SNE-ESHTER

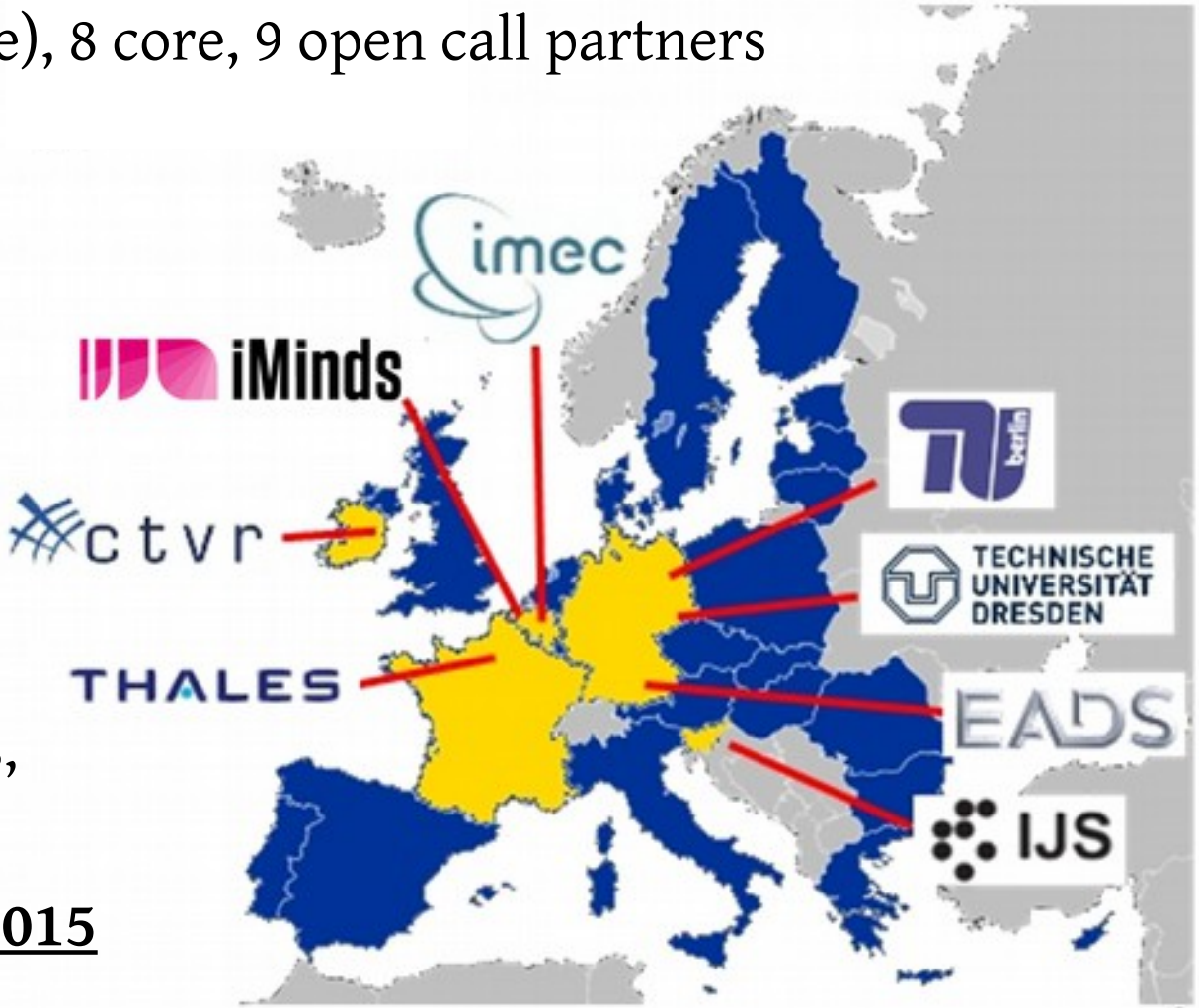
Cross Transceiver API implementation

Tomaž Šolc, Jožef Stefan Institute

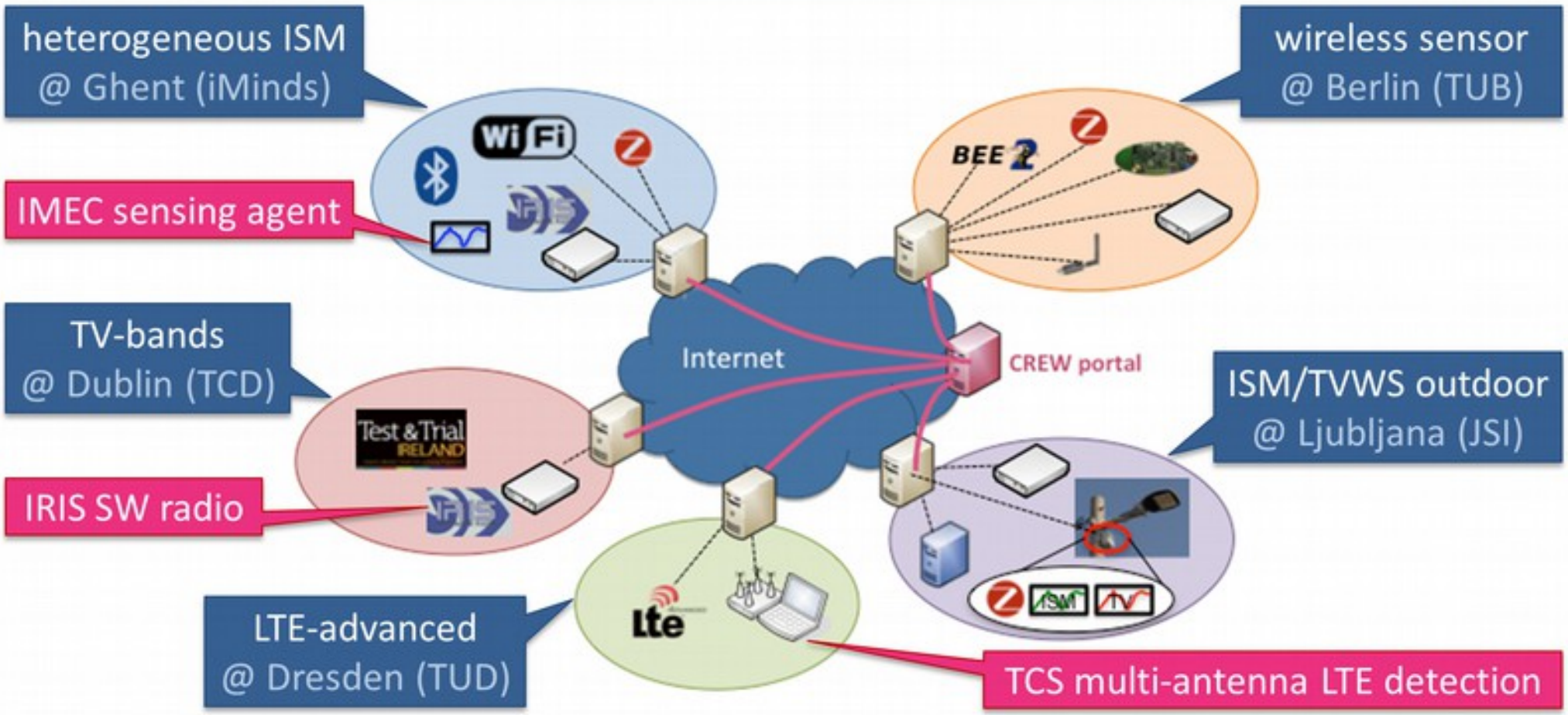
The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°258301 (CREW project).

