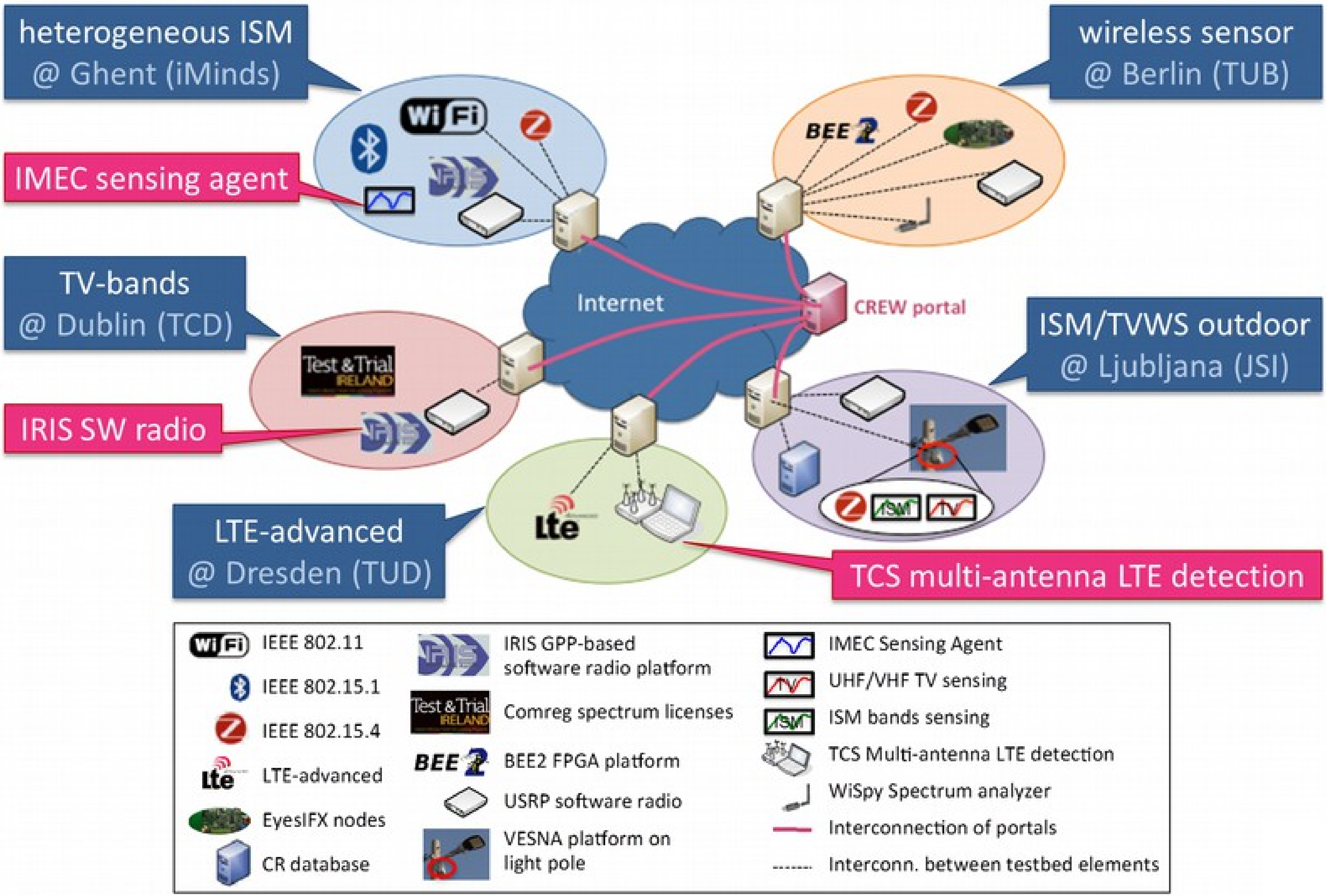
Adopting WINNF Transceiver Facility for Spectrum Sensors

Tomaž Šolc, Jožef Stefan Institute

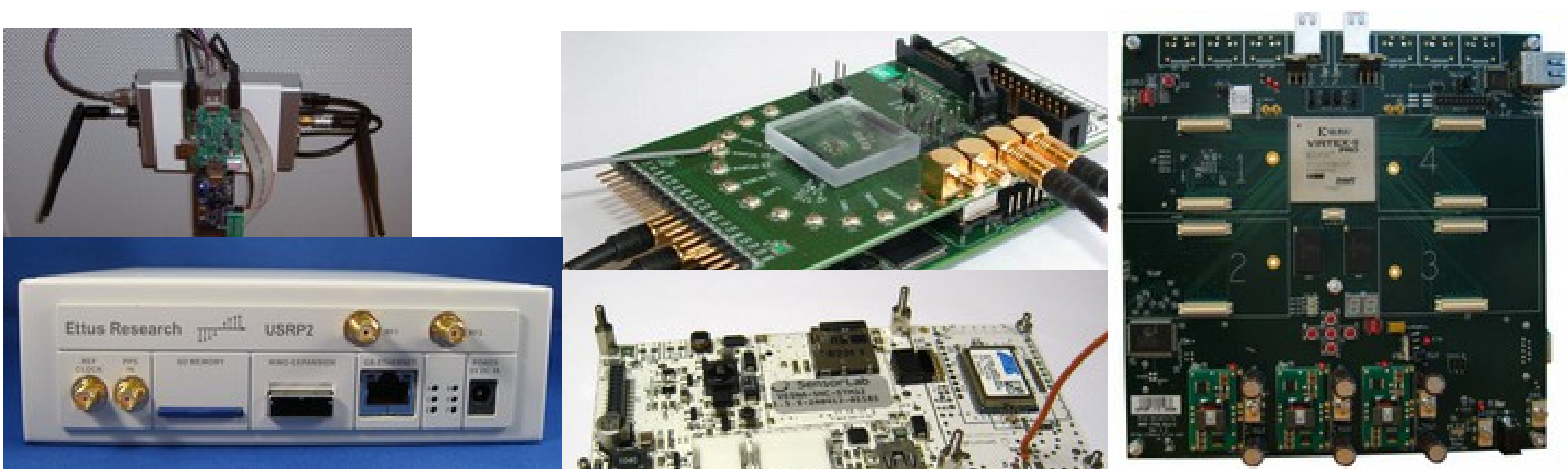
The research leading to these results has received funding from the European Union's
Seventh Framework Programme (FP7/2007-2013) under grant agreement n°258301 (CREW project).

