UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

OPTIMAL ANTENNA ARRAY SPACING IN THE UPLINK FOR DIFFERENT SMALL CELL SCENARIOS

A Degree's Thesis Submitted to the Faculty of the Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB)
at the

Universitat Politècnica de Catalunya (UPC)

in partial fulfillment of the requirements for the degree in Telecommunication Systems Engineering

by

Mario Gómez Arias

Supervisor: Jordi Romeu Robert
Professor

Barcelona, July 2014

Abstract

In this thesis, a testbed for MIMO channel measurements is designed, implemented and operated in different scenarios with line-of-sight (LOS) and non-line-of-sight (NLOS) propagation characteristics. We consider a 2x4 MIMO system with space and polarization diversity. Channel capacity for particular locations and antenna spacings is obtained using the eigenvalue decomposition of the measured MIMO channel matrix. This method estimates the number of independent channels between the transmitter and the receiver. This is particularly interesting in a rich scattering environment where fading may considerably degrade the performance of the system and as a result diversity techniques are employed. However, space diversity, where antennas are physically separated to obtain independent propagation paths, may be limited due to space constraints. This is the case of communication equipment for small cells. Thus the use of polarization diversity may be a more promising solution to obtain more compact antenna arrays. Finally, we demonstrate that for static channels even small changes in antenna spacings result in a change on capacity.

Resum

En aquesta tesi, s'ha dissenyat, implementat i operat en diferents escenaris una plataforma de proves per prendre mesures de canal MIMO. Es considera un sistema MIMO 2x4 amb diversitat en espai i polarització. La capacitat del canal és obtinguda per a unes localitzacions concretes a partir de la descomposició en valors singulars de la matriu de canal MIMO mesurada. Aquest mètode estima el nombre de canals independents entre el transmissor i el receptor. Això és especialment interessant en un entorn amb moltes reflexions on els esvaïments poden degradar considerablement el sistema i com a resultat s'utilitzen tècniques de diversitat. No obstant, la diversitat en espai, a on les antenes es troben físicament separades per obtenir camins de propagació independents, es pot veure limitada per restriccions d'espai. Aquest és el cas dels equips de comunicacions per a small cells. Com a resultat, l'ús de diversitat en polarització pot ser una solució més interessant per a obtenir arrays d'antenes més compactes. Finalment, es demostra que per a canals estàtics petits canvis en l'espaïat entre antenes modifiquen la capacitat.

Resumen

En esta tesis, se ha diseñado, implementado y operado en diferentes escenarios una plataforma de pruebas capaz de tomar medidas de canal MIMO. Se considera un sistema MIMO 2x4 con diversidad en espacio y polarización. La capacidad del canal es obtenida para unas localizaciones concretas a partir de la descomposición en valores singulares de la matrix de canal MIMO. Este método estima el numbre de canales independientes entre el transmisor y el receptor. Esto es especialmente interesante en un entorno con muchas reflexiones donde los desvanecimientos pueden degradar considerablemente el sistema y como resultado se emplean técnicas de diversidad. No obstante, diversidad en espacio, donde las antenas se encuentran físicamente separadas para obtener caminos de propagación independientes, puede verse limitada por restricciones de espacio. Este es el caso de los equipos de comunicación para small cells. Como resultado el uso de diversidad en polarización puede ser una solución más interesante para obtener arrays de antenas más compactas. Finalmente, se demuestra que para canales estáticos pequeños cambios en el espaciado entre antennas modifican la capacidad.

Acknowledgments

First of all, I would like to thank Professor Jordi Romeu for accepting my topic proposal and for his valuable support during the development of this thesis.

Also, I would like to thank Professor Sebastian Blanch, Joaquim Giner and Albert Marton for their advice and technical support.

And last but not least, I would like to thank Margarita, Hipólito, Diego and Martina for their unconditional love and support over all these years.

Revision history

Revision	Date	Purpose
1.0	04/07/2014	Preliminary results
2.0	10/07/2014	Corrected version from supervisor's comments
2.1	11/07/2014	Minor corrections

Document distribution list

Name	E-mail
Jordi Romeu	romeu@tsc.upc.edu

Written by:
Name Mario Gómez Arias
Date 11/07/2014

Reviewed and approved by:
Name Jordi Romeu Robert
Date 11/07/2014

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation

The anticipated growth in mobile data traffic represents a challenge for mobile network operators. It is generally agreed that the increase of the existing cell sites by means of *small cells* is an effective way to improve capacity. Small cell sites are expected to be located mainly outdoors separated by 50-300 m and at about 3-6 m above street level [1]. Since the environment seen from the mobile station (MS) is similar to that seen from the base station (BS), the techniques to mitigate multipath may differ from those used in traditional macro sites. *Space diversity* has been traditionally used at the BS to reduce fading. However, implementing separated receive antennas may be constrained. The use of *polarization diversity* may be a solution to obtain more compact antenna arrays. Thus communication equipment for the small cell scenario is anticipated to be considerably reduced. Then, equipment with integrated antennas that is mounted either on a wall or a lamppost is more attractive in this sense (Fig. 1-1).

(a) (b)

Figure 1-1: Small cell equipment. (a) Huawei AtomCell. Source: `http://www.huawei.com/minisite/mwc2012/en/solutionhighlights/detail24.html`. (b) Alcatel-Lucent lightRadio™9760 Metro Cell. Source: `http://www.launch3telecom.com/news/attsmall2015/`

1.2 Scope

This work demonstrates *experimentally* the effect of antenna array spacing on channel capacity. A testbed is designed, implemented and operated in different small cell scenarios with line-of-sight (LOS) and non-line-of-sight (NLOS) propagation characteristics. We consider a 2x4 MIMO system with polarization and space diversity at the BS. However, we constrain space diversity to illustrate typical deployment issues. Finally, the eigenvalue decomposition of the channel matrix serves as a benchmark to compare results. This method estimates the number of independent channels between the MS and the BS. Channel capacity can be then derived from the measured channel matrix.

The main goals are:

- To design and characterize an antenna consisting of a dual-polarized patch antenna.
- To design, implement, characterize and operate a testbed able to perform channel measurements.
- To observe the effect of antenna array spacing on capacity.

The project is carried out at the AntennaLab research group of the Teoria del Senyal i Comunicacions (TSC) department. This project was originally proposed by the author and discussed further in detail with the supervisor. Prof. Jordi Romeu made reference to the work of J.A. Valdesueiro [2]. It considered the modeling and experimental validation of wave propagation in subway tunnels with multiple antenna elements and the impact of different transmit/receive antenna configurations on the capacity.

Finally, this project was motivated by the research on antenna technologies for small cell communication equipment previously carried out by the author in an internship at Huber+Suhner, Switzerland.

1.3 System Requirements and Specifications

The system shall:

1. Be composed of 2 co-located, dual-polarized transmit antennas.
2. Be composed of 4 receive antennas. Specifically, the receiver subsystem shall:
   - 2.1. Be composed of 2 horizontally-separated, dual-polarized antennas.
   - 2.2. Be able to vary the horizontal spacing between antennas.
   - 2.3. Be implemented by means of software-defined radio.
   - 2.4. Be controlled by a software interface.
   - 2.5. Be able to acquire the signals coherently on all antenna ports.
3. Collect and store data for later postprocessing.
4. Demonstrate the effects of antenna array spacing in the uplink for different scenarios. Specifically,
   - 4.1. Demonstrate implies operating a testbed system with all subsystems working properly.
   - 4.2. A testbed shall be operated in two scenarios, namely:
     a. An scenario where multipath and non-line-of-sight (NLOS) dominate.
     b. An scenario where line-of-sight (LOS) dominates.
5. Be designed, built, tested and fully operational by July 2014.

System specifications are summarized in Table (1.1).

Subsystem	Requirement	Specification
Receive antenna	Frequency	2.45 GHz
	Type	Dual-polarized, microstrip-fed patch antenna
	Array layout	Two horizontally-separated (up to 2λ), dual-polarized ($\pm$ 45º) antennas
	Isolation	>20 dB
	Bandwidth	>50 MHz
	Return loss	>15 dB
	Cross-polarization discrimination	>25 dB
Transmitter	Setup	Co-located dual-polarized (0º/90º) antenna with R&S®SMB100A RF signal generator
Receiver	Setup	Array placed in an aluminum guide to allow horizontal displacement. Two Universal Software Radio Peripherals (USRP) with four XCVR2450 daugtherboards/transceivers.

Table 1.1: System specifications.

1.4 Work Packages, Tasks and Milestones

Table 1.2 contains the definitive work packages (WPs) and tasks of this project. Similarly, Table 1.3 contains the milestones and deliverables. Figure 1-2 shows the Gantt diagram.

The incidences and deviations from the initial proposal have been:

- The review of ray-tracing techniques has been omitted due to time constraints. Instead, a more extensive review of channel models has been done.
- Several problems with synchronization and frequency stability of the USRPs were encountered. This resulted in an overall delay in the development of the project.
- The effect of antenna array separation in channel capacity has been reduced to particular cases (i.e., for *static* channels only).

WP1 - Project management		
Tasks related with project management	**Planned start date:**	26/02/2014
	Planned end date:	11/07/2014
Internal tasks:	**Deliverables:**	**Dates:**
1. Requirements specification	Project proposal	10/03/2014
2. WPs maintenance	Critical review	23/04/2014
	Final report	11/07/2014

WP2 - Information gathering		
Review of literature and other resources	**Planned start date:**	26/02/2014
	Planned end date:	20/05/2014
Internal tasks: 1. Review of literature	**Deliverables:** None	**Dates:** –

WP3 - Antenna subsystem		
Design, manufacturing and characterization of antenna subsystem	**Planned start date:**	26/02/2014
	Planned end date:	07/04/2014
Internal tasks: 1. Design using Ansoft HFSS 2. Manufacturing 3. Characterization	**Deliverables:** Final report	**Dates:** 11/07/2014

WP4 - Receiver subsystem		
Implementation of receiver subsystem	**Planned start date:**	07/04/2014
	Planned end date:	20/06/2014
Internal tasks: 1. Configuration GNU Radio 2. Hardware modifications	**Deliverables:** Final report	**Dates:** 11/07/2014

WP5 - Experimental validation		
Testbed and measurement campaigns	**Planned start date:**	23/06/2014
	Planned end date:	30/06/2014
Internal tasks: 1. Setup measurement testbed 2. Measurement system validation 3. Measurement campaign scenario 1 4. Measurement campaign scenario 2 5. Data analysis	**Deliverables:** Final report	**Dates:** 11/07/2014

Table 1.2: Working packages (WPs) and tasks.

WP#	**Task#**	**Description**	**Milestone (M)/deliverable (D)**	**Date (week)**
1	1	Requirements specification	Project proposal (D)	10/03/2014
1	2	WPs maintenance	Critical review (D)	23/04/2014
3	1-3	Antenna subsystem	Antenna subsystem ready (M)	07/04/2014
5	1	Setup measurement testbed	Testbed ready (M)	23/06/2014
All	All	Project completed	Final report (D)	11/07/2014

Table 1.3: Milestones and deliverables.

