



UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in Informatica

ELABORATO FINALE

ANALISI DELLE PRESTAZIONI DI
RICEVITORI SDR A BASSO COSTO IN
AMBIENTE GNURADIO

Supervisore

Fabrizio Granelli

Laureando

Andrea Filippi

Anno accademico 2017/2018

Ringraziamenti

Desidero ringraziare tutte le persone che mi sono state vicine nei molteplici momenti di gioia/difficoltà. L'esperienza di questi tre anni all'università di Trento non la dimenticherò mai. Ho avuto modo di conoscere molte persone favolose e scoprire il Trentino in un modo diverso da come lo conoscevo in precedenza. Le alte montagne mi hanno aiutato a mantenere la serenità anche nei periodi di studio più intensi. La presenza di studenti provenienti da tutto il mondo mi ha permesso di tradurre una frase da un dialetto indiano che custodivo da anni come portafortuna.

- *Vorrei ringraziare di cuore il professor Fabrizio Granelli che ha permesso la mia laurea*
- *Un ringraziamento particolare a tutta la mia famiglia, alla mia ragazza, ai nonni, alla sorella, ai numerosi zii, agli infiniti cugini, tutti.*
- *Ringrazio i miei amici per esserci stati sia nei momenti di disperazione che in quelli di festa che di tradizione veniva organizzata al Bait dopo il superamento di un esame importante.*

“Possiamo avere tutti i mezzi di comunicazione del mondo, ma niente, assolutamente niente, sostituisce lo sguardo dell'essere umano.”

Paulo Coelho

Abstract

Il recente sviluppo nella capacità di calcolo dei computer sta permettendo il passaggio delle telecomunicazioni da hardware dedicato ad algoritmi software. Esistono vari tipi di SDR che forniscono l'hardware necessario ad un computer.

L'obiettivo del lavoro è stato quello di implementare una tecnica di comunicazione largamente utilizzata quale OFDM per osservare il comportamento di un SDR venduto come decoder per radio e televisione digitale a meno di 8 euro.

Per la parte di trasmissione è stato utilizzato un SDR da laboratorio (Ettus B210). L'implementazione di OFDM è stata fatta utilizzando l'ambiente Gnuradio su cui è stato successivamente progettato un modulo aggiuntivo per la crittografia rsa.

I test eseguiti hanno confermato la validità di questi SDR ma ne hanno evidenziato anche alcuni limiti legati principalmente al loro hardware limitato (RTL2832). La qualità della comunicazione instaurata ha permesso di confermare l'efficacia di questi dispositivi in applicazioni reali.

Indice

Sommario	3
0.0.1 Contesto	3
0.0.2 Obiettivo del lavoro	3
0.0.3 Problematiche affrontate	3
0.0.4 Organizzazione elaborato	3
1 Tecniche di trasmissione	5
1.1 Storia dello sviluppo	5
1.2 Modulazioni Analogiche	5
1.2.1 AM (Amplitude Modulation):	5
1.2.2 FM (Frequency Modulation):	5
1.2.3 PM (Phase Modulation):	5
1.3 Modulazioni Digitali	6
1.3.1 FSK (Frequency Shifting Key):	6
1.3.2 ASK (Amplitude Shift Keying):	6
1.3.3 PSK (Phase Shift Keying):	6
1.3.4 DPSK (Differential Phase Shift Keying):	7
1.3.5 QAM (Quadrature Amplitude Modulation):	7
2 OFDM (Orthogonal Frequency-division Multiplexing):	8
2.1 Principi di funzionamento	8
2.1.1 Ortogonalità delle sottoportanti:	8
2.1.2 Tempi di guardia	9
2.1.3 Equalizzazione	9
2.1.4 Recupero degli errori	10
2.1.5 Sincronizzazione in frequenza	11
2.2 Proprietà e campi di utilizzo	12
3 SDR (Software Defined Radio)	13
3.0.1 USRP (Universal Software Radio Peripheral)	14
3.0.2 RTL-SDR	14
4 Gnuradio	15
5 Crittografia RSA	15
5.1 Caratteristiche	16
5.1.1 Codifica asimmetrica con doppia chiave	16
5.1.2 Algoritmo unico	16
5.1.3 Chiavi intercambiabili	16
5.1.4 Procedimento creazione chiave monodirezionale	16
5.2 Algoritmo	16
5.2.1 Generazione delle chiavi	16
5.2.2 Cifratura	16
5.2.3 Decifratura	16