CREW platform



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 embedded devices]

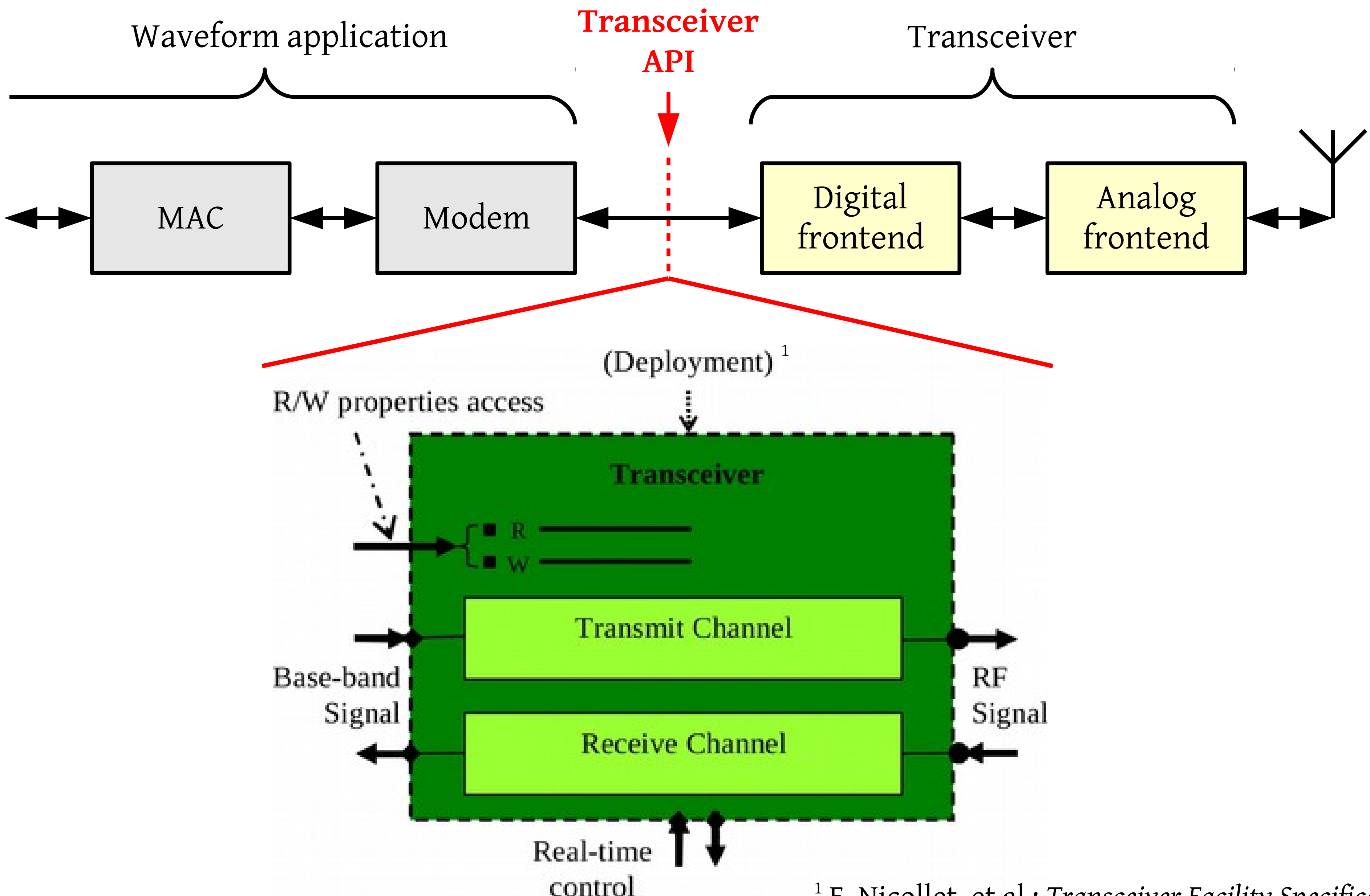


- <u>Each transceiver has its own native interface</u>

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,
  - could be also used for access to spectrum sensors?

WInnF Transceiver Facility



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

SDR vs. Spect. sensor

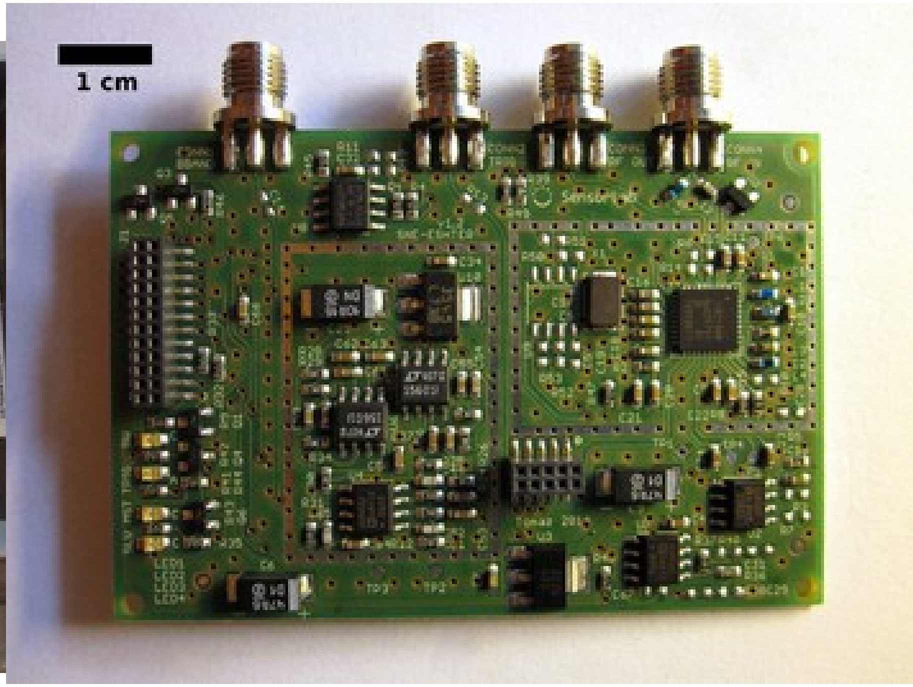
SDR	Spect. sensor
RX and TX	RX only
Continuous reception/ unlimited burst length	Limited buffer for a sample-process cycle
Frequency agile, low-latency	Uses a predetermined sequence
Fast turn on/turn off	Continuous scanning of a frequency band
Optimized for signal processing in software	Optimized for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

