RIS Detection Demo Notes

Aymen Khaleel, Laura Moreno
Lehrstuhl für Digitale Komunikations Systeme
Ruhr-Universität Bochum

Abstract

This document is intended to serve as a guide for the planned RIS detection demonstration, as well as to report the different scenarios of functioning and some observations to be taken into account. All the codes are at the GitHub repo: `https://github.com/lauravmorenoc/DKS/tree/main/RIS/demo_codes`, or in the same folder as this file.

1 General instructions

The general idea of the demonstration is to be able to detect the non-blocked RISs in a one transmitter - one receiver system, in which these last two correspond to two Pluto SDRs. In order to be able to run it, the computer has to have:

- ◇ The programs Visual Studio and MATLAB
- ◇ The drivers for using the Pluto SDR
- ◇ GNU Radio
- ◇ Python and its needed libraries: Libiio, pylibiio, numpy, among others.

The needed codes are in the "demo_codes" folder. Inside it, there should be two folders: **commented** and **uncommented**. Suggested is that the codes in the folder *commented* are used, as they are more organized and offer a clear path though the code. The original codes are however held in the folder *uncommented*, just as a back up. The GNU Radio program is held outside the both folders, as it's the only existing version.

1.1 Setting up

The physical set up depends on the scenario. All possible scenarios are described in section 2. In any case, the SDRs are always facing each other, and there's a distance of 60 cm between the Tx and RX antenna centers. The RIS 1 and 2 are placed exactly in the middle, and the distance between them is also 60 cm. In the case of the third scenario, the RIS 3 need so be placed a little further to minimize reflections from RIS 1 and 2, so its placed at 50 cm from the Rx antenna. The four (or five) devices are then connected to the USB extension, which is connected to the computer at its USB data port (in the case of the desktop PC, is the one with a noticeable blue color on the inside).

The following shows the instructions for running the demo step by step:

1. **Make sure that the device recognizes the SDRs:** Open the command console of the PC and write the command "ipconfig". If the SDRs are recognized by the system, there should be at least one connected LAN network with a device associated to the IP: 192.168.0.1

2. Remain in the command console of the PC and write the command "iio_info -s". This command shows the USB ports associated to each device. It is interesting for now just to now the ports of the two SDRs. Note which one corresponds to the transmitter, which in our case corresponds to the SDR whose serial number ends in 67 (it can be also the other one, but this is how we tested it). The other corresponds then to the receiver, whose serial number ends with 74.

3. Open the python code, which is different depending on the scenario (explained below). Go to the line 101 which corresponds to `sdr = adi.ad9361(uri='usb:1.12.5')`, and change the port to the corresponding Rx SDR port, which you noticed in the previous item.

4. Open the MATLAB code "msequence_twoRIS.m", or "msequence_RIS3.m" for the third scenario.

5. Go to the PC windows search and open the "Device Manager". Look for one of the options that says "COM" and see which ones corresponds to the RIS. After that, go to the MATLAB code and change the corresponding port name for all RIS in use (`risX = ris_init('COMXX', 115200);`). You may now test the code to check correct functioning. The RIS has to start turning on and off its leds quickly because of the sequence commands.

6. Open GNU Radio. This is the program used to transmit the signal, which corresponds to a cosine signal. The file is called "Transmitter_GNURadio.grc". You should see a **Variable** block with ID "USBport". Change the variable value according to the usb port associated to the transmitter SDR. Now, run the code. You should see a PSD plot appear with a relatively strong peak at 200 kHz.

7. Now, run the Python code. The first thing it will ask for is authorization for finding the threshold value associated to the first RIS. Once you agree (send 'y' through the console), it will start the process. The whole thing will repeat for RIS 2 (and RIS 3 in the third scneario). Whether if the RIS has to be put a blockage or not depends on each scenario and it's described below.

8. Once the threshold calculation process is finished, the code starts running normally and you will see the correlation peaks. You can run the MATLAB code now and see the change in the graphs.

