Wi-Five

Using GNU Radio and Python to detect EM emissions coming from the SiFive HiFive1 board

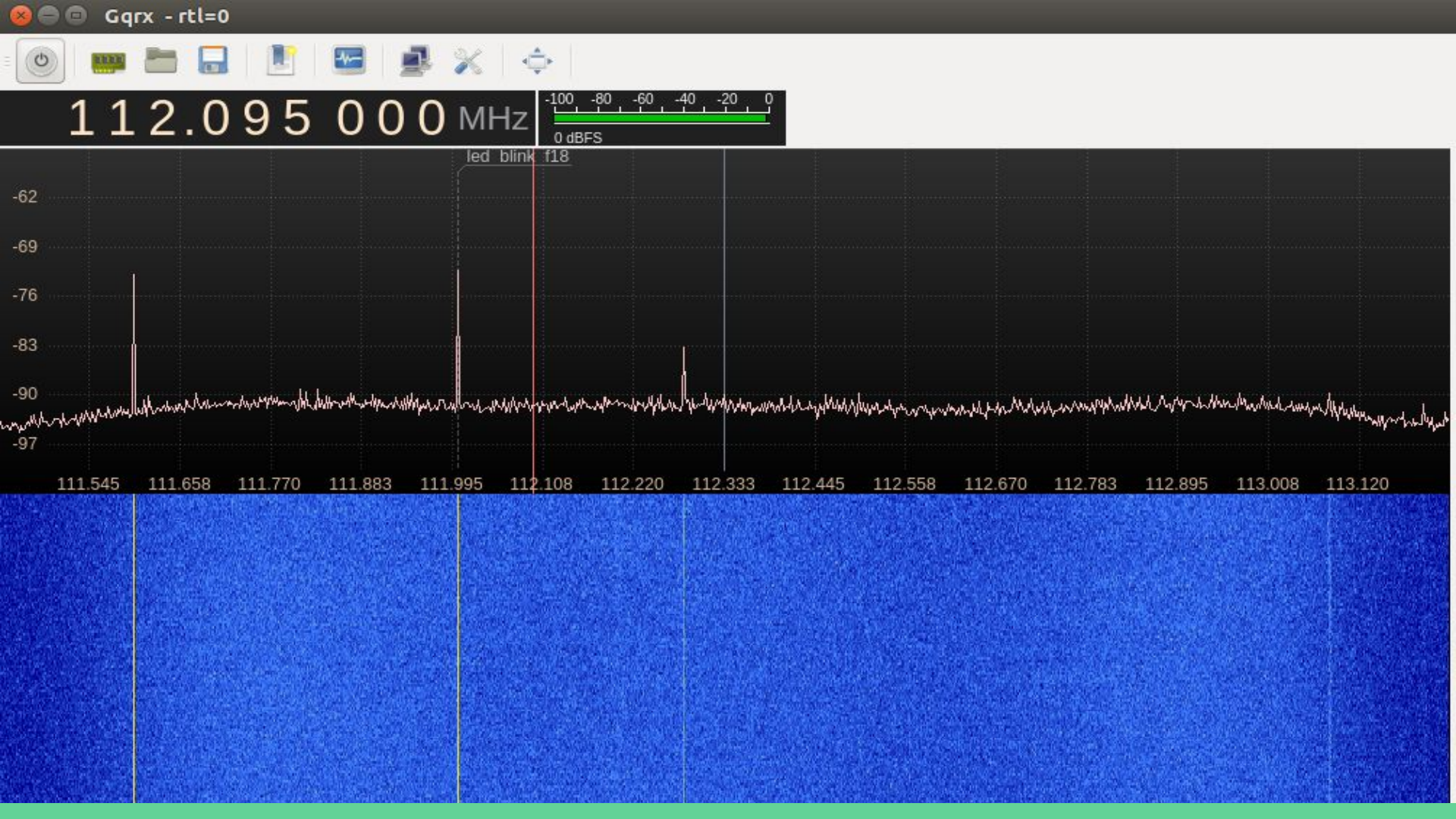
Student Valerio Lanieri
Supervisor Aurélien Francillon

EM emissions cover a wide range of frequencies

- When a program runs, unwanted EM emissions can cover a wide range of frequencies
- These emissions can constitute the basis of side-channel attacks: many papers already covered the subject and highlighted the risks
- It is therefore more and more important to counteract this issue
- But first, these emissions need to be identified
- Although rules of thumb can be followed, such as inspecting multiples of the clock frequency, EM information can come from unexpected frequencies

EM emissions cover a wide range of frequencies

- A preliminary test using gqrx, the RTL-SDR dongle and a telescopic antenna was performed
- Many frequency ranges were examined
- The board under inspection, the SiFive HiFive1, emits a carrier at 112 MHz regardless of the program which is running


Gqrx - rtl=0
112.095 000 MHz
-100 -80 -60 -40 -20 0
0 dBFS
led_blink_f18
-62
-69
-76
-83
-90
-97
111.545 111.658 111.770 111.883 111.995 112.108 112.220 112.333 112.445 112.558 112.670 112.783 112.895 113.008 113.120

EM emissions cover a wide range of frequencies

At least one EM emission was pinpointed...but is there a better way to find all the carriers other than manually searching for them, using the available instrumentation?