IP CREW

- **Cognitive Radio Experimentation World**
  - Facilitate research into cognitive radio by establishing an open federated test platform.
  - FP7 call 5 (FIRE Initiative), 8 core, 9 open call partners
  - **NOT** doing research nor developing new algorithms
  - Developing facilities for supporting research
  - Augmenting existing facilities
  - Offering better methods, validating solutions
  - **<u>Project ends October 2015</u>**



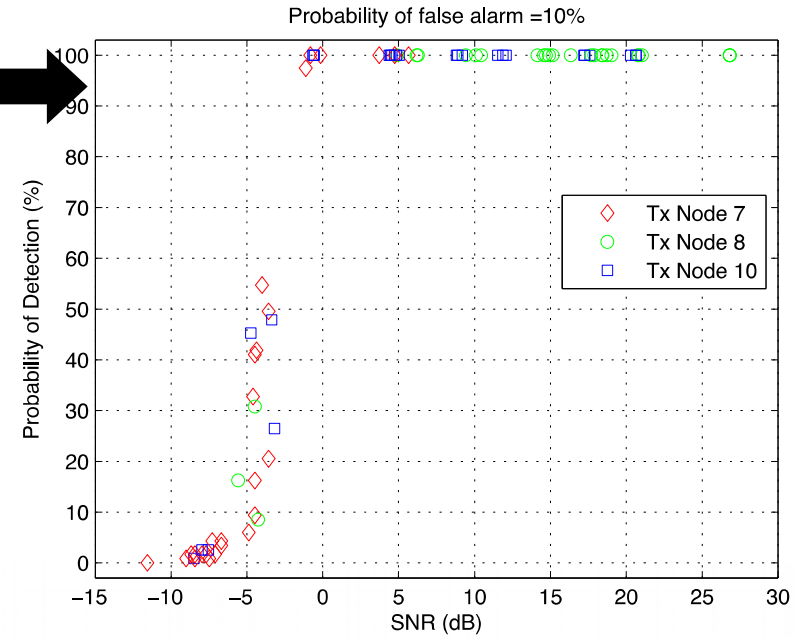
CREW platform

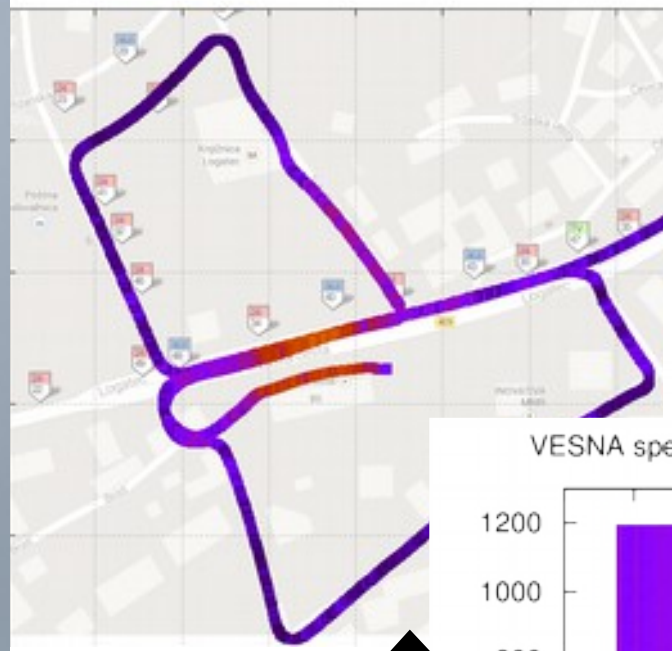


IEEE 802.11	IRIS GPP-based software radio platform	IMEC Sensing Agent
IEEE 802.15.1	Comreg spectrum licenses	UHF/VHF TV sensing
IEEE 802.15.4	BEE2 FPGA platform	ISM bands sensing
LTE-advanced	USRP software radio	TCS Multi-antenna LTE detection
EyesIFX nodes	VESNA platform on light pole	WiSpy Spectrum analyzer
CR database		Interconnection of portals
		Interconn. between testbed elements

LOG-a-TEC testbed

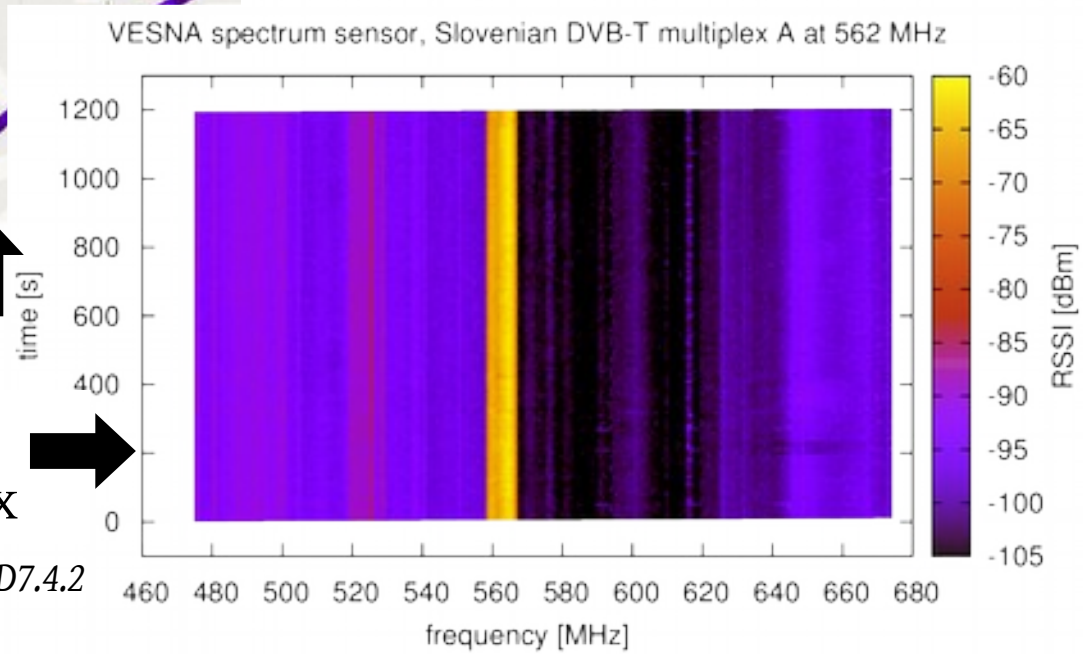
Wireless mic. detection in CREWTV experiment[1]





Transmitter localization[2]

Detection of DVB-T multiplex



[1] Dionísio, Ribeiro, Marques: *CREW project deliverable D7.4.2*
[2] Moerman, et al.: *CREW project deliverable D6.3*