6	OFDM in Gnuradio	17
6.1	Trasmittitore	17
6.1.1	Formazione pacchetti dal flusso in ingresso	17
6.1.2	Codifica informazioni in Simboli	18
6.1.3	Allocazione simboli sulle sottoportanti e creazione campioni per l'SDR	18
6.1.4	Passaggio dati all'SDR	19
6.2	Ricevitore	19
6.2.1	Ricezione informazioni, sincronizzazione	19
6.2.2	Equalizzazione e ottenimento simboli	20
6.2.3	Decodifica payload e scrittura file	20
6.3	Testing	21
6.3.1	Caratteristiche generali del sistema sotto test	21
6.3.2	Parametri specifici	21
7	Implementazione Rsa in Gnuradio	22
7.1	Sviluppo dei moduli	22
7.1.1	Definizione proprietà moduli per l'ambiente grafico	22
7.1.2	Algoritmo per cifratura e decifratura	23
7.1.3	Codice di test	24
7.2	Importazione in Gruradio	25
8	Conclusioni	26
	Sitografia	27

Sommario

Contesto

Lo sviluppo in software di apparati radio rappresenta un notevole passo in avanti nel mondo delle telecomunicazioni, finalmente è possibile programmare una scheda anzich  dover implementare circuiti dedicati. Un normale computer non   in grado da solo di trasmettere e ricevere segnali, per questo   necessario l'utilizzo di una scheda che permetta la conversione di onde elettromagnetiche in campioni e viceversa. Da anni sono disponibili varie tipologie di SDR sul mercato, il loro prezzo   ancora significativamente elevato trovando utilizzo specialmente in laboratori e rendendone proibitivo l'utilizzo su larga scala. In questi ultimi anni le cose stanno iniziando a cambiare con la disponibilit  di ricevitori SDR a ridotte prestazioni ma con costi stracciati.

Obiettivo del lavoro

L'obiettivo del lavoro svolto   proprio quello di osservare il comportamento di questi dispositivi low-cost in un ambiente di comunicazione avanzato cercando di evidenziare i loro limiti e le loro potenzialit . La scelta della tecnica di trasmissione per il lavoro   ricaduta su OFDM dato che viene largamente utilizzato per i sistemi di telecomunicazione odierni come ad esempio wifi, adsl, Powerline, televisione e radio digitali. Per l'implementazione   stato utilizzato l'ambiente open source Gnuradio che possiede tutti i requisiti per svolgere il lavoro desiderato. Infatti dispone di un insieme di blocchi appositi per l'implementazione di OFDM. Al fine di ottenere una comunicazione funzionante e completa   stato necessario acquisire esperienza nell'ambiente Gnuradio che ha in seguito permesso la progettazione e la creazione di un prototipo composto da due blocchi aggiuntivi per la crittografia RSA. Il lavoro di documentazione sul funzionamento OFDM di trasmettitore e ricevitore   molto utile a mio avviso per integrare la poca documentazione disponibile in rete presente principalmente nella spiegazione sul sito ufficiale.

Problematiche affrontate

La scarsa disponibilit  di documentazione su Gnuradio   stato senza dubbi l'ostacolo maggiore da affrontare. Prima di riuscire soltanto a simulare il trasferimento con la tecnica OFDM   stato necessario un grande sforzo e la conoscenza teorica del funzionamento.