VESNA SNE-ESHTER

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

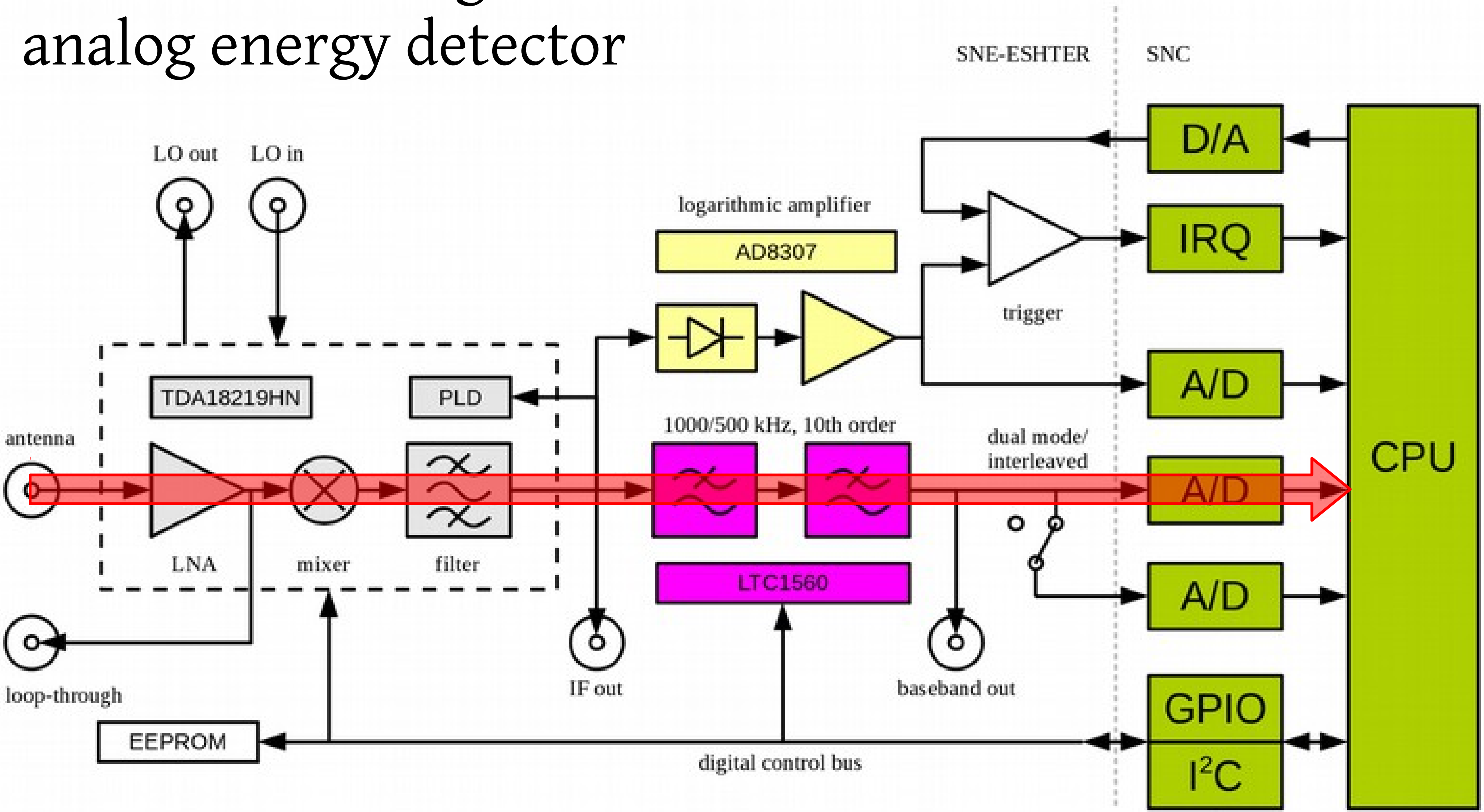
- Selectable sensing bandwidths from 8 MHz to 500 kHz
- Baseband signal capture (up to 2 Msample/s, 25000 samples)
- Statistical processing on sensor node CPU (covariance, Eigenv. det.)
- Low-latency programmable hardware trigger for energy detection
- Compressive and multi-antenna, multi-channel sensing

