



Connecting MATLAB® to USRP™ for Wireless System Design

Jeremy Twaits, NI



Vijayendra Kumar, MathWorks



MATLAB EXPO



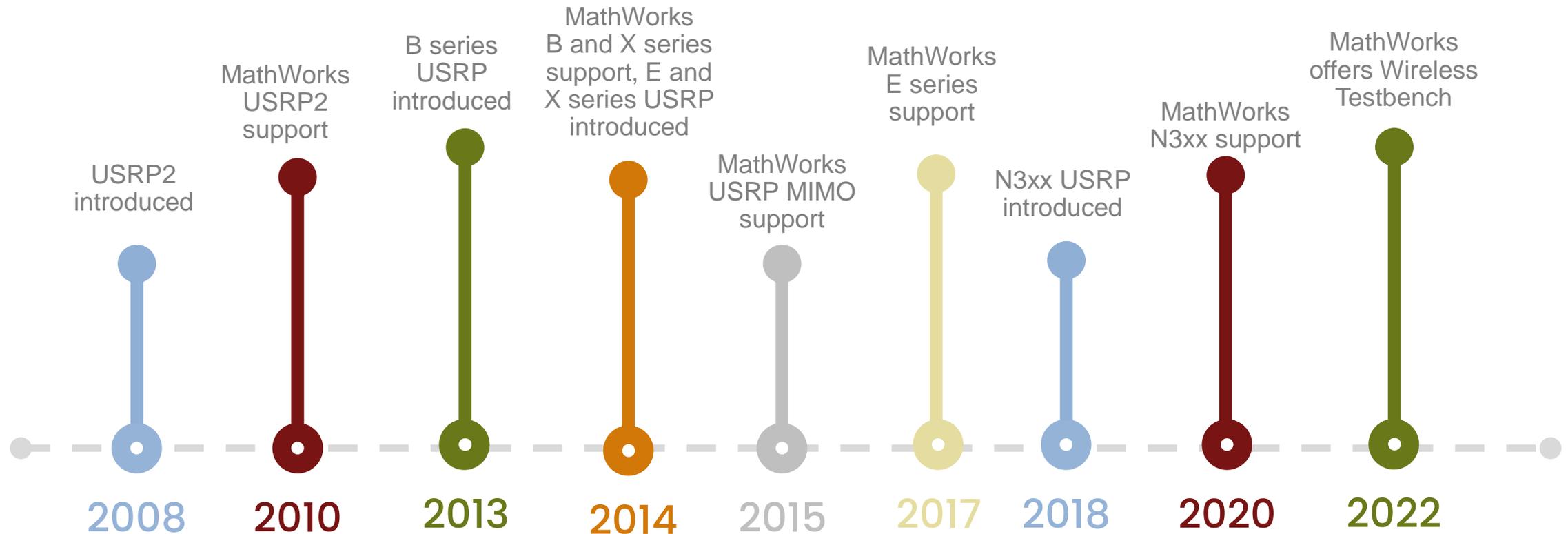
MathWorks 

@MathWorks

Share the EXPO experience
#MATLABEXPO



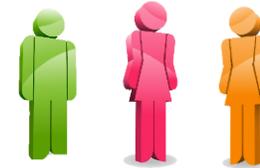
A History of MathWorks/NI SDR Collaboration



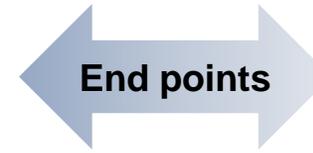
Signal Detection: A Wireless Design Requirement



Aero/Defense Company

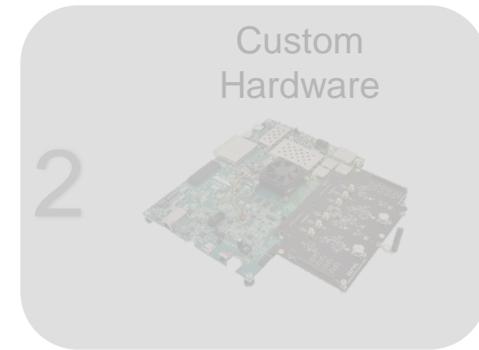


Wireless System Architect
Algorithm Developers



✓
Commercial off-the-shelf
Software Defined Radio
(COTS SDR Such as USRPs)

1



2

Custom
Hardware

Primary Goal: Signal Detection

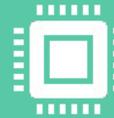
- Explore and test the feasibility of wireless system design
- Freeze the specs and prototype the **signal detector**