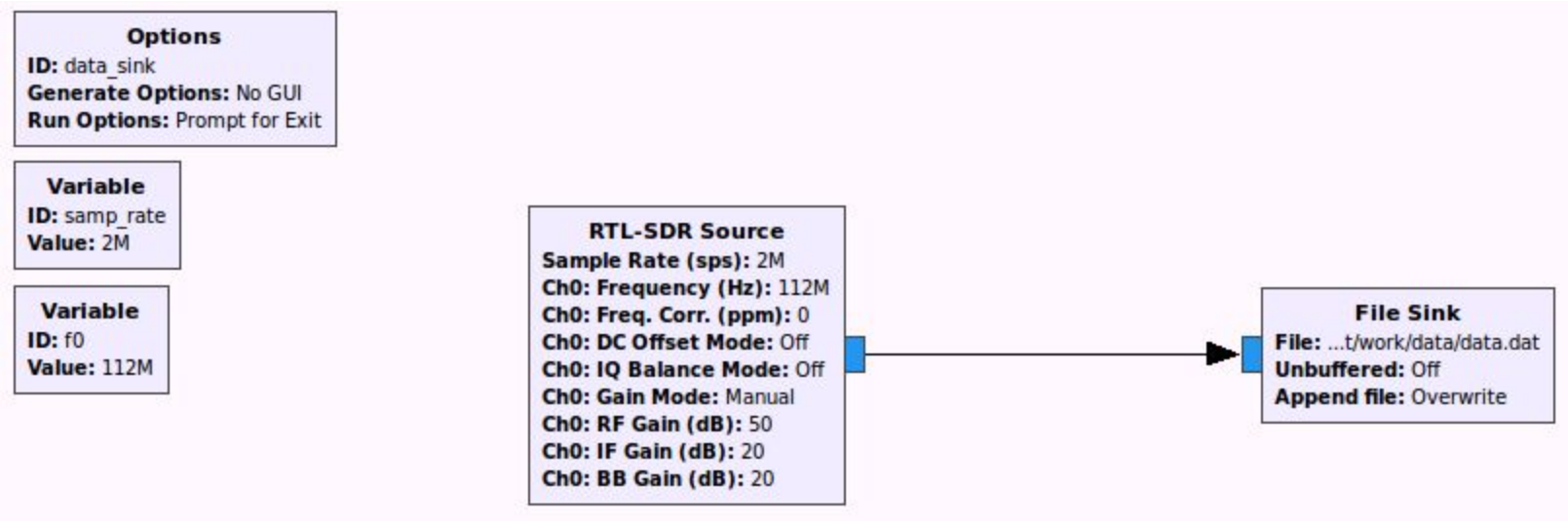
Idea:

realize a program that can do the job automatically.

