


| | | | | |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------|------------|----------------|----------------------------------------------------------------------------------------------------------------------------|
| Program Współpraca Polska-RPA | RAPORT CZĄSTKOWY z realizacji projektu w ramach programu międzynarodowego Polska-RPA | | |  Narodowe Centrum Badań i Rozwoju |
| Nr raportu | IR-RATfor5G-10 | | | |
| Data aktualizacji raportu: | | 2022.05.11 | Wersja | 7 |
| Numer umowy | PL-RPA2/02/RATfor5G+/2019 | | Akronim | RATfor5G+ |
| Okres realizacji projektu | od | 2019.01.01 | do | 2022.06.30 |
| Tytuł projektu | Technologie dostępu radiowego dla standardu 5G i przyszłych generacji sieci bezprzewodowych | | | |
| Tytuł raportu | Testy USRP | | | |

1. Testy QPSK

Pierwsze testy polegały na sprawdzeniu kompatybilności radia wraz z Matlabem. W tym celu zrobiono QPSK Transmitter oraz QPSK Receiver na dwóch osobnych USRP oraz komputerach. Aby było to możliwe potrzeba pakietów Communications Toolbox Support Package for USRP Radio oraz Simulink.

Oficjalna dokumentacja Matlaba stworzyła skrypty, które to umożliwiają, aczkolwiek nie są one do końca kompatybilne z urządzeniem, więc trzeba je było zmodyfikować.

QPSK Transmitter

Skrypt matlabowy jest w wersji live - jest to dość nowa modyfikacja, która umożliwia nie tylko pisanie samego kodu, ale również warstwę wizualną tworząc praktycznie kod wraz z dokumentacją. Rozwiązanie podobne jest do znanego z Pythona Jupyter Notebook.

W miejscach pokazanych poniżej potrzebna jest zmiana platformy na B200.

Prepare USRP B210

Discover Radio

Discover radio(s) connected to your computer. This example uses the first USRP® radio found using the `findsdru` function. Check if the radio is available and record the radio type. If no available radios are found, then this function will throw an error.

```
radios = findsdru;
radio = [];
for n = 1:length(radios)
    if strcmp(radios(n).Platform, 'B200') && strcmp(radios(n).Status, 'Success')
        radio = radios(n);
        break;
    end
end
if isempty(radio)
    error('USRP B200 radio not found.');
```

Create a USRP B210 transmitter System object.

```
hRadioTx = comm.SDRuTransmitter;
hRadioTx.Platform = 'B200';
hRadioTx.SerialNum = radio.SerialNum;
hRadioTx.ChannelMapping = 1;
hRadioTx.CenterFrequency = 5.8e9;
hRadioTx.MasterClockRate = 20e6;
hRadioTx.InterpolationFactor = 100;
hRadioTx.Gain = 89.75;
```

Obtain hardware info and verify hardware settings.

```
hRadioTxInfo = info(hRadioTx)
```

Rys. 1 Fragment skryptu QPSK Transmitter

Prepare QPSK Transmitter

Create and configure the QPSK transmitter System object.

```
21 params = usrpQPSKTxParams('B200', false);
22 hTx = QPSKTransmitter('UpsamplingFactor', 4, ...
23     'MessageLength', params.MessageLength, ...
24     'MessageBits', params.MessageBits, ...
25     'NumberOfMessage', params.NumberOfMessage, ...
26     'RolloffFactor', 0.5, ...
27     'RaisedCosineFilterSpan', 10, ...
28     'ScramblerBase', 2, ...
29     'ScramblerPolynomial', [1 1 0 1], ...
30     'ScramblerInitialConditions', [0 0 0 0]);
```

Construct spectrum analyzer object.

In our radio, for the choice of windowing, we use Kaiser window with sidelobe attenuation = 160 dB.

```
31 hSpectrum = dsp.SpectrumAnalyzer;
32 hSpectrum.SampleRate = hRadioTxInfo.BasebandSampleRate;
33 hSpectrum.SpectralAverages = 10;
34 hSpectrum.FrequencyResolutionMethod = 'WindowLength';
35 hSpectrum.WindowLength = 8192;
36 hSpectrum.Window = 'Kaiser';
37 hSpectrum.SidelobeAttenuation = 160;
```

Rys. 2 Fragment skryptu QPSK Transmitter cz. 2

Reszta kodu pozostaje bez zmian.

Main Streaming Loop.

```
38 for iter = 1:1e9
39
40     % Generate QPSK pulse-shaped complex baseband I/Q samples.
41     x = hTx();
42
43     % Plot PSD.
44     hSpectrum(x);
45
46     % Use USRP radio to tx one frame of complex baseband I/Q samples.
47     underrun = hRadioTx(x);
48     if underrun ~= 0
49         fprintf('underrun = %d.\n', underrun);
50     end
51
52 end
```

Release objects.

```
53 release(hRadioTx)
54 release(hTx)
55 release(hSpectrum)
```

Rys. 3 Fragment skryptu QPSK Transmitter cz. 3

Po włączeniu, na drugim komputerze z podłączonym drugim niezależnym USRP B200 włączany jest skrypt dla odbiornika.

2. QPSK Receiver

W tych dwóch miejscach kod wymaga ponownie zmiany na platformę B200.

1
2
3
4
5
6
7
8
9
10
11

Prepare USRP B210

Discover Radio

Discover radio(s) connected to your computer. This example uses the first USRP® radio found using the `findsdru` function. Check if the radio is available and record the radio type. If no available radios are found, then this function will throw an error.

```
radios = findsdru;  
radio = [];  
for n = 1:length(radios)  
    if strcmp(radios(n).Platform, 'B200') && strcmp(radios(n).Status, 'Success')  
        radio = radios(n);  
        break;  
    end  
end  
if isempty(radio)  
    error('USRP B200 radio not found.');
```

12
13
14
15
16
17
18
19
20
21
22

Create a USRP B210 receiver System object.

```
hRadio = comm.SDRuReceiver;  
hRadio.Platform = 'B200';  
hRadio.SerialNum = radio.SerialNum;  
hRadio.ChannelMapping = 1;  
hRadio.CenterFrequency = 5.8e9 - (50e3 - 68.92e3);  
hRadio.MasterClockRate = 20e6;  
hRadio.DecimationFactor = 100;  
hRadio.SamplesPerFrame = 22452;  
hRadio.Gain = 76 - 20;  
hRadio.TransportDataType = 'int16';  
hRadio.OutputDataType = 'double';
```

Rys. 4 Fragment skryptu QPSK Receiver

23

Obtain hardware info and verify hardware settings.

```
hRadioInfo = info(hRadio)
```