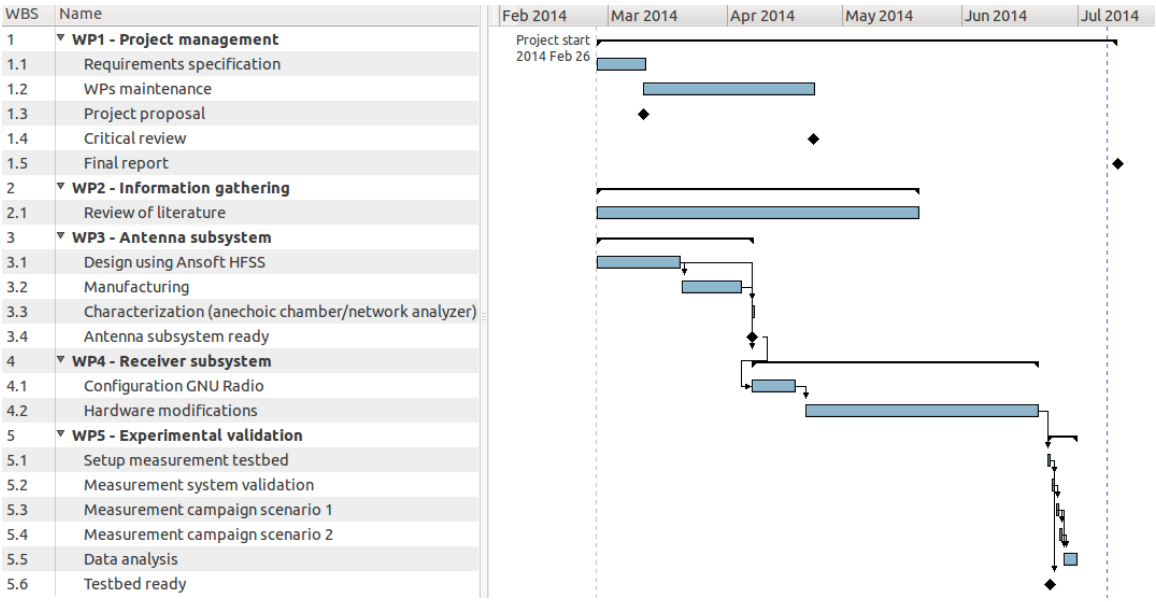

Figure 1-2: Gantt diagram.

2

State of the Art

2.1 System Model

Multiple-input multiple-output (MIMO) systems, where multiple antennas can be used to increase data rates through *multiplexing* or to improve system performance through *diversity*, have been a topic of interest in wireless communications. A *multiplexing gain* can be achieved when the MIMO channel matrix can be decomposed into independent channels resulting in an increase in spectral efficiency [3], [4]. However, to achieve these spectral efficiency gains, accurate knowledge of the channel matrix is required. This is commonly referred to as channel side information (CSI). Channel side information at the receiver (CSIR) occurs when this information is available only at the receiver. If a feedback path to the transmitter is available, the receiver can send the information back to the transmitter and then provide channel side information at the transmitter (CSIT). When the channel is not known to either the transmitter or the receiver, a zero-mean spatially white (ZMSW) model is considered, where the entries are assumed to be i.i.d. zero-mean, unit-variance, complex circularly-symmetric Gaussian random variables [5].

2.1.1 Narrowband Multiple-Input Multiple-Output (MIMO) Model

A discrete-time model describing a narrowband[1] communication system of M transmit and N receive antennas is the following:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} h_{11} & \dots & h_{1M} \\ \vdots & \ddots & \vdots \\ h_{N1} & \dots & h_{NM} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix} + \begin{bmatrix} n_1 \\ \vdots \\ n_N \end{bmatrix} \tag{2.1}$$

or in compact matrix notation:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{2.2}$$

where $\mathbf{x}$ represents the transmitted symbol, $\mathbf{n}$ is the noise vector, and $\mathbf{H}$ is the $N \times M$ channel matrix with $[h_{ij}]$ representing the channel gain from transmit antenna j to receive antenna i. We assume complex Gaussian noise with i.i.d. components and identical power at each receiver, i.e., $\mathrm{E}\left\{\mathbf{n}\mathbf{n}^H\right\} = \sigma^2\mathbf{I}_N$. The total transmitter power is constrained to $\mathrm{E}\left\{\mathbf{x}\mathbf{x}^H\right\} = P_t$. Finally, the average signal-to-noise ratio (SNR) at each receiver branch is

[1] The channel is assumed to be flat over the frequency band of interest.

$\rho = P_t/\sigma^2$.

2.1.2 Parallel Decomposition of the MIMO Channel

The MIMO channel in Eq. (2.2) can be decomposed into a number of parallel independent channels. As a result, we can send independent data across each of the parallel channels. This is referred to as *multiplexing gain*.

To obtain these independent channels, suppose that $\mathbf{H}$ is an $N \times M$ matrix of rank k. We can obtain its singular value decomposition (SVD) as [5]:

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{2.3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are two unitary matrices[2] and $\mathbf{\Sigma}$ is an $M \times N$ diagonal matrix of singular values $\{\sigma_i\}$ of $\mathbf{H}$. These singular values have the property that $\sigma_i = \sqrt{\lambda_i}$, for λ_i the ith eigenvalue of $\mathbf{H}\mathbf{H}^H$, and k of these singular values are nonzero. Because k cannot exceed the number of columns or rows of $\mathbf{H}$, $k \leq \min(M, N)$. If $\mathbf{H}$ is full rank, then $k = \min(M, N)$, which is referred to as a *rich scattering environment*.

Transmit precoding and *receiver shaping*[3] transforms the MIMO channel into k parallel independent channels, where the ith channel has channel gain σ_i. However, this requires knowledge of the channel matrix at both trasmitter and receiver.

2.1.3 MIMO Channel Capacity

The capacity of a MIMO channel is the maximum data rate that can be transmitted over the channel with some error probability. Capacity depends on the knowledge of the channel matrix, i.e., whether its *distribution* or a *realization* of it is known at either the transmitter or the receiver. We assume CSIR only.

Ergodic capacity is used to characterize capacity in time-varying channels [5]. With CSIR and no CSIT, the ergodic capacity is given by:

$$C = \mathrm{E}\left\{\log_2 \det\left[\mathbf{I}_N + \frac{\rho}{M}\mathbf{H}\mathbf{H}^H\right]\right\}, \text{ [bits/Hz]} \tag{2.4}$$

For the special case $\mathbf{H} = \mathbf{I}_n$:

$$C = n\log_2\left(1 + \frac{\rho}{n}\right) \to \rho/\log(2) \text{ as } n \to \infty \tag{2.5}$$

i.e., capacity scales linearly rather than logarithmically with the number of parallel channels or number of antenna array elements, n. Moreover, in the low SNR regime [26]:

$$C \approx \log_2\left(1 + \frac{\rho}{M}\|\mathbf{H}\|^2\right) \tag{2.6}$$

i.e., for low SNR values the capacity of a MIMO channel is equivalent to that of a single-input single-output (SISO) channel with power gain $\|\mathbf{H}\|^2$, where $\|\mathbf{H}\|^2 = \sum_{i=1}^{M}\sum_{j=1}^{N}|h_{ij}|^2$ is the Frobenius norm.

[2] $\mathbf{U}$ and $\mathbf{V}$ unitary imply that $\mathbf{U}^H\mathbf{U} = \mathbf{I}_N$ and $\mathbf{V}^H\mathbf{V} = \mathbf{I}_M$.

[3] In transmit precoding the input $\mathbf{x}$ to the antennas is generated by a linear transformation on input vector $\tilde{\mathbf{x}}$ as $\mathbf{x} = \mathbf{V}\tilde{\mathbf{x}}$. Receiver shaping performs a similar operation at the receiver by multiplying the channel output $\mathbf{y}$ by $\mathbf{U}^H$.

The expresion in Eq. (2.4) can be written in terms of the eigenvalues $\lambda_1, \ldots, \lambda_k$ of $\mathbf{H}\mathbf{H}^H$, with $k = \min(M, N)$, as [4]:

$$C = \mathrm{E}\left\{\sum_{i=1}^{k} \log_2\left(1 + \frac{\rho}{M}\lambda_i\right)\right\}, \text{ bits/Hz} \tag{2.7}$$

From Eq. (2.7) it can be seen that capacity grows linearly with the number of channel eigenvalues. Equivalently, the ith term in the right side of Eq. (2.7) is the capacity of the ith independent parallel channel after its decomposition.

Optimization of the Capacity of Multi-Antenna Gaussian Channels

From Eq. (2.7), it can be seen that capacity depends on the eigenvalues which in turn depend on $\mathbf{H}$. Chiurtu et al. [6] discuss the problem of choosing $\lambda_1, \ldots, \lambda_k$ for $k \leq \min(M, N)$ so as to maximize the capacity expression

$$C = \sum_{i=1}^{k} \log_2\left(1 + \frac{P_i}{\sigma^2}\lambda_i\right) \tag{2.8}$$

under the following constraints:

$$\begin{cases} \sum_{i=1}^{k} P_i \leq P & \text{(power constraint)} \\ \sum_{i=1}^{k} \lambda_i = L & \text{(channel constraint)} \end{cases}$$

Equation (2.8) can be maximized using only the power constraint. Solving the optimization leads to a *water-filling* power allocation [6], [5]. However, a joint optimization problem is proposed in [6] where the channel constraint reflects that the number and values of the $k \leq \min(M, N)$ nonzero eigenvalues can vary depending on the the antenna locations while the sum is constant. This problem can be solved using Lagrange multipliers by differentiating

$$\begin{aligned} J(P_1, \cdots, P_n, \lambda_1, \cdots, \lambda_n) = &\sum_{i=1}^{n} \log_2(1 + \frac{P_i}{\sigma^2}\lambda_i) \\ &+ a\sum_{i=1}^{k} P_i + b\sum_{i=1}^{k} \lambda_i \end{aligned} \tag{2.9}$$

with respect to P_i and λ_i.

For a set of $k \leq \min(M, N)$ nonzero λ_i and the corresponding P_i, the unique solution of the above maximization problem is:

$$\begin{cases} P_1 = P_2 = \cdots = P_k = \dfrac{P}{k} \\ \lambda_1 = \lambda_2 = \cdots = \lambda_k = \dfrac{L}{k} \end{cases}$$

and the resulting capacity after optimization is:

$$C_k = k\log_2\left(1+\frac{LP}{k^2\sigma^2}\right) \tag{2.10}$$

Finally, the number k^* of parallel channels to achieve the maximum is obtained by solving:

$$C_{\max} = \max_k C_k = \max_{1\leq k\leq \min(M,N)} k\log_2\left(1+\frac{LP}{k^2\sigma^2}\right) \tag{2.11}$$

The number k^* is not necessarily $k=\min(M,N)$. The solution to Eq. (2.11) is $k^* = 0.504976\sqrt{a}$ where $a = LP/\sigma^2$ and this solution is unique.

2.1.4 MIMO Diversity Gain: Beamforming

So far multiple antennas have been regarded as a technique to achieve a *capacity* gain. However, they can be used to obtain array and diversity gain instead. This scheme, where the same symbol is weighted and sent over each transmit antenna, is referred to as *MIMO beamforming*.

If precoding and shaping with column vectors (see Section 2.1.2) is performed, the received signal is given by

$$y = \mathbf{u}^H\mathbf{H}\mathbf{v}x + \mathbf{u}^H\mathbf{n} \tag{2.12}$$

i.e, antennas at the receiver combine coherently the received signal.

2.1.5 Diversity-Multiplexing Trade-offs

Previous sections suggest whether multiple antennas in MIMO systems should be intended for diversity gain, multiplexing gain or both. In *block fading channels* with CSIR only, full diversity gain and multiplexing gain can be obtained simultaneously with coding techniques (e.g., D-BLAST encoding, [5]). However, for finite coding blocklengths it is not possible to achieve full diversity and full multiplexing gain simultaneously, in which case there is a trade-off between these gains [7].