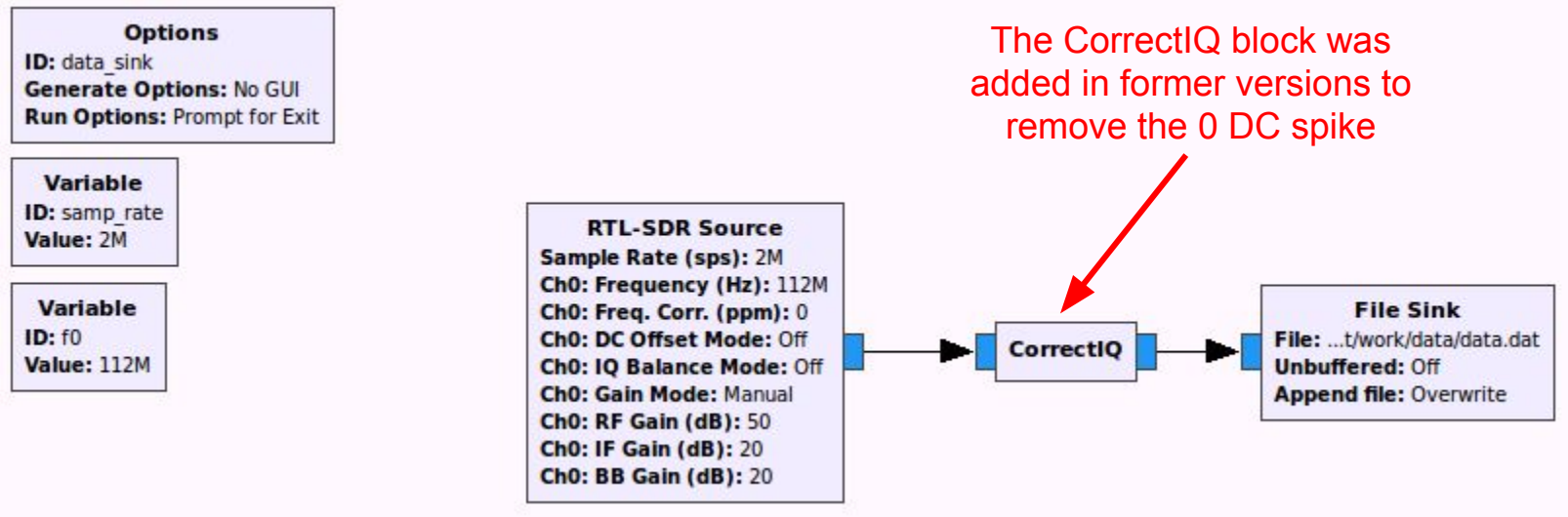
Building the scripts: GNU Radio

- GNU Radio was chosen as a basis to develop these scripts
- Since GNU Radio is largely written in Python, the scripts were developed in Python as well
- The script generated via the following .grc file was used as a starting point

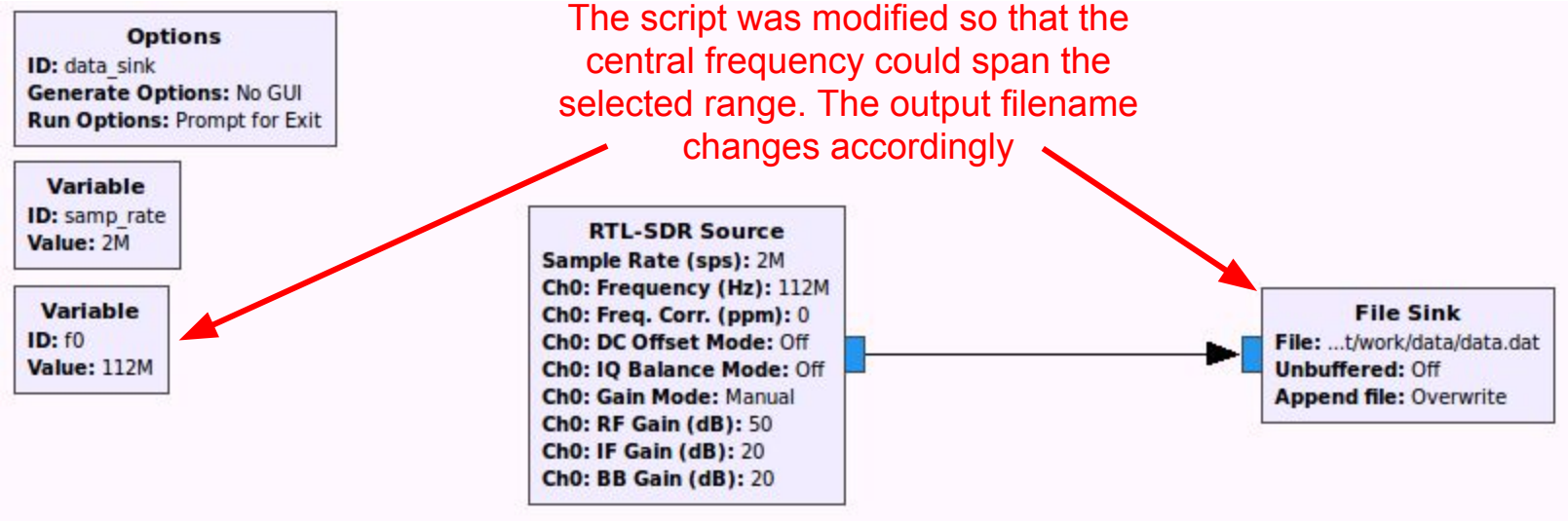
Building the scripts: GNU Radio



Building the scripts: GNU Radio



Building the scripts: GNU Radio

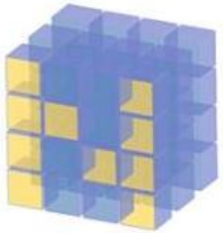


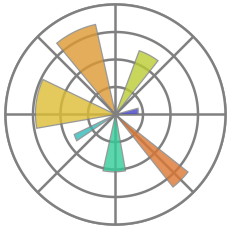
Building the scripts: time domain data

- Every acquisition in the time domain is dumped to a raw binary file
- Each sample consists in a floating point complex number
- In the current version, each acquisition lasts ca. 1 second with a 2 MHz sampling frequency
- Therefore, ca. 2 million samples are acquired for each frequency range
- Due to the chosen sampling frequency, each time domain set covers frequencies from f0 - 1 MHz to f0 + 1 MHz