24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

Prepare QPSK Receiver

Create and configure the QPSK receiver System object.

```
params = usrpQPSKRxParams('B200', false);  
hRx = QPSKReceiver(...  
    'ModulationOrder', params.ModulationOrder, ...  
    'SampleRate', params.Fs, ...  
    'DecimationFactor', params.Decimation, ...  
    'FrameSize', params.FrameSize, ...  
    'HeaderLength', params.HeaderLength, ...  
    'NumberOfMessage', params.NumberOfMessage, ...  
    'PayloadLength', params.PayloadLength, ...  
    'DesiredPower', params.DesiredPower, ...  
    'AveragingLength', params.AveragingLength, ...  
    'MaxPowerGain', params.MaxPowerGain, ...  
    'RolloffFactor', params.RolloffFactor, ...  
    'RaisedCosineFilterSpan', params.RaisedCosineFilterSpan, ...  
    'InputSamplesPerSymbol', params.Interpolation, ...  
    'MaximumFrequencyOffset', params.MaximumFrequencyOffset, ...  
    'PostFilterOversampling', params.Interpolation/params.Decimation, ...  
    'PhaseRecoveryLoopBandwidth', params.PhaseRecoveryLoopBandwidth, ...  
    'PhaseRecoveryDampingFactor', params.PhaseRecoveryDampingFactor, ...  
    'TimingRecoveryDampingFactor', params.TimingRecoveryDampingFactor, ...  
    'TimingRecoveryLoopBandwidth', params.TimingRecoveryLoopBandwidth, ...  
    'TimingErrorDetectorGain', params.TimingErrorDetectorGain, ...  
    'PreambleDetectorThreshold', params.PreambleDetectorThreshold, ...  
    'DescramblerBase', params.ScramblerBase, ...  
    'DescramblerPolynomial', params.ScramblerPolynomial, ...  
    'DescramblerInitialConditions', params.ScramblerInitialConditions, ...  
    'BerMask', params.BerMask, ...  
    'PrintOption', true);
```

Rys. 5 Fragment skryptu QPSK Receiver cz. 2

Reszta kodu pozostaje bez zmian i po tych modyfikacjach jest gotowa do włączenia.

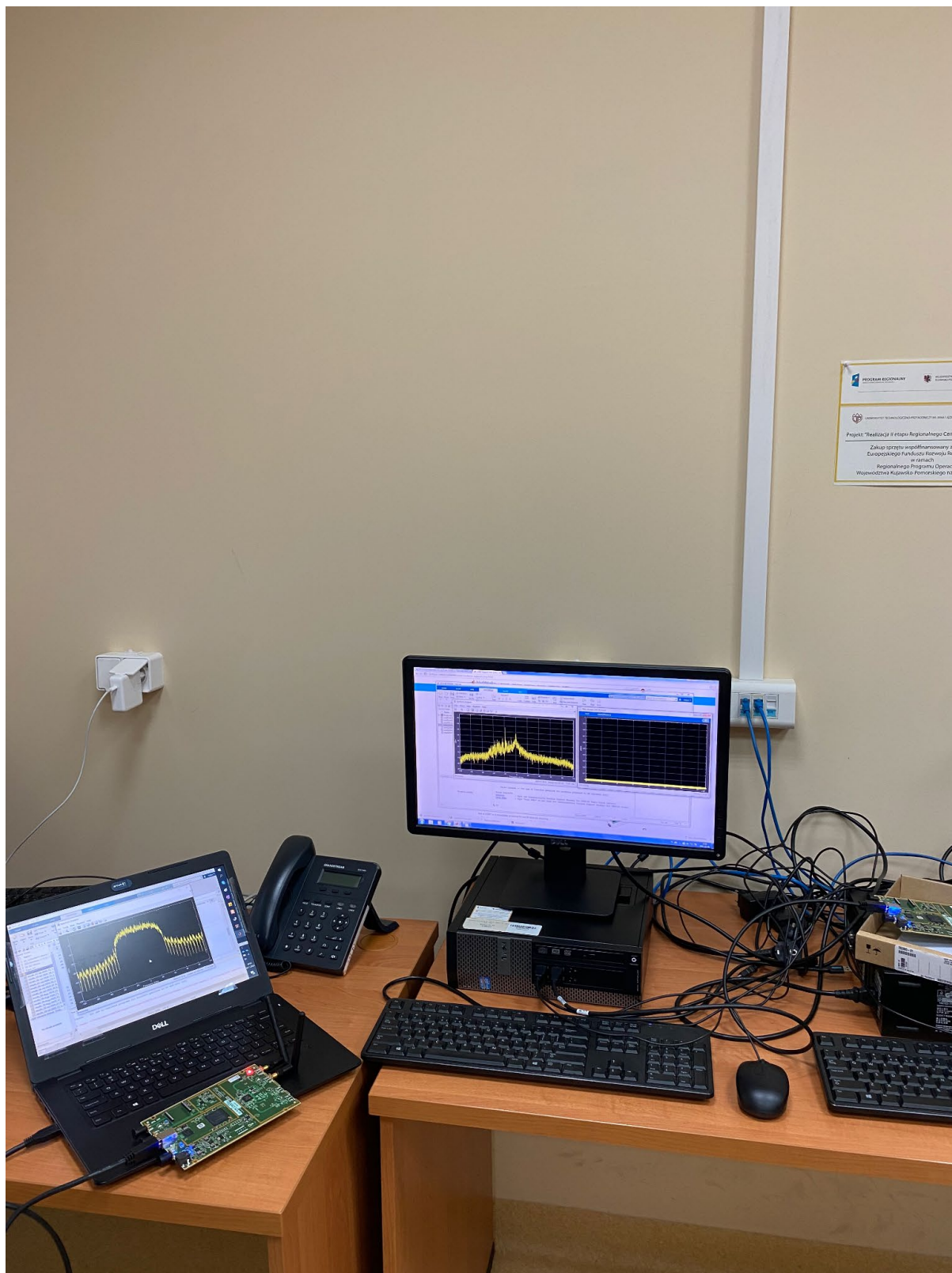
| | |
|-----------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| Construct spectrum analyzer object. | |
| In our radio, for the choice of windowing, we use Kaiser window with sidelobe attenuation = 160 dB. | |
| 52 | hSpectrum = dsp.SpectrumAnalyzer; |
| 53 | hSpectrum.SampleRate = hRadioInfo.BasebandSampleRate; |
| 54 | hSpectrum.SpectralAverages = 10; |
| 55 | hSpectrum.FrequencyResolutionMethod = 'WindowLength'; |
| 56 | hSpectrum.WindowLength = 8192; |
| 57 | hSpectrum.Window = 'Kaiser'; |
| 58 | hSpectrum.SidelobeAttenuation = 160; |
| Construct time-domain plot object. | |
| 59 | hTimePlot = dsp.ArrayPlot; |
| 60 | hTimePlot.Name = 'Time-domain plot of abs(x)'; |
| 61 | hTimePlot.YLabel = 'abs(x)'; |
| 62 | hTimePlot.YLimits = [0 1]; |

Rys. 6 Fragment skryptu QPSK Receiver cz. 3

| | |
|--------------------------------|--------------------------------------------------------------------|
| Main Streaming Loop. | |
| 63 | for iter = 1:1e9 |
| 64 | |
| 65 | % Keep accessing the SDRu System object output until it is valid |
| 66 | len = 0; |
| 67 | while len <= 0 |
| 68 | [corruptSignal, len, overrun] = hRadio(); |
| 69 | if overrun ~= 0 |
| 70 | fprintf('overrun = %d. ', overrun); |
| 71 | fprintf('*****\n'); |
| 72 | end |
| 73 | end |
| 74 | |
| 75 | % When the SDRu System object output is valid, decode the received |
| 76 | % message |
| 77 | [~, ~, BER] = hRx(corruptSignal); |
| 78 | |
| 79 | % Plot PSD. |
| 80 | hSpectrum(corruptSignal); |
| 81 | |
| 82 | % Plot abs(x). |
| 83 | hTimePlot(abs(corruptSignal)); |
| 84 | |
| 85 | end |
| Release System objects. | |
| 86 | release(hRx); |
| 87 | release(hRadio); |
| 88 | release(hTimePlot); |
| 89 | release(hSpectrum); |

Rys. 7 Fragment skryptu QPSK Receiver cz. 4

Po włączeniu skryptu TX na jednym urządzeniu i RX na drugim, sprawdzono działanie. Zdjęcie poniżej pokazuje włączone dwa radia oraz analizę widma, na którym widać zmiany po przybliżeniu jednego urządzenia do drugiego.