Per l'utilizzo del ricevitore dvb-t/dab come SDR   necessario fare uso di driver alternativi che hanno la tendenza a litigare con quello originale installato in automatico dal sistema operativo. Risolti questi problemi l'aggiunta in Gnuradio non ha rappresentato una grande difficolt ,   bastato prestare attenzione alla configurazione dei parametri affin  che non superassero i vincoli dichiarati per il funzionamento. Per la parte che riguarda lo sviluppo dei due moduli per la crittografia sono state riscontrate problematiche relative a conflitti fra librerie gi  installate nel sistema,   stata richiesta molta pazienza per risolverli e poter finalmente aggiungere i moduli rsa all'ambiente Gnuradio. L'ultima problematica ha riguardato la fase di testing del sistema completo, il vincolo di rimanere nel piccolo laboratorio per utilizzare le apparecchiature ha influito negativamente sulla qualit  e sulla tipologia dei test. Sarebbe stato interessante effettuare nuovamente i test in un ambiente aperto per vedere ad esempio quanto il fenomeno del "multipath propagation" abbia influito sulla perdita di informazioni.

Organizzazione elaborato

L'elaborato propone una documentazione teorica sulle modulazioni principali ponendo particolare attenzione alla tecnica OFDM utilizzata, successivamente viene introdotto il mondo degli SDR spiegando le differenze in termini tecnici fra le due tipologie di SDR utilizzati. In conclusione della sezione teorica viene brevemente discussa la codifica RSA riportando anche l'algoritmo. L'obiettivo   facilitare la comprensione della progettazione finale dei blocchi per la crittografia situata nell'ultima parte dello scritto.

La sezione pratica comprende nella sua parte iniziale la documentazione sul funzionamento di OFDM nell'ambiente Gnuradio ordinata al fine di seguire il flusso logico delle informazioni attraverso tutto il loro percorso da sorgente a destinatario. Segue una sezione in merito ai risultati ottenuti. Viene successivamente documentata la procedura di creazione di nuovi blocchi in ambiente Gnuradio seguendo passo passo lo sviluppo dei due prototipi per l'aggiunta della crittografia RSA.

Le problematiche di un'integrazione nel sistema OFDM di questi blocchi vengono discusse nella sezione delle conclusioni assieme ai risultati ottenuti dagli SDR e i possibili sviluppi futuri del lavoro svolto.

Tecniche di trasmissione

Storia dello sviluppo

La comunicazione fra gli esseri umani è una delle abilità che hanno permesso all' uomo di evolversi. La prima forma di comunicazione è stata quella verbale, molto rapida ed efficace che però non garantisce la durata delle informazioni trasmesse. Altri metodi di comunicazione vennero sviluppati, alcuni fra i più interessanti erano le nuvole di fumo che venivano utilizzate dagli indigeni d'America e dai cinesi per comunicare lungo la grande muraglia cinese. Analogamente le tribù africane utilizzavano i tamburi. La scrittura apparve circa 7000 anni fa favorendo l'inizio di un progresso che porterà l'uomo verso il ruolo centrale che ha ora sulla terra. Già nell'epoca della nascita di Cristo l'uomo aveva instaurato una rete di comunicazione in forma scritta che interessava tutto il vecchio continente e lo collegava anche al mondo indiano ed orientale. Altri metodi un po' particolari vennero sfruttati prima dell' invenzione dell' elettricità come ad esempio: l'addestramento di piccioni viaggiatori che dimostrano avere un packet loss di solo il 17% oppure l'utilizzo di una lingua formata da fischi fra le lunghe valli nelle isole dell'arcipelago delle Canarie. Con la scoperta della corrente elettrica si è aperto per noi un nuovo mondo di possibilità fra le quali quella di trasferire immense quantità di informazioni velocemente e su lunghe distanze. Dapprima il telegrafo e poi il telefono fino ad arrivare alle trasmissioni analogiche seguite dall'avvento dell'era digitale. Nelle telecomunicazioni moderne per trasmettere si utilizza una portante (segnale elettrico oppure onda elettromagnetica) alla quale vengono aggiunte le informazioni secondo diverse tecniche dette anche modulazioni.