Building the scripts: Numpy, Scipy and MatplotLib

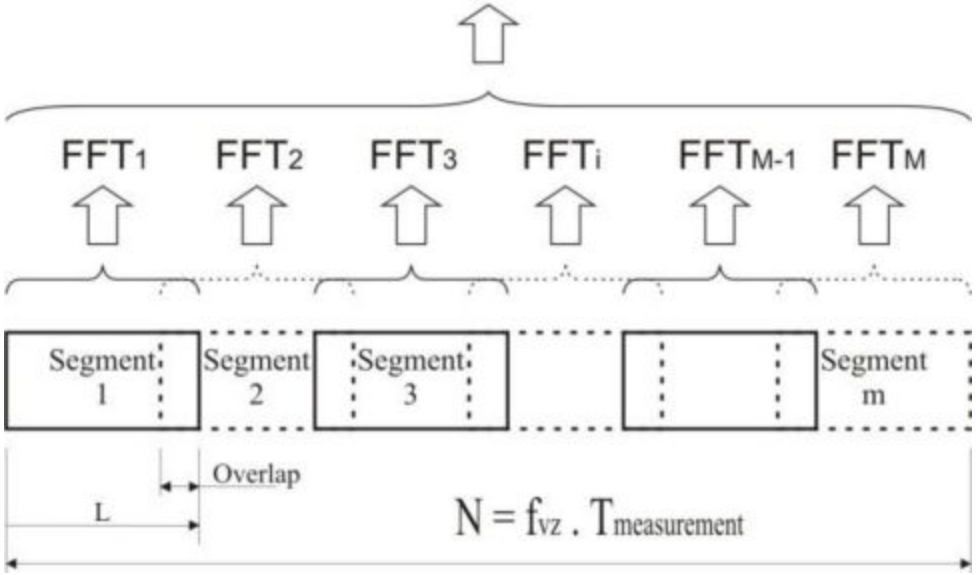
- These powerful and easy to use Python libraries were used to compute the frequency domain data
- Two functions were mainly used: **welch** and **spectrogram**