Rys. 8 Działanie QPSK

3. Testy LTE

Po weryfikacji działania radia wraz z Matlabem postanowiono przejść do systemu LTE i przetestowania testbedu w tej technologii. Aby rozpocząć pracę trzeba posiadać w tym przypadku Communications Toolbox Support Package for USRP Radio oraz LTE Toolbox. Tutaj Matlab również posiada dokumentację, która nie jest jednak ponownie zgodna z urządzeniem, co trzeba było zmienić.

LTE Transmitter

Dokumentacja pokazuje przykład wysyłania poprzez dwie anteny i nie przewiduje wybranego modelu, ponieważ USRP B200 posiada tylko jeden port TX i jeden RX, dlatego w kilku miejscach potrzebne są zmiany.

LTE SIB1 Transmission over Two Antennas

This example uses both channels of USRP® B210, X300 or X310 to transmit an LTE downlink signal that requires two antennas. The signal is generated by the LTE Toolbox™ and random bits are inserted into the SIB1 field, the first of the System Information Blocks. The accompanying example [sdruLTE2x2SIB1Rx.m](#) receives this signal with two antennas, recovers the SIB1 data, and checks the CRC.

This example uses the SDRu Transmitter System object™. The ChannelMapping property of the object is set to [1 2] to enable use of both channels. The step method takes a two-column matrix in which the first column is the signal for 'RF A' of the radio and the second column is the signal for 'RF B' of the radio.

After starting this example, please run [sdruLTE2x2SIB1Rx.m](#) in a new MATLAB session. In Windows, if two B210 radios are used for these examples, each radio must be connected to a separate computer.

Please refer to the Setup and Configuration section of [Documentation for USRP® Radio](#) for details on configuring your host computer to work with the SDRu Transmitter System object.

Generate LTE Signal

```
1 % Check for presence of LTE Toolbox
2 if isempty(ver('lte'))
3     error(message('sdru:examples:NeedLST'));
4 end
5
6 % Generate LTE signal
7 rmc = lteRMCDL('R.12'); % Base RMC configuration
8 rmc.CellRefP = 2; % 2 transmit antennas
9 rmc.PDSCH.NLayers = 2; % 2 layers
10 rmc.NCellID = 64; % Cell identity
11 rmc.NFrame = 100; % Initial frame number
12 rmc.TotSubframes = 8*10; % Generate 8 frames. 10 subframes per frame
13 rmc.OCNGPDSCHEnable = 'On'; % Add noise to unallocated PDSCH resource elements
14 rmc.PDSCH.RNTI = 61;
15 rmc.SIB.Enable = 'On';
16 rmc.SIB.DCIFormat = 'Format1A';
17 rmc.SIB.AllocationType = 0;
18 rmc.SIB.VRBStart = 0;
19 rmc.SIB.VRBLength = 6;
20 rmc.SIB.Gap = 0;
21 rmc.SIB.Data = randi([0 1],144,1); % Use random bits in SIB data field. This is not a valid SIB message
22 trData = [1;0;0;1];
23 [eNodeBOutput,txGrid,rmc] = lteRMCDLTool(rmc,trData);
```

Rys. 9 Fragment skryptu LTE Transmitter

Zmiana platformy na B200.

```
Connect to Radio

31 radioFound = false;
32 radiolist = findsdru;
33 for i = 1:length(radiolist)
34     if strcmp(radiolist(i).Status, 'Success')
35         if strcmp(radiolist(i).Platform, 'B200')
36             radio = comm.SDRuTransmitter('Platform', 'B200' ...
37                 'SerialNum', radiolist(i).SerialNum);
38             radio.MasterClockRate = 1.92e6 * 4; % Need to exceed 5 MHz minimum
39             radio.InterpolationFactor = 4;      % Sampling rate is 1.92 MHz
40             radioFound = true;
41             break;
42         end
43         if (strcmp(radiolist(i).Platform, 'X300') || ...
44             strcmp(radiolist(i).Platform, 'X310'))
45             radio = comm.SDRuTransmitter('Platform', radiolist(i).Platform, ...
46                 'IPAddress', radiolist(i).IPAddress);
47             radio.MasterClockRate = 184.32e6;
48             radio.InterpolationFactor = 96;      % Sampling rate is 1.92 MHz
49             radioFound = true;
50         end
51         if (strcmp(radiolist(i).Platform, 'N300') || ...
52             strcmp(radiolist(i).Platform, 'N310'))
53             radio = comm.SDRuTransmitter('Platform', radiolist(i).Platform, ...
54                 'IPAddress', radiolist(i).IPAddress);
55             radio.MasterClockRate = 122.88e6;
56             radio.InterpolationFactor = 64;      % Sampling rate is 1.92e6
57             radioFound = true;
58         end
59         if (strcmp(radiolist(i).Platform, 'N320/N321'))
60             radio = comm.SDRuTransmitter('Platform', radiolist(i).Platform, ...
61                 'IPAddress', radiolist(i).IPAddress);
62             radio.MasterClockRate = 245.76e6;
63             radio.InterpolationFactor = 128;      % Sampling rate is 1.92e6
64             radioFound = true;
65         end
66     end
67 end
```

Rys. 10 Fragment skryptu LTE Transmitter cz. 2

Tutaj potrzeba zmienić ChannelMapping, który domyślnie ustawiony jest na dwa TX.

```
68
69 if ~radioFound
70     error(message('sdr:examples:NeedMIMORadio'));
71 end
72
73 radio.ChannelMapping = [1]; % Use both TX channels
74 radio.CenterFrequency = 900e6;
75 radio.Gain = 25;
76 radio.UnderrunOutputPort = true;
77
78 radio
```

Rys. 11 Fragment skryptu LTE Transmitter cz. 3

W pętli również trzeba zmienić wielkość macierzy, ponieważ oryginalnie jest większego rozmiaru, gdyż zakłada większą ilość portów.