Agenda



Workflows

Supported Workflows



Hardware

Supported USRPs



Workflows

Supported Workflows



Hardware

Supported USRPs

Wireless Testbench: Supported Hardware



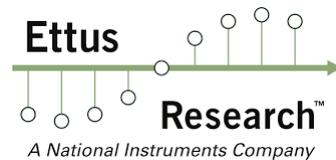
N310



N320



N321



USRP



USRP N310 Software Defined Radio Device

Features

- Channels: up to 4x4 per device
- 100 MHz bandwidth/channel
- 10 MHz – 6 GHz
- Embedded ARM processor for stand-alone operation
- Large user-programmable FPGA, Zynq-7100
- 2 x 10 GbE streaming support
- Remote management support
- Rack mountable, half wide, 1U
- Support for Open Source – GNU Radio, UHD, RFNoC
- L = 14.06”; W = 8.31”; H = 1.72”
- Weight 6.9 lb = 3.13 kg



Applications

- Communications System Design/Prototyping
- 802.11, LTE Research
- 5G NR FR1
- SATCOM
- UE emulation
- Massive MIMO
- SIGINT/EW
- Spectrum Monitoring
- Record & Playback

USRP N320/N321 Feature Highlights

Common:

- 3 MHz – 6 GHz range
- 200 MHz BW per channel
- 2x2 MIMO
- 200/245.76/250 MHz sample rates
- Preselection filters
- Dual SFP+ ports (1 GbE, 10 GbE, Aurora)
- QSFP+, RJ45
- GPSDO
- Ethernet-based sync (White Rabbit)
- Stand-alone operation

N320:

- Zynq XC7Z100-2FFG9001
- External LO input ports

N321:

- LO Distribution for up to 128x128 MIMO

Applications:

- Phase Coherent Wireless Testbeds
- Wideband Spectrum Monitoring
- Radar Prototyping
- Direction Finding



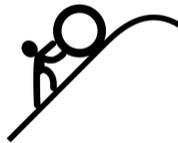
SDR Use Cases, Applications and Current Challenges

Traditional Use Cases and Applications



- Transmit/Capture of standard-based and custom wireless signals
- Stream wireless signals to the host with live data processing
- Run prebuilt wireless applications on the FPGA/SOC of SDR for early design exploration and testing
- Run any custom wireless application on the FPGA/SOC on SDR hardware

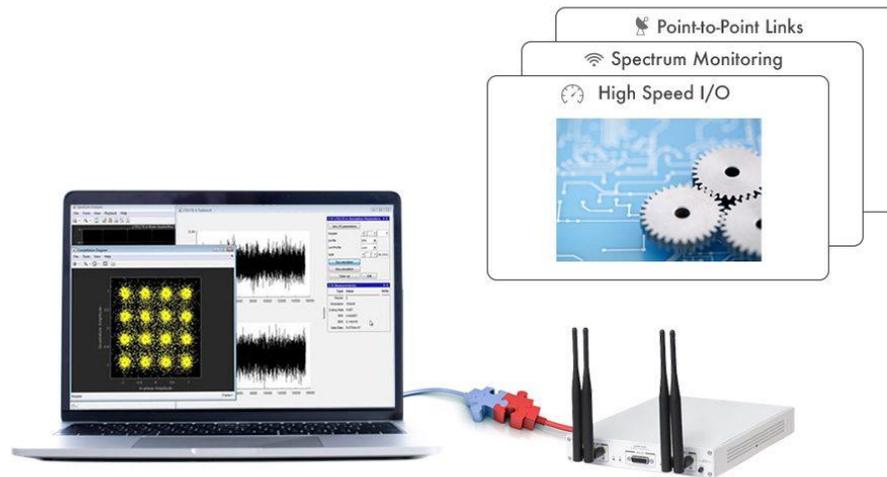
Current Challenges



- Bandwidth hungry applications require high-speed transmit and capture solutions
- Real time and near real time processing for wireless applications require
 - Optimal use of hardware resources
 - Intelligent signal detection and data capture
 - Efficient host processing
- Large amount of real time data is needed for training of AI models

New Solution and Supported Workflows

Wireless Testbench



Explore and test wireless designs using high speed and intelligent data transmit and capture

Wireless Testbench Workflows



High-Speed data transmit and capture



Intelligent signal detection



Workflows

Supported Workflows



Hardware

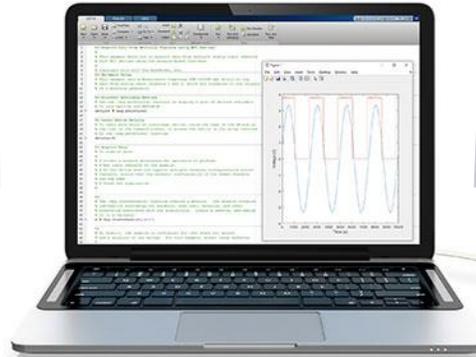
Supported USRPs

Workflow: High-Speed Transmit and Capture

- *Channel effects*
- *Frontend validation*
- *Loopback*
- *End-to-end transceiver design*