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

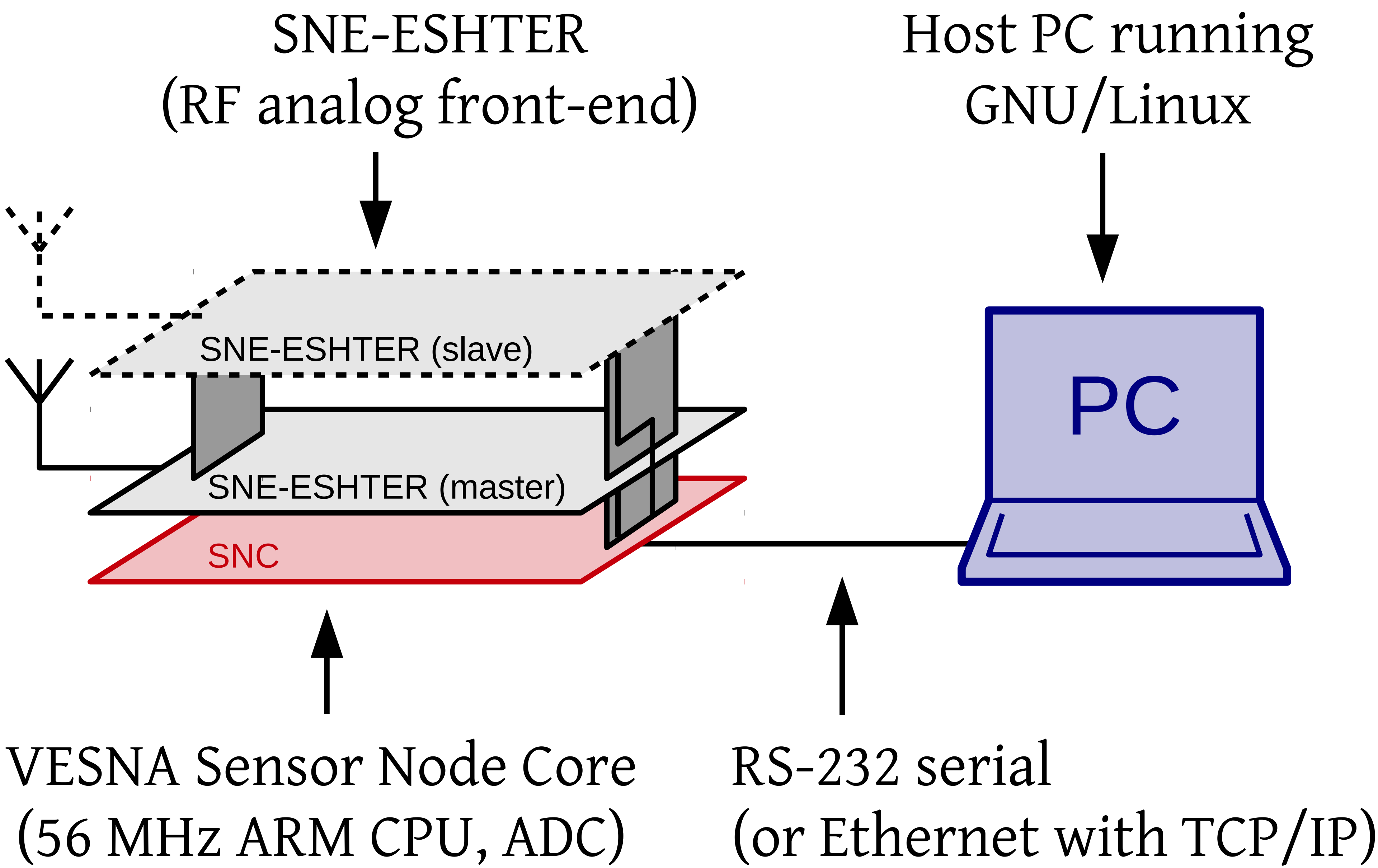


RF analog front-end

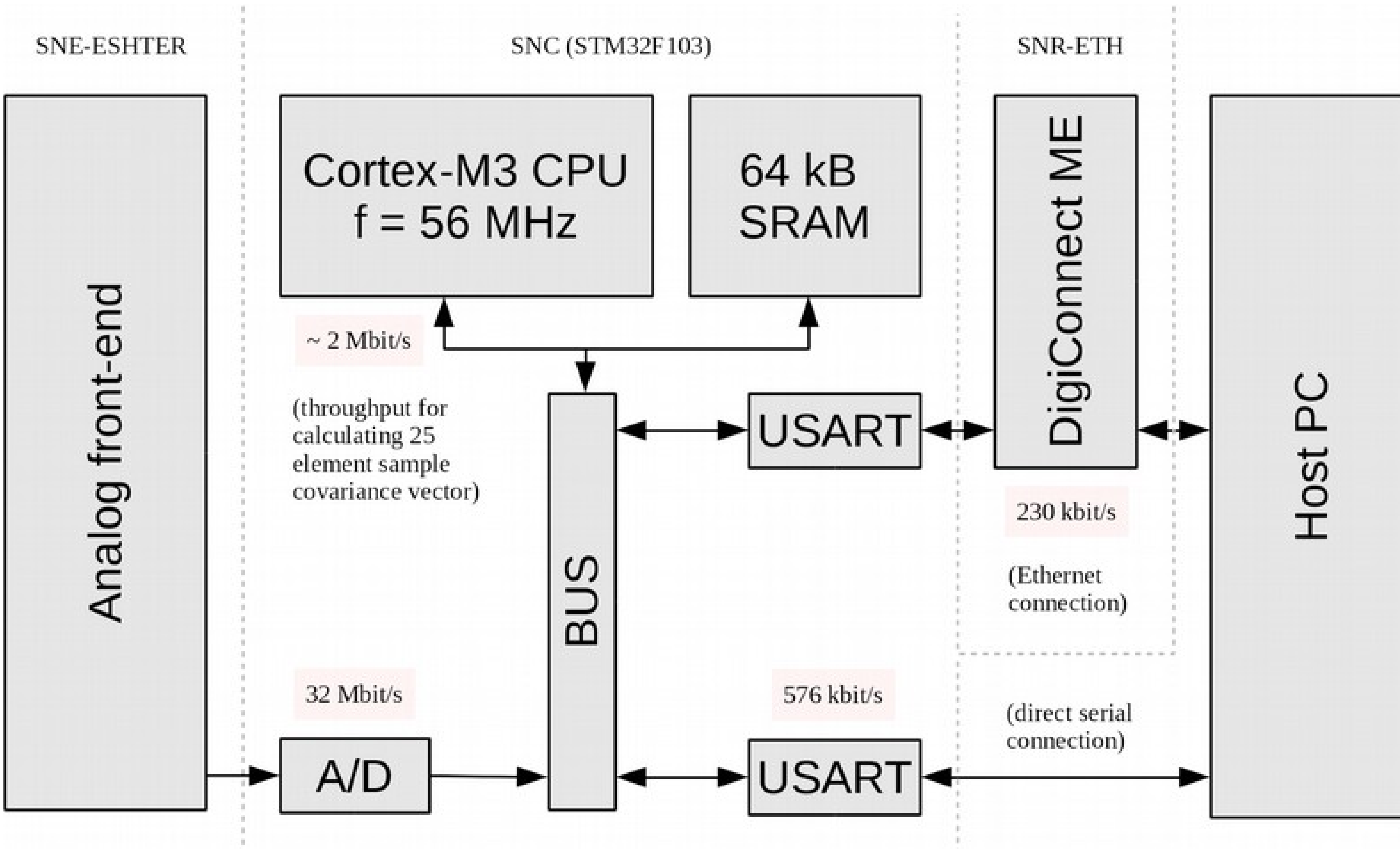
42 – 870 MHz single-conversion, low-IF receiver
analog energy detector