Send Signal over Two Antennas

```
79 % Scale signal to make maximum magnitude equal to 1
80 eNodeBOutput = eNodeBOutput/max(abs(eNodeBOutput(:)));
81
82 % Reshape signal as a 3D array to simplify the for loop below
83 % Each call to step method of the object will use a two-column matrix
84 samplesPerFrame = 10e-3*rmc.SamplingRate; % LTE frames are 10 ms long
85 numFrames = length(eNodeBOutput)/samplesPerFrame;
86 txFrame = permute(reshape(permute(eNodeBOutput,[1 3 2]), ...
87     samplesPerFrame,numFrames,rmc.CellRefP),[1 3 2]);
88
89 disp('Starting transmission');
90 disp('Please run sdrLTE2x2SIB1Rx.m in a new MATLAB session');
91
92 currentTime = 0;
93 while currentTime < 300 % Run for 5 minutes
94     for n = 1:numFrames
95         % Call step method to send a two-column matrix
96         % First column for TX channel 1. Second column for TX channel 2
97         bufferUnderflow = step(radio,txFrame(:,n));
98         if bufferUnderflow~=0
99             warning('sdr:examples:DroppedSamples','Dropped samples')
100         end
101     end
102     currentTime = currentTime+numFrames*10e-3; % One frame is 10 ms
103 end
104 release(radio);
105 disp('Transmission finished')
```

Rys. 12 Fragment skryptu LTE Transmitter cz. 4

Po takich zmianach można włączyć skrypt i przejść do skryptu odbiornika

LTE Receiver

Ponownie dokumentacja nie przewiduje wybranego modelu, więc w pokazanych miejscach trzeba zmodyfikować kod. Reszta skryptu pozostaje niezmienniona .

```
LTE Cell Search, MIB and SIB1 Recovery with Two Antennas

This example uses both channels of USRP® B210, X300 or X310 to receive an LTE downlink signal. The LTE Toolbox™ is used to synchronize, demodulate and decode the signal sent by the accompanying example sdrLTE2x2SIB1Tx.m. Since the transmitted signal uses a transmit diversity scheme, orthogonal space frequency block code (OSFBC) decoding is performed by the function lteTransmitDiversityDecode. In the end, the SIB1 field, the first of the System Information Blocks, is recovered and the CRC is checked.

This example uses the SDRu Receiver System object™. The ChannelMapping property of the object is set to [1 2] to enable use of both channels. The step method outputs a two-column matrix in which the first column is the signal from 'RF A' of the radio and the second column is the signal from 'RF B' of the radio.

Before starting this example, please run sdrLTE2x2SIB1Tx.m in a separate MATLAB session. In Windows, if two B210 radios are used for these examples, each radio must be connected to a separate computer.

Please refer to the Setup and Configuration section of Documentation for USRP® Radio for details on configuring your host computer to work with the SDRu Receiver System object.

Connect to Radio

1  radioFound = false;
2  radiolist = findsdr;
3  for i = 1:length(radiolist)
4      if strcmp(radiolist(i).Status, 'Success')
5          if strcmp(radiolist(i).Platform, 'B200')
6              radio = comm.SDRuReceiver('Platform', 'B200', ...
7                  'SerialNum', radiolist(i).SerialNum);
8              radio.MasterClockRate = 1.92e6 * 4; % Need to exceed 5 MHz minimum
9              radio.DecimationFactor = 4;         % Sampling rate is 1.92e6
10             radioFound = true;
11             break;
12         end
13         if (strcmp(radiolist(i).Platform, 'X300') || ...
14             strcmp(radiolist(i).Platform, 'X310'))
15             radio = comm.SDRuReceiver('Platform', radiolist(i).Platform, ...
16                 'IPAddress', radiolist(i).IPAddress);
17             radio.MasterClockRate = 184.32e6;
18             radio.DecimationFactor = 96;         % Sampling rate is 1.92e6
19             radioFound = true;
20         end
21     end
```

Rys. 13 Fragment skryptu LTE Receiver

```
21     if (strcmp(radiolist(i).Platform, 'N300') || ...
22         strcmp(radiolist(i).Platform, 'N310'))
23         radio = comm.SDRuReceiver('Platform', radiolist(i).Platform, ...
24             'IPAddress', radiolist(i).IPAddress);
25         radio.MasterClockRate = 122.88e6;
26         radio.DecimationFactor = 64;         % Sampling rate is 1.92e6
27         radioFound = true;
28     end
29     if (strcmp(radiolist(i).Platform, 'N320/N321'))
30         radio = comm.SDRuReceiver('Platform', radiolist(i).Platform, ...
31             'IPAddress', radiolist(i).IPAddress);
32         radio.MasterClockRate = 245.76e6;
33         radio.DecimationFactor = 128;         % Sampling rate is 1.92e6
34         radioFound = true;
35     end
36 end
37 end
38
39 if ~radioFound
40     error(message('sdr:examples:NeedMIMOradio'));
41 end
42
43 radio.ChannelMapping = [1]; % Receive signals from both channels
44 radio.CenterFrequency = 900e6;
45 radio.Gain = 30;
46 radio.SamplesPerFrame = 19200; % Sampling rate is 1.92 MHz. LTE frames are 10 ms long
47 radio.OutputDataType = 'double';
48 radio.EnableBurstMode = true;
49 radio.NumFramesInBurst = 4;
50 radio.OverflowOutputPort = true;
51
52 radio
```

Rys. 14 Fragment skryptu LTE Receiver cz. 2

Aby można było przetestować działanie najlepiej posiadać dwa komputery, jednak pojawia się tutaj problem z licencją Matlab. Rozwiązanie znalezione na forach mówi o tym, żeby włączyć Matlab w dwóch osobnych instancjach i USRP podłączyć do dwóch różnych od siebie portów, np. jeden na USB 2.0, drugi na USB 3.0. Taki krok poczyniono. Po włączeniu najpierw transmisji, a później odbioru, zaprezentowano poniżej wyniki.

Najpierw wykryto transponder i pokazano, żeby włączyć osobny skrypt odbiornika.

```
Checking radio connections...
Win32; Microsoft Visual C++ version 14.2; Boost_107200; UHD_3
----- see libuhd version information above this line -----

radio =
  comm.SDRuTransmitter with properties:

    Platform: 'B200'
    SerialNum: '321587C'
    ChannelMapping: 1
    CenterFrequency: 900000000
    LocalOscillatorOffset: 0
    Gain: 25
    PPSSource: 'Internal'
    ClockSource: 'Internal'
    MasterClockRate: 7680000
    InterpolationFactor: 4
    TransportDataType: 'int16'
    EnableBurstMode: false
Starting transmission
Please run sdrulTE2x2SIB1Rx.m in a new MATLAB session
```

Rys. 15 Informacja o urządzeniu

Z racji tego, że skrypt włączono chwilę później, zaczęła wypisywać się krótko informacja o utracie pakietów, ale finalnie transmisja zakończyła się pomyślnie.

```
Starting transmission
Please run sdrulTE2x2SIB1Rx.m in a new MATLAB session
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Warning: Dropped samples
Transmission finished
```

Rys. 16 Informacja o transmisji

Równoległe po włączeniu odbiornika, został on zidentyfikowany poprawnie jako osobne drugie urządzenie, mimo podłączenia do tego samego komputera, co widać na przykład po innym numerze seryjnym.