Building the scripts: Welch's method

$$\widehat{G_{xx}}(k) = \frac{1}{M}\sum_{i=1}^{M}\frac{2}{L}[FFT_i]^2$$



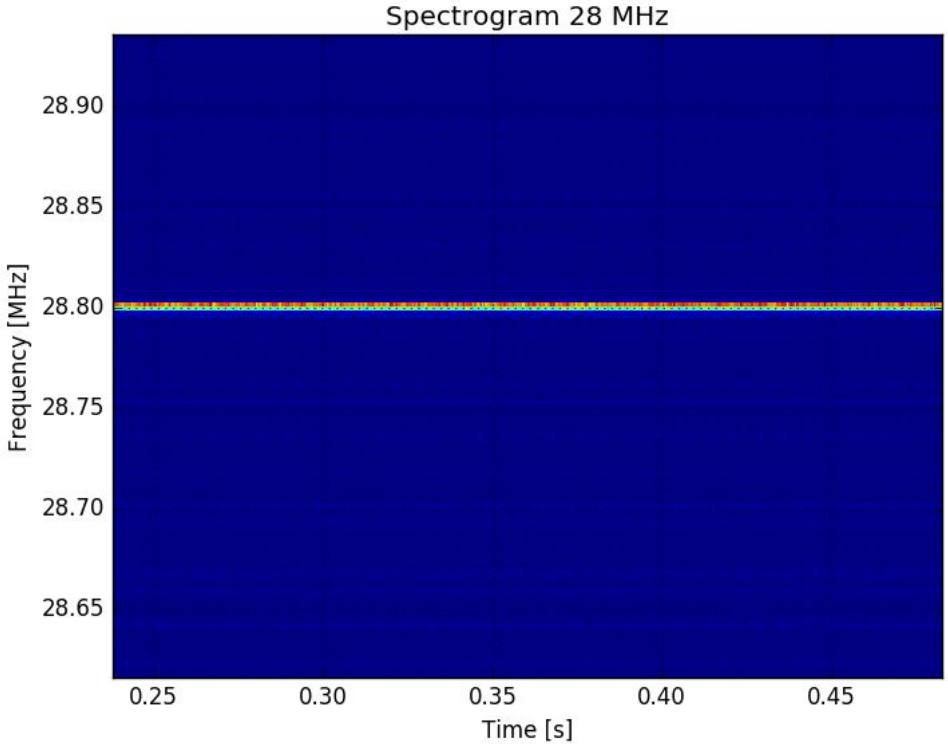
Building the scripts: Welch’s method

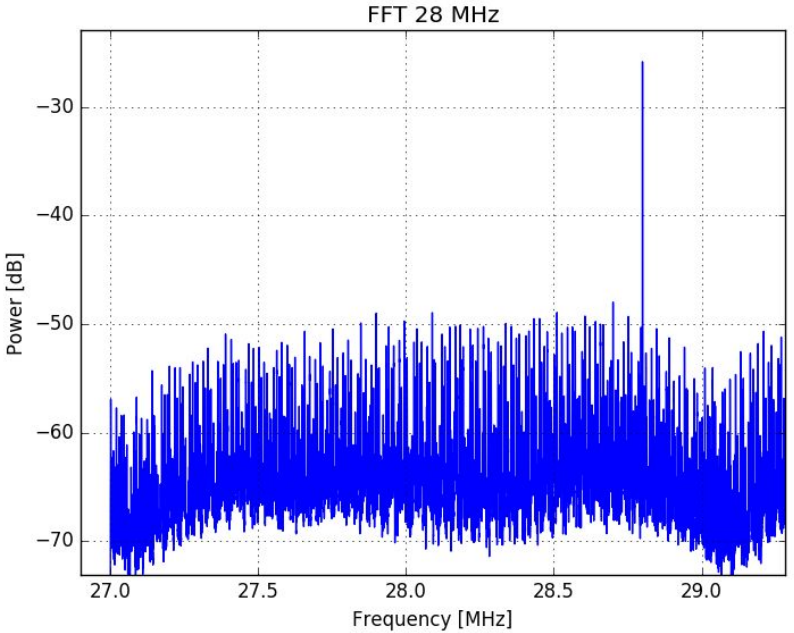
- Welch’s method allows to compute the average power spectrum by using several overlapping time domain segments and a window function
- The function also applies a linear detrending to remove the undesired 0 DC spike
- This is particularly useful because frequency domain data changes over time: an average power spectrum yields more information on the EM emissions caused by the execution of some code
- Also, it filters out random noise
- In one word: SNR is increased

Building the scripts: the spectrogram

- The spectrogram function is used to show the time evolution of frequency domain data
- It is particularly useful to perform in depth analysis of the pinpointed EM emissions