Typical setup in a testbed

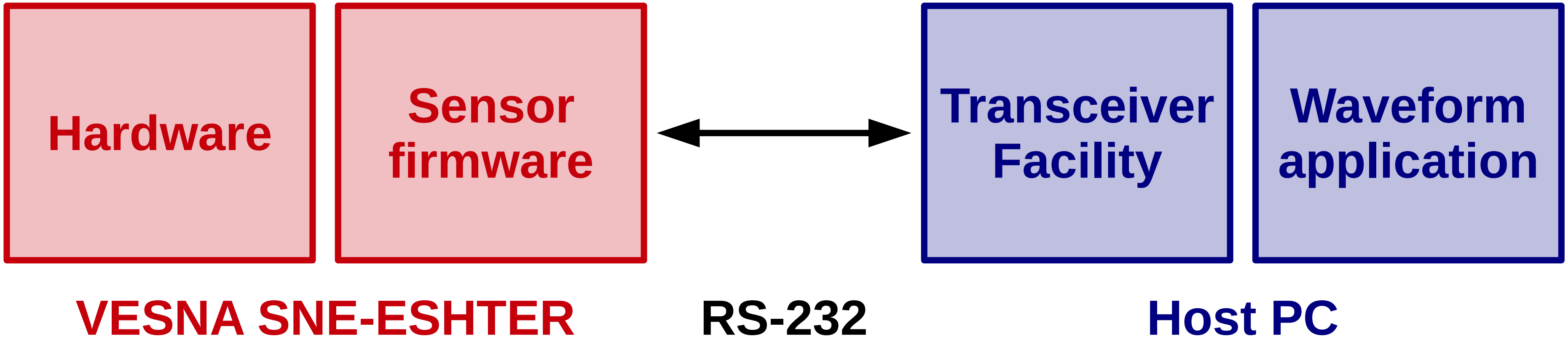


Throughputs



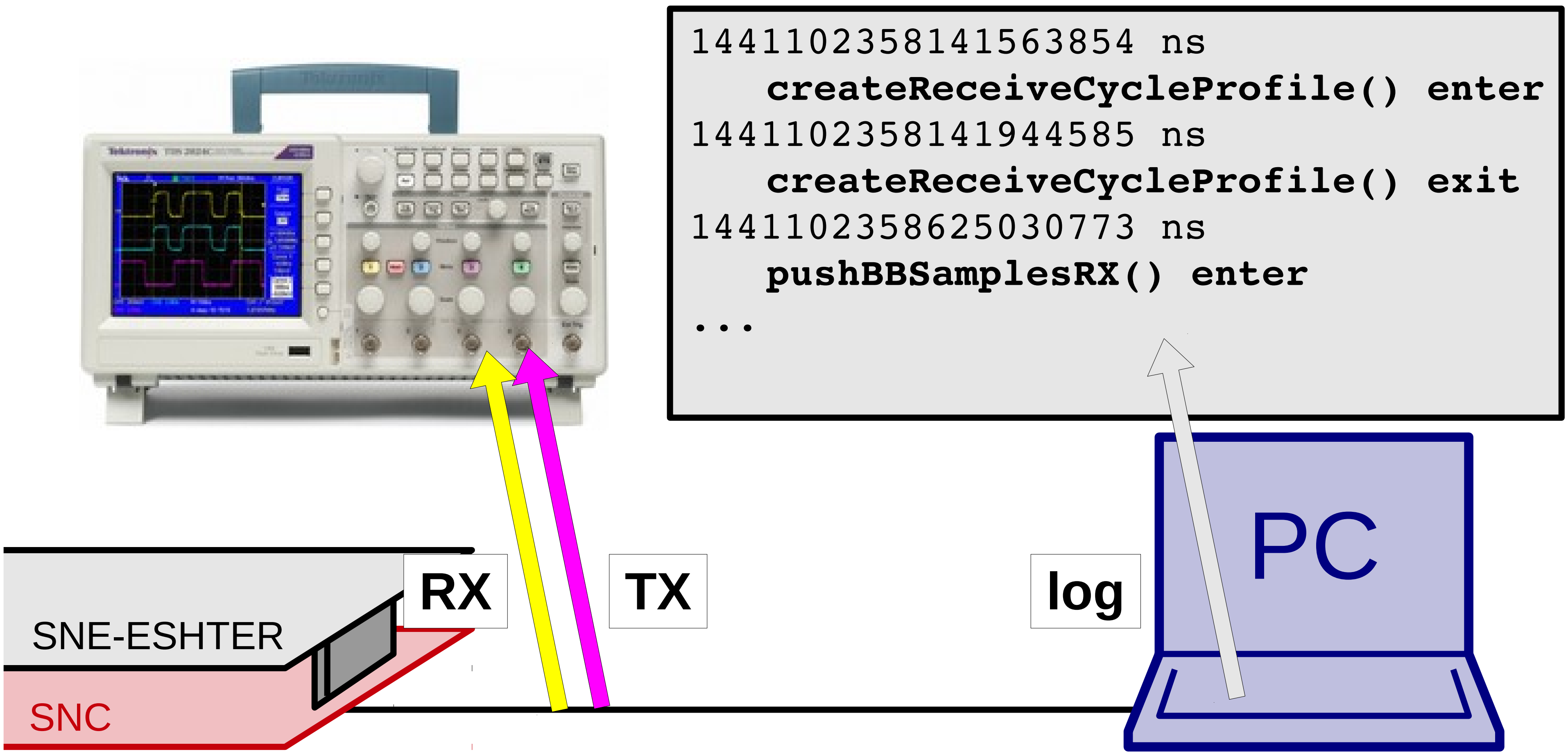
Implementation overview

- C++ library, linked with user application
  - Transceiver Facility specification is language agnostic.
  - When changing hardware, just link with a different library.
- Running on the Host PC
  - No need to reprogram sensor's firmware,
  - but limited by the serial interface, host PC clock.