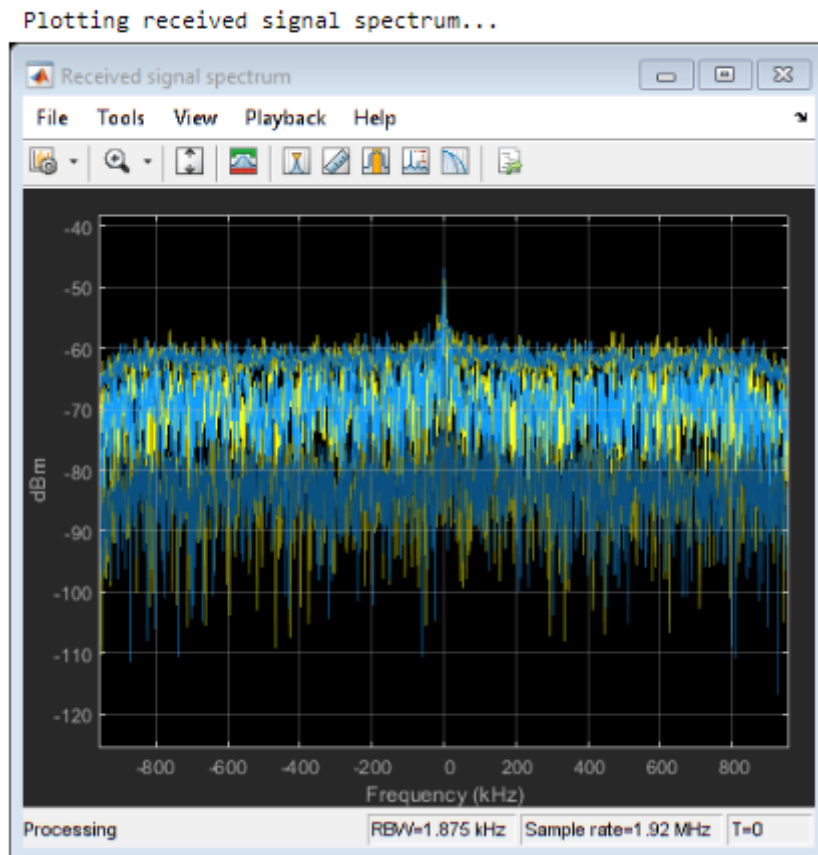
```
Checking radio connections...
Win32; Microsoft Visual C++ version 14.2; Boost_107200; UHD_3
----- see libuhd version information above this line -----

radio =
  comm.SDRuReceiver with properties:

    Platform: 'B200'
    SerialNum: '3215892'
    ChannelMapping: 1
    CenterFrequency: 900000000
    LocalOscillatorOffset: 0
    Gain: 30
    PPSSource: 'Internal'
    ClockSource: 'Internal'
    MasterClockRate: 7680000
    DecimationFactor: 4
    TransportDataType: 'int16'
    OutputDataType: 'double'
    SamplesPerFrame: 19200
    EnableBurstMode: true
    NumFramesInBurst: 4
```

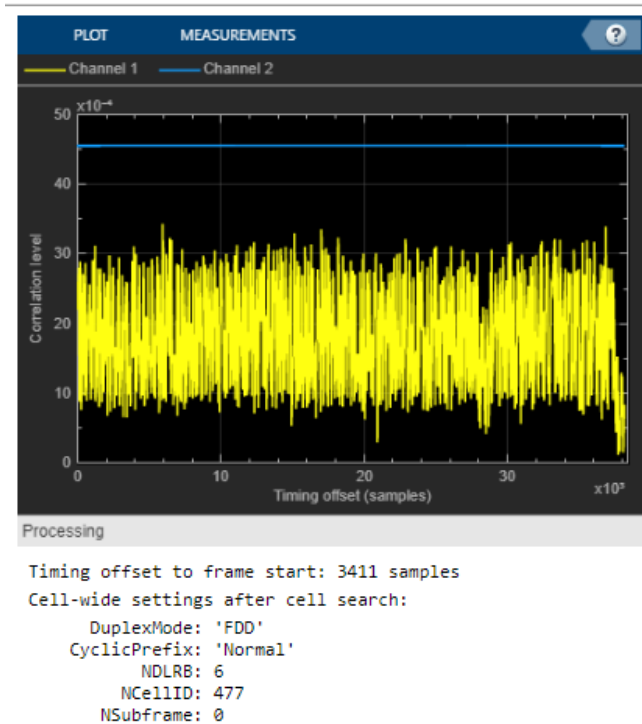
Rys. 17 Informacja o drugim urządzeniu

Następnie utworzył się wykres spektrum sygnału generujący się na żywo.



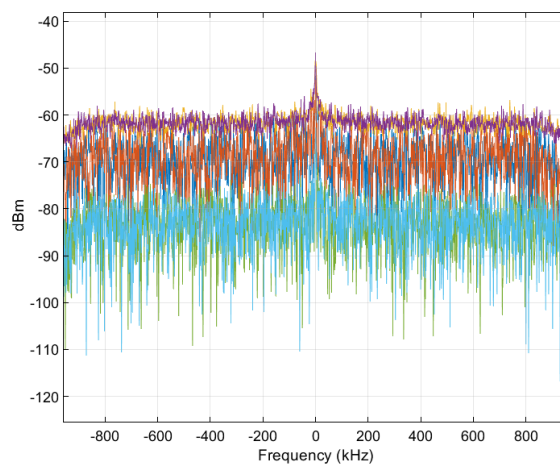
Rys. 18 Spektrum sygnału LTE

Na końcu wygenerował się wykres korelacji. Jest on poprawnie pokazany w tym przypadku tylko dla jednego kanału, ponieważ urządzenie drugiego nie posiada.



Rys. 19 Wykres korelacji

Oraz wygenerował się również wykres ostateczny wykres z analizatora widma na odbiorniku. Na środku widać duży wzrost w momencie przechwycenia pakietu.



RBW=1.875 kHz, Sample rate=1.92 MHz

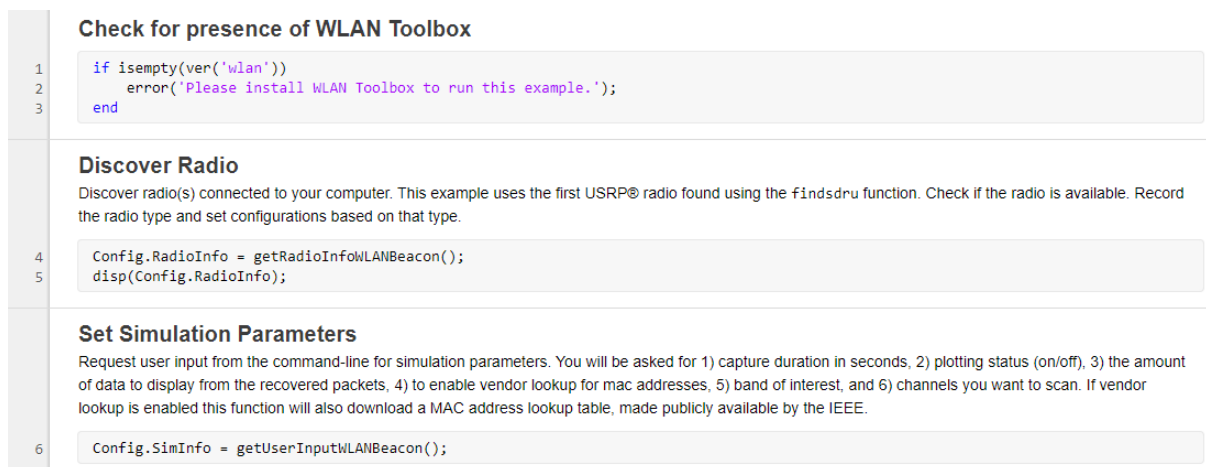
Rys. 20 Wykres z analizatora widma na odbiorniku.