Unified transceiver API

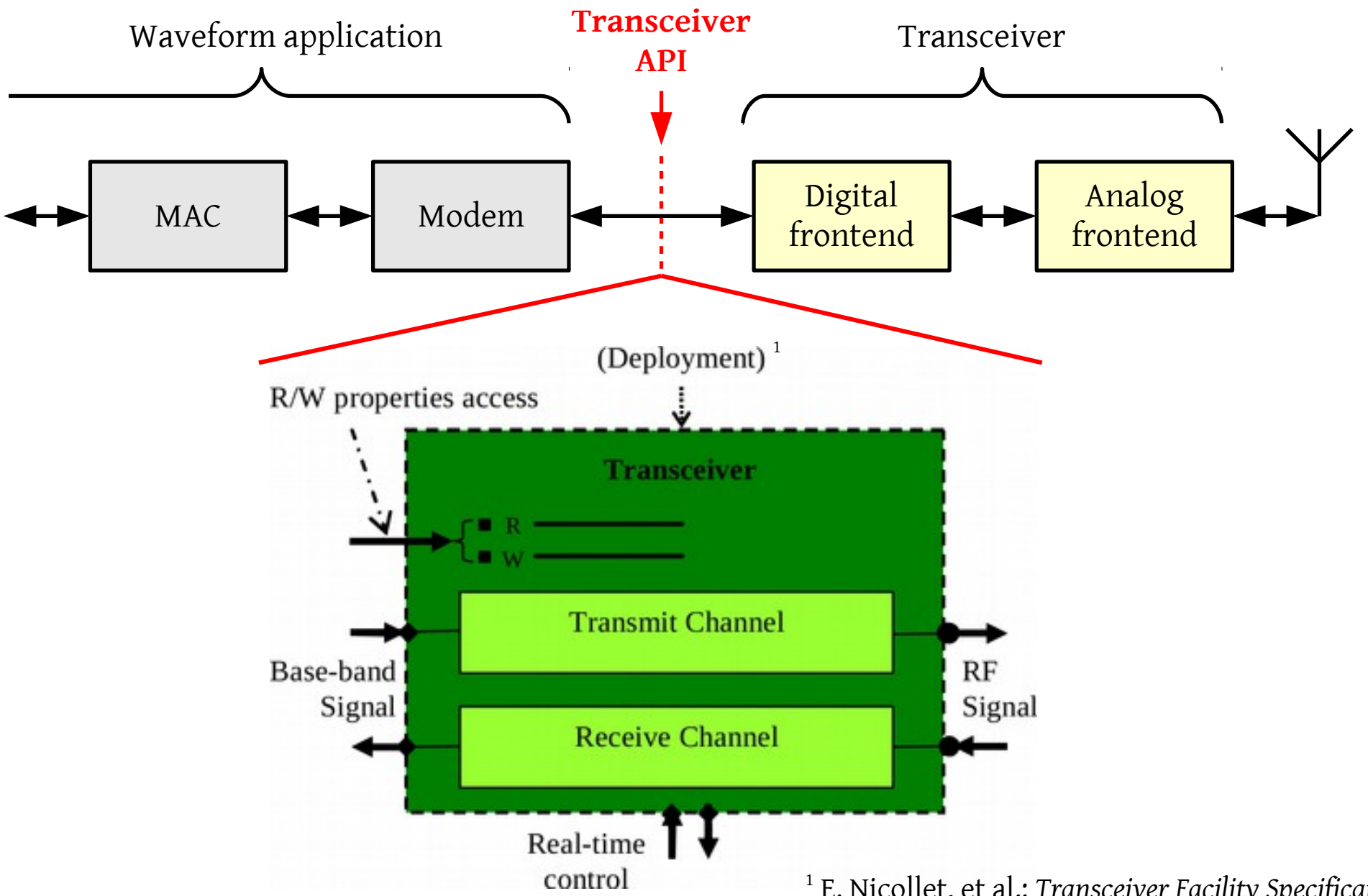
- CREW uses a large variety of transceiver hardware
  - True SDR nodes (Ettus Research USRP, WARP)
  - Spectrum sensors (SNE-ESHTER, Imec Sensing Engine)
  - [Low-power, narrow band radios on sensor nodes]

- <u>Each transceiver has its own native interface</u>

Unified transceiver API (cont.)

- To perform an experiment, testbed users:
  - develop an application running on nodes,
  - remotely upload and start the application and
  - perform measurements using test instrumentation.
- One API for all radio hardware would simplify
  - application development for testbed users and
  - portability of an experiment between testbeds
- WINNF Transceiver Facility
  - selected as an API for SDR nodes early in the project,
  - <u>could be also used for access to spectrum sensors?</u>

WInnF Transceiver Facility



[1] E. Nicollet, et al.: *Transceiver Facility Specification*

WInnF Transceiver Facility

- Baseband RX/TX sample streaming
  - Asynchronous callback (*PushBBSamplesXX)*
- Time handling
  - <u>Events</u>: Receive Start and Stop, Transmit Start and Stop
  - <u>Discriminators</u>: immediate, absolute, eventBased, undef.
- Tuning presets
  - Sampling frequency, channel bandwidth, gain, filter characteristics, etc.
- Central frequency, ...

Application code example (C++)

```
class Receiver : public Transceiver::I_ReceiveDataPush
{
   public:
      void pushBBSamplesRx(
            Transceiver::BBPacket* thePushedPacket,
            Transceiver::Boolean endOfBurst)
      {
         ...
      };
};


Transceiver::Time start(Transceiver::immediateDiscriminator);
Transceiver::Time stop(Transceiver::undefinedDiscriminator);
Transceiver::ULong packet_size = 2048;
Transceiver::UShort tuning_preset = 2;
Transceiver::Frequency freq = 700000000;

device->receiveChannel.createReceiveCycleProfile(
      start, stop, packet_size, tuning_preset, freq);
```

SDR vs. Spect. sensor

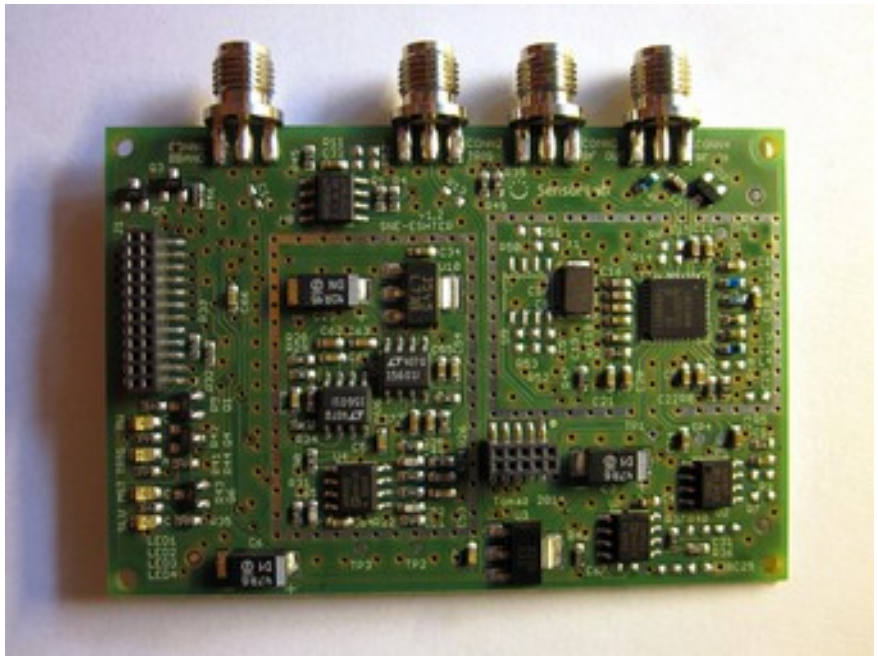
SDR	Spect. sensor
• RX and TX	• RX only
• Continuous reception/ unlimited burst length	• Limited buffer for a sample-process cycle
• Frequency agile, low-latency	• Uses a predetermined sequence
• Fast turn on/turn off	• Continuous scanning of a frequency band
• Optimized for signal processing in software	• Optimized for on-board signal processing