Modulazioni Analogiche

Le modulazioni analogiche sono utilizzate per inviare un segnale la cui variazione avviene all'interno di un intervallo teoricamente con infiniti impossibili valori. Sono solitamente meno complesse da implementare per il ricevitore rispetto alle versioni digitali.

- **1.2.1 AM (Amplitude Modulation):**

La modulazione in ampiezza è stata una delle prime modulazioni utilizzate grazie alla sua facilità di implementazione in hardware. Il segnale viene direttamente sommato alla portante in modo analogico, la sua semplicità è ormai l'unico vantaggio in quanto una soluzione di questo tipo è soggetta ad interferenze di qualsiasi origine, è sufficiente infatti una semplice attenuazione del segnale per influire direttamente sui dati ricevuti. Questa modulazione viene ancora utilizzata per la trasmissione radio che grazie a frequenze molto basse (khz) e potenze elevate (kw) permette di comunicare su distanze mondiali

- **1.2.2 FM (Frequency Modulation):**

La modulazione viene effettuata variando la frequenza del segnale portante alzandola o abbassandola in relazione alle informazioni da trasmettere, è più efficiente della modulazione in ampiezza in quanto non necessita di variare la potenza. Richiede però dei circuiti più complessi che siano in grado di svolgere il compito di codifica/decodifica. La modulazione in frequenza FM è tuttora utilizzata per la trasmissione della radio anche se sta venendo progressivamente sostituita dalla radio digitale "DAB"

- **1.2.3 PM (Phase Modulation):**

Le informazioni vengono modulate modificando la fase della portante in relazione al valore di ampiezza del segnale analogico in ingresso. Viene raramente utilizzata a causa della complessità

richiesta nei ricevitori, tuttavia il principio di funzionamento alla base è lo stesso utilizzato da PSK e QAM.

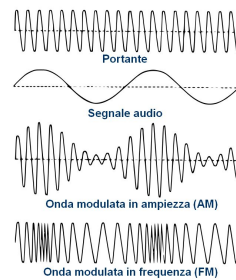


Figura 1.1: rappresentazione segnale modulato AM e FM nel tempo

Modulazioni Digitali

• 1.3.1 FSK (Frequency Shifting Key):

Questa tecnica di modulazione codifica l'informazione variando la frequenza della portante in valori predefiniti, ad esempio per ottenere una codifica binaria alterna due frequenze diverse. Questa tecnica ha il vantaggio di essere facile nell'implementazione e poco soggetta ad interferenze, tuttavia necessita di una maggiore larghezza di banda rispetto ad altre modulazioni digitali quali psk o ask. [14]

• 1.3.2 ASK (Amplitude Shift Keying):

La modulazione viene effettuata variando l'ampiezza del segnale portante alzandola o abbassandola in relazione alle informazioni da trasmettere, richiede un canale più affidabile in grado di ricevere anche i livelli di ampiezza più bassi. Trova ancora utilizzo nelle fibre ottiche e la sua versione a codifica binaria (solo due livelli di potenza della portante presente/non presente) che prende il nome di OOF(on/off keying) veniva in passato utilizzata per trasmettere messaggi in codice morse. [2]

• 1.3.3 PSK (Phase Shift Keying):

Questo tipo di modulazione codifica le informazioni in ingresso variando la fase della portante, ne esistono varie versioni che differiscono per il numero di valori diversi. La versione più semplice è la binaryPSK che varia di metà periodo mentre versioni come la 4PSK di un quarto e così via per la variante 8PSK e 16PSK. Tali possibili sfasature della portante vengono dette costellazione e vengono di norma rappresentate come coordinate complesse su un grafico. Sulle ascisse si trova la portante, mentre sulle ordinate si trova in quadratura ovvero sfasata di 90° . La lunghezza del vettore fra l'origine e uno dei punti della costellazione rappresenta l'ampiezza del segnale modulato mentre l'angolo rappresenta la sfasatura rispetto alla portante. Esiste una variante di 4PSK detta QPSK che differisce per una disposizione della costellazione ruotata di 45° .