4. Testy Wi-Fi

Finalnym etapem pracy było przetestowanie testbedu na łączności Wi-Fi. Początkowe etapy nie przebiegały pomyślnie ze względu na niezgodne częstotliwości. Po weryfikacji udało się jednak to skorygować.

Matlab w swojej dokumentacji udostępnia kod, który postanowiono zaimplementować. W tym przypadku nie trzeba było nic zmieniać, ponieważ urządzenie jest w skrypcie wykrywane, a nie zdefiniowane na początku.

Najpierw skrypt sprawdza czy jest zainstalowany WLAN Toolbox. Następnie sprawdzane są parametry radia.



Rys. 21 Fragment kodu skryptu WLAN

Po włączeniu radio zostało poprawnie wykryte.

```
Checking radio connections...
      Platform: 'B200'
      Address: '3215892'
      MasterClockRate: 200000000
      USRPDecimationFactor: 1
      USRPGain: 50
```

Rys. 22 Informacja o wykryciu urządzenia

Użytkownik ma do wyboru dwa pasma:

- 5 GHz - domyślne,
- 2.4 GHz.

Po zatwierdzeniu którejś z opcji, wybiera się kanały z danym pasmem. Domyślnie jest to 153 i 157 dla 5 GHz lub 1 i 6 dla 2.4GHz.

```
To find valid channel numbers in your geographic location, please refer to the documentation.

7  % Get channel number
8  reply = input(['What is the band you want to scan?\n', ...
9  '1 == 5 GHz band (default)\n',...
10 '2 == 2.4 GHz band\n[1: '], 's');
11 if isempty(reply)
12     reply = '1';
13 end
14 if strcmp(reply, '1')
15     bandToScan = 5;
16     validChannels = [...
17         ' 7-16   (5.035-5.080 GHz)\n' ...
18         ' 34-64   (5.170-5.320 GHz)\n' ...
19         '100-144 (5.550-5.720 GHz)\n' ...
20         '149-165 (5.745-5.825 GHz)\n' ...
21     ];
22     defaultChannels = '[153 157]';
23 else
24     bandToScan = 2.4;
25     validChannels = [...
26         ' 1-13   (2.412-2.472 GHz)\n'...
27         ' 14     (2.484 GHz)\n'...
28     ];
29     defaultChannels = '[1 6]';
30 end
31
32 % Get channels to scan
33 reply = input(['Valid channel numbers are:\n' validChannels ...
34 'Which channels do you want to scan? ' defaultChannels ':'], 's');
35 if isempty(reply)
36     reply = defaultChannels;
37 end
38 channelsToScan = reshape(str2num(reply), [], 1); %#ok<ST2NM>
```

Rys. 23 Fragment skryptu WLAN cz. 2

Przeprowadzono dwa testy dla osobnych pasm.

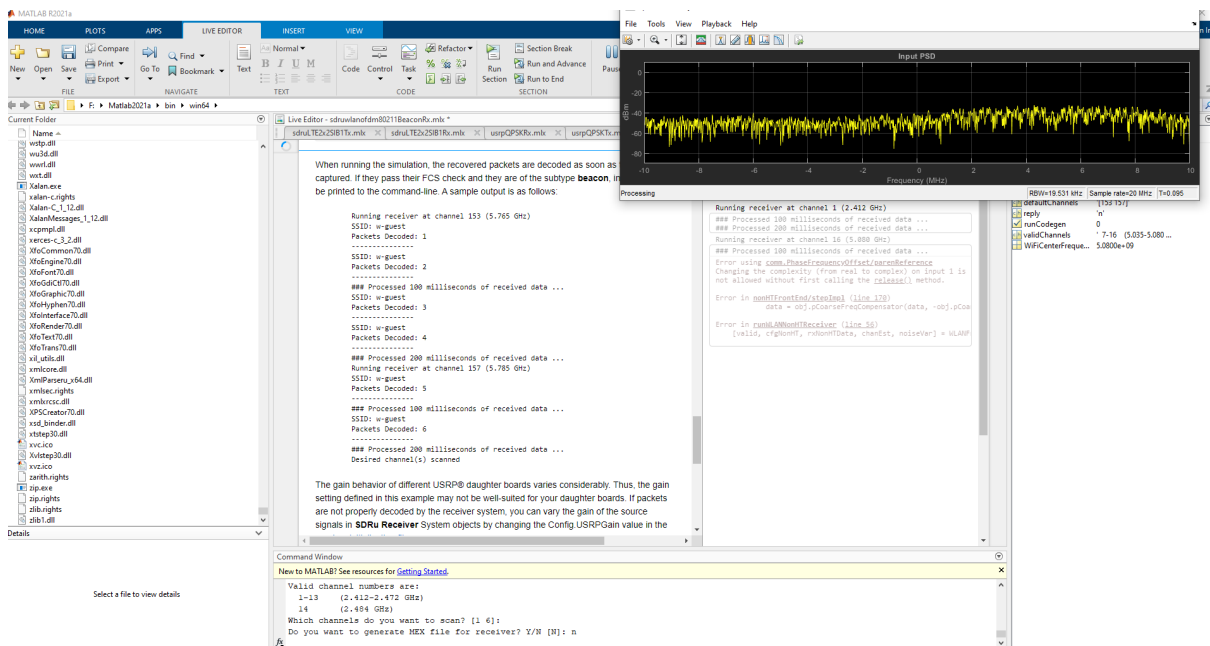
4. Test 2.4 GHz

Pierwszy test był na paśmie 2.4 GHz. Tester pytany jest o kilka opcji. Na początku wybrano czas wychwytywania danych. Później skorzystano z opcji generowania wykresów. Następnie wybrano opcję pokazywania wszystkich pakietów oraz odrzucono wyświetlanie dostawcy przy adresach MAC. Kolejno ustawiono pasmo na 2.4 GHz i wybrano domyślne kanały [1 6]. Ostatecznie odrzucono również opcję generowania pliku MEX, ponieważ przy wcześniejszych próbach generował się zbyt długo.

```
Seconds of data to capture? [0.2]: 0.1
Do you want to enable scopes? Y/N [N]: y
Do you want to display packet info?
1 == SSID only (default)
2 == Additional beacon packet info
3 == Show all packets (includes non-beacons)
[1]: 3
Do you want display vendors for MAC addresses?
(Vendor list must be downloaded once, this may take a minute or two) Y/N [N]: n
What is the band you want to scan?
1 == 5 GHz band (default)
2 == 2.4 GHz band
[1]: 2
Valid channel numbers are:
  1-13      (2.412-2.472 GHz)
  14        (2.484 GHz)
Which channels do you want to scan? [1 6]:
Do you want to generate MEX file for receiver? Y/N [N]: n
```

