA & CEMA



Source: Far-field Correlation Electromagnetic Analysis attacks against AES in real-world applications F. van Tienen

CEMA attack on AES



5.8.4 A Recursive Key Expansion

We addressed the requirements discussed above by adopting a recursive structure. After the first n_k bits of the expanded key have been initialized with the cipher key, each subsequent bit is computed in terms of bits that have previously been generated.

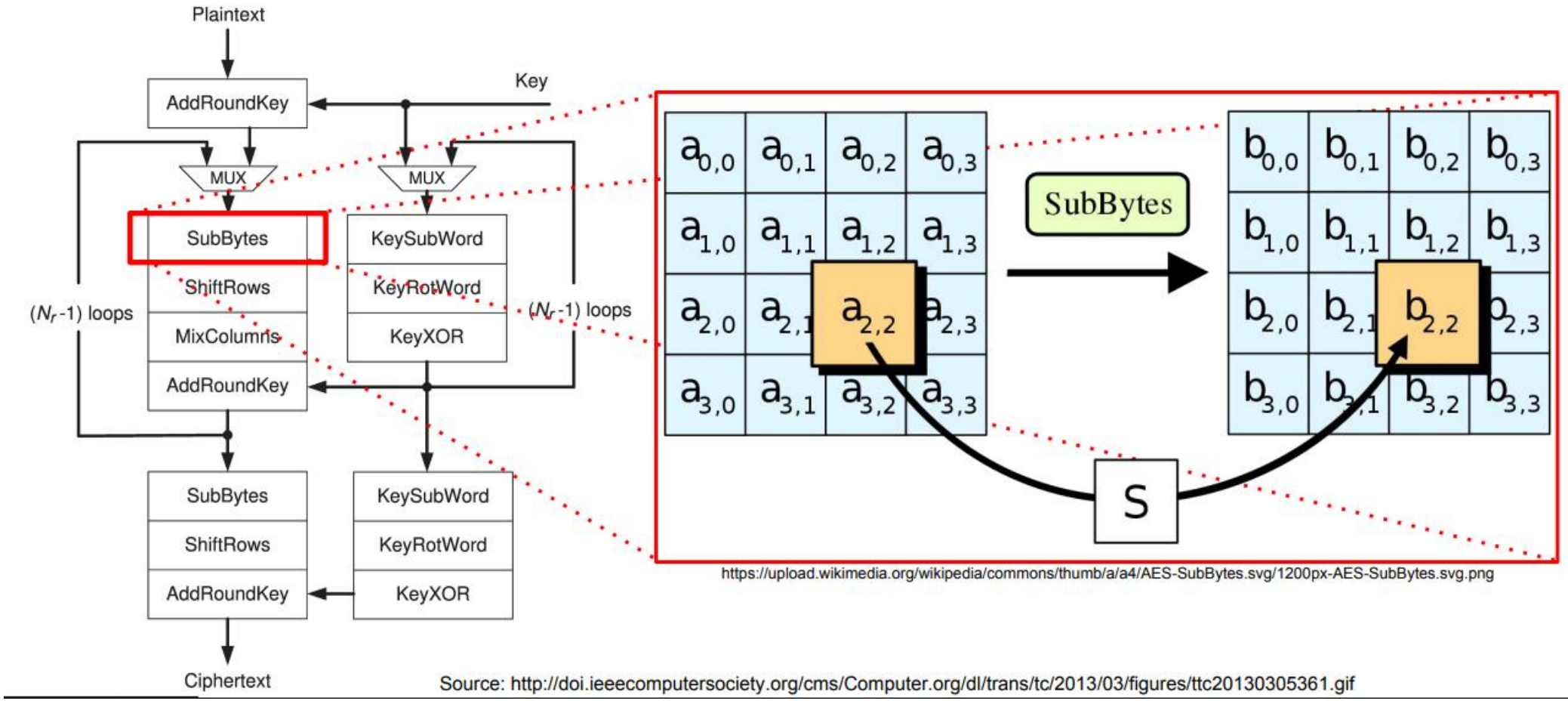
Source: The Design of Rijndael, Joan Daemen and Vincent Rijmen, Springer, 2002.

dRoundKey.svg/2000px-AES-AddRoundKey.svg.png

Source: http://doi.ieeecomputersociety.org/cms/Computer.org/dl/trans/tc/2013/03/figures/ttc20130305361.gif

Source: https://archive.fosdem.org/2019/schedule/event/sdr_em_sidechannel_attacks/attachments/slides/2931/export/events/attachments/sdr_em_sidechannel_attacks/slides/2931/robyns2019fosdem.pdf

CEMA attack on AES (2)



Source: http://doi.ieeecomputersociety.org/cms/Computer.org/dl/trans/tc/2013/03/figures/ttc20130305361.gif

Source:
https://archive.fosdem.org/2019/schedule/event/sdr_em_sidechannel_attacks/attachments/slides/2931/export/events/attachments/sdr_em_sidechannel_attacks/slides/2931/robyns2019fosdem.pdf

EMMA framework

- Let’s see: https://github.com/rpp0/emma

Conclusion

Lazy people gonna hate

- There is a magical algorithm for all datasets and purposes
  - Still a lot of work
  - Improve things within an Ensemble
  - It’s good also to generate clean signals! → less noise to learn + more samples
- But we can also take some inspiration from some great work:
  - Enhanced Low SNR Radio Signal Classification using Deep Learning, AI Wireless 2020 in National Chiao Tung University
  - And other → lot of GitHub projects

Thank You

Please contact us:

contact@penthertz.com

+33 1 73 13 82 79

penthertz.com