Baseband Transceiver



Baseband Transmitter

- *Custom signal*
- *Noise*
- *Interfering waveform*
- *Receiver design*



Baseband Receiver

- *5G, WLAN, Satellite*
- *Custom signal*
- *Cognitive radio*
- *Spectrum sensing*

Key Features

- Full rate transmit/capture (up to 250 MSPS) Tx/Rx
- Baseband receiver, transmitter, and transceiver workflows

High-Speed Transmit and Capture: Baseband Receiver Demo

The screenshot displays the MATLAB R2022b Live Editor interface. The main window shows a script titled "Capture from Frequency Band with Multiple Antennas". The script includes the following sections and code:

Capture from Frequency Band with Multiple Antennas

This example shows how to use a software-defined radio (SDR) to capture data from a specified frequency band using multiple antennas. The example then plots the frequency spectrum of the captured data.

Set Up Radio

Call the `radioConfigurations` function. The function returns all available radio setup configurations that you saved using the Radio Setup wizard. For more information, see [Connect and Set Up NI\(R\) USRP\(R\) Radios](#).

```
1 radios = radioConfigurations;
```

Specify the name of a saved radio setup configuration to use with this example.

```
2 radioName = radios(1).Name;
```

Specify Frequency Band

Specify the start and the end of the frequency band. By default, this example captures the 470–694 MHz frequency band, typically allocated to TV broadcasting channels 21–48.

```
3 frequencyBand.Start = 470000000;
4 frequencyBand.End = 694000000;
5 frequencyBand.Width = frequencyBand.End - frequencyBand.Start;
6 frequencyBand.MidPoint = frequencyBand.Start + frequencyBand.Width/2;
```

Configure Baseband Receiver

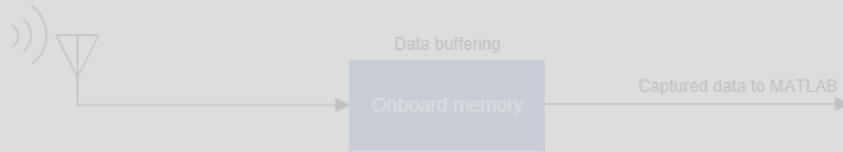
Create a baseband receiver object with the specified radio. To speed up the execution time of this example upon subsequent runs, reuse the workspace object from the first run of the example.

```
7 if ~exist("bbrx", "var")
8     bbrx = basebandReceiver(radioName);
9 end
```

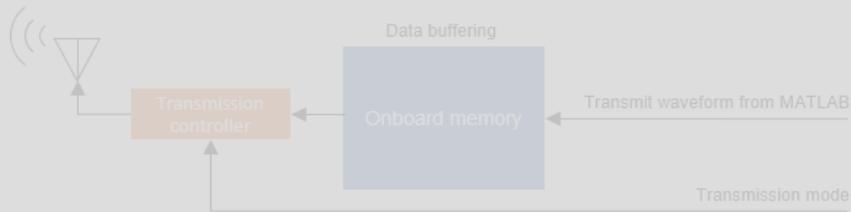
The Command Window at the bottom left shows the prompt `f1 >>`. The Workspace window at the bottom right shows the following variables:

Name	Value
antenna	2
bbrx	1x1 basebandReceiver
captureLength	1x1 duration
carrierGen	1x1 SineWave
data	12288000x2 complex do...
frequencyBand	1x1 struct
PlotPowerLimits	[-120, -60]