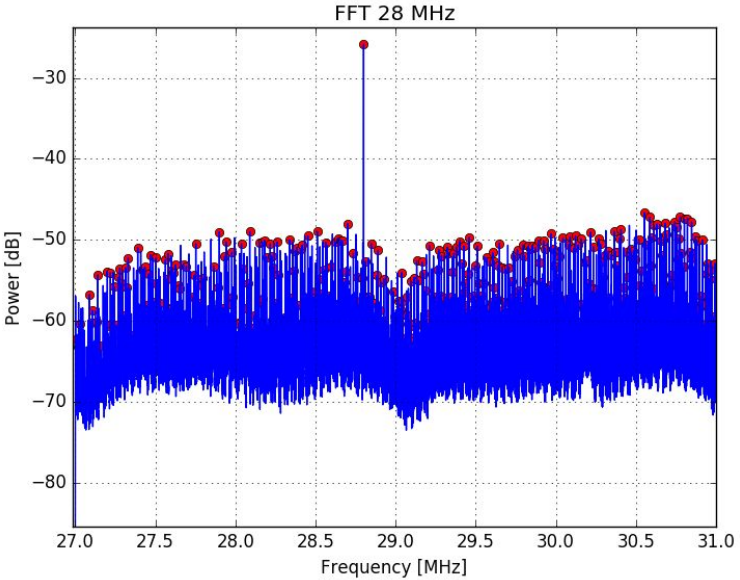
Building the scripts: the spectrogram

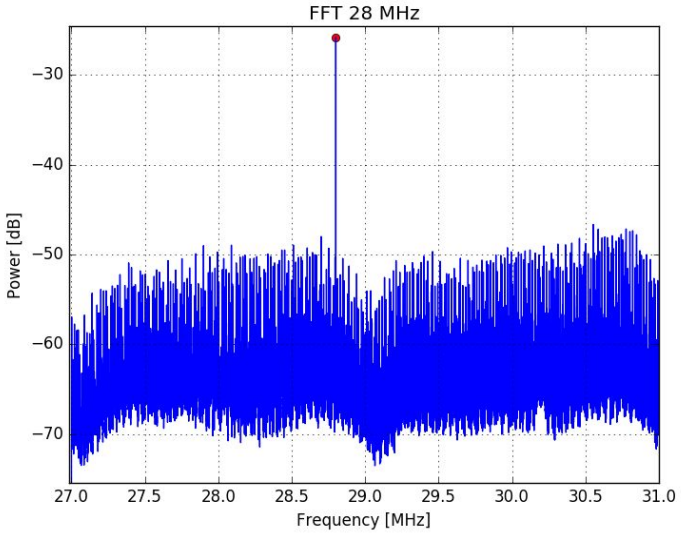




Building the scripts: detect_peaks.py

- This useful Python script was used since it is highly customizable
- However, freedom is a double edged sword: if the input parameters are not properly set, not enough peaks or too many peaks will be detected





Building the scripts: filtering out unwanted EM waves

- These scripts are mostly useful when expensive equipment is not available, i.e. outside of an anechoic chamber
- Consequence: background frequencies will interfere with the search
- Solution: filter out the background frequencies by comparing two power spectra:
  - One in which the board is off
  - Another one in which the board is executing a program

Building the scripts: filtering out unwanted EM data

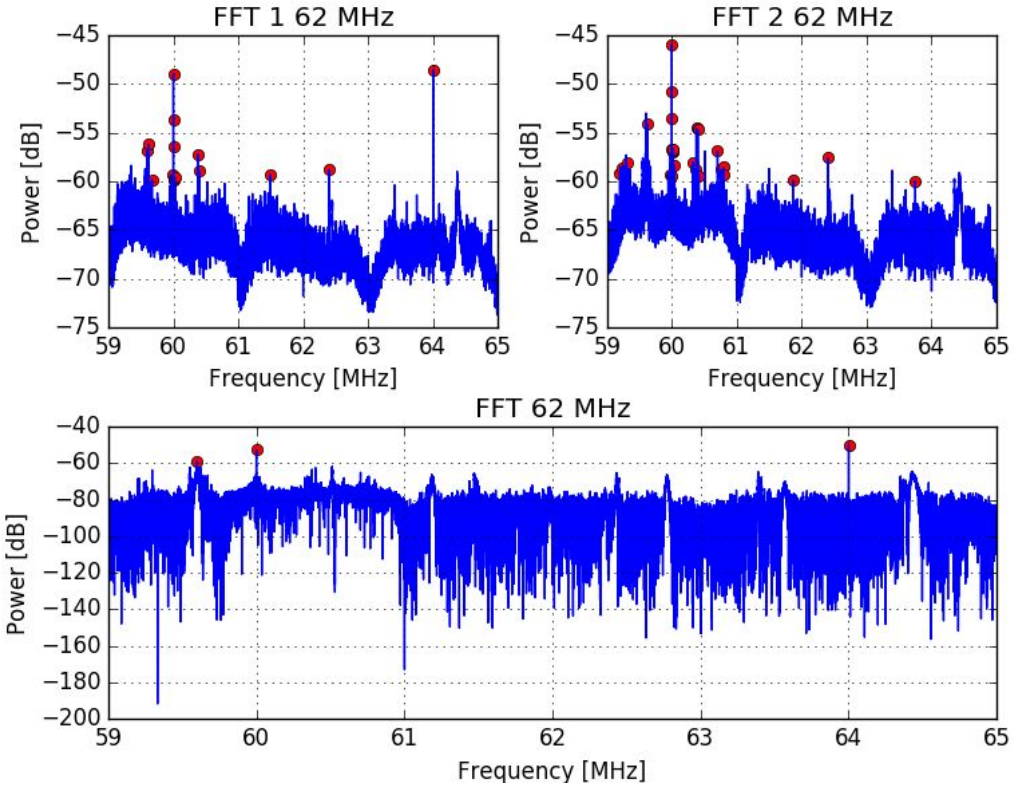
- In order to do so, a rudimentary method was used:
  1) Compute the two power spectra using Welch's method
  2) Extract the magnitudes by computing the square root of the power spectral densities
  3) Subtract the magnitudes
  4) Elevate the obtained magnitudes to two: in the new power spectrum the unwanted EM frequencies are filtered out

