Wireless Communication Systems
@CS.NCTU

Lab2: OFDM over USRP

2018.03.30

Outline

- Background
  - USRP
  - Environment
- ToDo
  - Tx / Rx (C++ for USRP)
  - Decoding (MATLAB)
- Grading Criteria

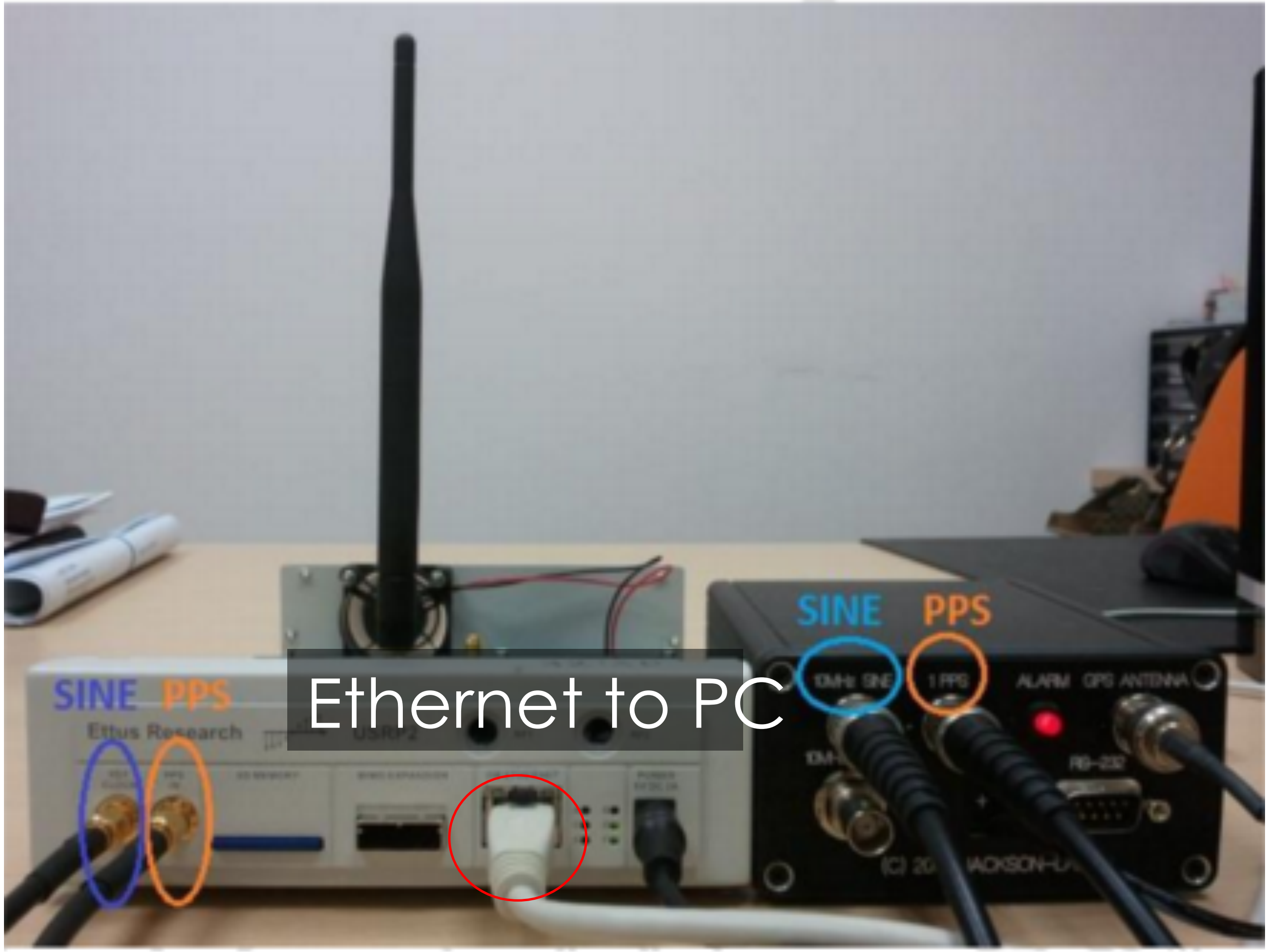
What is USRP?

- Software Defined Radio
- Use software to program how a radio operates

USRP

- Universal Software Defined Radio
  - Expensive! (~2,000USD)
  - Use C++/ python/GUI to define the radio!!