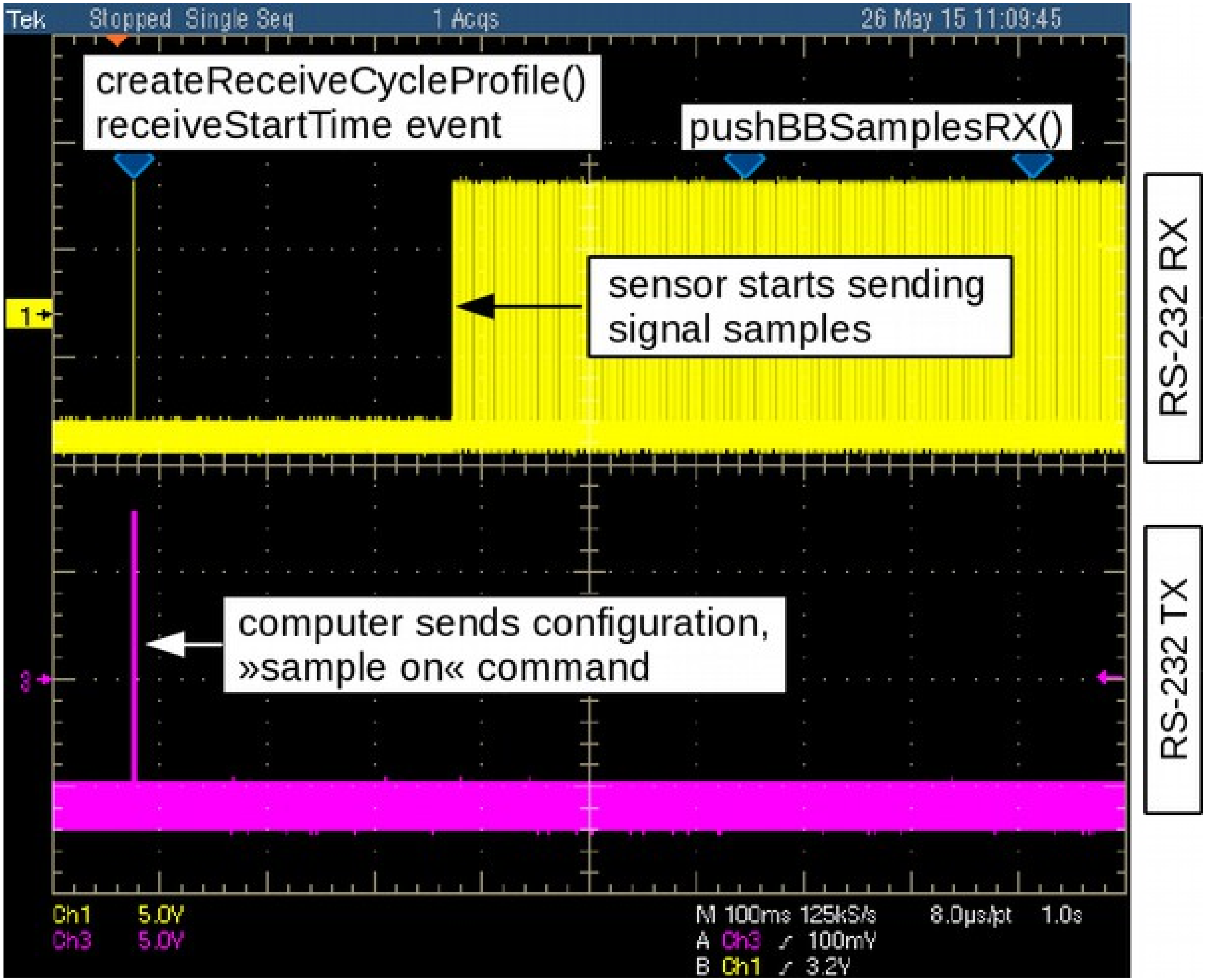


Latency benchmarks

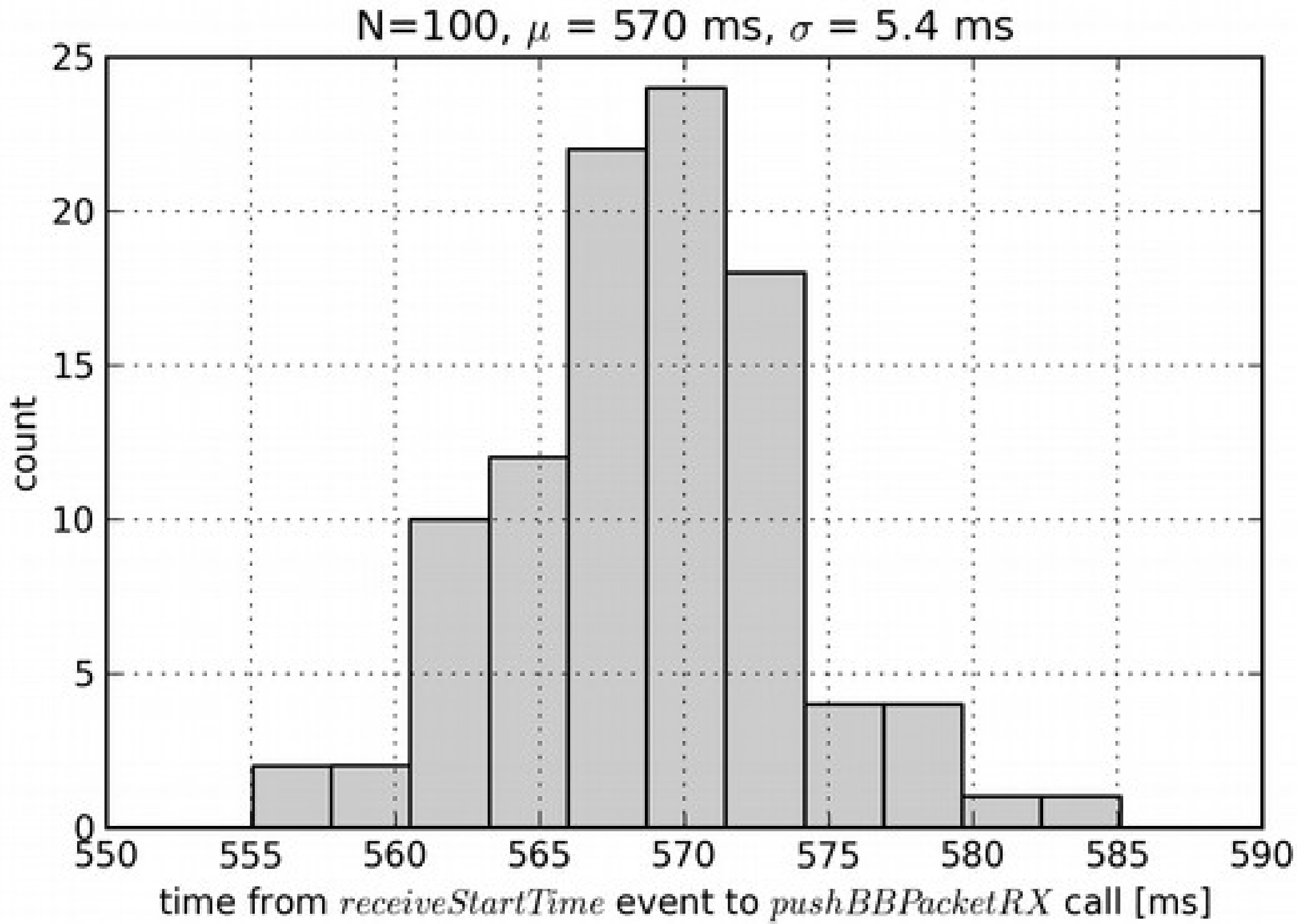
- From *receiveStartTime* event to *pushBBSamples* call