VESNA SNE-ESHTER

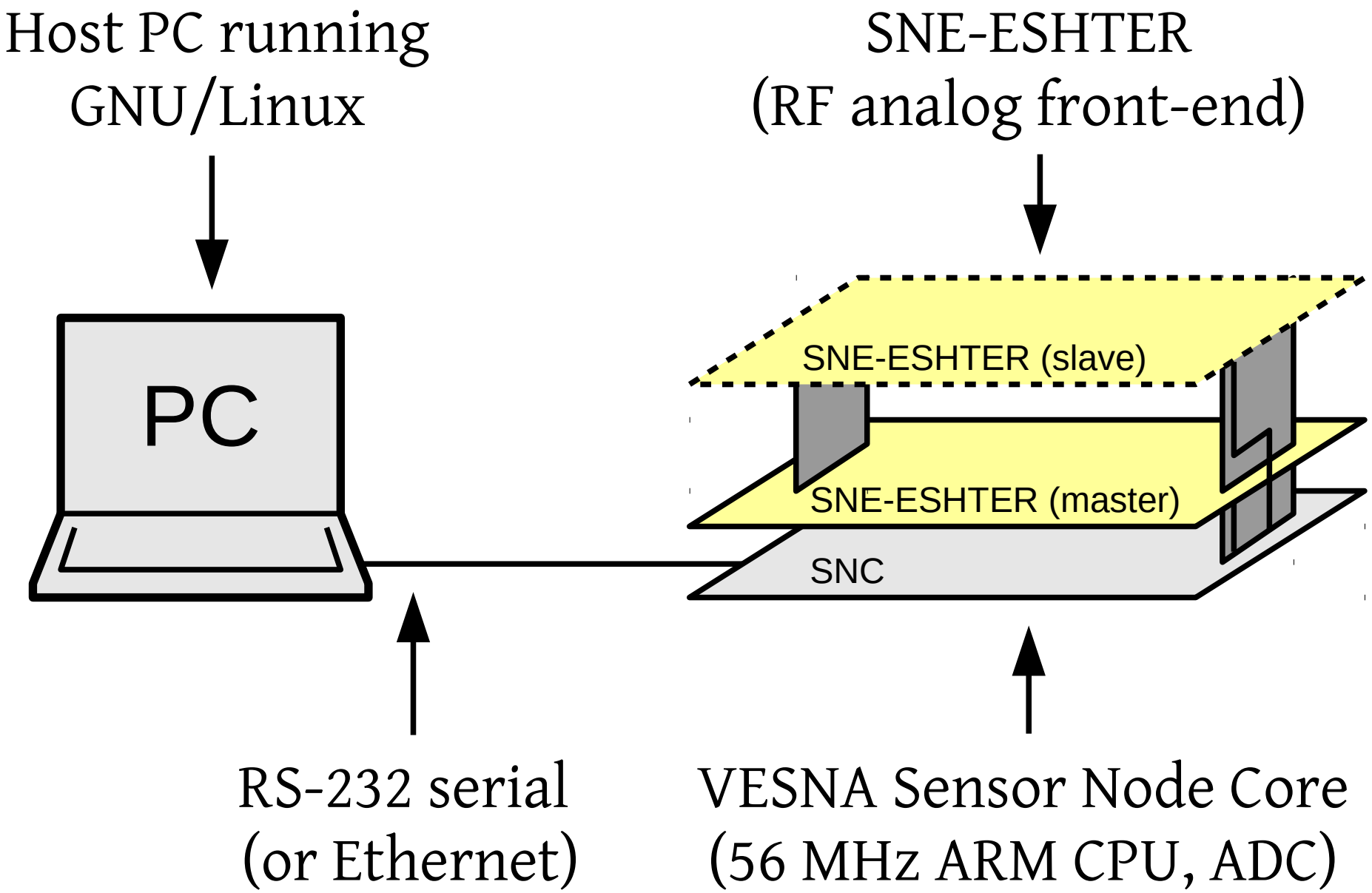
Compact, low-cost spectrum sensor for VHF/UHF bands

- Selectable sensing bandwidths from 8 MHz to 500 kHz
- Baseband signal capture (up to 2 Msample/s, 25000 samples)
- Statistical processing on sensor node CPU (covariance, Eigenv.)
- Low-latency programmable hardware trigger for energy det.
- Compressive and multi-antenna sensing.

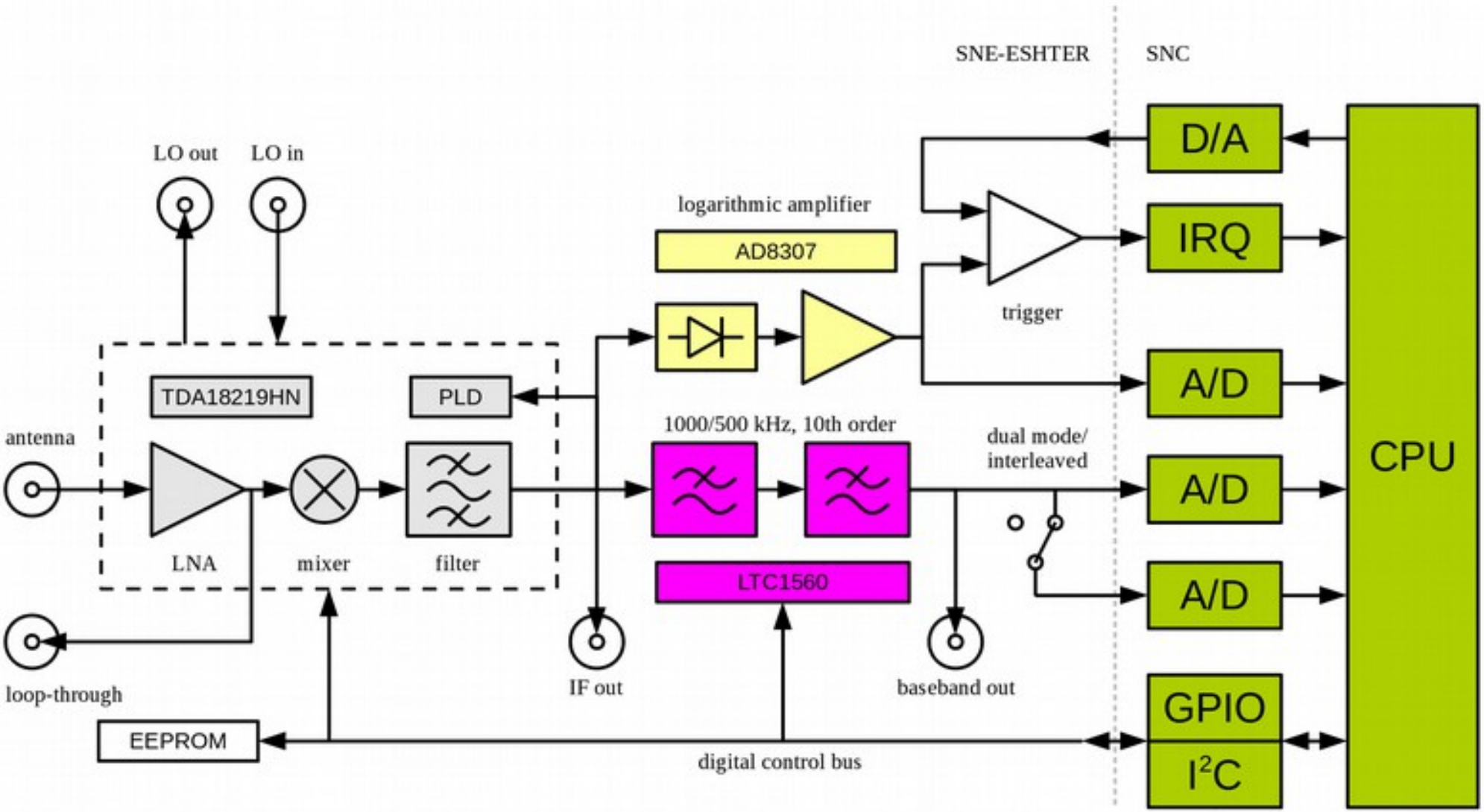
SNE-ESHTER
(embedded sensing hardware for TVWS experimental radio)



