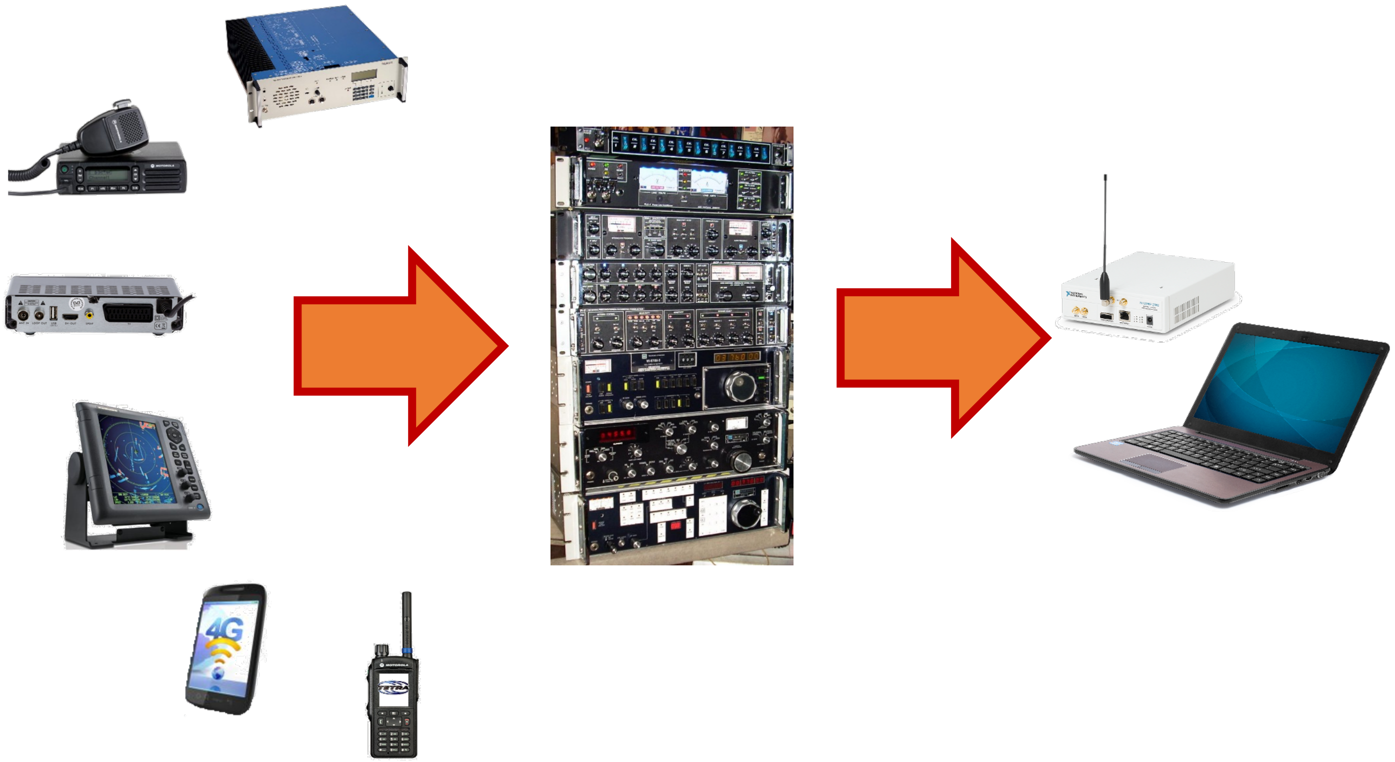
Software defined radio

SDR concept

- Radio components such as modulators, demodulators and tuners are traditionally implemented in analog hardware components.
- The advent of modern computing and analog to digital converters (ADCs) allows most of these HW based components to be implemented in SW instead.



SDR paradigm

- In a SDR receiver the incoming signal is converted to a digital format and then the signal is processed digitally.
- Most of the HW in a SDR is programmable so that it can be completely configured by software.
  - SDR can be easily reconfigured: it is sufficient to update the SW to keep up with new modulation formats, new algorithms and new applications.
- Common hardware platform can be used across a variety of different products and applications, thereby reducing costs, whilst maintaining or improving the performance.