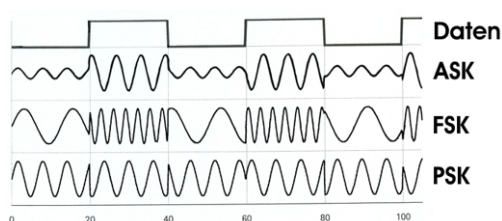


Figura 1.2: rappresentazione segnale modulato nel tempo ASK, FSK e PSK [15]

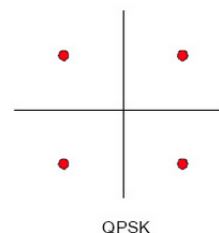


Figura 1.3: Costellazione QPSK [21]

- **1.3.4 DPSK (Differential Phase Shift Keying):**

Questa tecnica differisce da PSK solo per la particolarità di codificare il simbolo non utilizzando una costellazione fissa, le informazioni sono espresse come cambio di fase rispetto al simbolo precedente. Tale caratteristica rende questa modulazione robusta sia contro variazioni di ampiezza come psk sia contro distorsioni della fase del segnale ricevuto.

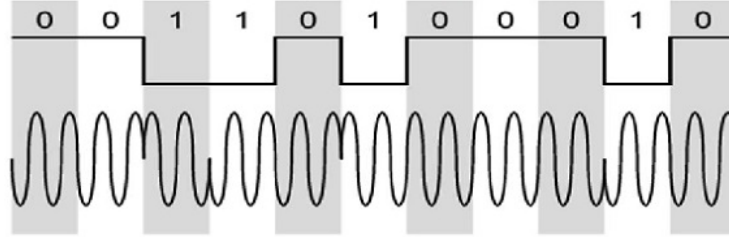


Figura 1.4: rappresentazione segnale modulato DPSK [9]

- **1.3.5 QAM (Quadrature Amplitude Modulation):**

E' una tecnica di modulazione simile a PSK ma introduce la modulazione anche in ampiezza. La costellazione risulta avere punti non più equidistanti dall'origine. QAM come PSK presenta varianti che differiscono per il numero di punti sulla costellazione, in sistemi moderni si utilizzano anche 256 punti. Una particolarità comune a psk consiste nel fatto che due punti della costellazione adiacente differiscono per un solo bit, ciò incrementa l'efficacia di un eventuale error recovery. QAM viene anche utilizzato per trasferire più flussi analogici contemporaneamente, questo particolare utilizzo fa sì che QAM venga considerato anche come una tecnica di modulazione analogica. [20]

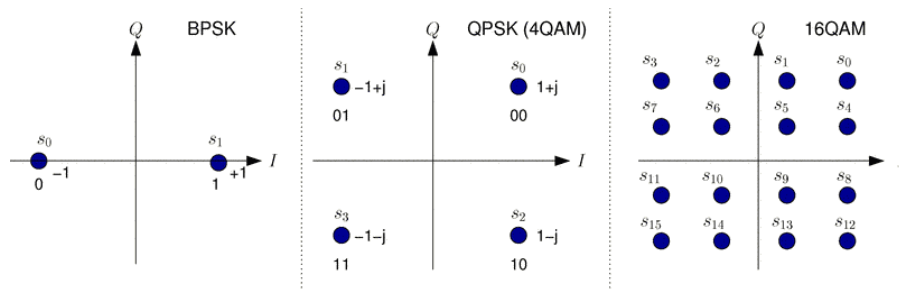


Figura 1.5: Costellazioni BPSK, QPSK(4QAM), 16PSK [3]

OFDM (Orthogonal Frequency-division Multiplexing):

OFDM è una tecnica di codifica digitale su portanti multiple, su ogni sotto portante vengono modulate le informazioni utilizzando una delle varie tecniche digitali disponibili (solitamente si utilizza una versione di PSK oppure di QAM). OFDM trova svariati utilizzi nei sistemi di comunicazione moderni come ad esempio adsl, fibra ottica, 4G, wi-fi(802.11a/g/n/ac), radio/televisione digitale e WiMAX. [16] [17]