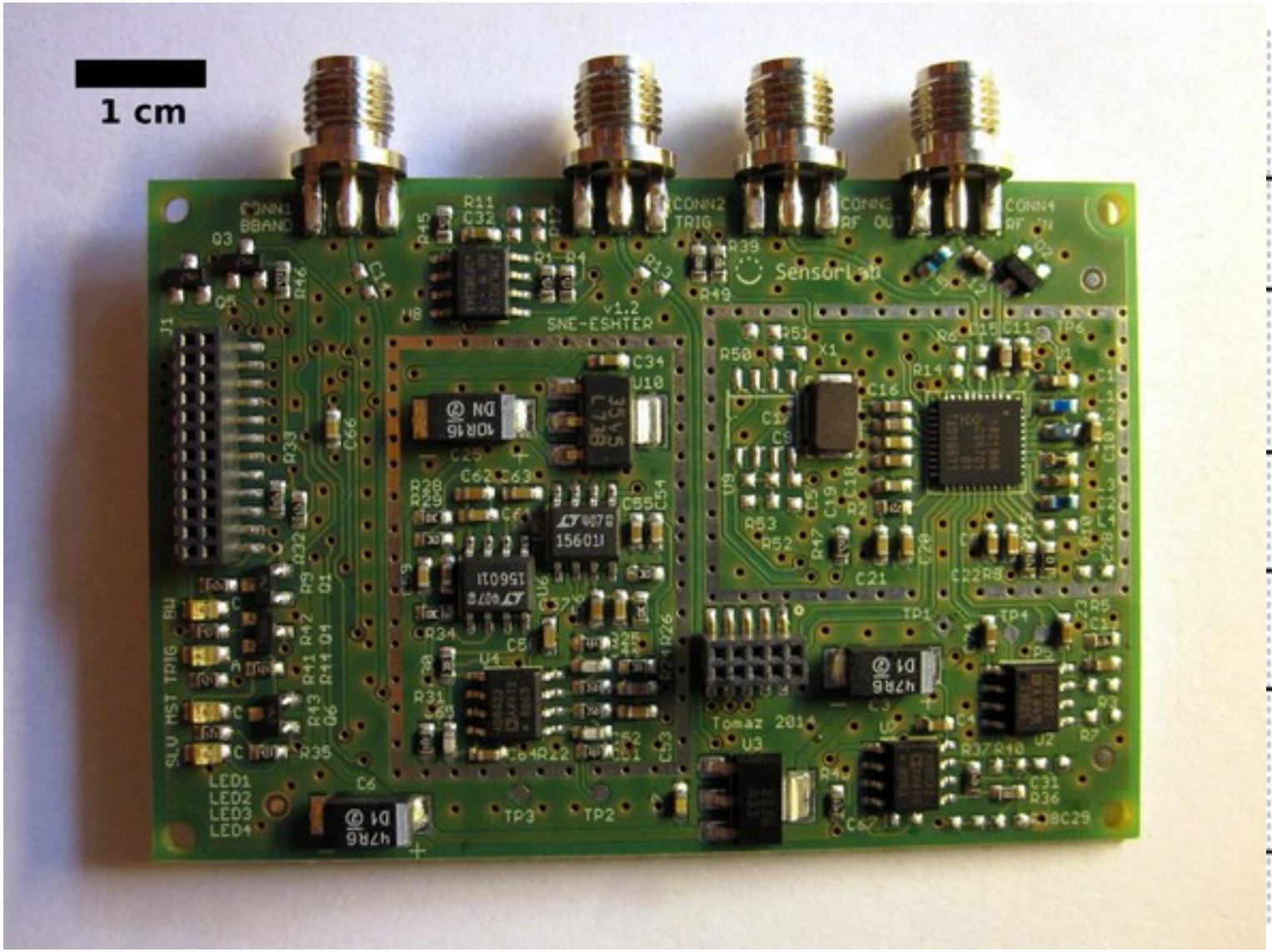
Typical setup in a testbed

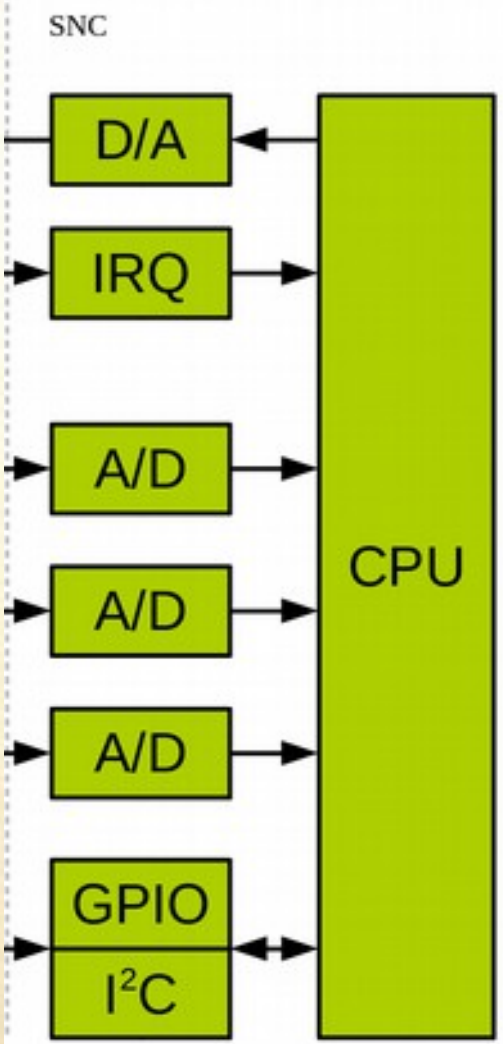


RF analog front-end

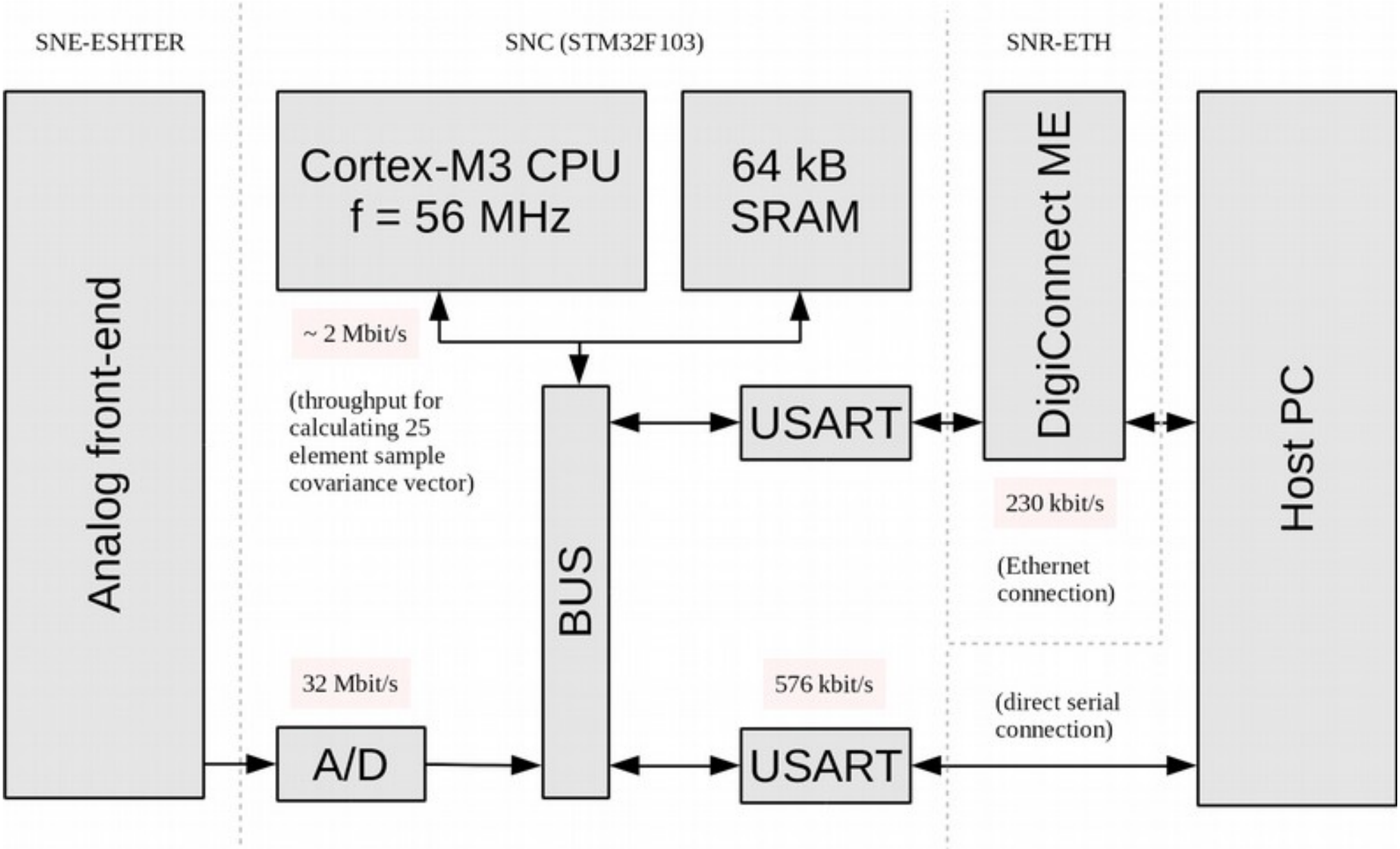


RF analog front-end





Throughputs



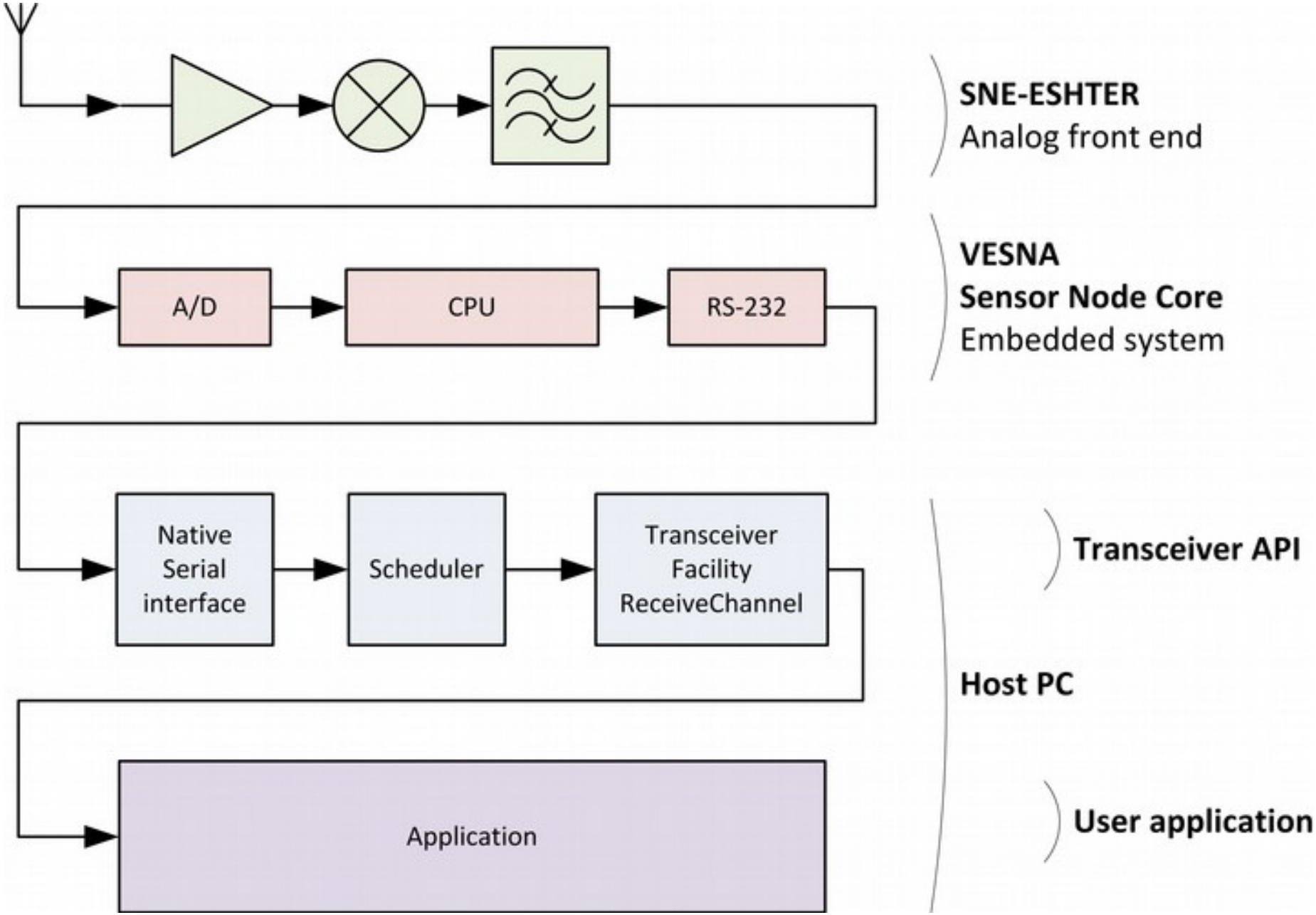
Serial line protocol

HOST PC

```
-> select channel 650000:1:650001 config 0,2
<- ok

-> samples 1024
<- ok

-> sample on
<- TS 0.001 CH 650000 DS 2042 2045 2053 ...
<- TS 0.136 CH 650000 DS 2053 2056 2042 ...
<- ...

-> sample off
<- ok
```

SENSOR

System overview



Application code example

```
class Receiver : public Transceiver::I_ReceiveDataPush
{
    public:
        void pushBBSamplesRx(
                Transceiver::BBPacket* thePushedPacket,
                Transceiver::Boolean endOfBurst)
        {
            ...
        };
};

...
```