Latency measurement with scope

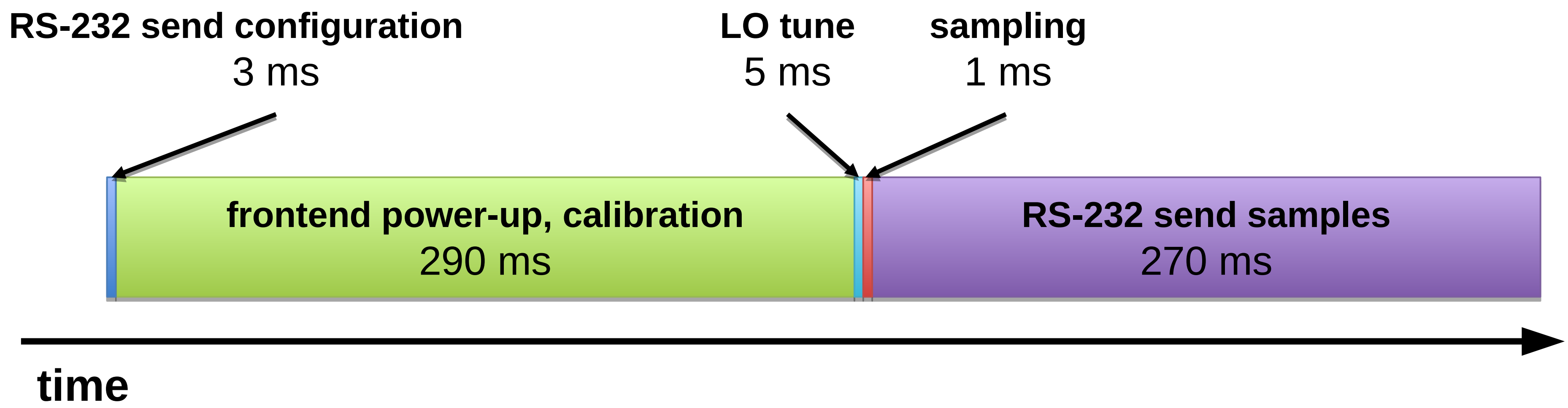


Histogram of measurements



- Measured using the host PC clock
  - inaccurate due to task switching, etc.

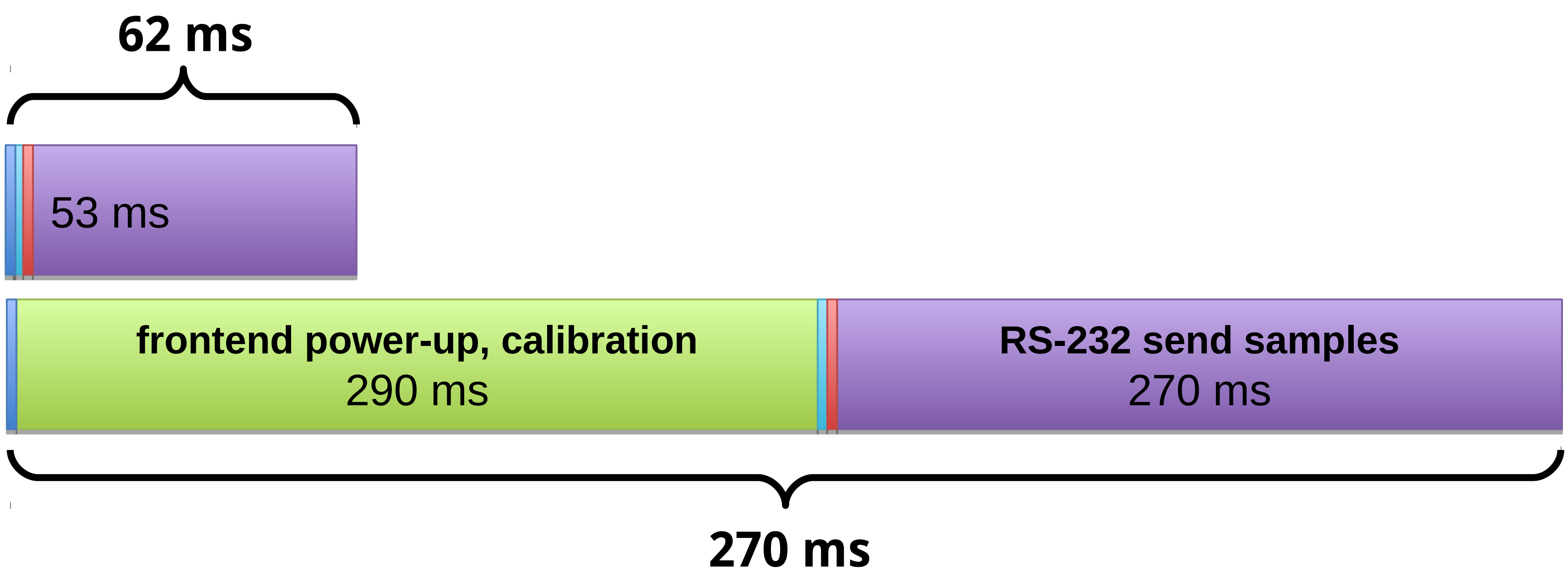
Analysis



- **Latency approximately 570 ms** (for 2048 samples)
  - Frontend slow to resume from power saving mode.
  - Sending ASCII formatted samples over 576 kbps RS-232 connection – latency depends on packet size
- Blind time of sensor in this mode > 99.5%

Improving latency

- Keep front-end constantly powered-up
  - No calibration for each RX cycle, but increased power usage.
- Send samples in binary format instead of ASCII
  - Higher bitrate due to lower CPU usage, no ASCII overhead,
  - but complicates sensor interface, backwards incompatible.