SDR – Historical milestones

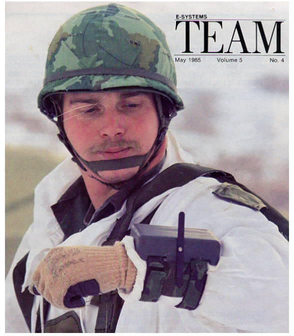
1984 - E-Systems Inc. (Raytheon) coined the term 'software radio'

1992 - J.Mitola publishes the paper: 'Software Radio: Survey, Critical Analysis and Future Directions',

2011 - A finnish hacker discovered that the baseband RTL2832U chip could be forced to operate in test mode to continuously output 8-bit unsigned samples of baseband I/Q data.

1984 1991 1992 2005 2011

SPEAKeasy

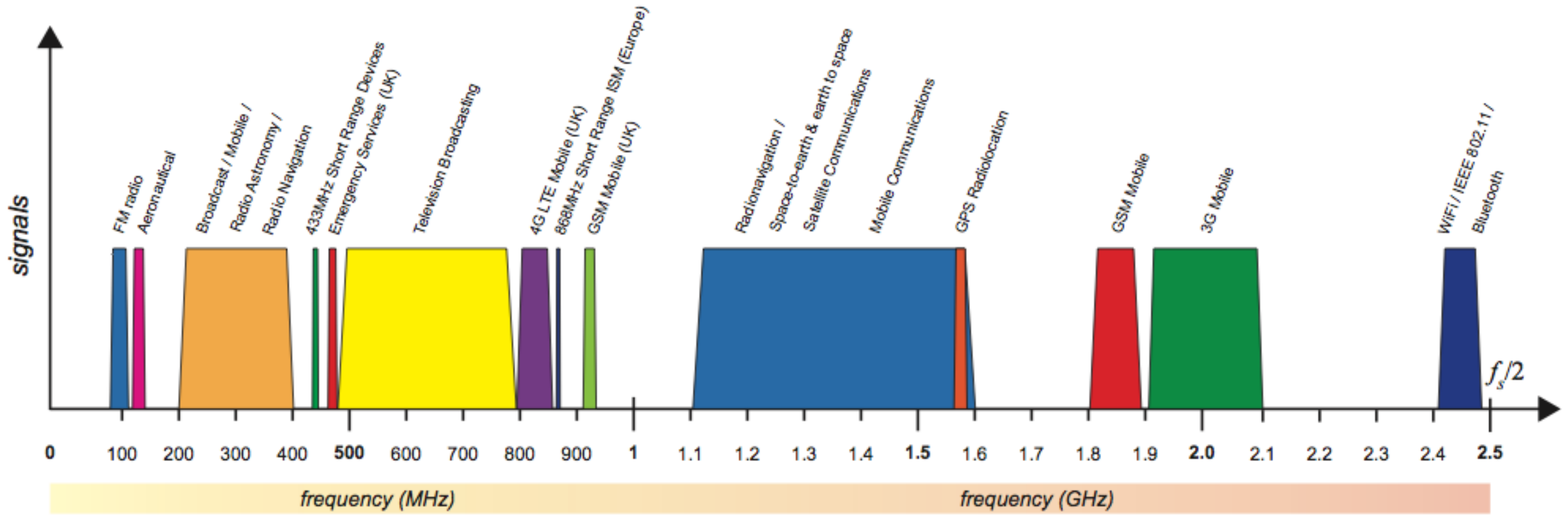


1991 - SPEAKeasy is the first military program that specifically required a radio to have its physical layer components. The objective was a single radio that could support ten different military radio protocols