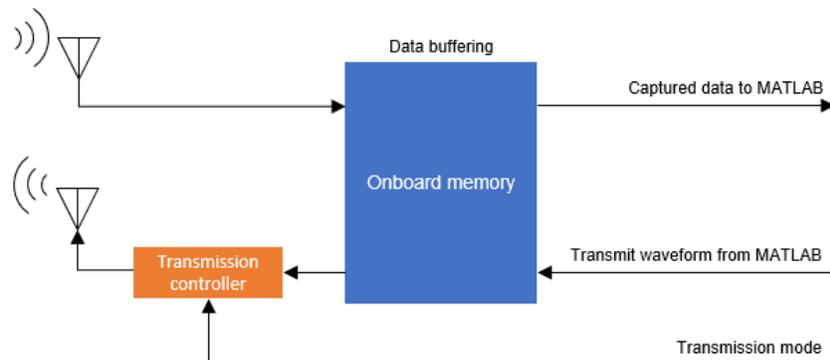
High-Speed Transmit and Capture: Receiver, Transmitter, Transceiver



```
bbrx = basebandReceiver("MyRadio")
bbrx.SampleRate = 122.88e6;
bbrx.CenterFrequency = 2.2e9
[data,~] = capture(bbrx,milliseconds(3));
```



```
bbtx = basebandTransmitter("MyRadio")
bbtx.SampleRate = 122.88e6;
bbtx.CenterFrequency = 2.2e9;
transmit(bbtx,txWaveform,"continuous");
pause(5);
stopTransmission(bbtx);
```



```
bbtrx = basebandTransceiver("MyRadio")
bbtrx.SampleRate = 122.88e6;
bbtrx.TransmitCenterFrequency = 2.2e9;
bbtrx.CaptureCenterFrequency = 2.2e9;
txWaveform = complex(randn(1000,1),randn(1000,1));
transmit(bbtrx,txWaveform,"continuous");
[data,~] = capture(bbtrx,milliseconds(3));
stopTransmission(bbtrx);
```

High-Speed Transmit and Capture: Baseband Transceiver Demo

The screenshot displays the MATLAB R2022b Live Editor interface. The main window shows a script titled "Loopback Transmit and Capture" with the following content:

Loopback Transmit and Capture

This example shows how to use a software-defined radio (SDR) to transmit and capture a custom wireless waveform over the air.

Set Up Radio

Call the `radioConfigurations` function. The function returns all available radio setup configurations that you saved using the Radio Setup wizard. For more information, see [Connect and Set Up NI\(R\) USRP\(R\) Radios](#).

```
1 radios = radioConfigurations;
```

Specify the name of a saved radio setup configuration to use with this example.

```
2 radioName = radios(1).Name;
```

Specify Wireless Waveform

Use the attached `TestTone.mat` file to specify the transmit waveform. The `waveStruct` structure contains a complex sine tone that is generated by using the [Wireless Waveform Generator](#) app.

```
3 load("TestTone.mat")
```

Configure Baseband Transceiver

Create a baseband transceiver object with the specified radio. To speed up the execution time of this example upon subsequent runs, reuse the workspace object from the first run of the example.

```
4 if ~exist("bbtrx", "var")
5     bbtrx = basebandTransceiver(radioName);
6 end
```

Configure the baseband transceiver object using the parameters of the wireless waveform.

- Set the `SampleRate` property to the sample rate of the generated waveform.
- Set the `CenterFrequency` property to a value in the frequency spectrum indicating the position of the waveform transmission.

```
7 bbtrx.SampleRate = waveStruct.Fs;
8 bbtrx.TransmitCenterFrequency = 2.4e9;
9 bbtrx.CaptureCenterFrequency = bbtrx.TransmitCenterFrequency;
```

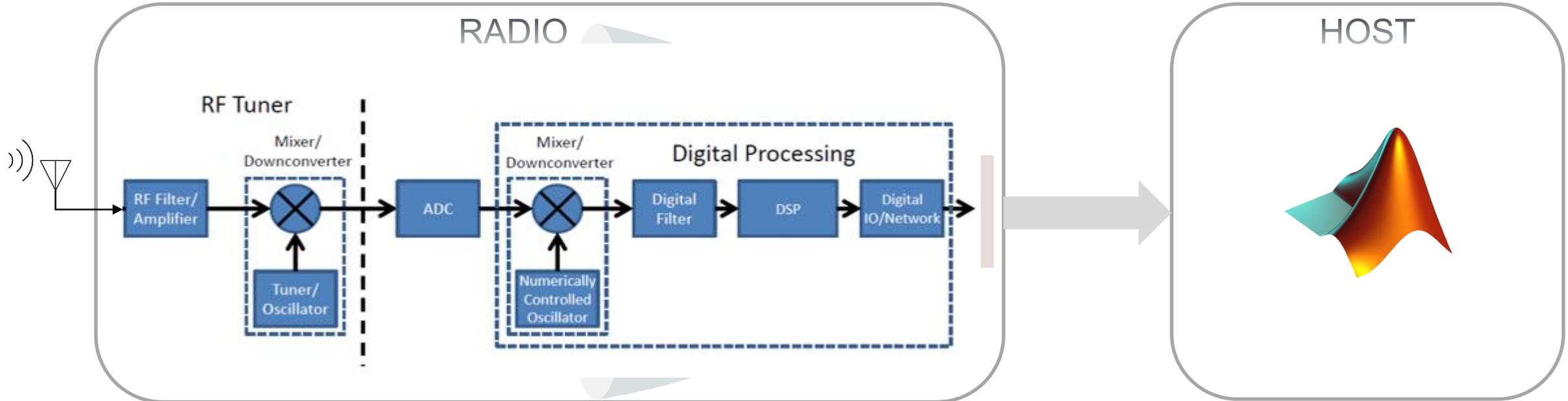
The Command Window shows the following commands:

```
>> clear
fs >>
```