- Official document
  - https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR.pdf



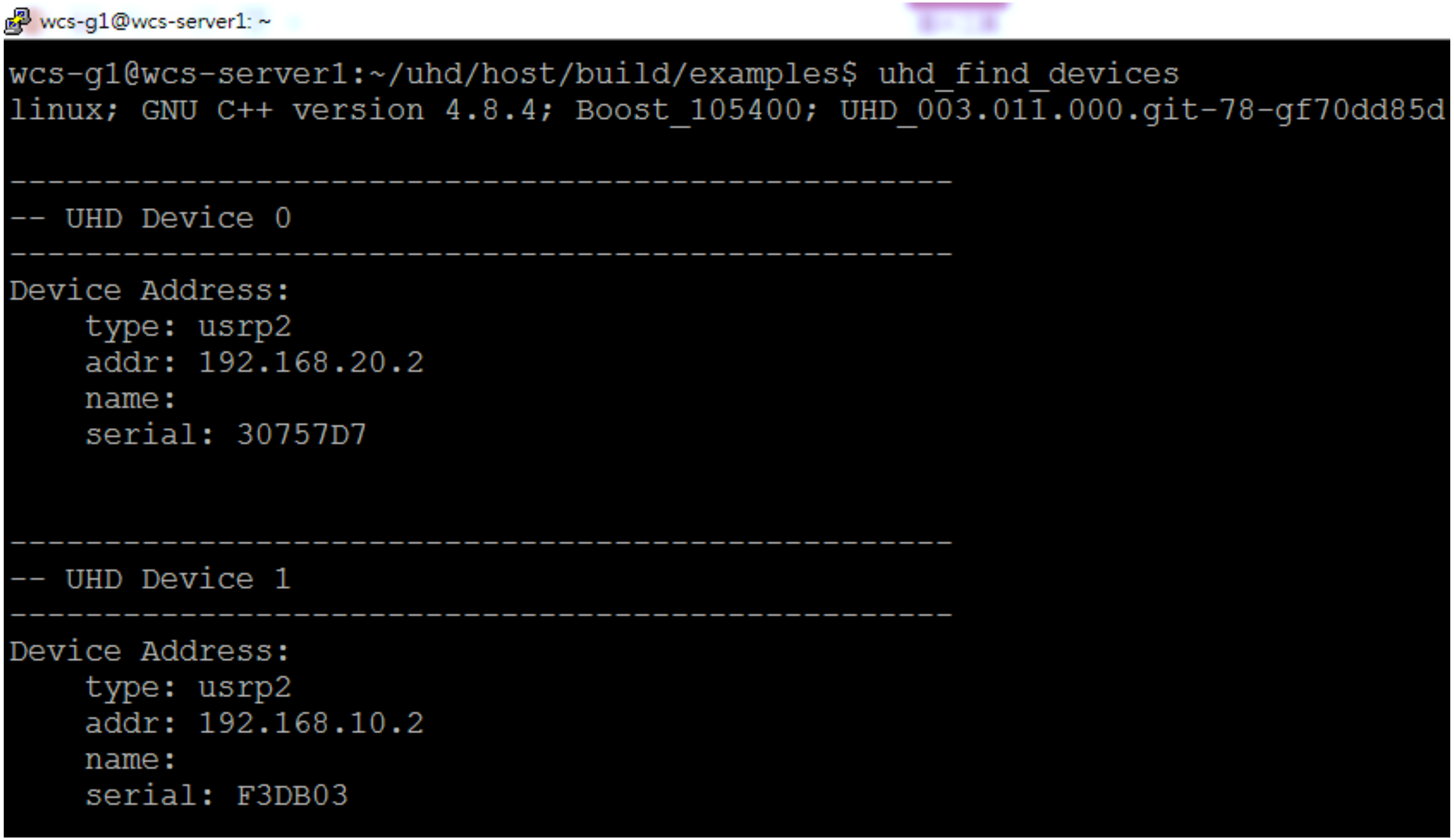
Ethernet to PC

USRP Driver (API)

- UHD
  - USRP Hardware Driver
  - C++ API
  - http://files.ettus.com/manual/
  - https://github.com/EttusResearch/uhd

- Installation
  - Done by TA

- Locating devices
  - host/build/utils/uhd_find_devices --args "addr=192.168.10.14"
  - This program scans the network for supported devices and prints out a list of discovered devices and their IP addresses