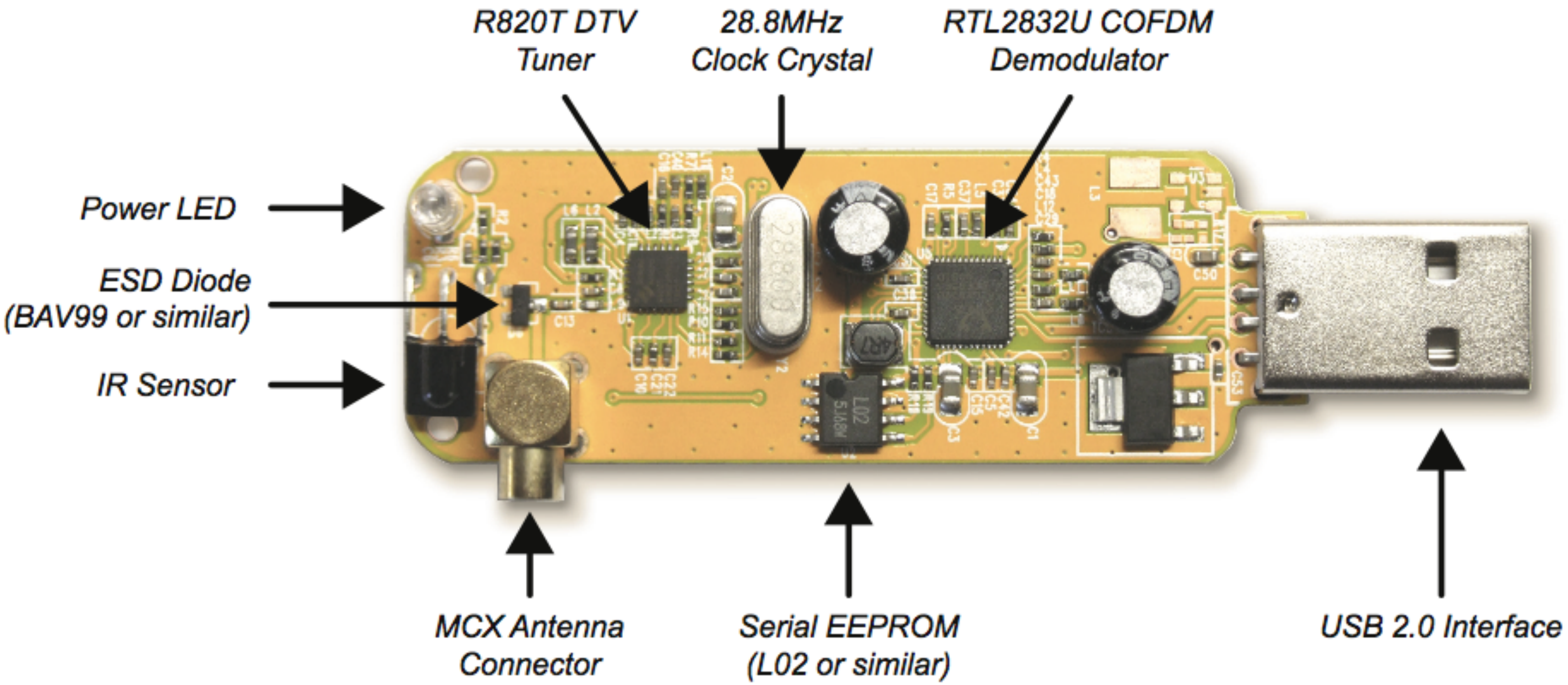
Ettus Research

2005 - Ettus research produces the first commercial SDR, USRP 1, initially based on the open source software GNU radio.

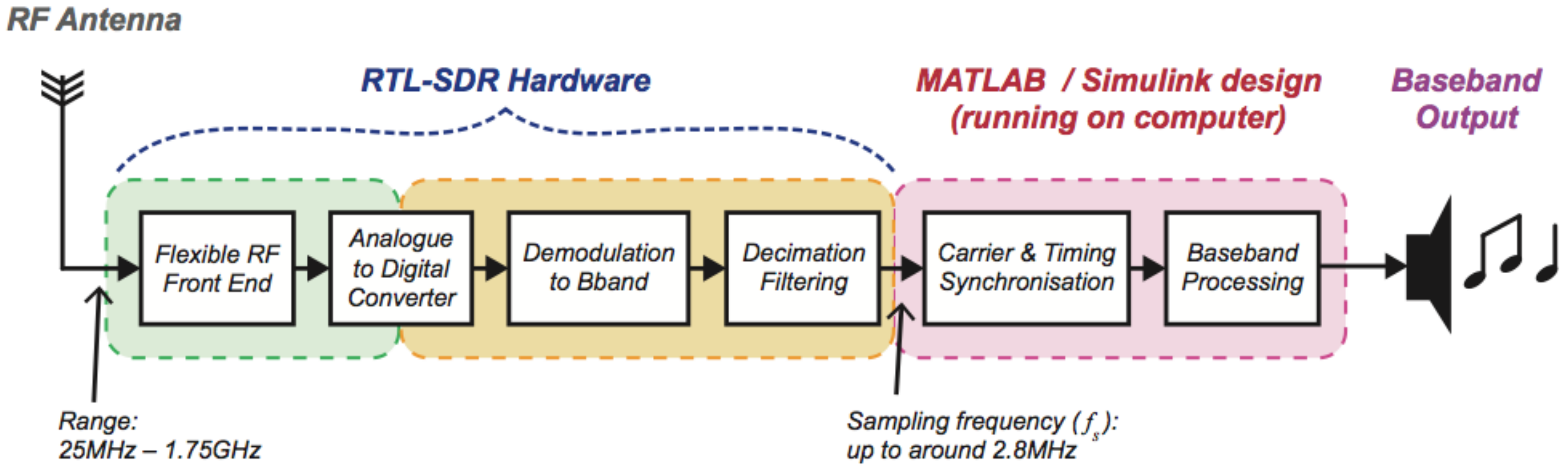
Telecom services in the 0.1-2.5 GHz band