The Workspace window shows the following variables:

Name	Value
bbtrx	1x1 basebandTransceiver
captureLength	1x1 duration
data	63.449000x2 complex int26
PlotPowerLimits	[-120 -60]
radioName	"MyN310"
radios	1x2 struct
spectrumScope	1x1 SpectrumAnalyzer

Workflow: Intelligent Signal Detection and Data Capture



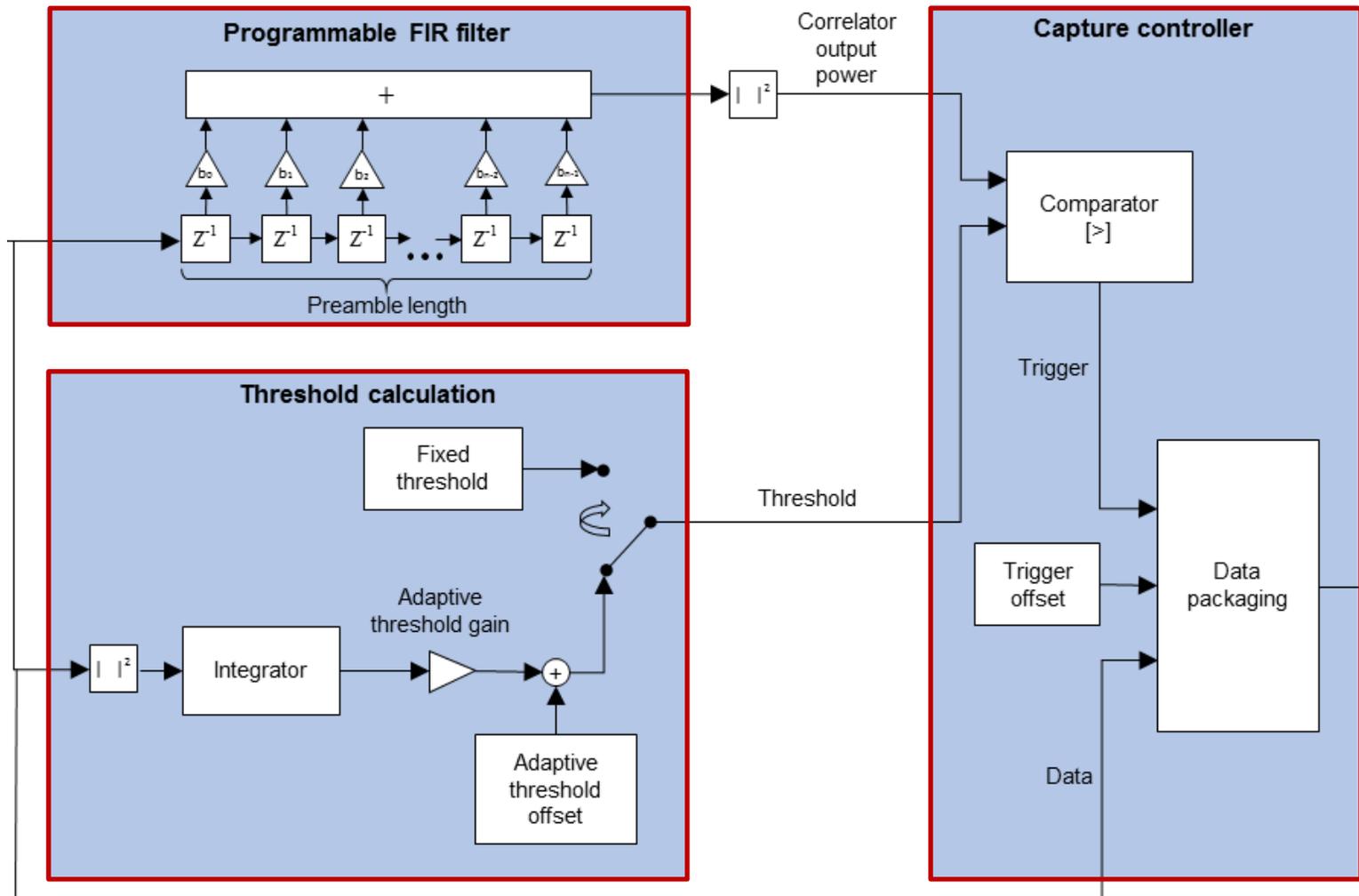
Features

- Intelligent Capture @ 250 MSPS
- Prebuilt application

Use cases/Applications

- Spectral conformance
- Signal detection
- Spectrum monitoring
- Signal classification
- Cognitive radio
- Signal intelligence
- Radar