$$S = (\sqrt{S_1} - \sqrt{S_2})^2$$

Building the scripts: filtering out unwanted EM data

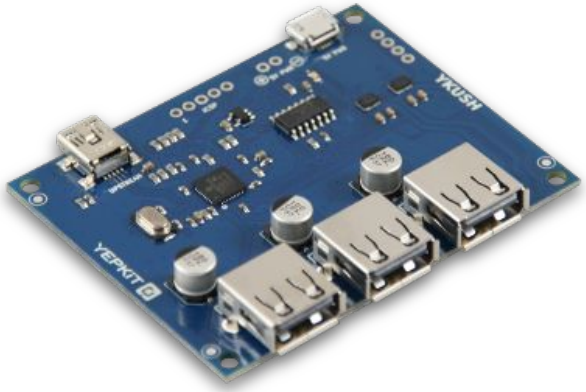
- This method is far from perfect, as many unwanted EM emissions still make it to the output spectrum with a non negligible magnitude due to many reasons (random noise, data sampling, etc.)
- The result, however, still allows to filter out most of the background emissions

Building the scripts: filtering out unwanted EM data



Building the scripts: switching the board on and off

- The Ykush Yepkit USB Switchable Hub was used in order to automatically switch the board on and off between acquisitions
- Since the program loaded on the HiFive1 is apparently stored inside a volatile memory, an additional script was used to upload the program every time the board was turned back on



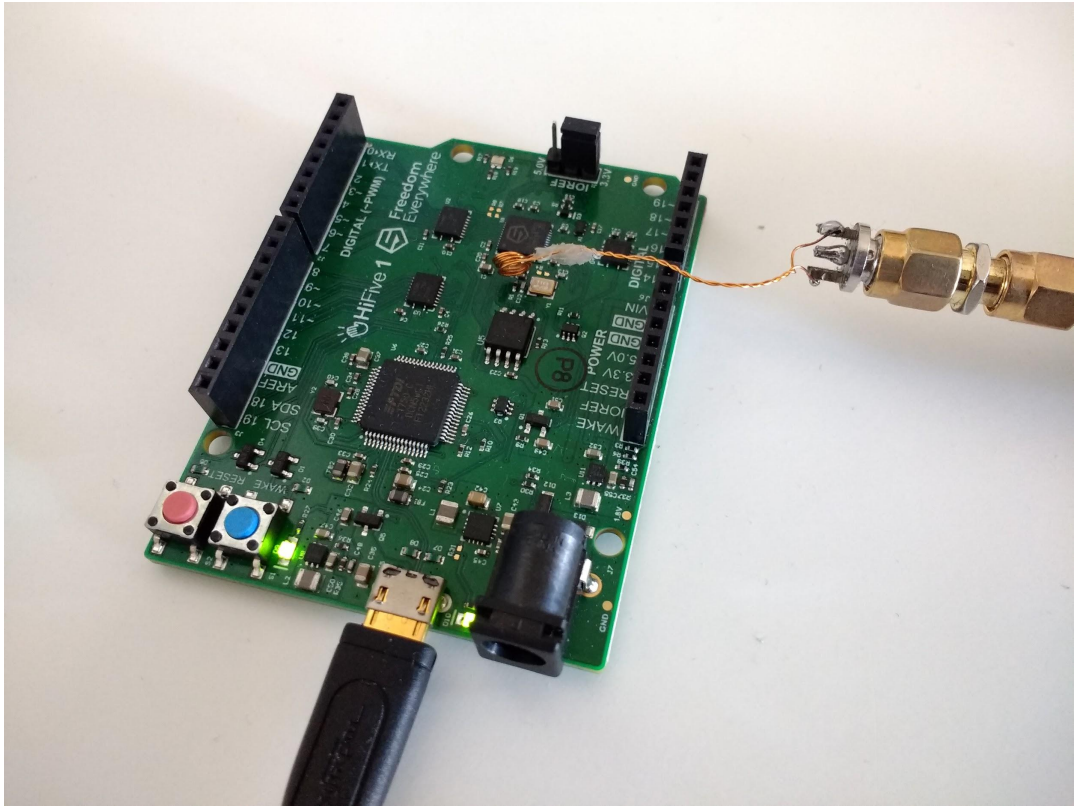
Building the scripts: additional features

- All the processed data can be exported with the -w command:
  - The peaks will be exported in a .txt file
  - Power spectral densities will be exported inside an .npz archive
- Pre processed data can be imported with the -l command
  - This is particularly useful because the processing time for large frequency spans can be considerable

Setup and tests

- The tests were performed by placing the antenna between the SPI flash memory and the RISC-V SoC
- Two acquisitions were performed: one without the board and one with the board running a simple program (i.e. a single RISC-V assembly instruction)
- The power spectrum of the difference was computed and the found peaks exported in a .txt file
- The frequencies pinpointed by the .txt file were inspected in real time with GNU Radio
- The pinpointed EM emissions where verified by moving the board away from the antenna