RTL-SDR architecture



SDR block diagram for an FM receiver

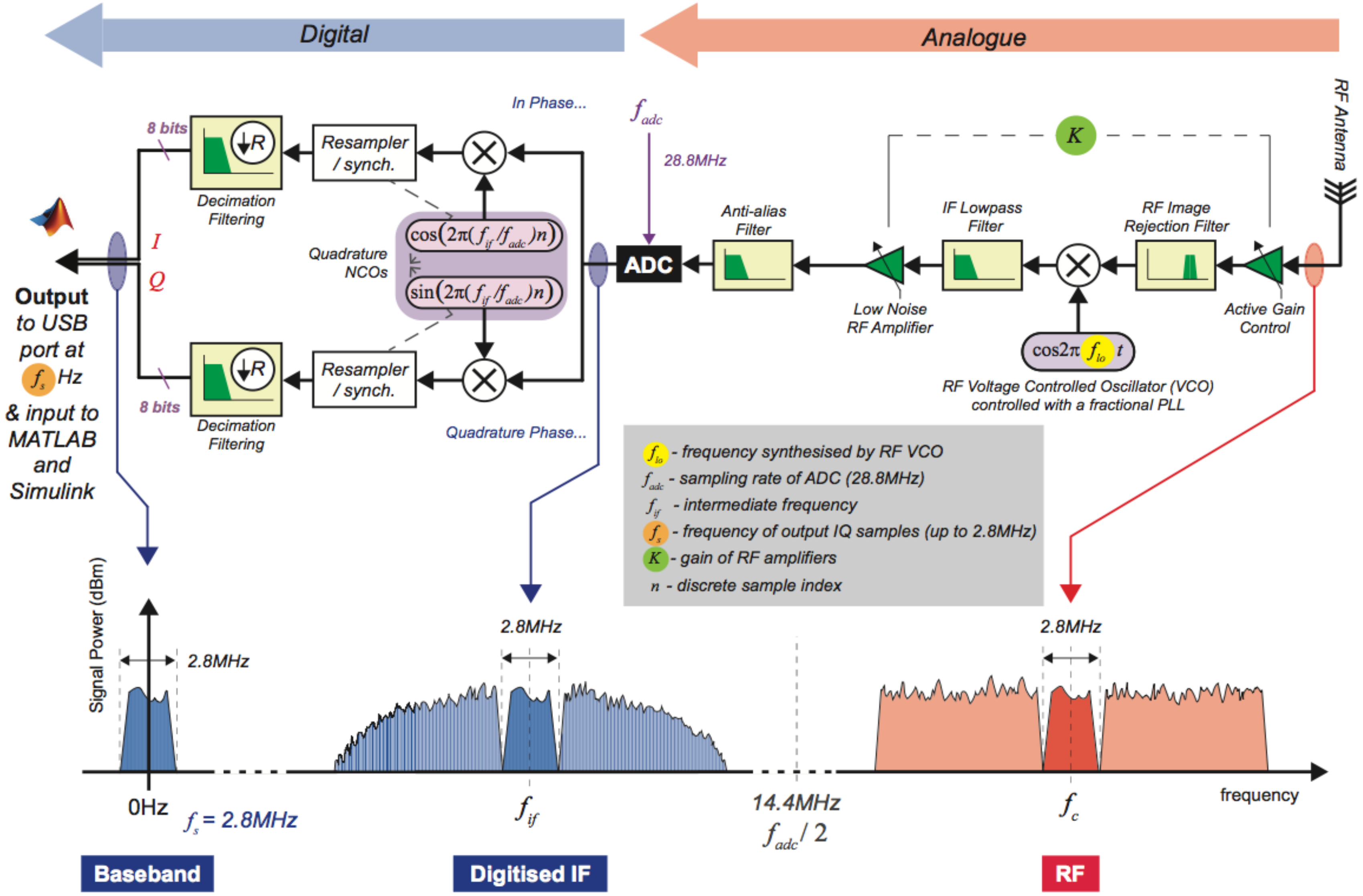


RTL-SDR receiving steps

Receiver chain (*superheterodyne* receiver)

1. Incoming signals are mixed down using a super-heterodine receiver to an *intermediate frequency* of **3.57** MHz.
2. The intermediate frequency analog signal is sampled by a 2 channel (baseband I/Q components) **28.8** MS/s 8-bit analog-to-digital converter (ADC).
3. The digitized I/Q data follows parallel paths through a digital downconversion process that mixes, filters, and decimates the input signal to a user-specified rate. The maximum rate is approximately **2.8** MS/s.
4. The downconverted samples are passed to the host computer over a standard USB connection.

From RF to baseband....



Programming a RTL-SDR with MATLAB

Classes and objects in MATLAB

- Object Oriented Programming (OOP) allows to create classes:
  - Description of the data type structure (fields or properties)
  - The set of operations (methods or functions) defined for this data type
- In MATLAB an object is a variable belonging to a specific class: before defining an object of a class it is necessary to know well the characteristics of the class.
- The operations that can be performed on a class are restricted to the methods defined for that class.