Intelligent Signal Detection and Data Capture: Internal Architecture



Programmable FIR filter

- Correlates the input signal with a known preamble sequence

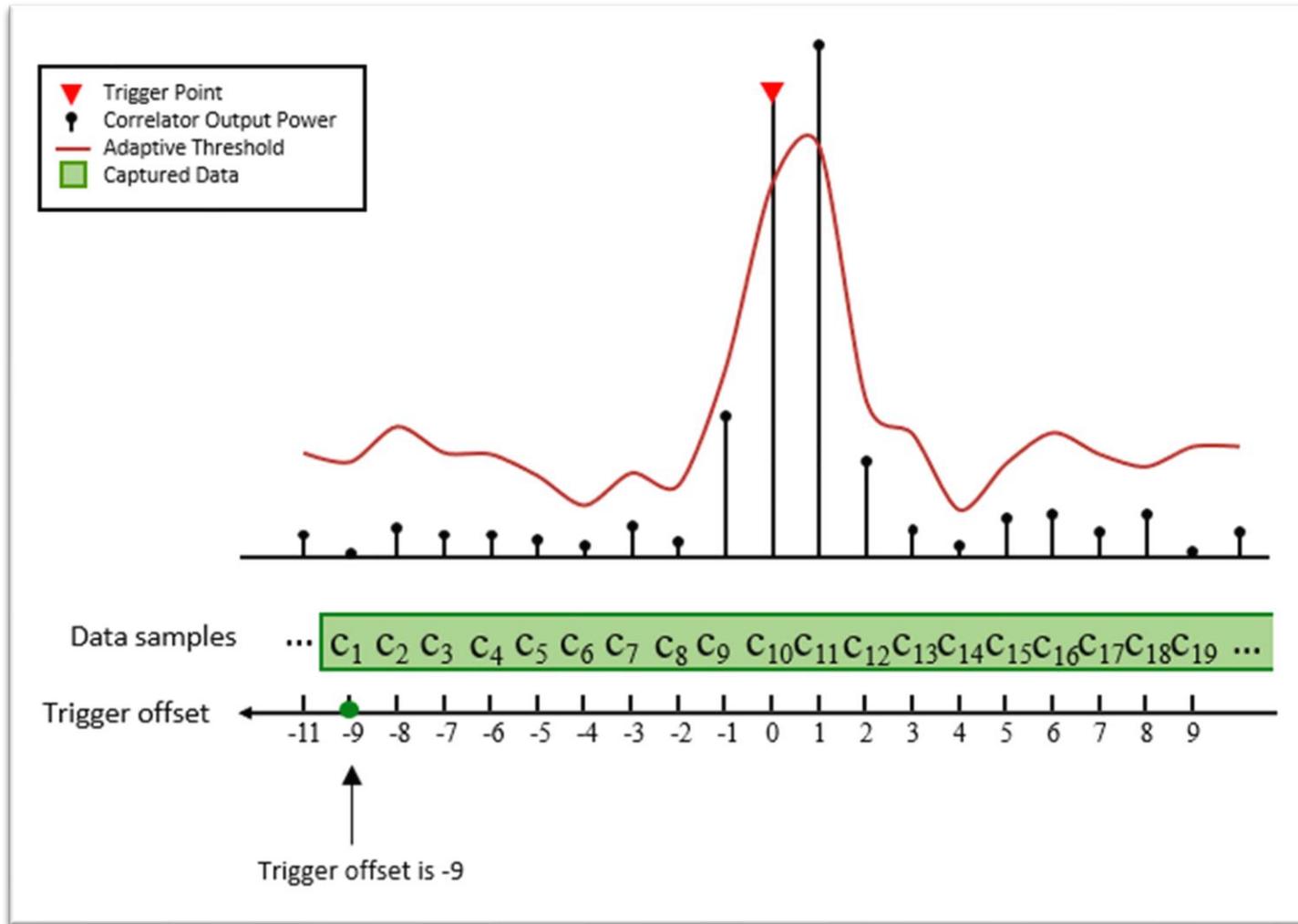
Threshold calculation

- Fixed and Adaptive Threshold

Capture Controller

- Calculates the trigger point
- Captures data

Intelligent Signal Detection and Data Capture: Thresholding, Triggering and Capturing