2.1.6 Review of Literature

The performance of *polarization diversity* schemes in small and micro cells at 1800 MHz was studied in [8]. The effect of cross-polarization discrimination (XPD), signal cross correlation and polarization diversity gain with horizontally and vertically-polarized antennas was evaluated. The performance of this later diversity scheme was compared with $\pm 45^{\text{o}}$ polarization and horizontal space diversity. The receive antenna consisted in a dual-polarized antenna with 28 dB of XPD. A signal generator was transmitting a continuous-wave of 1.0 W. Measurements showed that XPD values for horizontal and vertical polarizations vary between 5-15 dB and depend on the propagation path. Moreover, signal cross correlation is below 0.7. Diversity gain could be achieved in line-of-sight (LOS) and non-line-of-sight (NLOS) conditions. Moreover, $\pm 45^{\text{o}}$ polarization diversity is preferable to horizontal/vertical polarizations. Other diversity schemes for different antenna configurations have been studied in [9]-[11].

Similarly, a MIMO channel model for vertically-polarized and dual-polarized transmit antennas with dual-polarized receive antennas is proposed in [13]. The channel is decom-

posed into a LOS and NLOS component related with the Ricean K-factor as

$$\begin{aligned}\mathbf{H} &= \sqrt{\frac{K}{K+1}}\mathbf{H}_{\mathrm{F}} + \sqrt{\frac{1}{K+1}}\mathbf{H}_{v} \\ &= \alpha\left(\sqrt{\frac{K}{K+1}}\begin{bmatrix} e^{j\phi_{11}} & e^{j\phi_{12}} \\ e^{j\phi_{21}} & e^{j\phi_{22}} \\ e^{j\phi_{31}} & e^{j\phi_{32}} \end{bmatrix} + \sqrt{\frac{1}{K+1}}\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix}\right)\end{aligned} \tag{2.13}$$

where X_{ij} is a correlated zero-mean unit-variance complex Gaussian random variable of the Rayleigh matrix $\mathbf{H}_v$, α is a scalar related to the XPD, and K is the Ricean K-factor.

The model in [13] is validated with outdoor wideband channel measurements by using a 2x3 MIMO othogonal frequency-division multiplexing (OFDM) commercial system in the 2.5 GHz frequency band. The results show that higher capacity can be achieved with dual-polarization antennas when compared to the single-polarization, especially in locations closer to the base station (BS) where the MIMO channel is dominated by the LOS matrix component.

Finally, the experimental validation of the channel presented in [14] is performed in [15] considering the use of polarization diversity. The measurement set-up consisted of an antenna array at the mobile station (MS) moved along 11.8λ at a frequency of 2.05 GHz. The BS consisted of two dual-polarized $\pm 45^{\mathrm{o}}$ patch antennas with a separation of 3λ orientated at either $\pm 45^{\mathrm{o}}$ or $0^{\mathrm{o}}/90^{\mathrm{o}}$. The MIMO channel model is:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H_{VV}} & \mathbf{H_{HV}} \\ \mathbf{H_{VH}} & \mathbf{H_{HH}} \end{bmatrix} \tag{2.14}$$

where $\mathbf{H_{IJ}}$ is the subchannel matrix considering the polarization I and J at the MS and at the BS, respectively. Results over 89 measurement positions showed that the model reproduces accurately the realistic MIMO channel.

2.2 Channel Measurement Systems

Channel measurements aim to determine the performance of realistic wireless systems [20]. Together with channel modeling, channel measurements can provide information about the frequency, time and polarimetric characteristics of actual channels.

Channel measurement systems have been developed by both research institutions and industry. The commercial RUSK MIMO channel sounding system from MEDAV GmbH performs real-time channel impulse response measurements in the Universal Mobile Telecommunications Systems (UMTS) bands for multiple transmit and receive antenna element configurations [16] (Fig. 2-1).

Similarly, the EURECOM MIMO Open-Air Sounder (EMOS) is an outdoor measurement channel system for multiuser MIMO (MU-MIMO) channels [17]. The BS consists of 2 cross-polarized antennas. Each user equipment (UE) synchronizes to the BS using a synchronization symbol (SCH). The channel is estimated from a broadcast data channel (BCH) sequence consisting of 48 pilot symbols.

Finally, Ericsson Research has investigated the benefits of downlink cooperative MIMO systems in macrocell environments using a coherent measurement system [18], [19]. Three single-antenna BS transmit time-multiplexed OFDM symbols with 432 tones over a bandwidth of 20 MHz at 2.66 GHz. The MS consists of two dipole and two loop antennas mounted

on a van with a spacing of 30 cm. The MS sounds the 4x3 MIMO channel matrices in LOS, obstructed LOS and shadowed scenarios.

Figure 2-1: RUSK MIMO channel sounding system (Receiver Unit).

3

Methodology

3.1 Hardware Considerations

From Section 1.3, the receive antennas shall consist of two dual-polarized ($\pm 45^{o}$) antennas. We consider a 2-port microstrip-fed single radiating element similar to the Octogonal Reconfigurable Isolated Orthogonal Element (ORIOL) antenna in [21]. Clearly, polarization diversity can be achieved with a single radiating element. The antenna was simulated using ANSYS HFSS simulation software. HFSS offers different numerical solvers with the possibility to specify geometries and material properties. The substrate is RO4003C with dielectric constant $\epsilon_r = 3.55$, dissipation factor $\tan\delta = 0.0027$ and thickness of 1.524 mm. Figure 3-1 shows the antenna model in HFSS. Simulated and measured scattering parameters (s_{11} and s_{12}) are shown in Fig. 3-2. Measured complex impedance is 45.5 $\Omega \angle$ -0.303 rad = 43.4 - j13.6 Ω at 2.45 GHz. Simulated radiation pattern is shown in Fig. 3-3. Figure 3-4 shows the manufactured antennas. A block of aluminum is attached to the ground plane to allow the horizontal displacement of the antennas easily.

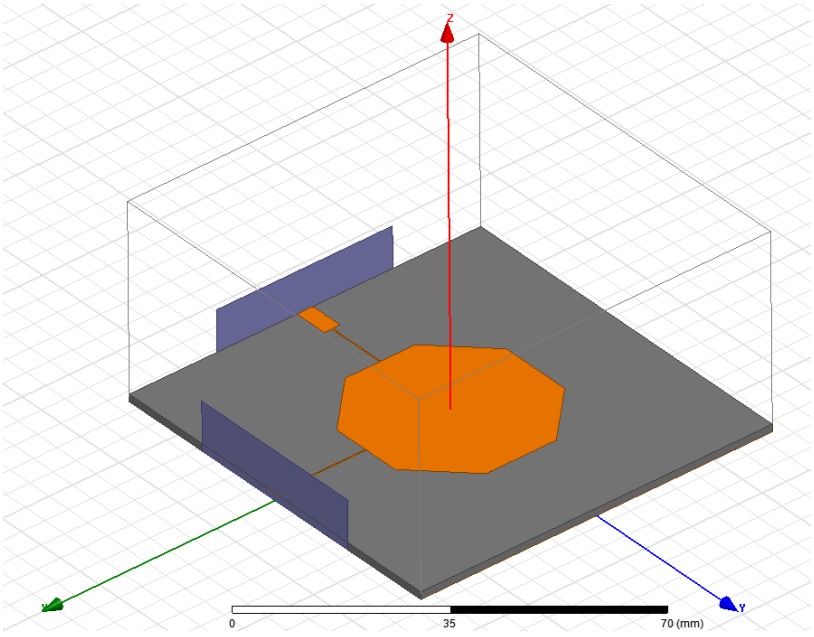

Figure 3-1: Dual-polarized antenna model.

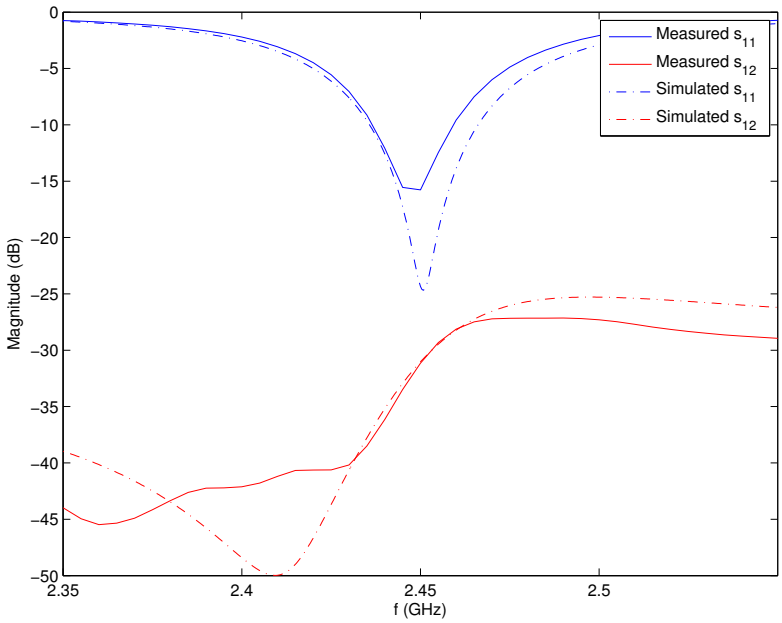

Figure 3-2: Simulated and measured scattering parameters (s_{11} and s_{12}).

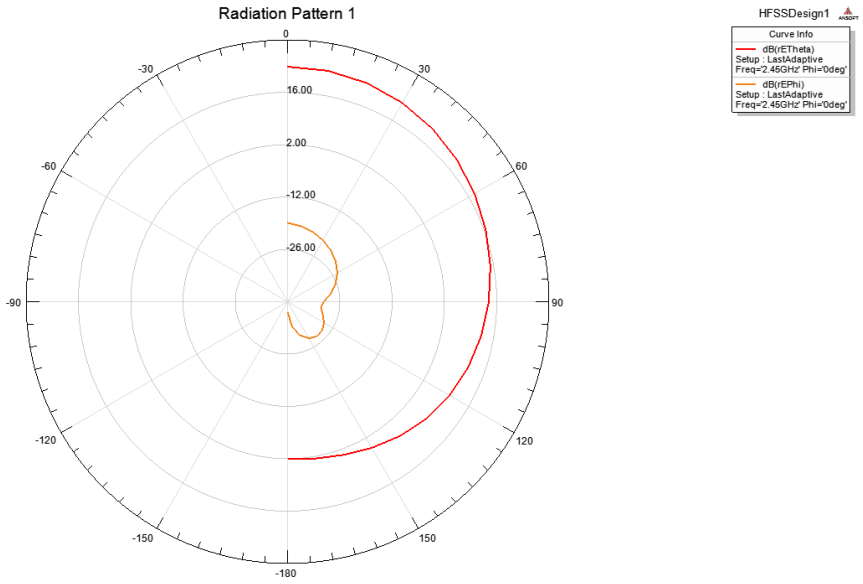

Figure 3-3: Simulated radiation pattern.