- For almost every class defined by MATLAB there is the `step` command, whose operation changes from class to class and depends on the class itself.

Example: creating a spectrum analyzer

- Command `obj = dsp.SpectrumAnalyzer` creates an object of the class `dsp.SpectrumAnalyzer`.
- To define the value of some fields, the object is treated as if it were a structure.
- In this example we give the value 'Spectrum Analyzer’ to the field ‘Name’ of our object

  ```
  obj = dsp.SpectrumAnalyzer('Name', 'Spectrum
  Analyzer’)
  ```

  or

  ```
  obj.Name = 'Spectrum Analyzer'
  ```

Example: creating a spectrum analyzer

- Create an object of the class `dsp.SpectrumAnalyzer`

```
scope =  dsp.SpectrumAnalyzer( ...
    'Name', 'Spectrum Analyzer’, ...
    'Title', 'Spectrum', ...
    'SpectrumType', 'Power', ...
    'FrequencySpan', 'Span and center frequency', ...
    'CenterFrequency',0, ...
    'Span',600, ...
    'ShowLegend', true, ...
    'SampleRate', Fs);
```

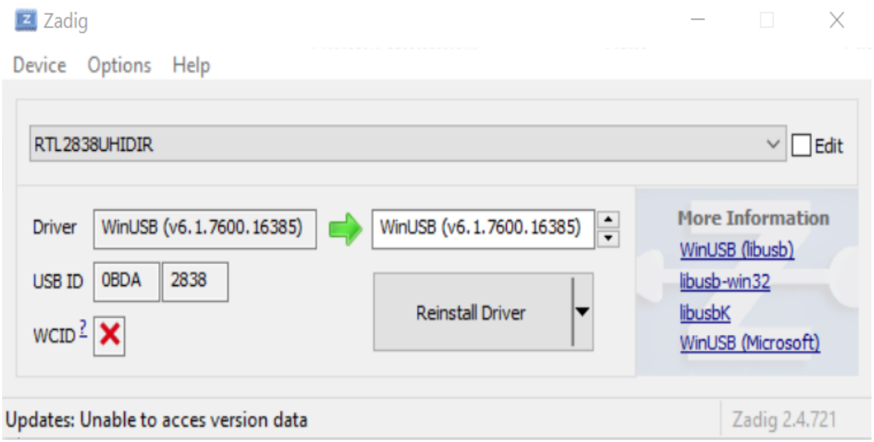
- The command `step(obj,X)` displays the frequency spectrum of double, single or fixed-point precision input `X`, in the Spectrum Analyzer figure. The columns of `X` are treated as independent channels.