9. Now, use the small blockage (or aluminum paper) to cover any of the RISs. The correlation peaks associated to that RIS will then threspass the threshold value and its circle will turn red, showing that that RIS is not available anymore.

2 Working scenarios

2.1 First scenario

Two RIS, each RIS running alone at a time. Instead of putting blockage we are turning on and off each RIS individually. Works fine with an aluminum sheet in between the RIS.

2.1.1 Finding threshold

The threshold is found by running the python code and saying yes to finding the threshold for each surface without covering the surfaces and without running the MATLAB codes yet.

2.1.2 Codes needed

:

- **Transmitter - GNU Radio:** Transmitter_GNURadio.grc
- **Python:** two_RISs.py
- **RIS control - MATLAB:** msequence_oneRIS.m

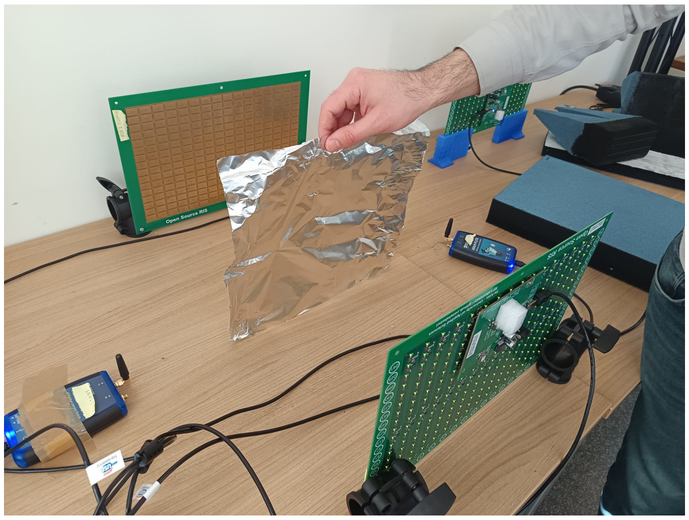

Figure 1: Scenario 1

2.2 Second scenario

Two RIS, both of them running together (implies that they work separatedly as well). A large sponge blockage was put in the middle to block the direct path between the Tx and the Rx antennas and to minimize reflection between the RISs. Another smaller sponge blockage was used to cover one or the other RIS from the receiver according to which RIS we want to detect. We also tried moving the receiver closer to the RIS which is not blocked (putting it in front of the RIS), which works very well. An explanation video is found by clicking here or going to the next url: `https://github.com/lauravmorenoc/DKS/tree/main/Media`.

2.2.1 Finding threshold

The threshold was found by running the Python code and covering the other RIS when calculating it. For example, the function will ask if it can proceed to find the threshold for the RIS 1. In that case, we covered the RIS2 using the large sponge blockage and then answered y (yes) on the command window. Works exactly the other way around for RIS2.

2.2.2 Codes needed

:

- ◇ **Transmitter - GNU Radio:** Transmitter_GNURadio.grc
- ◇ **Python:** two_RISs.py