The rest of the receiver is a Universal Software Radio Peripheral (USRP) from Ettus Research. The USRP is composed of several analog-to-digital converters (ADCs), digital-to-analog converters (DACs), digital down-converters (DDCs) and digital up-converters (DUCs) that allow baseband signal processing. *Daughterboards* perform the transceiver operations of the USRP. A wide range of daughterboards are available depending on the application and frequency range (DC to 6 GHz). Finally, a field programmable gate array

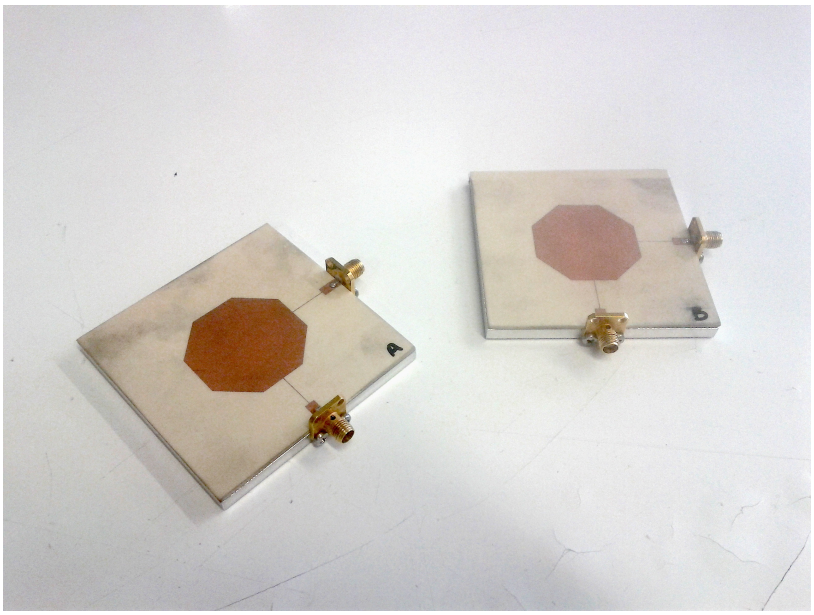

Figure 3-4: Manufactured antennas.

(FPGA) is responsible for conditioning digital data streams between the USRP and a PC.

The available model for this project is the USRP1 with two XCV2450 daugtherboards. Some specifications include 64 MS/s 12-bit ADCs, up to 16 MS/s data rate between the device and the host PC and USB 2.0 connectivity [22]. Each USRP1 includes two complex transmit/receive chains. For that reason two USRP1 were necessary. However, clock and sample synchronization between the two USRP1 are needed in order to build a MIMO system. The hardware modifications in [23] solve this problem by disabling the onboard clock of one of the USRP and using the second one as reference.

3.2 Software Development

The USRP hardware driver (UHD) serves as an interface between the USRP and the user-specific signal processing. GNU Radio eases this signal processing by including the UHD driver. GNU Radio is an open-source sofware development toolkit that provides signal processing operations with *blocks* [24]. The source code is written in Python and C++. Moreover, GNU Radio Companion (GRC) provides a graphical interface for creating signal flowgraphs by concatenating different blocks.

The receiver architecture implemented in GNU Radio is shown in Fig. 3-5. Previous to other functions, each channel is band-pass filtered at an intermediate frequency f_{IF}. A channel implements a digital PLL which outputs a carrier at $f_{IF} + f_0$ with $f_0 << f_{IF}$ the offset frequency due to clock drift of the USRP. This carrier is then the input of the mixers that bring the signal of interest to baseband. A graphical user interface (GUI) that allows the visualization of real-time spectrum was also implemented with GNU Radio. The complete GNU Radio code is in Appendix A.

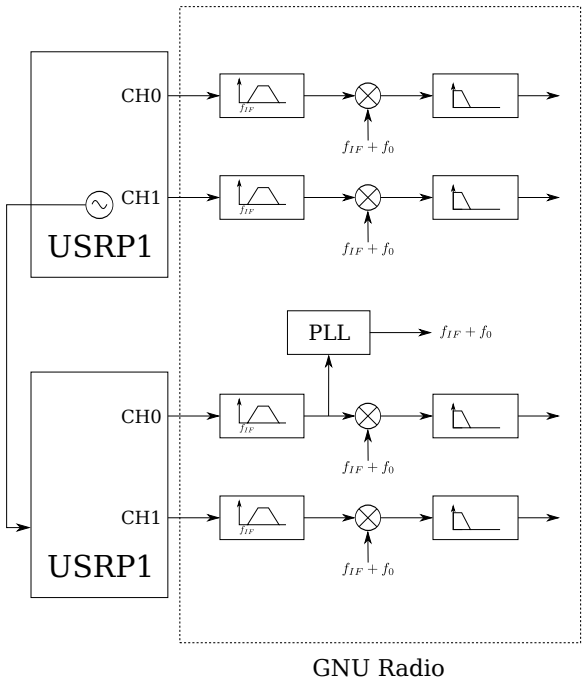

Figure 3-5: GNU Radio receiver implementation.

3.3 Channel Measurements

Measurements were carried out at two scenarios with different propagation characteristics. The effect of antenna array spacing on channel capacity is investigated. Figures 3-6 and 3-7 show the exact locations of these scenarios.

3.3.1 Assumptions

From Section 1.3, the system shall be composed of 2 co-located, dual-polarized transmit antennas transmitting simultaneously. This requires to transmit two orthogonal sequences in order to be able to separate the signals at the receiver. One possibility is to transmit two tones separated in frequency by $\Delta f < B_c$, where B_c is the coherence bandwidth of the channel [2]. We opted instead for performing *static* channel measurements, i.e., to assume that fading effects can be removed by averaging measurements. With this assumption, transmission can occur first in one polarization and then in the other. The eigenvalues resulting from the singular-value decomposition are then inserted into Eq. (2.7) without considering the expectation operator. As a result the transmitter consisted in a signal generator from Rohde&Schwarz (R&S®SMB100A) with a patch antenna transmitting first in vertical polarization and then in horizontal polarization. Time between measurements in vertical/horizontal polarization is 5 min. Slow fading effects (e.g., shadowing by walking pedestrians) can be then achieved by averaging over that time period.

Although we assume *static* channel measurements, the effects of fast fading must be also captured. For that reason, measurements were taken at sampling frequency $f_s = 100$ Hz. This value results from considering the channel coherence time as $\tau \approx .4/f_D$, where f_D is the Doppler shift [5]. Experimental channel measurements at 2.4 GHz indicate that $f_{D\max} \approx 20$ Hz [25]. Then $\tau_{\min} \approx 20$ ms. From Nyquist criterion, $f_s \geq 2/\tau_{\min} = 100$ Hz.

Figure 3-6: Scenario with LOS propagation.

Figure 3-7: Scenario with NLOS propagation.

3.3.2 Comparison and Validation of Results

In order to compare the performance of different antenna configurations in scenarios where transmit power and receive SNR may vary, the channel matrix must be normalized. The expression in Eq. (2.7) already assumes a normalization of the channel matrix $\mathbf{H}$, which is:

$$\|\mathbf{H}\|^2 = M \tag{3.1}$$

i.e., the channel matrix is normalized for a given SNR ρ. Under this normalization, the equivalent power gain for low SNR is (from Eq. (2.6)) N, i.e., the array gain. Different normalization methods of the MIMO channel matrix with their underlying physical meaning are discussed in [26].

Moreover, the channel measurement system needed to be validated to ensure that it accomplishes with its intended requirements. For that reason, a validation test consisting in performing MIMO channel measurements under strict static channel conditions with no reflections was performed in the anechoic chamber (Fig. 3-8). Under these conditions, we expect $\lambda_1 = \lambda_2 = 1$ for any antenna spacing. Figure 3-9 shows the capacity derived from measurements in the anechoic chamber compared to this expected capacity. Measurements and theoretic results are in agreement and the system can be validated.

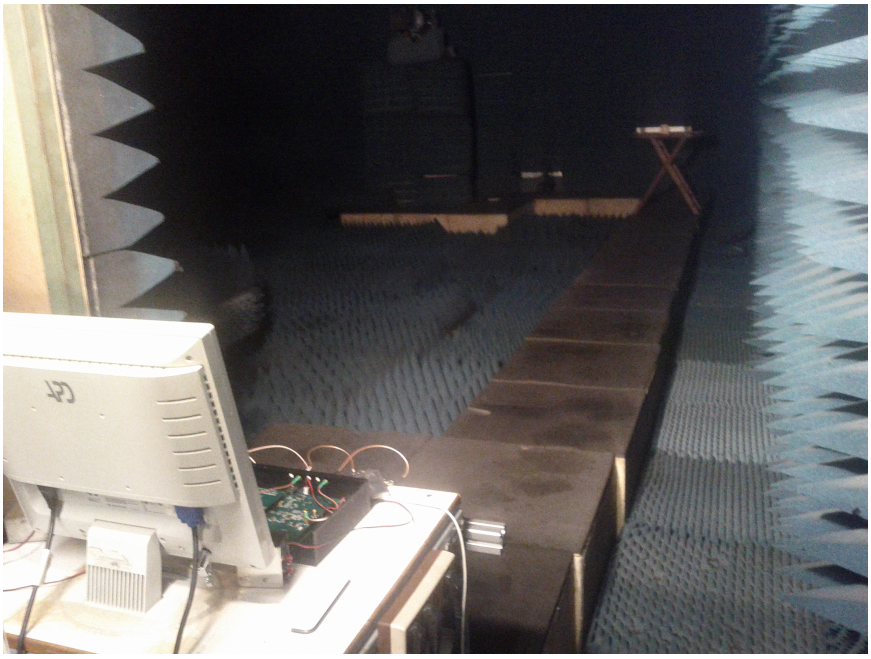

Figure 3-8: Measurement system validation in the anechoic chamber.

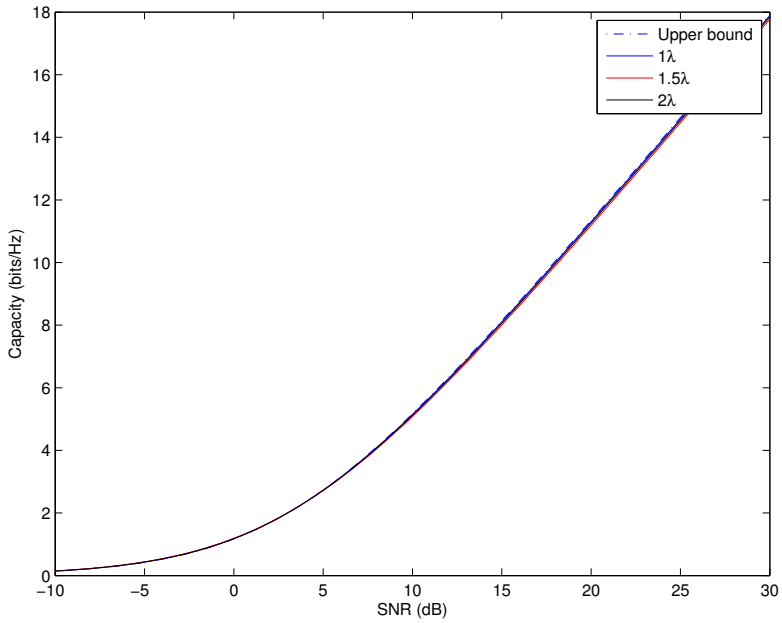

Figure 3-9: Capacity derived from channel measurements during validation.

4

Results

4.1 Channel Measurement System

Figure 4-1 shows the channel measurement system implemented in this thesis. Equipment is placed in a trolley for convenience. An aluminum guide contains the two dual-polarized antennas previously manufactured. The aluminum guide allows the horizontal displacement of the antennas up to 2λ.

Figure 4-1: Channel measurement system.

4.2 Effect of Antenna Spacing on Channel Capacity

Channel measurements were performed in two scenarios with different propagation characteristics. We obtain the MIMO channel capacity from Eq. (2.8) without taking into account the expectation operator. However, as explained in Section 3.3.1, channel matrix coefficients need to be averaged because we assume static channel conditions.