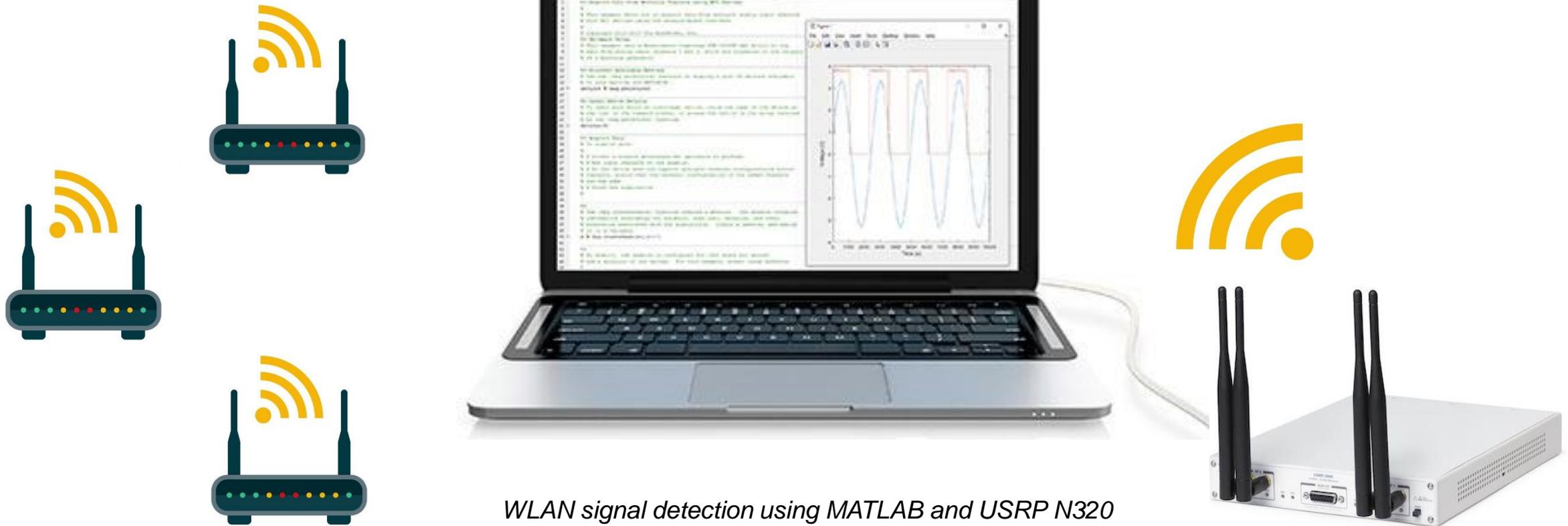
- Calibrate the thresholding and triggering



- Filter output power
- Scaled signal power signals

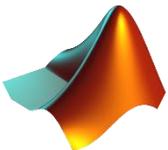
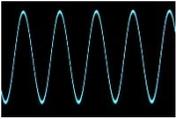
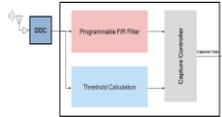
- Trigger delays are used to capture signals around the trigger point

Workflow: Intelligent Data Detection and Capture



Intelligent Data Detection and Capture: Steps

The scanning procedure comprises of 5 steps



Configure the `preambleDetector` object



Set the frequency band and channels for the preamble detector to scan.



Scan each specified channel and capture a waveform on successful detection



Process the waveform in MATLAB



Display key information about the signal.

Intelligent Data Detection and Capture: WLAN Scanner MATLAB Demo

OFDM WiFi Scanner Using SDR Preamble Detection

This example shows how to retrieve information about WiFi networks using a software-defined radio (SDR) and preamble detection. The example scans over the 2.4 GHz and 5 GHz channels and uses an SDR preamble detector to detect and capture orthogonal frequency-division multiplexing (OFDM) packets from the air. The example then decodes the OFDM packets to determine which packets are access point (AP) beacons. The AP beacon information includes the service set identifier (SSID), media access control (MAC) address (also known as the basic SSID, or BSSID), AP channel bandwidth, and 802.11 standard used by the AP.

Introduction

This example scans through a set of WiFi channels to detect AP beacons that are transmitted on 20 MHz subchannels. The scanning procedure uses a preamble detector on an NI™ USRP™ radio.

The scanning procedure comprises of these steps.