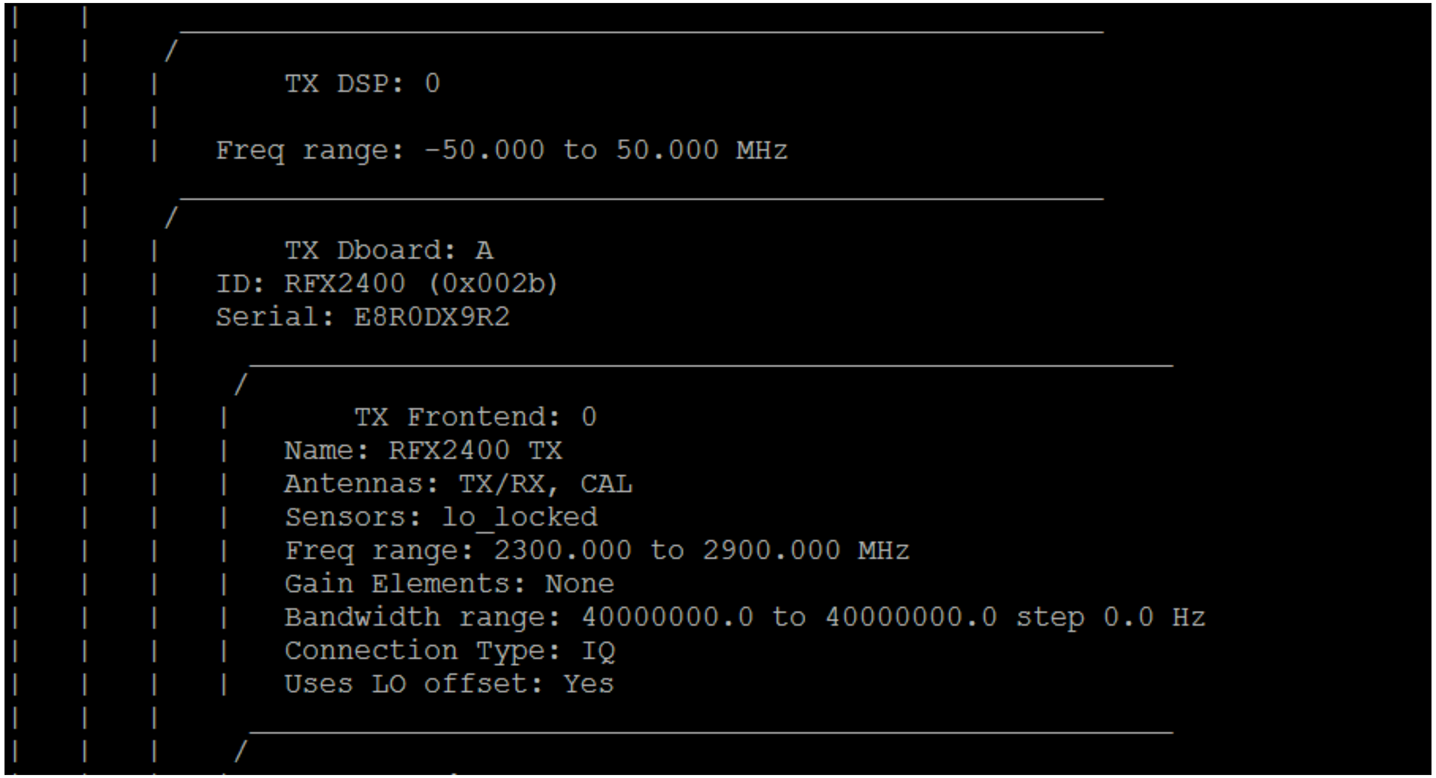


```
wcs-g1@wcs-server1:~/uhd/host/build/examples$ uhd_find_devices
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.011.000.git-78-gf70dd85d

--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
    type: usrp2
    addr: 192.168.20.2
    name:
    serial: 30757D7


--------------------------------------------------
-- UHD Device 1
--------------------------------------------------
Device Address:
    type: usrp2
    addr: 192.168.10.2
    name:
    serial: F3DB03
```

- host/build/utils/uhd_usrp_probe
  - This program constructs an instance of the device and prints out its properties, such as detected daughterboards, frequency range, gain ranges, etc