Figures 4-2 and 4-3 show the capacity derived from channel measurements in the LOS and NLOS scenario, respectively. For the LOS scenario in particular, we observe that

capacity is degraded for an array spacing of 2λ compared to λ and 1.5λ. This suggests that, for that geometry in particular between the receiver array, transmitter and scatterers (Figure 3-6), uncorrelated fading paths are already achieved with small horizontal spacings. However, the difference in capacity is only remarkable for SNR higher than 20 dB.

Similarly, for the NLOS scenario, we observe that capacity is degraded for an array spacing of 1.5λ. The differences in capacity are more important as the SNR increases. Finally, we observe again that small variations in the array spacing results in changes in the capacity.

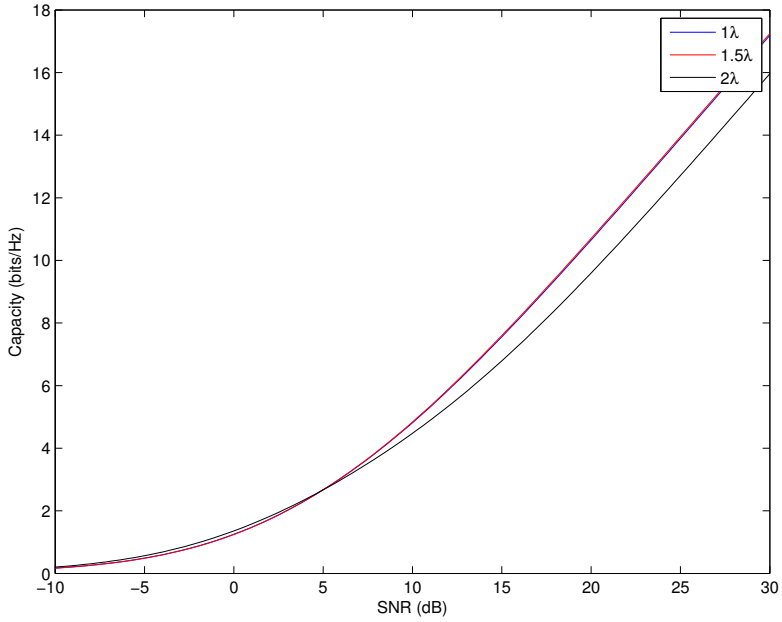

Figure 4-2: Capacity derived from channel measurements for LOS scenario.

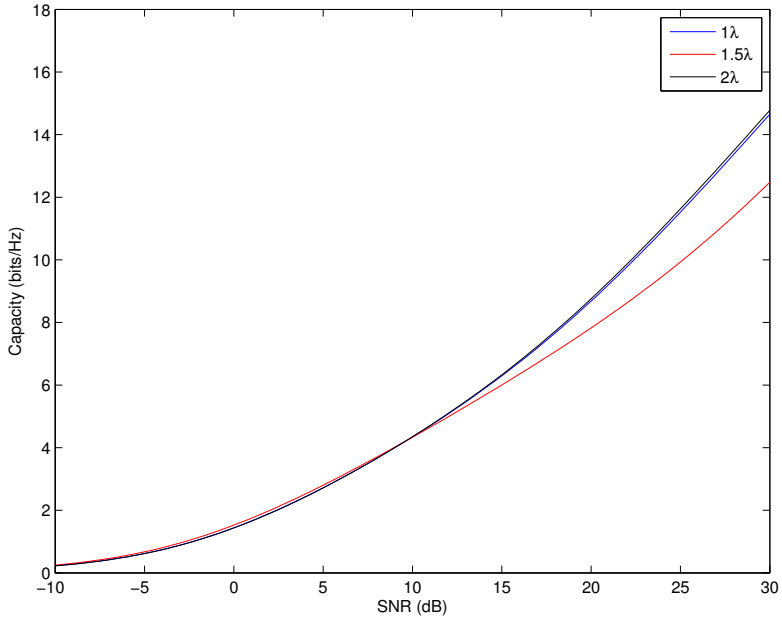

Figure 4-3: Capacity derived from channel measurements for NLOS scenario.

5

Cost Analysis

When estimating the cost of the system, the following assumptions have been made:

- *Long-lived assets* are depreciated according to the straight-line *depreciation* or *cost allocation* method [27]. This is a general accepted accounting principle (GAAP) that assumes an annual *depreciation expense* defined as:

$$\text{Depreciation expense} = \frac{\text{Depreciable base}}{\text{Useful life}} = \frac{\text{Acquisition cost} - \text{Residual value}}{\text{Useful life}} \tag{5.1}$$

  where *useful life* is expressed in years. The *depreciable base* is the amount of cost that is amortized over the asset's useful life. The *book value* is then

$$\text{Book value} = \text{Acquisition cost} - \text{Accumulated depreciation} \tag{5.2}$$

  where *accumulated depreciation* takes into account the actual life of the asset.
- The *residual value* and *useful life* are estimated based on the technical obsolenscence that the assets may have when compared to current technology.
- *Intangible assets* (software) are regarded as *amortizable* instead of *depreciable*.
- *Activities* are derived from the Gantt diagram in Section 1.4. Cost of activities are computed on an hourly-base. The cost of these activites is assumed 10,50 €/hour (*net* hourly wage of a junior engineer)
- Equipment that is not intended specifically for this system (e.g., network analyzer and signal generator) is considered as activities with a cost equivalent to rent it.

ASSETS		Units	Acquisition cost (€)	Useful life (years)	Residual value (€)	Depreciation base (€)	Depreciation expense (€)	Life (years)	Accumulated depreciation (€)	Book value (€)	Total
	Ettus Research USRP1	2	625,00	10	100,00	525,00	52,5	7	367,50	257,50	515,00 €
	Ettus Research XCVR2450	4	360,00	10	100,00	260,00	26,00	7	182,00	178,00	712,00 €
	Desktop PC	1	750,00	5	50,00	700,00	140,00	4	560,00	190,00	190,00 €
	Trolley	1	30,00	15	0,00	30,00	2,00	8	16,00	14,00	14,00 €
	Matlab Student Suite		69,00	5	0,00	69,00	13,80	0	0,00	69,00	69,00 €
MATERIALS											
	Connectors	5	9,50								47,50 €
	Rogers RO4003C substrate	1	35,00								35,00 €
	Other (PVC/aluminum)	1	20,00								20,00 €
ACTIVITIES		**Hours**	**Cost (€/hour)**								
	Project management	80	10,50								840,00 €
	Design, prototyping and simulations	60	10,50								630,00 €
	Implementation of hardware	180	10,50								1.890,00 €
	Implementation of software	40	10,50								420,00 €
	Measurements	40	10,50								420,00 €
	Data analysis	8	10,50								84,00 €
OVERHEAD											
Equipment renting											
	Agilent E5071C	1	20,00								20,00 €
	Rohde&Schwarz SMB100A	5	17,50								87,50 €
	Promax FAC-662B	10	7,00								70,00 €
Labor costs											
	Photoplotting	1	40,00								40,00 €
	Milling	1	20,00								20,00 €
											6.124,00 €

Table 5.1: Costs summary.

6

Conclusions

We have designed, implemented and operated a testbed for static channel measurements. Moreover, the testbed has been validated for static channel measurements. Two dual-polarized antennas have been previously designed, manufactured and characterized. The effect of antenna array spacing on channel capacity has been investigated. The eigenvalue decomposition of the channel covariance matrix has served as a benchmark. We have observed that small perturbations in the horizontal spacing have already an impact on the capacity. Depending on the propagation characteristics and geometry, this effect can be remarkable.

Finally, it is important to notice that the benefits of MIMO systems have to be compared with the cost of deploying multiple antennas with separate RF chains and the required multidimensional signal processing.

6.1 Future Work

Unfortunately, time constraints have limited considerably the original work proposal. The limited number of locations has made impossible to extract an optimal antenna array spacing. Nevertheless, the capabilities of this channel measurement system can be easily extended to obtain more significant results.

We suggest to consider ray-tracing techniques to validate the results presented in Section 4.2. Ray-tracing techniques capture the effects of the geometry between the BS and the MS (including buildings and objects) on received power variations. In addition to that, ray-tracing techniques have been already applied to predict the performance of MIMO systems (e.g., [28], [29]).

Finally, Section 2.1.6 has presented different MIMO channel models that consider transmit/receive polarization diversity. We propose to investigate the applicability of these models to small cell scenarios.

Bibliography

[1] J. Robson, "Small Cell Backhaul Requirements," NGMN Alliance White Paper, June 2012, Accessed: 02/07/2014. [Online]. Available: http://www.ngmn.org/uploads/media/NGMN_Whitepaper_Small_Cell_Backhaul_Requirements.pdf

[2] J. Valdesueiro, B. Izquierdo, and J. Romeu, "On 2x2 MIMO Observable Capacity in Subway Tunnels at C-Band: An Experimental Approach," *Antennas and Wireless Propagation Letters, IEEE*, vol. 9, pp. 1099–1102, 2010.

[3] G. Foschini and M. Gans, "On Limits of Wireless Communications in a Fading Environment when Using Multiple Antennas," *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, 1998. [Online]. Available: http://dx.doi.org/10.1023/A%3A1008889222784

[4] E. Telatar, "Capacity of Multi-Antenna Gaussian Channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, 1999. [Online]. Available: http://dx.doi.org/10.1002/ett.4460100604

[5] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[6] N. Chiurtu and B. Rimoldi, "Varying the Antenna Locations to Optimize the Capacity of Multi-Antenna Gaussian Channels," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 5, 2000, pp. 3121–3123 vol.5.

[7] L. Ordonez, D. Palomar, and J. Fonollosa, "Fundamental Diversity, Multiplexing, and Array Gain Tradeoff Under Different MIMO Channel Models," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 3252–3255.

[8] J. J. A. Lempiainen and J. Laiho-Steffens, "The Performance of Polarization Diversity Schemes at a Base Station in Small/Micro Cells at 1800 MHz," *Vehicular Technology, IEEE Transactions on*, vol. 47, no. 3, pp. 1087–1092, Aug 1998.

[9] P. C. F. Eggers, J. Toftgard, and A. Oprea, "Antenna Systems for Base Station Diversity in Urban Small and Micro Cells," *Selected Areas in Communications, IEEE Journal on*, vol. 11, no. 7, pp. 1046–1057, Sep 1993.

[10] A. M. d. Turkmani, A. Arowojolu, P. A. Jefford, and C. J. Kellett, "An Experimental Evaluation of the Performance of Two-Branch Space and Polarization Diversity Schemes at 1800 MHz," *Vehicular Technology, IEEE Transactions on*, vol. 44, no. 2, pp. 318–326, May 1995.

[11] R. Vaughan, "Polarization Diversity in Mobile Communications," *Vehicular Technology, IEEE Transactions on*, vol. 39, no. 3, pp. 177–186, Aug 1990.

[12] H. Bolcskei, R. Nabar, V. Erceg, D. Gesbert, and A. Paulraj, "Performance of Spatial Multiplexing in the Presence of Polarization Diversity," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, vol. 4, 2001, pp. 2437–2440 vol.4.

[13] V. Eiceg, H. Sampath, and S. Catreux-Erceg, "Dual-Polarization Versus Single-Polarization MIMO Channel Measurement Results and Modeling," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 1, pp. 28–33, Jan 2006.

[14] K. Pedersen, J. Andersen, J. Kermoal, and P. Mogensen, "A Stochastic Multiple-Input Multiple-Output Radio Channel Model for Evaluation of Space-Time Coding Algorithms," in *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd*, vol. 2, 2000, pp. 893–897 vol.2.