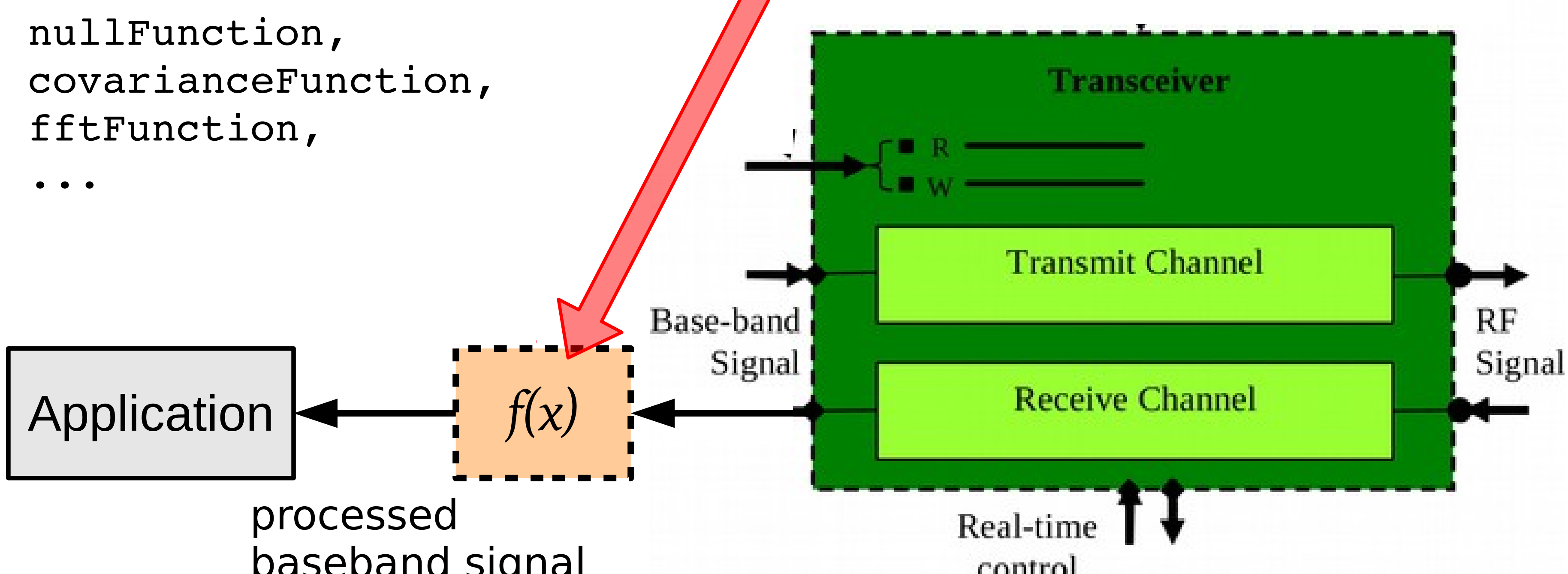


Exploiting on-board processing

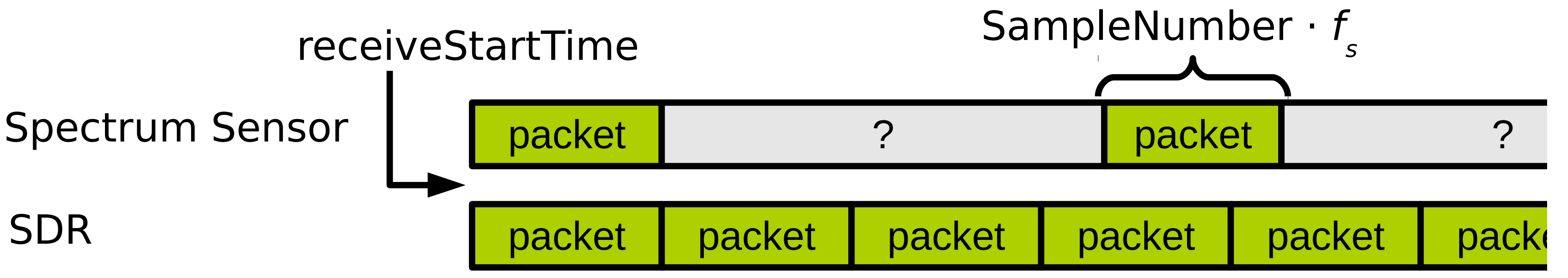
```
...
Transceiver::ULong packet_size = 2048;
Transceiver::UShort tuning_preset = 2;
Transceiver::Function func = Transceiver::covarianceFunction;

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

```
nullFunction,
covarianceFunction,
fftFunction,
...
```



Timing information



- Add a timestamp/stream offset to *BBPacket* class
  - defines timing in case of sample loss.

```
class BBPacket
{
    public:
        ULong SampleNumber;
        BBSample* packet;
        ULong time;
        BBPacket(ULong inSamplesNumber, BBSample* inPacket);
};
```

Conclusions

- Transceiver Facility has been adopted for spectrum sensors in CREW project testbeds.
- Latency with SNE-ESHTER hardware is ~570 ms.
- Support for spectrum sensors could be improved with minor changes of Transceiver Facility.
  - Support signal processing in hardware,
  - add timing information to packets of signal samples.
- Time will tell whether this abstraction is useful for use in testbeds.

Questions?

Tomaž Šolc

tomaz.solc@ijs.si

http://www.crew-project.eu