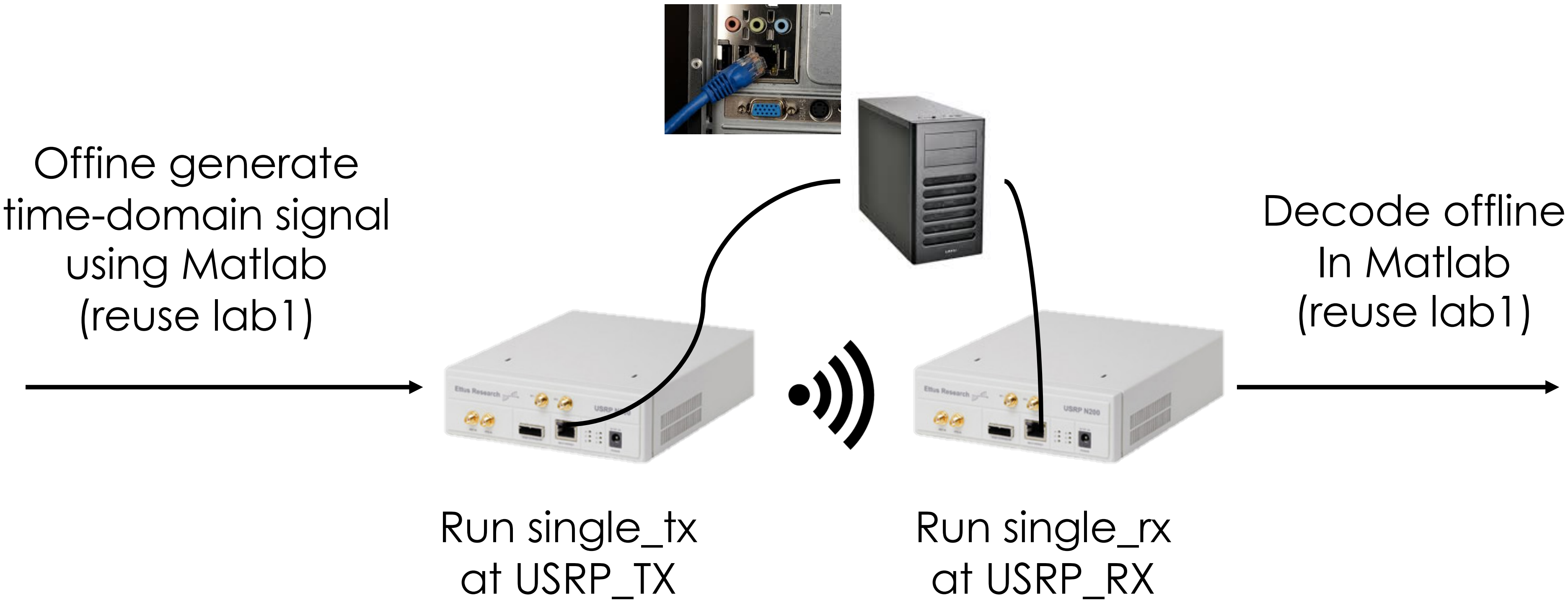
```
|   |     _____________________________________________________
|   |    /
|   |   |       TX DSP: 0
|   |   |
|   |   |   Freq range: -50.000 to 50.000 MHz
|   |
|   |     _____________________________________________________
|   |    /
|   |   |       TX Dboard: A
|   |   |   ID: RFX2400 (0x002b)
|   |   |   Serial: E8R0DX9R2
|   |   |
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       TX Frontend: 0
|   |   |   |   Name: RFX2400 TX
|   |   |   |   Antennas: TX/RX, CAL
|   |   |   |   Sensors: lo_locked
|   |   |   |   Freq range: 2300.000 to 2900.000 MHz
|   |   |   |   Gain Elements: None
|   |   |   |   Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: Yes
|   |   |
|   |   |     _____________________________________________________
|   |   |    /
```

How to Add a New File & Compile

- File (Source) Directory
  - Use built in Makefile
  - Put your files in ~/uhd/host/examples/
  - Add your filenames to the CmakeList.txt in ~/uhd/host/examples

- Compile (Binary) Directory
  - cd ~/uhd/host/build/examples
  - make
  - The executable bin file should be in this folder after compile

Environment



Offine generate
time-domain signal
using Matlab
(reuse lab1)

Decode offline
In Matlab
(reuse lab1)

Run single_tx
at USRP_TX

Run single_rx
at USRP_RX

- USRP Testbed in EC-538
- Access through ssh to test your UHD codes
  - single_tx.cpp and single_rx.cpp
- Run Matlab in your own machine