[15] J.-P. Kermoal, L. Schumacher, F. Frederiksen, and P. Mogensen, "Polarization Diversity in MIMO Radio Channels: Experimental Validation of a Stochastic Model and Performance Assessment," in *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th*, vol. 1, 2001, pp. 22–26 vol.1.

[16] "RUSK Multidimensional Channel Sounder," http://www.medav.de/fileadmin/redaktion/documents/English/rusk_mimo_e.pdf, Accessed: 05/07/2014.

[17] F. Kaltenberger, M. Kountouris, L. Cardoso, R. Knopp, and D. Gesbert, "Capacity of Linear Multi-User MIMO Precoding Schemes With Measured Channel Data," in *Signal Processing Advances in Wireless Communications, 2008. SPAWC 2008. IEEE 9th Workshop on*, July 2008, pp. 580–584.

[18] B. K. Lau, J. Medbo, and J. Furuskog, "Downlink Cooperative MIMO in Urban Macrocell Environments," in *Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE*, July 2010, pp. 1–4.

[19] M. Jensen, B. K. Lau, J. Medbo, and J. Furuskog, "Performance of Cooperative MIMO Based on Measured Urban Channel Data," in *Antennas and Propagation (EUCAP), Proceedings of the 5th European Conference on*, April 2011, pp. 2441–2444.

[20] N. Czink, A. Garcia Ariza, K. Haneda, M. Jacob, J. Kåredal, M. Käske, J. Medbo, J. Poutanen, J. Salmi, G. Steinböck, and K. Witrisal, "Channel Measurements," in *Pervasive Mobile and Ambient Wireless Communications*, ser. Signals and Communication Technology, R. Verdone and A. Zanella, Eds. Springer London, 2012, pp. 5–65. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-2315-6_2

[21] A. Grau, J. Romeu, L. Jofre, and F. De Flaviis, "On the Polarization Diversity Gain Using the ORIOL Antenna in Fading Environments," in *Antennas and Propagation Society International Symposium, 2005 IEEE*, vol. 4B, July 2005, pp. 14–17 vol. 4B.

[22] "USRP1 Datasheet," Ettus Research, Accessed: 04/07/2014. [Online]. Available: https://www.ettus.com/content/files/07495_Ettus_USRP1_DS_Flyer_HR.pdf

[23] "USRPClockingNotes," GNU Radio, Accessed: 04/07/2014. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/USRPClockingNotes

[24] "GNU Radio. The free & open-source radio ecosystem," http://gnuradio.org/redmine/projects/gnuradio/wiki, Accessed: 04/07/2014.

[25] H. MacLeod, C. Loadman, and Z. Chen, "Experimental Studies of the 2.4-GHz ISM Wireless Indoor Channel," in *Communication Networks and Services Research Conference, 2005. Proceedings of the 3rd Annual*, May 2005, pp. 63–68.

[26] S. Loyka and G. Levin, "On Physically-Based Normalization of MIMO Channel Matrices," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 3, pp. 1107–1112, March 2009.

[27] J. Pratt, *Financial Accounting in an Economic Context*. John Wiley & Sons, 2010. [Online]. Available: http://books.google.es/books?id=F2J7ZS_EClMC

[28] A. Kaya, W. Trappe, L. Greenstein, and D. Chizhik, "Predicting MIMO Performance in Urban Microcells Using Ray Tracing to Characterize the Channel," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 7, pp. 2402–2411, July 2012.

[29] C. Cerasoli, "The Use of Ray Tracing Models to Predict MIMO Performance in Urban Environments," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, Oct 2006, pp. 1–8.