Principi di funzionamento

• 2.1.1 Ortogonalità delle sottoportanti:

La divisione della larghezza di banda disponibile in sezioni più piccole non è una caratteristica unica di OFDM, ciò che lo distingue è la soluzione al problema di interferenza fra i sotto canali. La soluzione classica è quella di lasciare delle piccole bande di frequenza vuote dove non si trasmette fra un canale e quello adiacente (bande di guardia). Da notare che in questo tipo di approccio si ha una allocazione inefficiente della larghezza di banda disponibile. OFDM utilizza la proprietà di ortogonalità per riuscire a sovrapporre parzialmente canali adiacenti evitando sprechi di banda e aumentando così l'efficienza spettrale (indica la bontà del sistema nello sfruttare in maniera più o meno efficiente la banda disponibile [11]). La spiegazione matematica di questa tecnica è complessa e fa uso della trasformata di Fourier, è tuttavia possibile intuire il funzionamento dalla Figura 2.1. L'immagine rappresenta la disposizione delle sotto portanti, sulle ascisse troviamo le frequenze mentre sulle ordinate l'ampiezza. Ogni picco rappresenta un simbolo modulato sulla rispettiva portante. E' possibile notare che in questa particolare disposizione il rumore, che genererebbe ogni sottocanale all'esterno della propria banda, si annulla esattamente in corrispondenza delle frequenze di trasmissione dei simboli adiacenti non causando disturbo. Affinché l'ortogonalità garantisca che non ci siano interferenze fra le diverse sottoportanti è necessario che il tempo di trasmissione dei simboli sia uguale in tutto il sistema, in ambienti caratterizzati da una variazione dell'attenuazione sul mezzo trasmissivo (ad esempio sul doppino adsl) un eventuale algoritmo finalizzato ad aumentare il throughput non potrà agire sulla velocità di trasmissione dei simboli ma sul numero di sottoportanti oppure sulla tipologia di costellazione utilizzata per modulare (passando ad esempio da un bpsk che trasferisce un bit per simbolo a 16psk che ne trasferisce 4). Altro requisito fondamentale che verrà approfondito nella sezione dedicata alla sincronizzazione in frequenza è la necessità di avere un ottimo allineamento fra trasmettitore e ricevitore.

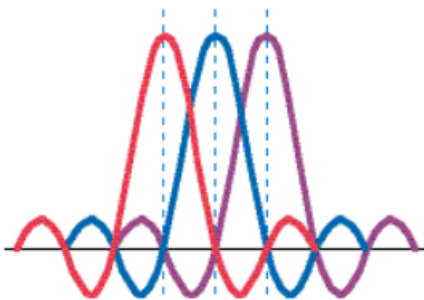


Figura 2.1: Ortogonalità sottoportanti OFDM [27]

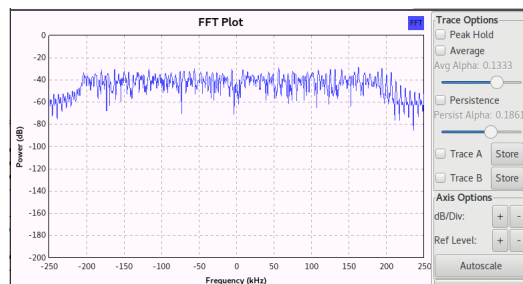


Figura 2.2: Trasmissione OFDM

• 2.1.2 Tempi di guardia

OFDM può soffrire di ISI (inter symbol interference). Questo problema avviene quando un simbolo trasmesso arriva al ricevitore assieme al precedente facendo fallire la decodifica, per risolvere questo problema viene aggiunto un tempo detto di guardia lungo solitamente attorno ad $1/10$ del simbolo. Inizialmente durante questo breve intervallo non veniva trasmesso nulla, successivamente è risultato più efficiente trasmettere l'ultimo pezzo del simbolo successivo (cyclic prefix) favorendo la corretta decodifica ricevitore.