USRP Server

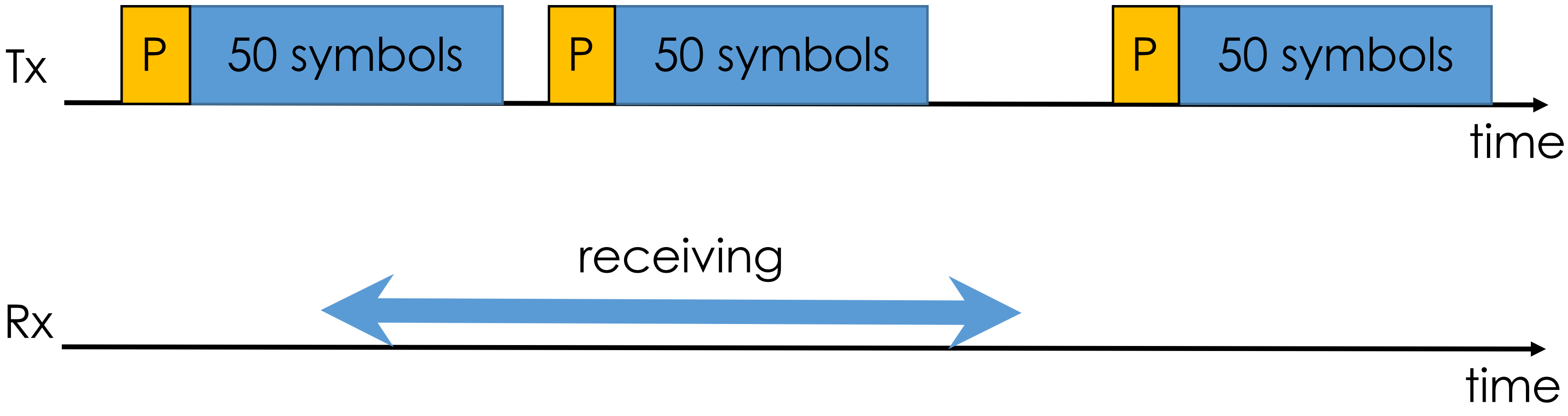
- ssh wcs-g#@140.113.203.6
  e.g., wcs-g1@140.113.203.6
  default password:
- USRP IP
  - Tx: 192.168.10.2 (connecting to eth0)
  - Rx: 192.168.20.2 (connecting to eth1)
- HW code put in **~/uhd/host/example**
  - single_tx.cpp/ single_tx.h
  - single_rx.cpp/ single_rx.h
- cd ~/uhd/host/build
- cmake .. (only the first time)
- make
- cd example (now in ~/uhd/host/build/example)

USRP Server

- mkdir wcs_trace
- Transimitter :
  - ./single_tx --f=2.49 --i=128
- Receiver:
  - ./single_rx --f=2.49 --i=128
- Received data in ./wcs_trace/rx_signals.bin

TODO

- Tx repetitively sends 50 symbols
  - USE_WARPLAB_TXRX = 0 to see the simulation result
  - Set MOD_ORDER = 2 to use BPSK modulation
- Rx receives at least one batch of 50 symbols
- Matlab offline decoding



Task 1: OFDM Symbol Generator

- modify your lab1 code: signal_gen.m
  - Change number of symbol to 50
  - Remove the codes related to ”interpolate”
  - signal_gen.m outputs
    - transmitted digital bits to tx_data.bin
    - transmitted frequency-domain samples to tx_syms_mat.bin

Task 2: USRP Transmitter

- Login to the testbed (page 12)
- Compile the example code and test (page.13)
- Sample code provided by the TA
  - Transmit on 2.49GHz
  - Please check the IP before transmission
    - Command: uhd_find_device
  - Launch the transmitter (USRP_TX) first
  - ./single_tx --f=2.49 --i=128
- TODO (single_tx.cpp/ single_tx.h)
  - Modify single_tx.cpp/ singal_tx.h to transmit the message you just generated
  - Use a while loop in Tx to continuously send batches

Task 3: USRP Receiver

- Sample code provided by the TA
  - Receive the upcoming signal
  - Save the data at wcs_trace/recv_singal.bin
  - Launch single_rx after single_tx
  - ./single_rx --f=2.49 --i=128
  - Press ^C to terminate after the receiver finishes receiving