Appendix A

GNU Radio code

```
#!/usr/bin/env python
##################################################
# Gnuradio Python Flow Graph
##################################################

from gnuradio import eng_notation
from gnuradio import gr
from gnuradio import uhd
from gnuradio import window
from gnuradio.eng_option import eng_option
from gnuradio.gr import firdes
from gnuradio.wxgui import fftsink2
from gnuradio.wxgui import forms
from gnuradio.wxgui import scopesink2
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import wx

class MIMOchannelRX(grc_wxgui.top_block_gui):

        def __init__(self):
                grc_wxgui.top_block_gui.__init__(self, title="Mimochannelrx")
                _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
                self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))

                ##################################################
                # Variables
                ##################################################
                self.samp_rate = samp_rate = .25e6
                self.norecord = norecord = True
                self.gain_slave = gain_slave = 1
                self.gain_CH_S = gain_CH_S = -.1
                self.gain_CH_M = gain_CH_M = 1
                self.gain = gain = 41
                self.f_c = f_c = 500
                self.f_IF = f_IF = 10e3
                self.f = f = 2.45e9
                self.N_inter = N_inter = 625
                self.N = N = 4
```

```
##################################################
# Blocks
##################################################
self.notebook_0 = self.notebook_0 = wx.Notebook(self.GetWin(), style
    =wx.NB_TOP)
self.notebook_0.AddPage(grc_wxgui.Panel(self.notebook_0), "fft")
self.notebook_0.AddPage(grc_wxgui.Panel(self.notebook_0), "mag/arg")
self.notebook_0.AddPage(grc_wxgui.Panel(self.notebook_0), "pll")
self.notebook_0.AddPage(grc_wxgui.Panel(self.notebook_0), "spectrum
    ")
self.Add(self.notebook_0)
self._norecord_check_box = forms.check_box(
        parent=self.notebook_0.GetPage(1).GetWin(),
        value=self.norecord,
        callback=self.set_norecord,
        label='norecord',
        true=True,
        false=False,
)
self.notebook_0.GetPage(1).Add(self._norecord_check_box)
_gain_sizer = wx.BoxSizer(wx.VERTICAL)
self._gain_text_box = forms.text_box(
        parent=self.notebook_0.GetPage(0).GetWin(),
        sizer=_gain_sizer,
        value=self.gain,
        callback=self.set_gain,
        label='gain',
        converter=forms.float_converter(),
        proportion=0,
)
self._gain_slider = forms.slider(
        parent=self.notebook_0.GetPage(0).GetWin(),
        sizer=_gain_sizer,
        value=self.gain,
        callback=self.set_gain,
        minimum=0,
        maximum=90,
        num_steps=90,
        style=wx.SL_HORIZONTAL,
        cast=float,
        proportion=1,
)
self.notebook_0.GetPage(0).Add(_gain_sizer)
self.wxgui_scopesink2_0_0 = scopesink2.scope_sink_f(
        self.notebook_0.GetPage(1).GetWin(),
        title="mag",
        sample_rate=samp_rate/N,
        v_scale=20,
        v_offset=-80,
        t_scale=0,
        ac_couple=False,
        xy_mode=False,
        num_inputs=4,
```

```
        trig_mode=gr.gr_TRIG_MODE_AUTO,
        y_axis_label="Counts",
        size=((400,45)),
)
self.notebook_0.GetPage(1).Add(self.wxgui_scopesink2_0_0.win)
self.wxgui_scopesink2_0 = scopesink2.scope_sink_f(
        self.notebook_0.GetPage(1).GetWin(),
        title="arg",
        sample_rate=samp_rate/N,
        v_scale=3.1415/2,
        v_offset=0,
        t_scale=0,
        ac_couple=False,
        xy_mode=False,
        num_inputs=3,
        trig_mode=gr.gr_TRIG_MODE_AUTO,
        y_axis_label="Counts",
        size=((400,45)),
)
self.notebook_0.GetPage(1).Add(self.wxgui_scopesink2_0.win)
self.wxgui_fftsink2_3 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(0).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT Master CH1",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(0).GridAdd(self.wxgui_fftsink2_3.win, 2, 1,
    1, 1)
self.wxgui_fftsink2_2 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(0).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT Slave CH0",
        peak_hold=False,
        size=((375,85)),
)
```

```
self.notebook_0.GetPage(0).GridAdd(self.wxgui_fftsink2_2.win, 1, 2,
    1, 1)
self.wxgui_fftsink2_1 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(0).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT Slave CH1",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(0).GridAdd(self.wxgui_fftsink2_1.win, 2, 2,
    1, 1)
self.wxgui_fftsink2_0_0_0_1 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(2).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT baseband",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(2).GridAdd(self.wxgui_fftsink2_0_0_0_1.win,
    2, 2, 1, 1)
self.wxgui_fftsink2_0_0_0_0_0_1 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(3).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="Spectrum (Slave CH0)",
        peak_hold=False,
        size=((375,85)),
)
```

```
self.notebook_0.GetPage(3).GridAdd(self.wxgui_fftsink2_0_0_0_0_0_1.
    win, 1, 2, 1, 1)
self.wxgui_fftsink2_0_0_0_0_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(3).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="Spectrum (Slave CH1)",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(3).GridAdd(self.wxgui_fftsink2_0_0_0_0_0_0
    .win, 2, 2, 1, 1)
self.wxgui_fftsink2_0_0_0_0_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(3).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="Spectrum (Master CH1)",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(3).GridAdd(self.wxgui_fftsink2_0_0_0_0_0_0.
    win, 2, 1, 1, 1)
self.wxgui_fftsink2_0_0_0_0_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(3).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="Spectrum (Master CH0)",
        peak_hold=False,
        size=((375,85)),
)
```

```
self.notebook_0.GetPage(3).GridAdd(self.wxgui_fftsink2_0_0_0_0_0.win
    , 1, 1, 1, 1)
self.wxgui_fftsink2_0_0_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(2).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="Spectrum (Master CH0)",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(2).GridAdd(self.wxgui_fftsink2_0_0_0_0.win,
    1, 1, 1, 1)
self.wxgui_fftsink2_0_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(2).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT PLL (after bandpass)",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(2).GridAdd(self.wxgui_fftsink2_0_0_0.win, 2,
     1, 1, 1)
self.wxgui_fftsink2_0_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(2).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT PLL",
        peak_hold=False,
        size=((375,85)),
)
```

```
self.notebook_0.GetPage(2).GridAdd(self.wxgui_fftsink2_0_0.win, 1,
    2, 1, 1)
self.wxgui_fftsink2_0 = fftsink2.fft_sink_c(
        self.notebook_0.GetPage(0).GetWin(),
        baseband_freq=0,
        y_per_div=10,
        y_divs=10,
        ref_level=0,
        ref_scale=2.0,
        sample_rate=samp_rate/N,
        fft_size=1024/2,
        fft_rate=15,
        average=False,
        avg_alpha=None,
        title="FFT Master CH0",
        peak_hold=False,
        size=((375,85)),
)
self.notebook_0.GetPage(0).GridAdd(self.wxgui_fftsink2_0.win, 1, 1,
    1, 1)
self.uhd_usrp_source_1 = uhd.usrp_source(
        device_addr="serial=46242c2f,type=usrp1",
        stream_args=uhd.stream_args(
                cpu_format="fc32",
                channels=range(2),
        ),
)
self.uhd_usrp_source_1.set_subdev_spec("A:0 B:0", 0)
self.uhd_usrp_source_1.set_samp_rate(samp_rate)
self.uhd_usrp_source_1.set_center_freq(f-f_IF, 0)
self.uhd_usrp_source_1.set_gain(gain+gain_slave, 0)
self.uhd_usrp_source_1.set_center_freq(f-f_IF, 1)
self.uhd_usrp_source_1.set_gain(gain+gain_slave+gain_CH_S, 1)
self.uhd_usrp_source_0 = uhd.usrp_source(
        device_addr="serial=4c7468c1,type=usrp1",
        stream_args=uhd.stream_args(
                cpu_format="fc32",
                channels=range(2),
        ),
)
self.uhd_usrp_source_0.set_subdev_spec("A:0 B:0", 0)
self.uhd_usrp_source_0.set_samp_rate(samp_rate)
self.uhd_usrp_source_0.set_center_freq(f-f_IF, 0)
self.uhd_usrp_source_0.set_gain(gain, 0)
self.uhd_usrp_source_0.set_center_freq(f-f_IF, 1)
self.uhd_usrp_source_0.set_gain(gain+gain_CH_M, 1)
self.low_pass_filter_0_0_0_0 = gr.fir_filter_ccf(1, firdes.low_pass(
        1, samp_rate/N, f_c, f_c/4, firdes.WIN_HAMMING, 6.76))
self.low_pass_filter_0_0_0 = gr.fir_filter_ccf(1, firdes.low_pass(
        1, samp_rate/N, f_c, f_c/4, firdes.WIN_HAMMING, 6.76))
self.low_pass_filter_0_0 = gr.fir_filter_ccf(1, firdes.low_pass(
        1, samp_rate/N, f_c, f_c/4, firdes.WIN_HAMMING, 6.76))
self.low_pass_filter_0 = gr.fir_filter_ccf(1, firdes.low_pass(
        1, samp_rate/N, f_c, f_c/4, firdes.WIN_HAMMING, 6.76))
```

```
self.gr_throttle_1 = gr.throttle(gr.sizeof_gr_complex*1, samp_rate)
self.gr_throttle_0_1 = gr.throttle(gr.sizeof_gr_complex*1, samp_rate
    )
self.gr_throttle_0_0 = gr.throttle(gr.sizeof_gr_complex*1, samp_rate
    )
self.gr_throttle_0 = gr.throttle(gr.sizeof_gr_complex*1, samp_rate)
self.gr_sub_xx_0_0_0 = gr.sub_ff(1)
self.gr_sub_xx_0_0 = gr.sub_ff(1)
self.gr_sub_xx_0 = gr.sub_ff(1)
self.gr_pll_refout_cc_0 = gr.pll_refout_cc(3.1415/200, 1.75*f_IF/(
    samp_rate/N)*(2*3.1415), 1.5*f_IF/(samp_rate/N)*(2*3.1415))
self.gr_nlog10_ff_0_0_0_0 = gr.nlog10_ff(20, 1, -3)
self.gr_nlog10_ff_0_0_0 = gr.nlog10_ff(20, 1, -3)
self.gr_nlog10_ff_0_0 = gr.nlog10_ff(20, 1, -3)
self.gr_nlog10_ff_0 = gr.nlog10_ff(20, 1, -3)
self.gr_mute_cc_0 = gr.mute_ff(bool(norecord))
self.gr_multiply_xx_0_0_0_0 = gr.multiply_vcc(1)
self.gr_multiply_xx_0_0_0 = gr.multiply_vcc(1)
self.gr_multiply_xx_0_0 = gr.multiply_vcc(1)
self.gr_multiply_xx_0 = gr.multiply_vcc(1)
self.gr_keep_one_in_n_2_1_1_2 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_1_1_1 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_1_1_0 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_1_1 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_1_0 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_1 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2_0 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_2 = gr.keep_one_in_n(gr.sizeof_float*1,
    N_inter)
self.gr_keep_one_in_n_1 = gr.keep_one_in_n(gr.sizeof_gr_complex*1, N
    )
self.gr_keep_one_in_n_0_1 = gr.keep_one_in_n(gr.sizeof_gr_complex*1,
     N)
self.gr_keep_one_in_n_0_0 = gr.keep_one_in_n(gr.sizeof_gr_complex*1,
     N)
self.gr_keep_one_in_n_0 = gr.keep_one_in_n(gr.sizeof_gr_complex*1, N
    )
self.gr_interleave_0 = gr.interleave(gr.sizeof_float*1)
self.gr_file_sink_0 = gr.file_sink(gr.sizeof_float*1, "/home/ubuntu/
    Documents/gnuradio-filesink/4RX")
self.gr_file_sink_0.set_unbuffered(False)
self.gr_conjugate_cc_0 = gr.conjugate_cc()
self.gr_complex_to_mag_0_0_0_0 = gr.complex_to_mag(1)
self.gr_complex_to_mag_0_0_0 = gr.complex_to_mag(1)
self.gr_complex_to_mag_0_0 = gr.complex_to_mag(1)
self.gr_complex_to_mag_0 = gr.complex_to_mag(1)
self.gr_complex_to_arg_0_0_0_0 = gr.complex_to_arg(1)
```

```
self.gr_complex_to_arg_0_0_0 = gr.complex_to_arg(1)
self.gr_complex_to_arg_0_0 = gr.complex_to_arg(1)
self.gr_complex_to_arg_0 = gr.complex_to_arg(1)
self.band_pass_filter_1_1_0 = gr.fir_filter_ccf(1, firdes.band_pass(
       1, samp_rate/N, f_IF, 2*f_IF, 2.5e3, firdes.WIN_HAMMING,
           6.76))
self.band_pass_filter_0_2 = gr.fir_filter_ccf(1, firdes.band_pass(
       1, samp_rate, f_IF, 2*f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76)
           )
self.band_pass_filter_0_1 = gr.fir_filter_ccf(1, firdes.band_pass(
       1, samp_rate, f_IF, 2*f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76)
           )
self.band_pass_filter_0_0 = gr.fir_filter_ccf(1, firdes.band_pass(
       1, samp_rate, f_IF, 2*f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76)
           )
self.band_pass_filter_0 = gr.fir_filter_ccf(1, firdes.band_pass(
       1, samp_rate, f_IF, 2*f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76)
           )

##################################################
# Connections
##################################################
self.connect((self.uhd_usrp_source_1, 0), (self.gr_throttle_1, 0))
self.connect((self.uhd_usrp_source_0, 0), (self.gr_throttle_0, 0))
self.connect((self.uhd_usrp_source_0, 1), (self.gr_throttle_0_0, 0))
self.connect((self.low_pass_filter_0_0_0, 0), (self.wxgui_fftsink2_3
    , 0))
self.connect((self.uhd_usrp_source_1, 1), (self.gr_throttle_0_1, 0))
self.connect((self.gr_complex_to_mag_0_0, 0), (self.gr_nlog10_ff_0_0
    , 0))
self.connect((self.gr_nlog10_ff_0_0, 0), (self.wxgui_scopesink2_0_0,
     1))
self.connect((self.gr_nlog10_ff_0, 0), (self.wxgui_scopesink2_0_0,
    0))
self.connect((self.low_pass_filter_0, 0), (self.
    wxgui_fftsink2_0_0_0_1, 0))
self.connect((self.gr_complex_to_arg_0_0_0, 0), (self.gr_sub_xx_0_0,
     1))
self.connect((self.gr_interleave_0, 0), (self.gr_mute_cc_0, 0))
self.connect((self.gr_keep_one_in_n_2_1, 0), (self.gr_interleave_0,
    2))
self.connect((self.gr_keep_one_in_n_2_1_0, 0), (self.gr_interleave_0
    , 3))
self.connect((self.gr_keep_one_in_n_2_0, 0), (self.gr_interleave_0,
    1))
self.connect((self.gr_multiply_xx_0_0_0, 0), (self.
    low_pass_filter_0_0_0, 0))
self.connect((self.gr_complex_to_arg_0, 0), (self.gr_sub_xx_0_0, 0))
self.connect((self.gr_complex_to_arg_0_0_0_0, 0), (self.
    gr_keep_one_in_n_2_1_1_2, 0))
self.connect((self.gr_complex_to_mag_0, 0), (self.gr_nlog10_ff_0, 0)
    )
self.connect((self.gr_multiply_xx_0_0, 0), (self.low_pass_filter_0_0
    , 0))
```

```
self.connect((self.gr_nlog10_ff_0_0_0, 0), (self.
    wxgui_scopesink2_0_0, 2))
self.connect((self.gr_complex_to_mag_0_0_0, 0), (self.
    gr_nlog10_ff_0_0_0, 0))