Before starting…….

- Install Support Package for RTL-SDR Radio
    - On the MATLAB Home tab click **Add-Ons > Get Hardware Support Packages**.
    - In Add-On Explorer, browse or search for the **Communications Toolbox™ Support Package for RTL-SDR Radio**.
    - Select the support package and then click Install.
    - During support package installation, you will be prompted to install the drivers needed for the RTL-SDR Radio software.

How to install drivers for RTL-SDR in MATLAB

- After having installed the **Communications Toolbox Support Package for RTL-SDR Radio**
- Open **File Explorer** on your PC.
- Follow this path: C:\ProgramData\MATLAB\SupportPackages\<version>\3P.instrset\zadig.instrset\zadig\zadig-XX.exe
  - <version> is the MATLAB version
  - XX is the Zadig version number
- Connect the RTL-SDR to the PC
- Open **Zadig** application, select RTL from the list
- Install **Driver**



Hardware Setup

1. Plug the RTL-SDR into your computer
2. Start MATLAB, at the MATLAB command prompt, call the `sdrsetup` function.
3. To get information for all radios connected to your computer, call the `sdrinfo` function.

```
        hwinfo = sdrinfo
hwinfo =
        RadioName: 'Generic RTL2832U OEM'
        RadioAddress: '0'
        RadioIsOpen: 0
        TunerName: 'R820T'
        Manufacturer: 'Realtek'
        Product: 'RTL2838UHIDIR'
        GainValues: [29×1 double]
        RTLCrystalFrequency: 28800000
        TunerCrystalFrequency: 28800000
        SamplingMode: 'Quadrature'
        OffsetTuning: 'Disabled'
```