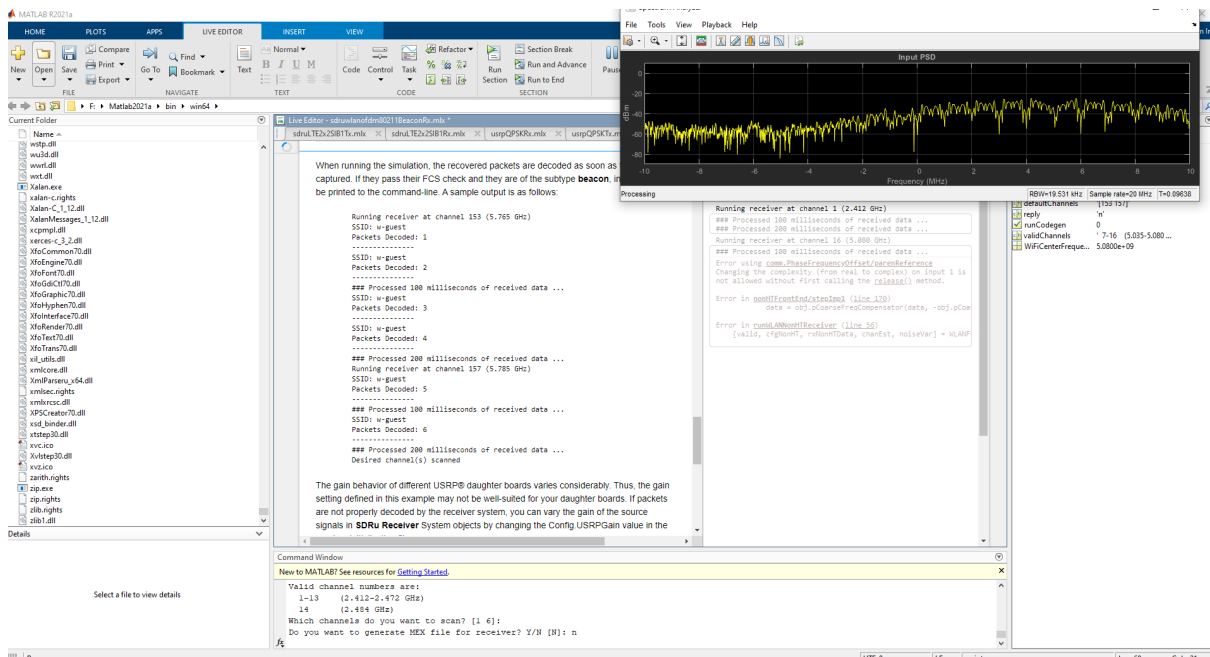
Rys. 24 Informacje do wyboru w skrypcie

Po włączeniu, pokazuje się analizator widma, który na bieżąco sprawdza pasmo.



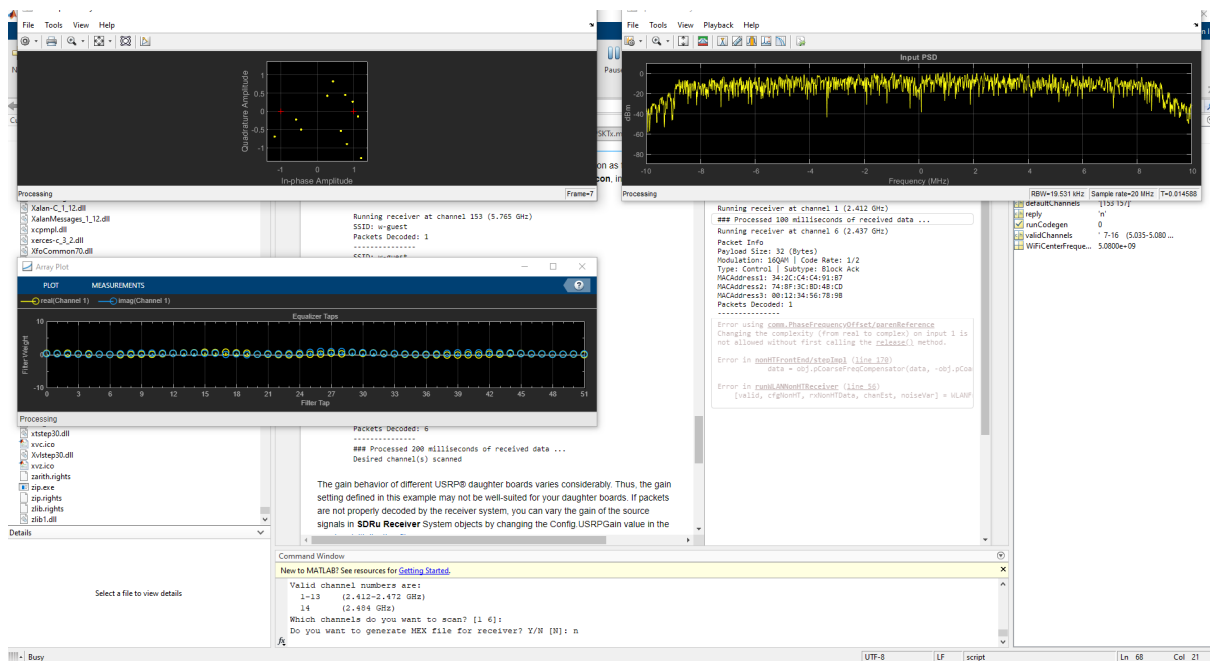
Rys. 25 Włączony skrypt wraz z analizatorem widma

Następnie podłączono się komputerem do routera i włączono pingowanie adresu ip oraz pobieranie kilku dużych plików.



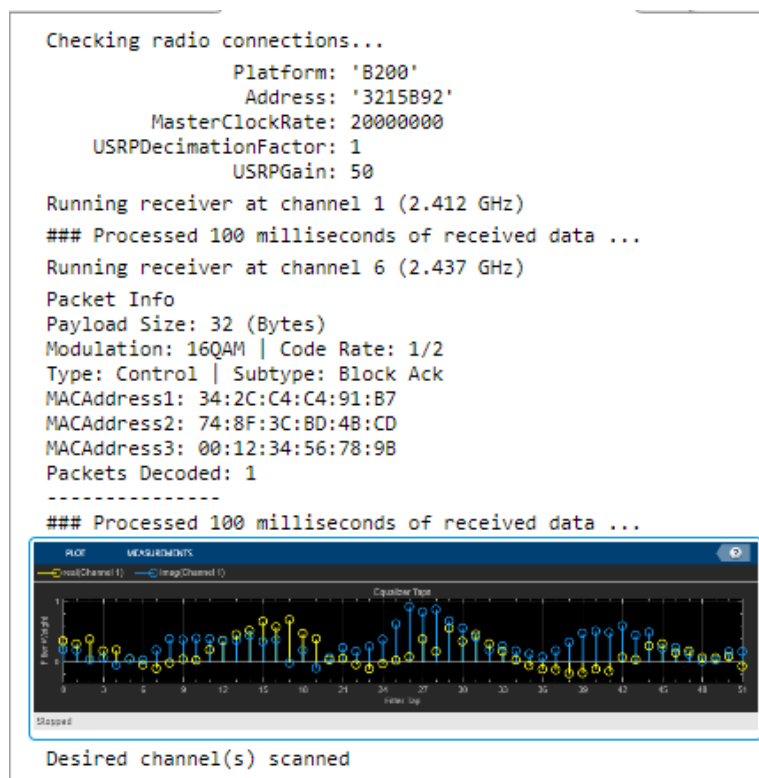
Rys. 26 Włączony skrypt wraz z analizatorem widma cz. 2

Po przybliżeniu komputera do radia, wykres na analizatorze się zmienił.