self.connect((self.gr_complex_to_mag_0_0_0_0, 0), (self.
    gr_nlog10_ff_0_0_0_0, 0))
self.connect((self.gr_nlog10_ff_0_0_0_0, 0), (self.
    wxgui_scopesink2_0_0, 3))
self.connect((self.low_pass_filter_0_0, 0), (self.wxgui_fftsink2_2,
    0))
self.connect((self.gr_conjugate_cc_0, 0), (self.
    gr_multiply_xx_0_0_0_0, 0))
self.connect((self.gr_keep_one_in_n_0, 0), (self.
    wxgui_fftsink2_0_0_0_0, 0))
self.connect((self.gr_keep_one_in_n_0, 0), (self.
    wxgui_fftsink2_0_0_0_0_0, 0))
self.connect((self.gr_keep_one_in_n_0_0, 0), (self.
    wxgui_fftsink2_0_0_0_0_0_0, 0))
self.connect((self.gr_keep_one_in_n_1, 0), (self.
    wxgui_fftsink2_0_0_0_0_0_1, 0))
self.connect((self.gr_keep_one_in_n_0_1, 0), (self.
    wxgui_fftsink2_0_0_0_0_0_0_0, 0))
self.connect((self.gr_throttle_1, 0), (self.band_pass_filter_0, 0))
self.connect((self.band_pass_filter_0, 0), (self.gr_keep_one_in_n_1,
     0))
self.connect((self.gr_throttle_0_1, 0), (self.band_pass_filter_0_0,
    0))
self.connect((self.band_pass_filter_0_0, 0), (self.
    gr_keep_one_in_n_0_1, 0))
self.connect((self.gr_throttle_0, 0), (self.band_pass_filter_0_1, 0)
    )
self.connect((self.band_pass_filter_0_1, 0), (self.
    gr_keep_one_in_n_0, 0))
self.connect((self.gr_throttle_0_0, 0), (self.band_pass_filter_0_2,
    0))
self.connect((self.band_pass_filter_0_2, 0), (self.
    gr_keep_one_in_n_0_0, 0))
self.connect((self.gr_mute_cc_0, 0), (self.gr_file_sink_0, 0))
self.connect((self.gr_pll_refout_cc_0, 0), (self.wxgui_fftsink2_0_0,
     0))
self.connect((self.low_pass_filter_0, 0), (self.wxgui_fftsink2_0, 0)
    )
self.connect((self.gr_complex_to_arg_0, 0), (self.gr_sub_xx_0, 0))
self.connect((self.gr_keep_one_in_n_0_1, 0), (self.
    gr_multiply_xx_0_0_0_0, 1))
self.connect((self.gr_sub_xx_0, 0), (self.wxgui_scopesink2_0, 0))
self.connect((self.gr_complex_to_arg_0_0, 0), (self.gr_sub_xx_0, 1))
self.connect((self.gr_sub_xx_0_0, 0), (self.wxgui_scopesink2_0, 1))
self.connect((self.gr_pll_refout_cc_0, 0), (self.
    band_pass_filter_1_1_0, 0))
self.connect((self.gr_complex_to_arg_0, 0), (self.gr_sub_xx_0_0_0,
    0))
self.connect((self.gr_keep_one_in_n_1, 0), (self.gr_pll_refout_cc_0,
     0))
```

```
self.connect((self.gr_keep_one_in_n_2, 0), (self.gr_interleave_0, 0)
    )
self.connect((self.gr_complex_to_mag_0, 0), (self.gr_keep_one_in_n_2
    , 0))
self.connect((self.low_pass_filter_0, 0), (self.gr_complex_to_mag_0,
     0))
self.connect((self.band_pass_filter_1_1_0, 0), (self.
    wxgui_fftsink2_0_0_0, 0))
self.connect((self.gr_conjugate_cc_0, 0), (self.gr_multiply_xx_0, 0)
    )
self.connect((self.gr_conjugate_cc_0, 0), (self.gr_multiply_xx_0_0_0
    , 0))
self.connect((self.gr_keep_one_in_n_0_0, 0), (self.
    gr_multiply_xx_0_0_0, 1))
self.connect((self.gr_complex_to_arg_0, 0), (self.
    gr_keep_one_in_n_2_0, 0))
self.connect((self.low_pass_filter_0, 0), (self.gr_complex_to_arg_0,
     0))
self.connect((self.gr_multiply_xx_0, 0), (self.low_pass_filter_0, 0)
    )
self.connect((self.gr_keep_one_in_n_1, 0), (self.gr_multiply_xx_0,
    1))
self.connect((self.gr_complex_to_mag_0_0, 0), (self.
    gr_keep_one_in_n_2_1, 0))
self.connect((self.low_pass_filter_0_0, 0), (self.
    gr_complex_to_mag_0_0, 0))
self.connect((self.gr_complex_to_arg_0_0, 0), (self.
    gr_keep_one_in_n_2_1_0, 0))
self.connect((self.low_pass_filter_0_0, 0), (self.
    gr_complex_to_arg_0_0, 0))
self.connect((self.gr_keep_one_in_n_0, 0), (self.gr_multiply_xx_0_0,
     1))
self.connect((self.gr_conjugate_cc_0, 0), (self.gr_multiply_xx_0_0,
    0))
self.connect((self.low_pass_filter_0_0_0_0, 0), (self.
    wxgui_fftsink2_1, 0))
self.connect((self.gr_complex_to_arg_0_0_0_0, 0), (self.
    gr_sub_xx_0_0_0, 1))
self.connect((self.low_pass_filter_0_0_0_0, 0), (self.
    gr_complex_to_arg_0_0_0_0, 0))
self.connect((self.gr_sub_xx_0_0_0, 0), (self.wxgui_scopesink2_0, 2)
    )
self.connect((self.gr_complex_to_mag_0_0_0, 0), (self.
    gr_keep_one_in_n_2_1_1, 0))
self.connect((self.gr_keep_one_in_n_2_1_1_0, 0), (self.
    gr_interleave_0, 7))
self.connect((self.gr_keep_one_in_n_2_1_1, 0), (self.gr_interleave_0
    , 6))
self.connect((self.low_pass_filter_0_0_0, 0), (self.
    gr_complex_to_mag_0_0_0, 0))
self.connect((self.gr_complex_to_arg_0_0_0, 0), (self.
    gr_keep_one_in_n_2_1_1_0, 0))
self.connect((self.low_pass_filter_0_0_0, 0), (self.
    gr_complex_to_arg_0_0_0, 0))
```

```
        self.connect((self.gr_keep_one_in_n_2_1_1_1, 0), (self.
            gr_interleave_0, 4))
        self.connect((self.gr_keep_one_in_n_2_1_1_2, 0), (self.
            gr_interleave_0, 5))
        self.connect((self.gr_complex_to_mag_0_0_0_0, 0), (self.
            gr_keep_one_in_n_2_1_1_1, 0))
        self.connect((self.low_pass_filter_0_0_0_0, 0), (self.
            gr_complex_to_mag_0_0_0_0, 0))
        self.connect((self.gr_multiply_xx_0_0_0_0, 0), (self.
            low_pass_filter_0_0_0_0, 0))
        self.connect((self.band_pass_filter_1_1_0, 0), (self.
            gr_conjugate_cc_0, 0))

def get_samp_rate(self):
        return self.samp_rate

def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.wxgui_fftsink2_0_0_0_1.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_scopesink2_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_scopesink2_0_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_3.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_1.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_2.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_0_0_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_0_0_0_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_0_0_0_0_0_0_0.set_sample_rate(self.samp_rate/
            self.N)
        self.wxgui_fftsink2_0_0_0_0_0_0.set_sample_rate(self.samp_rate/self.
            N)
        self.wxgui_fftsink2_0_0_0_0_0_1.set_sample_rate(self.samp_rate/self.
            N)
        self.wxgui_fftsink2_0_0_0_0_0.set_sample_rate(self.samp_rate/self.N)
        self.wxgui_fftsink2_0_0.set_sample_rate(self.samp_rate/self.N)
        self.band_pass_filter_0_1.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
        self.band_pass_filter_0.set_taps(firdes.band_pass(1, self.samp_rate,
             self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76))
        self.band_pass_filter_0_0.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
        self.low_pass_filter_0_0.set_taps(firdes.low_pass(1, self.samp_rate/
            self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
        self.band_pass_filter_1_1_0.set_taps(firdes.band_pass(1, self.
            samp_rate/self.N, self.f_IF, 2*self.f_IF, 2.5e3, firdes.
            WIN_HAMMING, 6.76))
        self.low_pass_filter_0_0_0_0.set_taps(firdes.low_pass(1, self.
            samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
            6.76))
        self.band_pass_filter_0_2.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
```

```
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate/
            self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
        self.low_pass_filter_0_0_0.set_taps(firdes.low_pass(1, self.
            samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
            6.76))
        self.uhd_usrp_source_1.set_samp_rate(self.samp_rate)
        self.uhd_usrp_source_0.set_samp_rate(self.samp_rate)

    def get_norecord(self):
        return self.norecord

    def set_norecord(self, norecord):
        self.norecord = norecord
        self._norecord_check_box.set_value(self.norecord)
        self.gr_mute_cc_0.set_mute(bool(self.norecord))

    def get_gain_slave(self):
        return self.gain_slave

    def set_gain_slave(self, gain_slave):
        self.gain_slave = gain_slave
        self.uhd_usrp_source_1.set_gain(self.gain+self.gain_slave, 0)
        self.uhd_usrp_source_1.set_gain(self.gain+self.gain_slave+self.
            gain_CH_S, 1)

    def get_gain_CH_S(self):
        return self.gain_CH_S

    def set_gain_CH_S(self, gain_CH_S):
        self.gain_CH_S = gain_CH_S
        self.uhd_usrp_source_1.set_gain(self.gain+self.gain_slave+self.
            gain_CH_S, 1)

    def get_gain_CH_M(self):
        return self.gain_CH_M

    def set_gain_CH_M(self, gain_CH_M):
        self.gain_CH_M = gain_CH_M
        self.uhd_usrp_source_0.set_gain(self.gain+self.gain_CH_M, 1)

    def get_gain(self):
        return self.gain

    def set_gain(self, gain):
        self.gain = gain
        self.uhd_usrp_source_1.set_gain(self.gain+self.gain_slave, 0)
        self.uhd_usrp_source_1.set_gain(self.gain+self.gain_slave+self.
            gain_CH_S, 1)
        self.uhd_usrp_source_0.set_gain(self.gain, 0)
        self.uhd_usrp_source_0.set_gain(self.gain+self.gain_CH_M, 1)
        self._gain_slider.set_value(self.gain)
        self._gain_text_box.set_value(self.gain)

    def get_f_c(self):
```

```
        return self.f_c

    def set_f_c(self, f_c):
        self.f_c = f_c
        self.low_pass_filter_0_0.set_taps(firdes.low_pass(1, self.samp_rate/
            self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
        self.low_pass_filter_0_0_0_0.set_taps(firdes.low_pass(1, self.
            samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
            6.76))
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate/
            self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
        self.low_pass_filter_0_0_0.set_taps(firdes.low_pass(1, self.
            samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
            6.76))

    def get_f_IF(self):
        return self.f_IF

    def set_f_IF(self, f_IF):
        self.f_IF = f_IF
        self.band_pass_filter_0_1.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
        self.band_pass_filter_0.set_taps(firdes.band_pass(1, self.samp_rate,
             self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING, 6.76))
        self.band_pass_filter_0_0.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
        self.band_pass_filter_1_1_0.set_taps(firdes.band_pass(1, self.
            samp_rate/self.N, self.f_IF, 2*self.f_IF, 2.5e3, firdes.
            WIN_HAMMING, 6.76))
        self.band_pass_filter_0_2.set_taps(firdes.band_pass(1, self.
            samp_rate, self.f_IF, 2*self.f_IF, 2.5e3, firdes.WIN_HAMMING,
            6.76))
        self.uhd_usrp_source_1.set_center_freq(self.f-self.f_IF, 0)
        self.uhd_usrp_source_1.set_center_freq(self.f-self.f_IF, 1)
        self.uhd_usrp_source_0.set_center_freq(self.f-self.f_IF, 0)
        self.uhd_usrp_source_0.set_center_freq(self.f-self.f_IF, 1)

    def get_f(self):
        return self.f

    def set_f(self, f):
        self.f = f
        self.uhd_usrp_source_1.set_center_freq(self.f-self.f_IF, 0)
        self.uhd_usrp_source_1.set_center_freq(self.f-self.f_IF, 1)
        self.uhd_usrp_source_0.set_center_freq(self.f-self.f_IF, 0)
        self.uhd_usrp_source_0.set_center_freq(self.f-self.f_IF, 1)

    def get_N_inter(self):
        return self.N_inter

    def set_N_inter(self, N_inter):
        self.N_inter = N_inter
```

```
                self.gr_keep_one_in_n_2.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_0.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1_0.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1_1_1.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1_1_2.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1_1.set_n(self.N_inter)
                self.gr_keep_one_in_n_2_1_1_0.set_n(self.N_inter)

        def get_N(self):
                return self.N

        def set_N(self, N):
                self.N = N
                self.wxgui_fftsink2_0_0_0_1.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_scopesink2_0.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_scopesink2_0_0.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_3.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_1.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_2.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_0_0_0.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_0.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_0_0_0_0.set_sample_rate(self.samp_rate/self.N)
                self.wxgui_fftsink2_0_0_0_0_0_0_0.set_sample_rate(self.samp_rate/
                    self.N)
                self.wxgui_fftsink2_0_0_0_0_0_0.set_sample_rate(self.samp_rate/self.
                    N)
                self.wxgui_fftsink2_0_0_0_0_0_1.set_sample_rate(self.samp_rate/self.
                    N)
                self.wxgui_fftsink2_0_0_0_0_0.set_sample_rate(self.samp_rate/self.N)
                self.gr_keep_one_in_n_1.set_n(self.N)
                self.gr_keep_one_in_n_0_1.set_n(self.N)
                self.wxgui_fftsink2_0_0.set_sample_rate(self.samp_rate/self.N)
                self.gr_keep_one_in_n_0.set_n(self.N)
                self.gr_keep_one_in_n_0_0.set_n(self.N)
                self.low_pass_filter_0_0.set_taps(firdes.low_pass(1, self.samp_rate/
                    self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
                self.band_pass_filter_1_1_0.set_taps(firdes.band_pass(1, self.
                    samp_rate/self.N, self.f_IF, 2*self.f_IF, 2.5e3, firdes.
                    WIN_HAMMING, 6.76))
                self.low_pass_filter_0_0_0_0.set_taps(firdes.low_pass(1, self.
                    samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
                    6.76))
                self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate/
                    self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING, 6.76))
                self.low_pass_filter_0_0_0.set_taps(firdes.low_pass(1, self.
                    samp_rate/self.N, self.f_c, self.f_c/4, firdes.WIN_HAMMING,
                    6.76))

if __name__ == '__main__':
        parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
        (options, args) = parser.parse_args()
        tb = MIMOchannelRX()
        tb.Run(True)
```