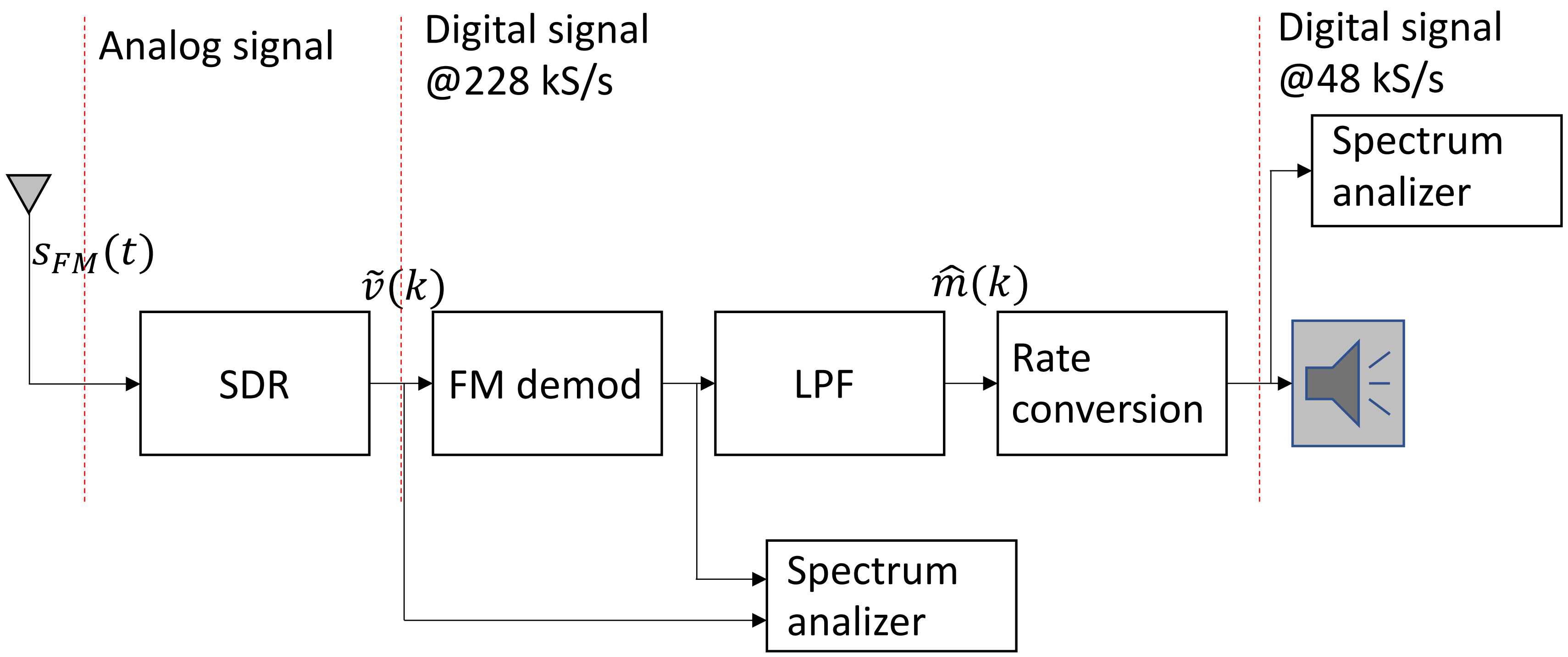
Load RTL-SDR driver

- Construct an RTL-SDR receiver System object:

```
obj_rtlsdr = comm.SDRRTLReceiver
obj_rtlsdr =
        comm.SDRRTLReceiver with properties:
        RadioAddress: '0'
        CenterFrequency: 102500000
        EnableTunerAGC: true
        SampleRate: 250000
        OutputDataType: 'int16'
        SamplesPerFrame: 1024
        FrequencyCorrection: 0
        EnableBurstMode: false
```

CenterFrequency: 102500000	Carrier frequency of the received signal
SampleRate: 250000	Bandwidth of the received signal
SamplesPerFrame: 1024	Number of samples passed to MATLAB with each call to `step` function

FM receiver – practical implementation



FM receiver

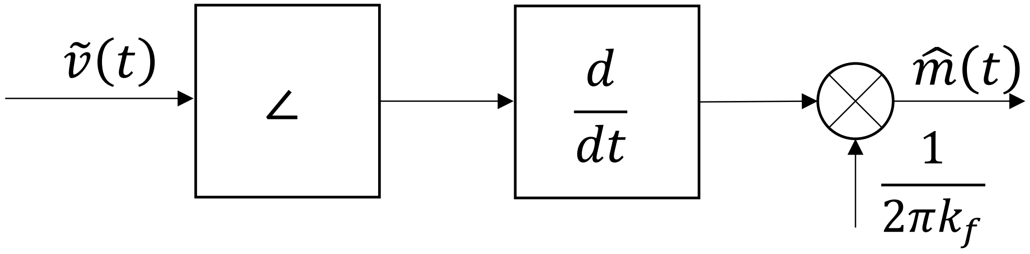
- Neglecting the effect of noise and channel, the complex envelope of the received signal is

$$\tilde{v}(t) = A_c e^{j2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau}$$

- The modulating signal can be recovered by differentiating the phase of $\tilde{v}(t)$

$$\hat{m}(t) = \frac{1}{2\pi k_f}\frac{d}{dt}\angle\tilde{v}(t)$$

Conceptual FM baseband receiver



FM receiver – practical implementation

- At the SDR output there is the signal complex envelope sampled at frequency $f_s = \frac{1}{T_s}$

$$\tilde{v}(k) = \tilde{v}(t)|_{t=kT_s} = A_c e^{j2\pi k_f \int_{-\infty}^{kT_s} m(\tau)d\tau} \approx A_c e^{j2\pi k_f \sum_{\ell=-\infty}^{k} m(\ell)T_s}$$

- The product of two consecutive baseband samples yields

$$\begin{aligned} \tilde{v}(k)\tilde{v}^*(k-1) &\approx A_c e^{j2\pi k_f \sum_{\ell=-\infty}^{k} m(\ell)T_s} A_c e^{-j2\pi k_f \sum_{\ell=-\infty}^{k-1} m(\ell)T_s} \\ &= A_c^2 e^{j2\pi k_f m(k)T_s} \end{aligned}$$

- An estimate of $m(k)$, the k-th sample of $m(t)$, is

$$\hat{m}(k) = \frac{1}{T_s}\frac{1}{2\pi\Delta f}\angle\tilde{v}(k)\tilde{v}^*(k-1)$$

FM mono

- The first (1945-60) FM transmissions were *mono*, i.e.

$$m(t) = L(t) + R(t)$$

and $m(t)$ normalized to 1 and $k_f = 75$ kHz/V so that

$$\Delta f = k_f \max|m(t)| = 75 \text{ kHz.}$$