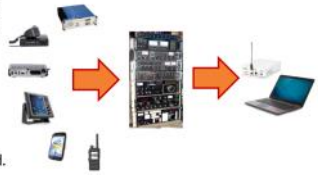
Software defined radio

3rd lesson
Monday, 5 Oct

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.

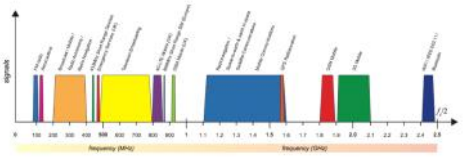
Programmable HW for SDR

1. Reconfigurable Radio (FPGA-Based)

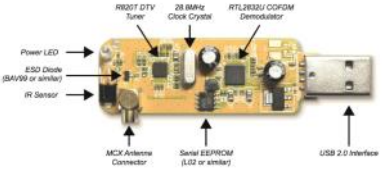
2. Programmable Radio (DSP-Based or FPGA/DSP Hybrid)

3. Fully Software Radio (GPP-Based)

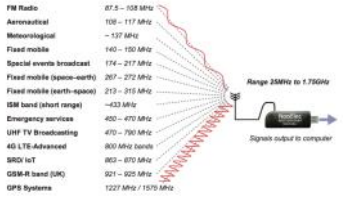
Telecom services in the 0.1-2.5 GHz band



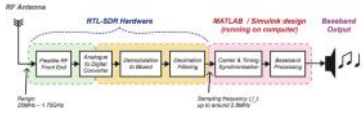
RTL-SDR architecture



RTL-SDR services



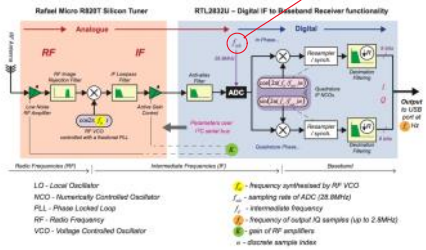
SDR block diagram for an FM receiver



Signal processing flow diagram

$f_{if} = 3.57$ MHz intermediate frequency

$f_{adc} = f_s$ sampling frequency



The superheterodine receiver performs the frequency conversion in two steps: ① from $f_c \rightarrow f_{if}$ and ② from $f_{if} \rightarrow 0$

Regarding the step ①, the local oscillator frequency f_{LO} is set to $f_{LO} = f_c - f_{if}$ so that after the first mixer the signal's spectrum will be centered around f_{if}

Let's assume that we want to receive the passband signal

$$s(t) = s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin 2\pi f_c t$$

Let's assume that we want to receive the passband signal

$$s(t) = s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin 2\pi f_c t$$

After multiplication by $2\cos 2\pi f_{LO} t = 2\cos(2\pi(f_c - f_{if})t)$

$$2\,s(t)\cos(2\pi f_{LO} t) = 2s_I(t)\cos(2\pi f_c t)\cos(2\pi(f_c - f_{if})t) - 2s_Q(t)\sin(2\pi f_c t)\cos(2\pi(f_c - f_{if})t)$$

Taking advantage of these relationships:

$$\cos(2\pi f_c t)\cos(2\pi(f_c - f_{if})t) = \frac{1}{2}\cos(2\pi f_{if} t) + \frac{1}{2}\cos(2\pi(2f_c - f_{if})t)$$

$$\sin(2\pi f_c t)\cos(2\pi(f_c - f_{if})t) = \frac{1}{2}\sin(2\pi f_{if} t) + \frac{1}{2}\sin(2\pi(2f_c - f_{if})t)$$

each filtering $s(t)\cos(2\pi f_{LO} t)$ with a low pass filter will yield

$s(t)$ → (×) $2\cos(2\pi f_{LO} t)$ → LPF → $s_{if}(t) = s_I(t)\cos(2\pi f_{if} t) - s_Q(t)\sin(2\pi f_{if} t)$

* This filter band includes f_{if}

RTL-SDR receiving steps

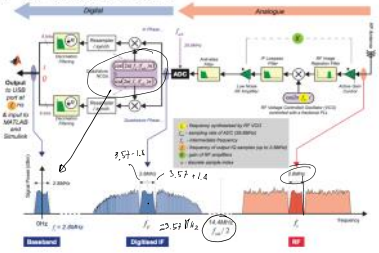
$B \leq \frac{1}{2T_s}$ $B \leq 14.4$ MHz

$f_s = \frac{1}{T_s} = 28.8$ MHz

Receiver chain (*superheterodyne* receiver)

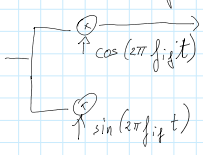
1. Incoming signals are mixed down using a super-heterodine receiver to an *intermediate frequency* of **3.57** MHz.
2. The intermediate frequency analog signal is sampled by a 2 channel (baseband I/Q components) **28.8** MS/s 8-bit analog-to-digital converter (ADC).
3. The digitized I/Q data follows 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....