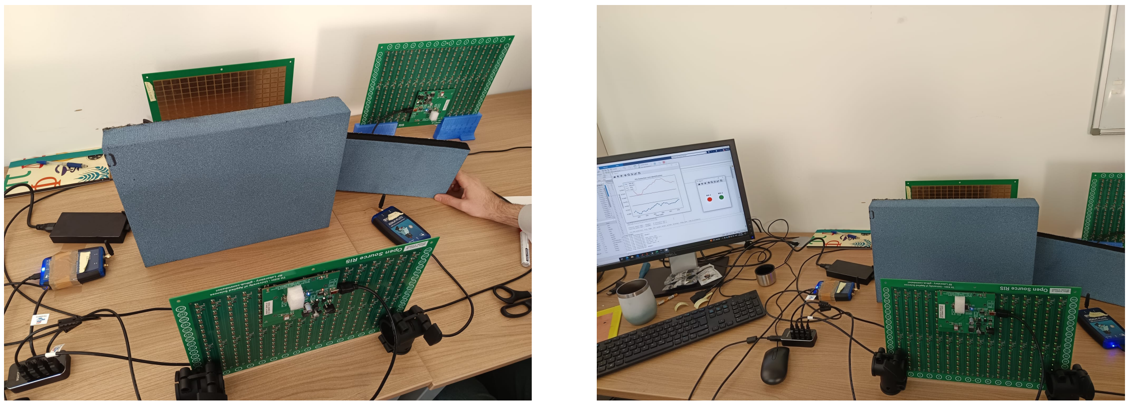
Figure 2: Scenario 2

- ◇ **RIS control - MATLAB:** msequence_twoRIS.m

2.3 Third scenario

Three RIS. Running each of them separatedly or at the same time. A large sponge blockage was put in the middle of the RIS 1 and 2 to block the direct path between the Tx and the Rx antennas and to minimize reflection between these two RISs. Another sponge blockage was used to simulate an obstacle in front of the third RIS. Another smaller sponge blockage was used to cover RIS 1 or 2 from the receiver according to which RIS we want to detect. We also tried moving the receiver closer to the RIS that we want to detect is not blocked (putting it in front of the RIS), which works well but depends a lot on the position of the receiver relative to the RIS center.

When detecting each RIS, the high autocorrelation of each sequence makes its correlation peaks very high, enough to detect it correctly. However, there's also a high (although less) correlation between the different sequences, specially when detecting the RIS3. That makes the correlation peaks of the other two RIS increase as well. This issue can be controlled by modifying the threshold factors accordingly.

2.3.1 Codes needed

:

- ◇ **Transmitter - GNU Radio:** Transmitter_GNURadio.grc
- ◇ **Python:** three_RISs.py

◇ **RIS control - MATLAB:** For running each RIS separately: `msequence`$_{o}neRIS.m.Forrunningthethree$

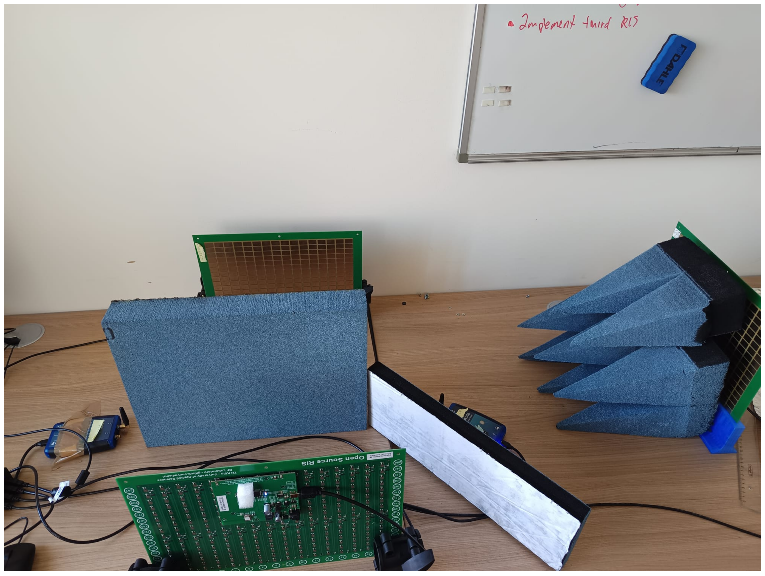
Figure 3: Scenario 3

3 Observations

To take in account:

- ◇ The RIS work normally fine but sometimes after a while we get a response error on the MATLAB command window saying that getting an answer is taking too long. In that case, it is enough to rest each RIS by pressing the white button on the back.
- ◇ As far as we noticed, using aluminum sheets might work as well as the sponge blockages, in case it's too difficult to carry them around.
- ◇ Using tape to fix the Tx SDR might be a good idea. Carry a ruler as well.
- ◇ Avoid a lot of movements or distubances both whilst finding the thresholds and running the demo. It's a very sensible system.
- ◇ The threshold factors are very dependent on the environment. Adjust them properly once while preparing the scenarios and maybe once more if the environment changes (a lot of people around, more devices around, etc).