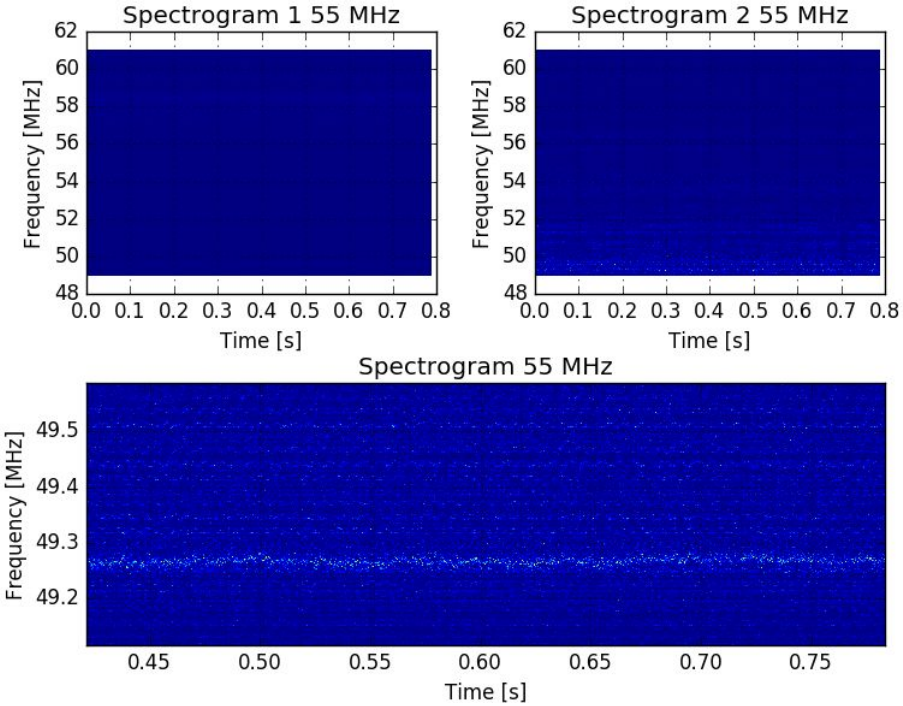
Setup and tests

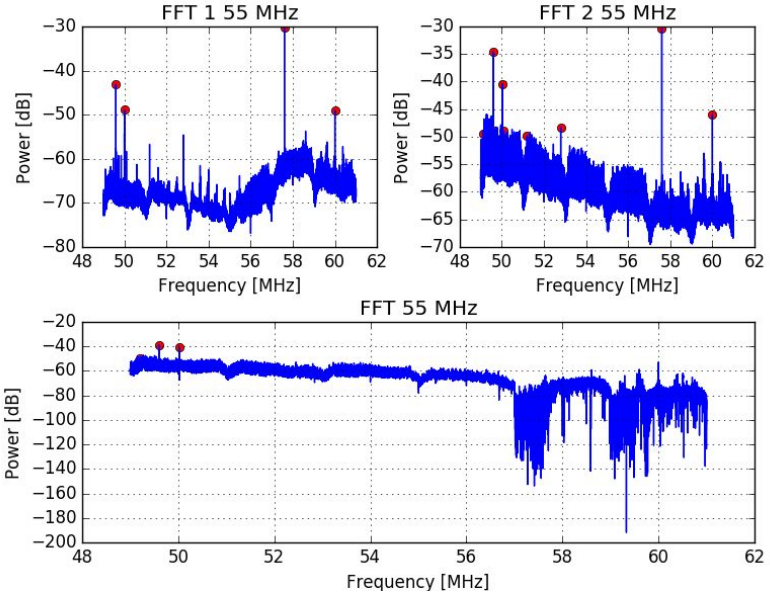


Conclusions

- Due to time constraints, only two frequencies were isolated with confidence
- The script is far from being perfect
- However, it still provides a useful set of tools for frequency domain analysis
- If the highlighted issues will be solved this could be a valuable instrument for the study of the EM side channel of any embedded system

DEMO


Spectrogram 1 55 MHz
Frequency [MHz]
Time [s]
Spectrogram 2 55 MHz
Frequency [MHz]
Time [s]
Spectrogram 55 MHz
Frequency [MHz]
Time [s]


FFT 1 55 MHz
Power [dB]
Frequency [MHz]
FFT 2 55 MHz
Power [dB]
Frequency [MHz]
FFT 55 MHz
Power [dB]
Frequency [MHz]

Questions?