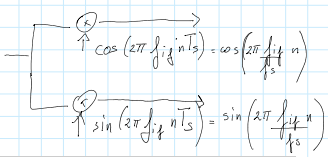


In baseband we down-convert the received signal from f_{if} to 0. This conversion is done in a NUMERIC quadrature receiver

Analog

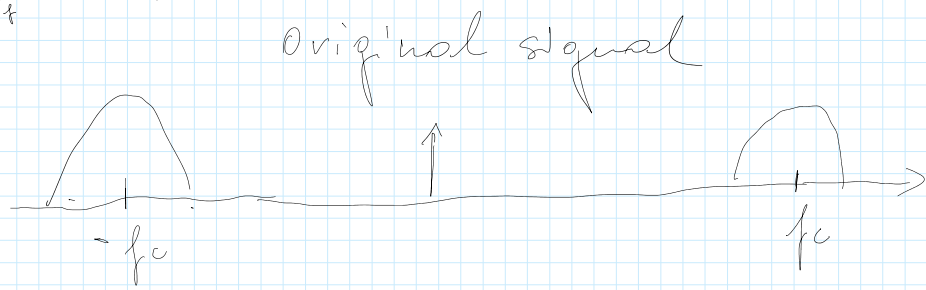


Numeric

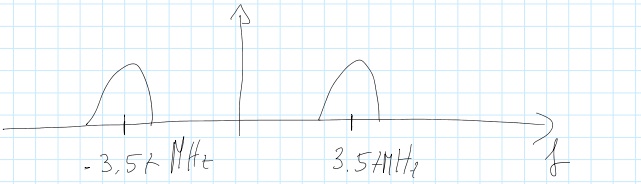


Programming a RTL-SDR with MATLAB

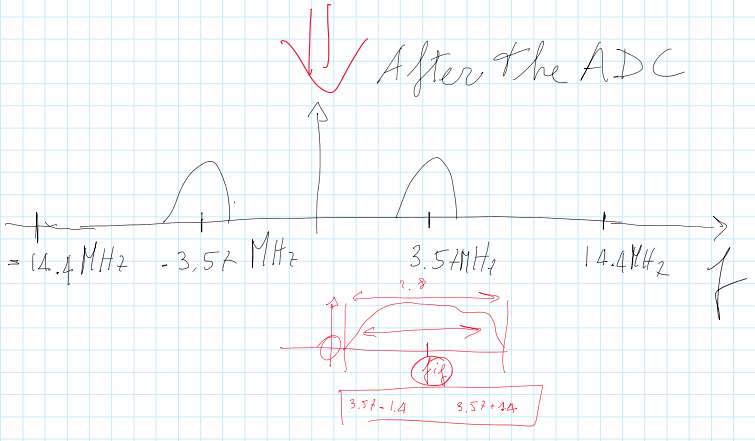
Original signal



After the RF part



After the ADC



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.

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'
        RadioInUse: 0
        TunerName: 'R820T'
        Manufacturer: 'Realtek'
        Product: 'RTL2838UHIDIR'
        GainValues: [29×1 double]
        RTLCrystalFrequency: 28800000
        TunerCrystalFrequency: 28810000
        SamplingMode: 'Quadrature'
        OffsetTuning: 'Disabled'
```

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 (1)

- 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 (2)

- 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.

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        Carrier frequency of the received signal
        EnableTunerAGC: true
        SampleRate: 250000                Bandwidth of the received signal
        OutputDataType: 'int16'
        SamplesPerFrame: 1024             Number of samples passed to MATLAB with each call to step function
        FrequencyCorrection: 0
        EnableBurstMode: false
```

Using RTL-SDR with MATLAB

- **Main parameters**
  - Carrier frequency
    `CenterFrequency:` = [24-1766]MHz
  - Signal bandwidth
    `SampleRate:` = [225,300]kHz U [900,2560]kHz
  - Buffer data length
    `SamplesPerFrame:` = 256*m:m