Specific to SNE-ESHTER

```
SpectrumSensor *sensor = new SpectrumSensor(device);
DeviceImp* eshter = new DeviceImp(receiver, sensor);

eshter->receiveChannel.createReceiveCycleProfile(
        start, stop, packet_size, tuning_preset, freq);
```

Supported features

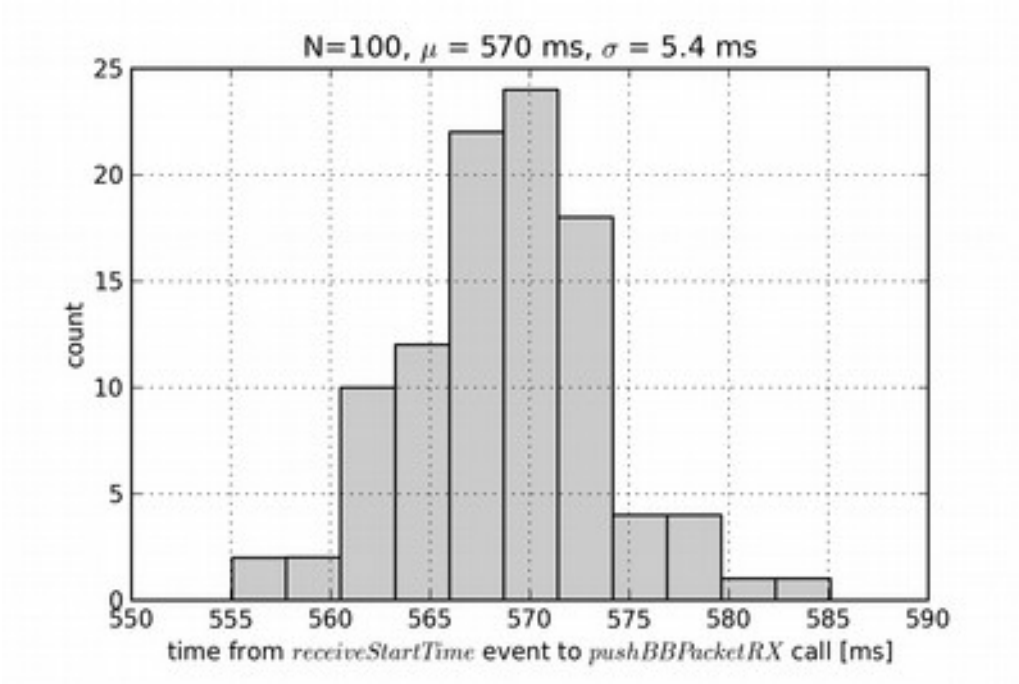
- **Two hardware modes supported**
  - B=1 MHz, fs=2 Msample/s
  - B=500 kHz, fs=1 Msample/s
- **Arbitrary packet length (up to 25000 samples)**
  - SNE-ESHTER does not use quadrature sampling (Q=0)
  - sample loss occurs between packets
- ***createReceiveCycleProfile()***
  - defined or undefined packet length - *setReceiveStopTime()*
  - *configureReceiveCycle()* not supported
- **scheduling using discriminators supported**
  - *absolute*, *eventBased*, *immediate*, *undefined*
  - *eventBased* takes at most one event in the past as reference
  - host PC clock used as reference

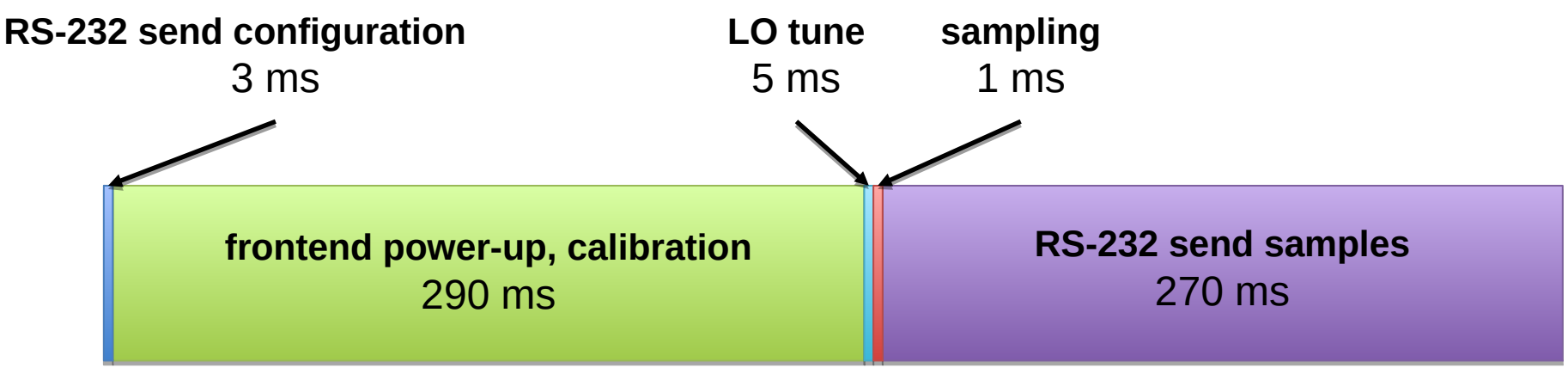
Unsupported features & problems

- **Advanced hardware features unsupported**
  - Using multiple analog-frontends,
  - analog energy detector and programmable trigger,
  - on board statistical signal processing.
- **High latency, long receive start and stop times**
  - Transceiver Facility is only a façade over a slow serial protocol
- **Dropped samples**
  - Bandwidth restrictions in SNE-ESHTER prevent uninterrupted streaming of samples.
  - Impossible to describe streaming limitation in API
  - Impossible for application to know about the gap between packets pushed to the application
- **Paper discussing this accepted at WInnComm Europe 2015**

Benchmarks

- **Latency – *receiveStart* to *pushBBSamples*** (2048 samples) **approx. <u>570 ms</u>**
  - Frontend slow to resume from power saving mode.
  - Sending ASCII formatted samples over 576 kbps RS-232 connection.
- **Blind time of sensor in this mode > 99.5%**