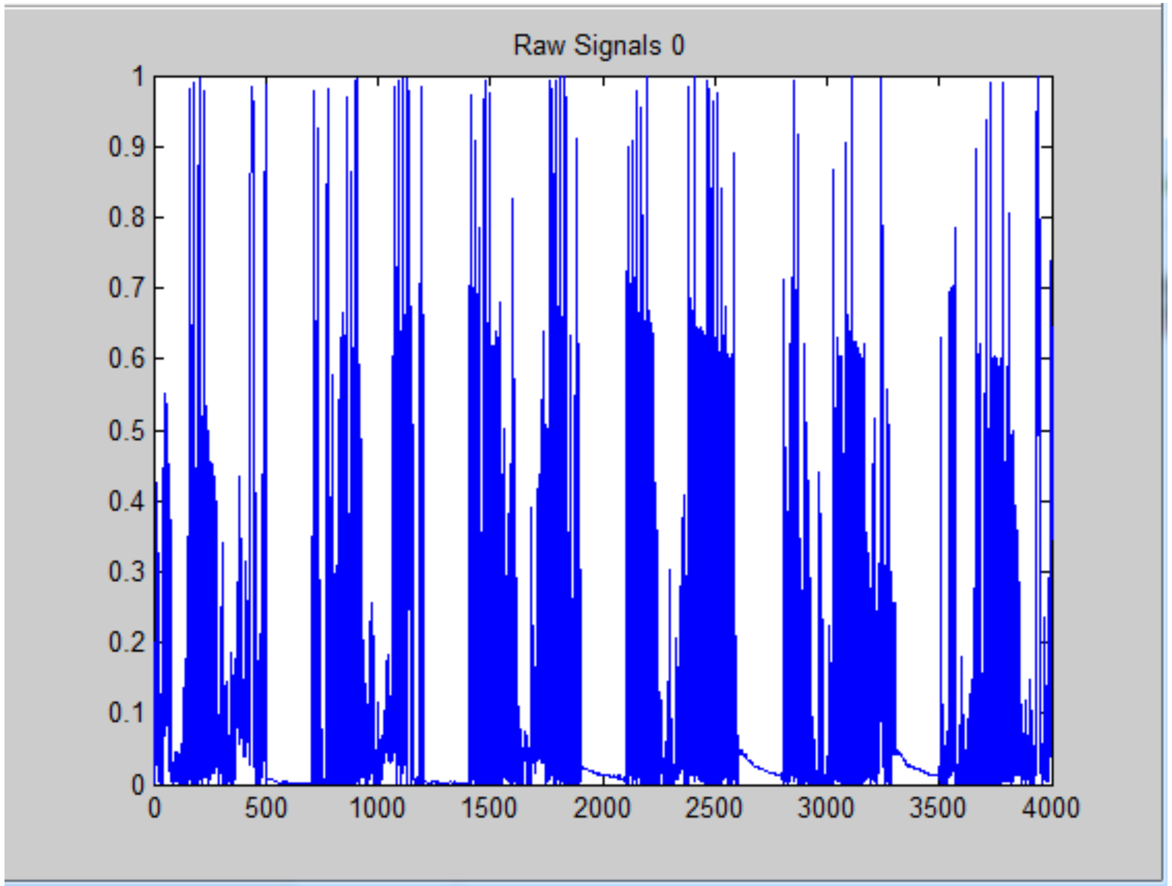
- TODO (single_rx.h)
  - Modify single_rx.h to ensure at least receiving one batch of 500 symbols for offline decoding

Task 4: Matlab Decoding

- Download wcs_trace/recv_singal.bin
- Read the above received signals to your lab1 decoder
- Remove "decimate"
  - raw_rx_dec = filter(interp_filt2, 1, rx_vec_air);
  - raw_rx_dec = raw_rx_dec(1:2:end);
- The most difficult part should be *packet detection*
  - Visually check whether the detected packet index actually matches the location of a preamble
  - If you cannot find the location of preamble correctly, try to adjust the parameter "LTS_CORR_THRESH" and see if detection can be successful
    - (default LTS_CORR_THRESH = 0.8)