- Configure the `preambleDetector` object with a preamble that is generated from the legacy long training field (L-LTF).
- Set the frequency band and channels for the preamble detector to scan.
- Scan each specified channel and with each successful detection of an OFDM packet, capture a waveform for a set duration.
- Process the waveform in MATLAB® by searching for beacon frames in the captured waveform and extracting relevant information from each successfully decoded beacon frame.
- Display key information about the detected APs.

Set Up Radio

Call the `radioConfigurations` function. The function returns all available radio setup configurations that you saved using the Radio Setup wizard. For more information, see [Connect and Set Up NI® USRP® Radios](#).

```
radios = radioConfigurations;
```

Specify the name of a saved radio setup configuration to use with this example.

```
radioName = radios(1).Name;
```

Configure Preamble Detector

Create a preamble detector object with the specified radio. To speed up the execution time of this example upon subsequent runs, reuse the workspace object from the first run of the example.

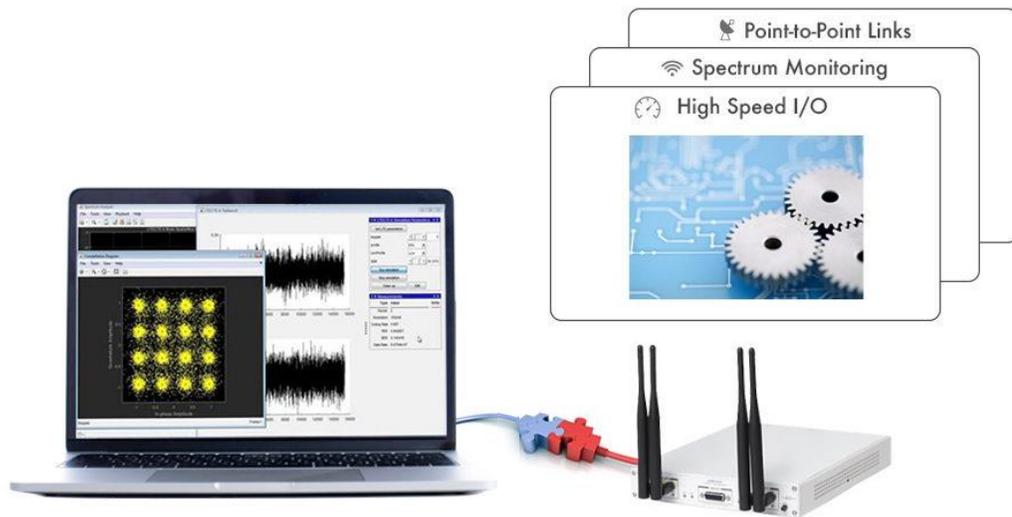
```
if ~exist("pd","var")
    pd = preambleDetector(radioName);
end
```

The screenshot also shows the MATLAB Live Editor interface with the following components:

- Toolbar:** Includes menus for FILE, PLOTS, APPS, LIVE EDITOR, INSERT, and VIEW. It contains various icons for file operations (Open, Save, Print, Export), navigation (Go To, Find), text formatting (Normal, Bold, Italic, Underline), code control (Code Control, Task, Refactor), and execution (Run, Run and Advance, Run to End, Step, Stop).
- File Browser:** Shows the current folder structure, including a Microsoft Excel Comma Separated Values file (oui.csv), a Function folder (helperNoiseEstimate.m, recoverPreamble.m), and a Live Script folder (OFDMWiFiScannerUsi...).
- Workspace:** A table with columns for Name and Value, currently empty.

Wireless Testbench

Explore and test wireless designs using high speed and intelligent data transmit and capture



Other Features

- USRP SDR support
- Simple radio set up
- Prebuilt reference applications

Transmit and capture wideband signals at up to 250 MSPS

End-to-end transceiver design, Standard-based and custom signal transmitter/receiver design

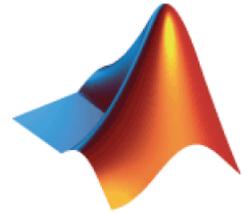
Intelligent data capture using preamble detector

Reduce data requirements by intelligently capturing only waveforms of interest by preamble detection.

Arbitrary sample rate

Work with 5G, WLAN, and DVB-S2 and custom signals using MATLAB and supported SDR

Summary



MathWorks®

- Close collaboration between NI and MathWorks for SDR solutions
- Tight integration between MATLAB and USRP radio
- Accelerating transition from simulation to prototyping
- N310, N320, N321 provide capabilities for apps like high-speed transmit/capture, spectrum monitoring

MATLAB EXPO

Thank you



© 2022 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.