Rys. 27 Włączony skrypt wraz z analizatorem widma po przechwyceniu pakietu

Ostatecznie pakiet został przechwycony i zdekodowany, co widać w informacji oraz wykresach, a analizator widma znacznie się zmienił, co kończy test na tym paśmie.



Rys. 28 Informacja o przechwyceniu pakietu

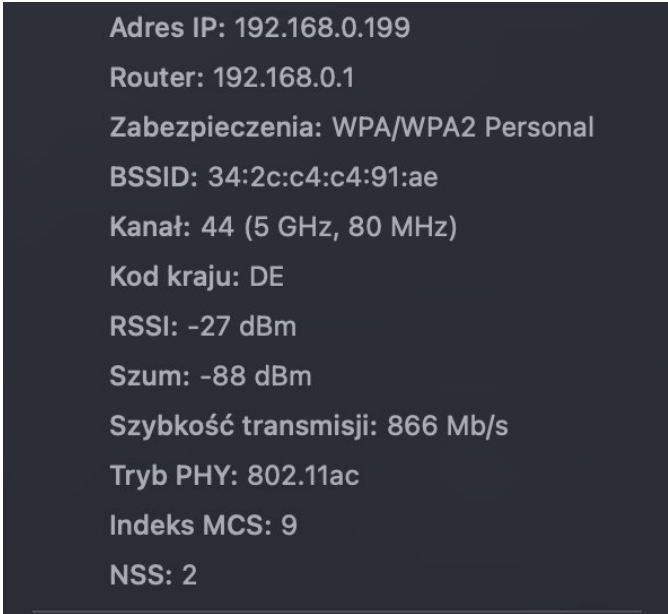
5. Test 5 Ghz

Drugim testem było sprawdzenie tego samego skryptu na paśmie 5 GHz. Wprowadzone parametry prezentują się następująco:

```
Seconds of data to capture? [0.2]: 0.1
Do you want to enable scopes? Y/N [N]: y
Do you want to display packet info?
1 == SSID only (default)
2 == Additional beacon packet info
3 == Show all packets (includes non-beacons)
[l]: 3
Do you want display vendors for MAC addresses?
(Vendor list must be downloaded once, this may take a minute or two) Y/N [N]: n
What is the band you want to scan?
1 == 5 GHz band (default)
2 == 2.4 GHz band
[l]: 1
Valid channel numbers are:
  7-16    (5.035-5.080 GHz)
 34-64    (5.170-5.320 GHz)
100-144  (5.550-5.720 GHz)
149-165  (5.745-5.825 GHz)
Which channels do you want to scan? [153 157]:44
Do you want to generate MEX file for receiver? Y/N [N]: n
```

Rys. 29 Wprowadzone parametry przy teście 5 GHz

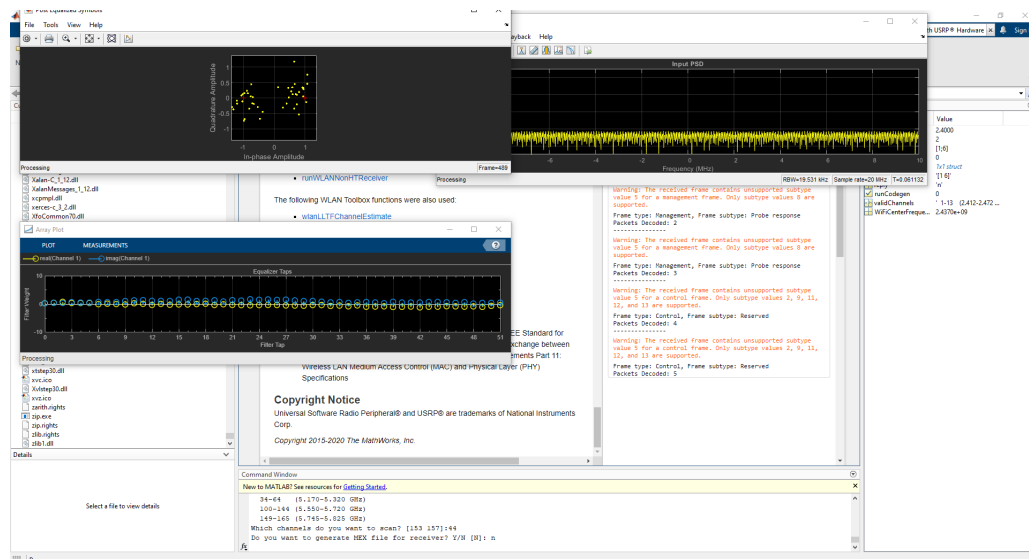
Tutaj zmianami są tylko: pasmo 5 GHz oraz kanał 44, który sprawdzono na komputerze.



```
Adres IP: 192.168.0.199
Router: 192.168.0.1
Zabezpieczenia: WPA/WPA2 Personal
BSSID: 34:2c:c4:c4:91:ae
Kanał: 44 (5 GHz, 80 MHz)
Kod kraju: DE
RSSI: -27 dBm
Szum: -88 dBm
Szybkość transmisji: 866 Mb/s
Tryb PHY: 802.11ac
Indeks MCS: 9
NSS: 2
```

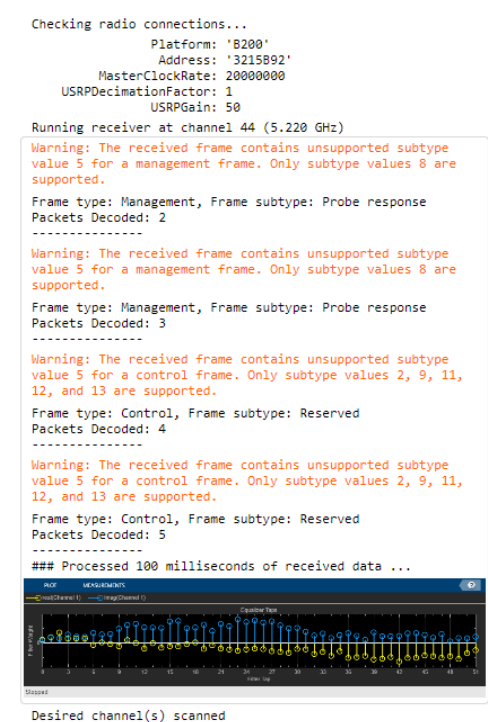
Rys. 30 Parametry access pointa sprawdzone na komputerze

Wyniki prezentują się znacznie lepiej. Analizator widma zmienił swoją wartość na około -90 dB, co zgadza się z danymi z komputera.



Rys. 31 Włączony skrypt 5 GHz po przechwyceniu pakietu

Po rozpoczęciu pingowania, pakiety zostały przechwycone. Ich typ nie jest poprawnie wyłapywany przez skrypt, aczkolwiek testy przeszły i zakończyły się pomyślnie.



Rys. 32 Przechwycone pakiety