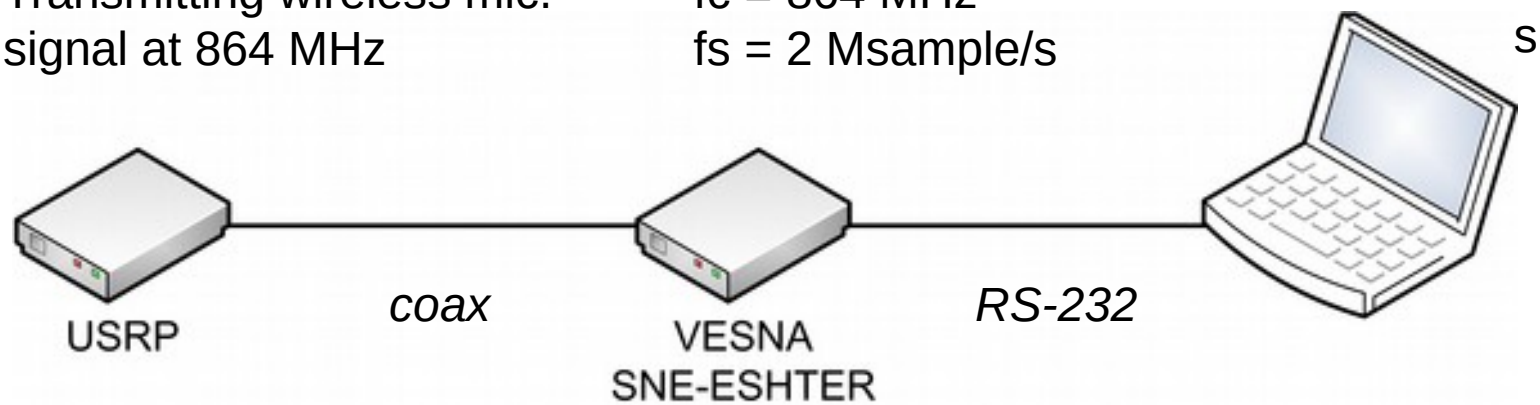


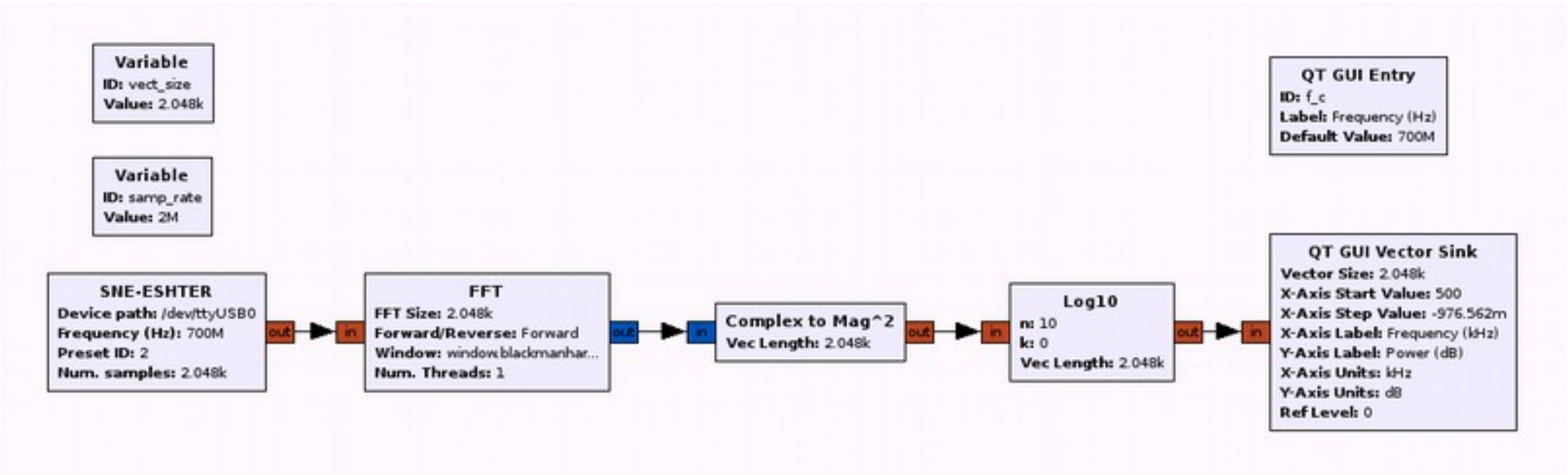
Demo setup

Transmitting wireless mic. signal at 864 MHz

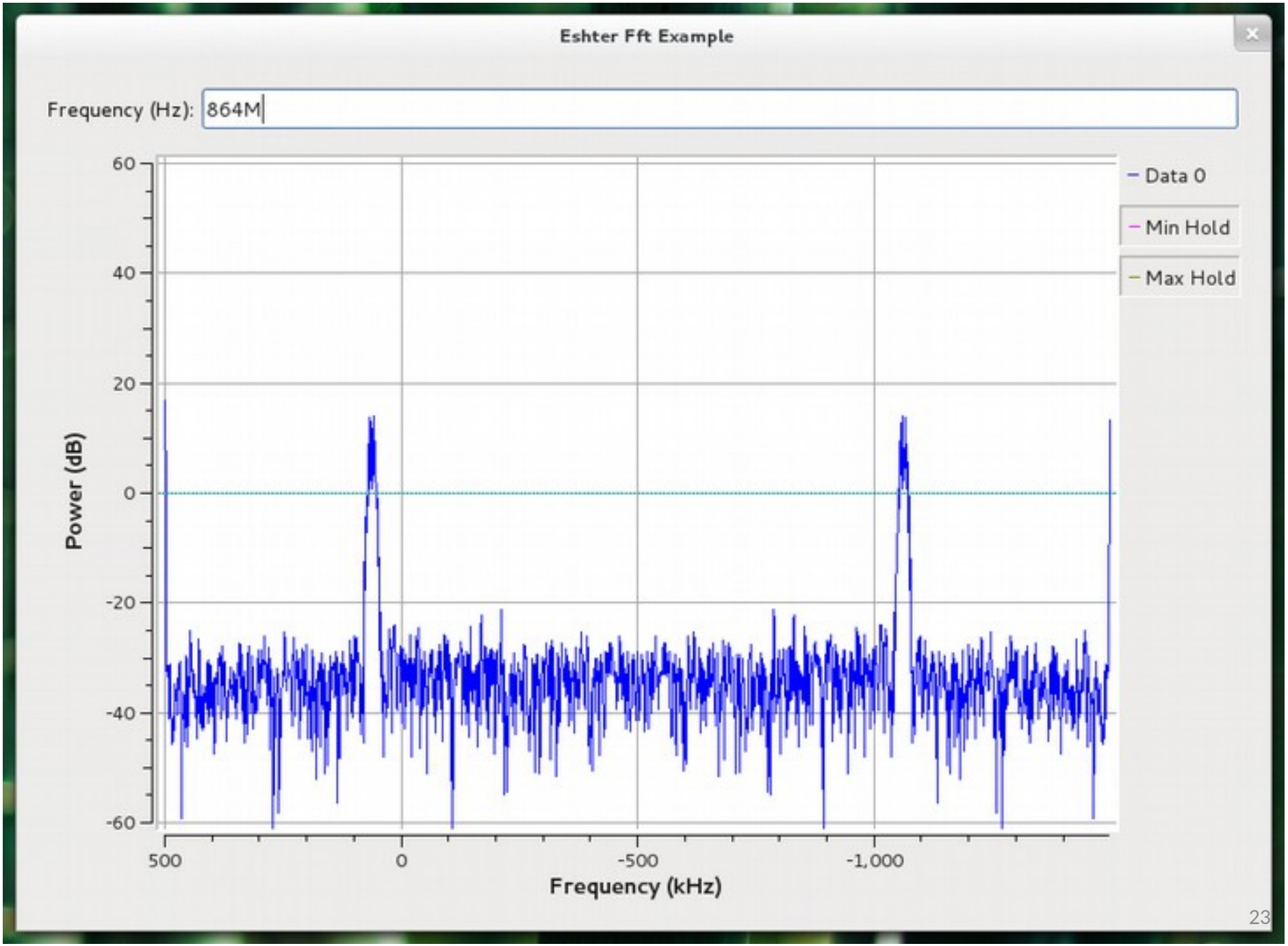
fc = 864 MHz
fs = 2 Msample/s

Calculate and display power spectral density





Demo


Eshter Fft Example
Frequency (Hz): 864M
Data 0
Min Hold
Max Hold
Power (dB)
60
40
20
0
-20
-40
-60
500
0
-500
-1,000
Frequency (kHz)

Questions?

Tomaž Šolc
tomaz.solc@ijs.si