TA's sample code

- Send random integers
- Plot of the abs of signals in r_signal.bin

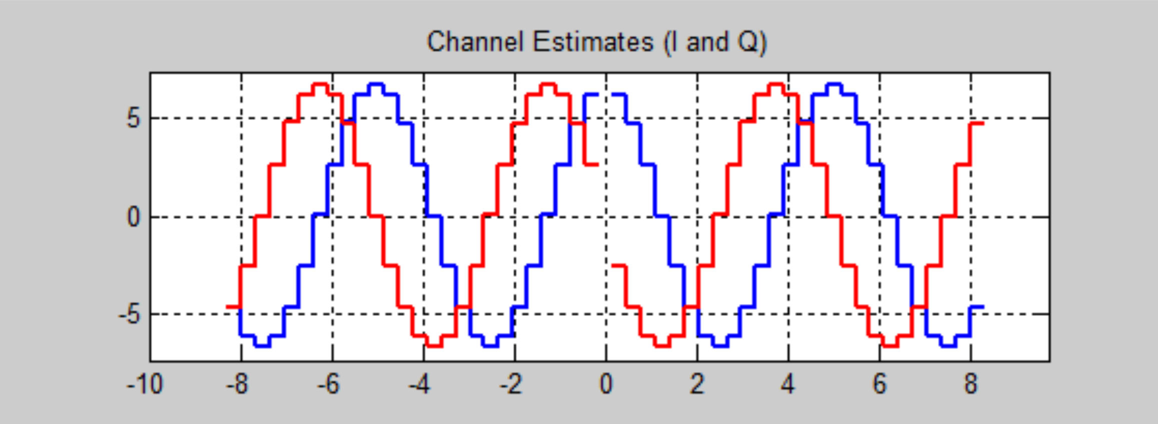


Task 5: Results

- Plot the figures
- Calculate the SNR and BER

Required figures

- Figure 1: Channel Estimation H[k] (WARP figure 4-1)

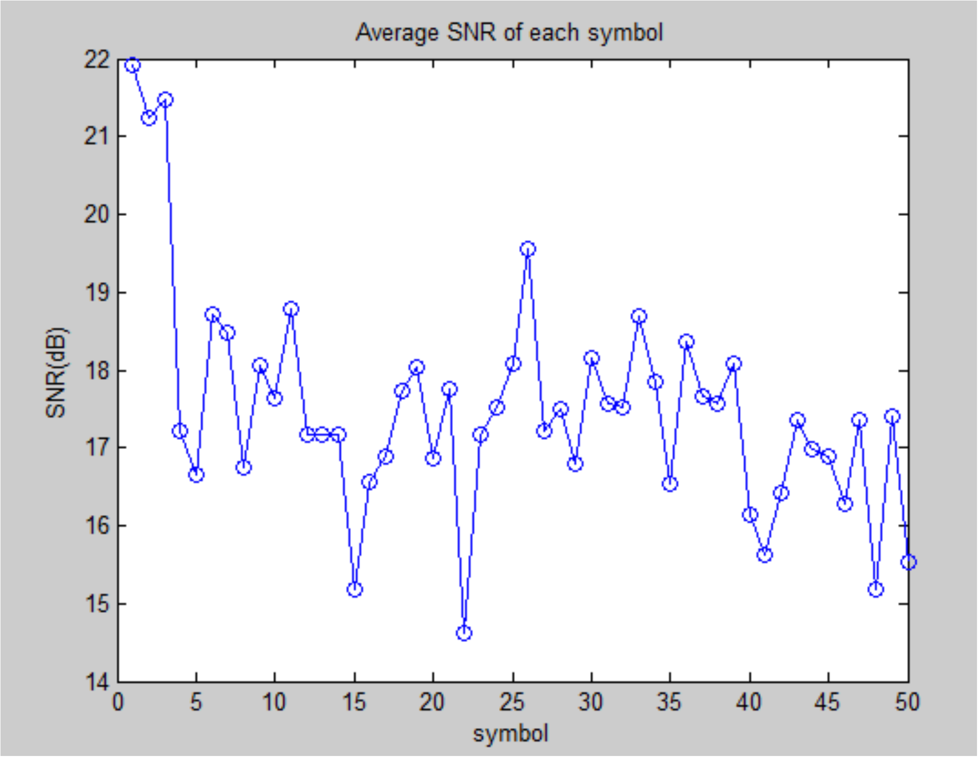


Required figures

- Figure 2: subcarrier SNR
  - average SNR of each data subcarrier among all symbols (bar graph)
  - With and without phase track
- Observation
  - Check if there exists deep fading

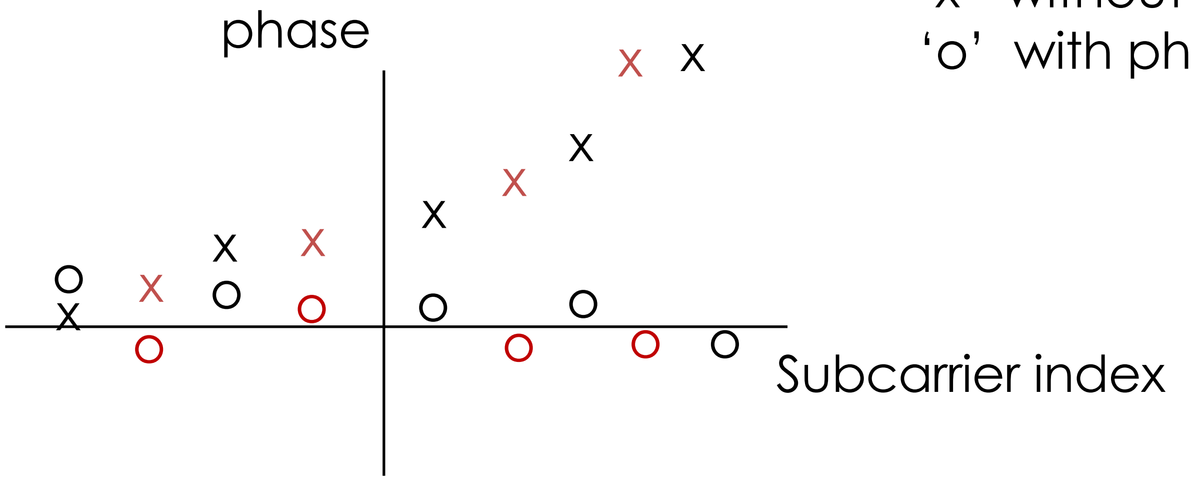
Required figures

- Figure 3: symbol SNR
  - average SNR of all subcarriers for symbols over time (line graph or scatter plot)
  - With and without phase track
- Observation
  - Check if SNR drops over time if phase track is disabled



Required figures

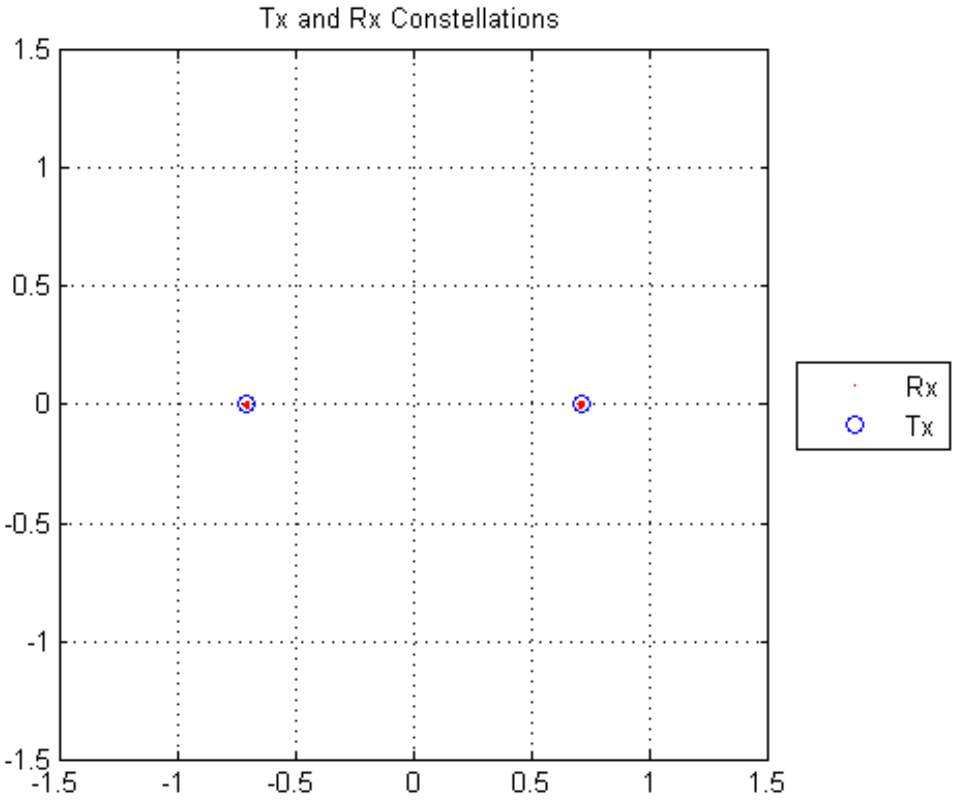
- Figure 4: Phases of decoded signal of different subcarriers in the first symbol
  - with and without phase track



'x' without phase tracking
'o' with phase tracking

Required figures

- Figure 5: constellation points (WARP figure 6)



Grading

- Tx/Rx: 30%
- decode.m: 40%
    - Each figure: 8%
- Report: 20%

Code Submission

- Deadline: Apr. 17 (Tue.) 23:59
- Submit to E3
  - source code: signal_gen.m, decode.m, single_tx.cpp, single_tx.h, single_rx.cpp, single_rx.h
  - Report (.pdf): include all figures and your discussion/ovservation