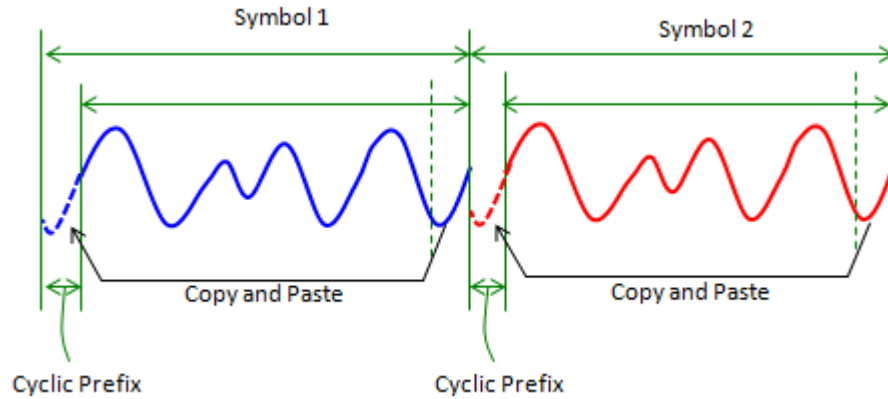


Figura 2.3: Cyclic guard [6]

• 2.1.3 Equalizzazione

L'equalizzazione del segnale ricevuto è una procedura fondamentale per la corretta decodifica delle informazioni. L'obiettivo di questa procedura è quello di modificare il segnale ricevuto cercando di agire esattamente nel modo opposto di come è stato distorto dal canale in modo da annullarne la distorsione, per ottenere tale risultato esistono numerosi algoritmi di equalizzazione che differiscono per gli approcci diversi in relazione alle informazioni disponibili sul mezzo trasmissivo oppure alle informazioni ottenute durante la trasmissione stessa. Esistono due tipi di equalizzazione possibile uno nel dominio del tempo quindi analizzando lo scorrere dei simboli e uno nel dominio delle frequenze che analizza il comportamento del canale nelle varie sottoportanti. All'inizio di una trasmissione OFDM vengono inviati dei preamboli noti al ricevitore che li utilizza per stimare la distorsione del canale di trasmissione, il ricevitore aggiusta le proprie previsioni anche mentre sta ricevendo grazie ai piloti contenuti in ogni simbolo OFDM. Da notare che l'equalizzatore opera variando in egual modo sia il segnale che il rumore.

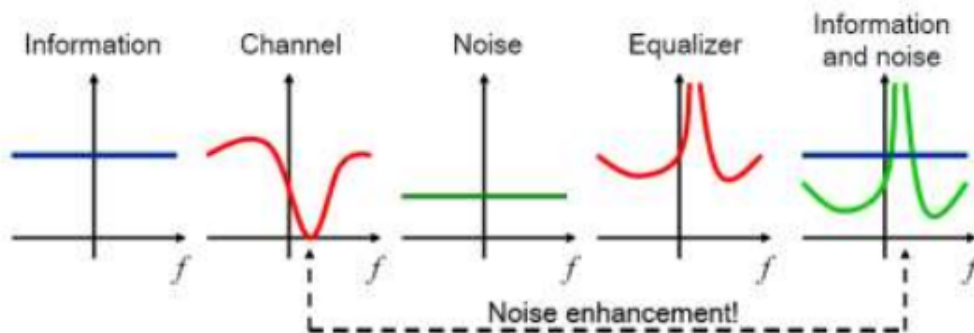


Figura 2.4: principio funzionamento equalizzatore [12]

• 2.1.4 Recupero degli errori

Individuare e correggere gli errori avvenuti nella trasmissione è un compito complesso. Un sistema OFDM necessita la correzione di errore per essere robusto.

CRC (Cyclic Redundancy Check)

CRC è una tecnica di controllo dell'errore, il funzionamento si basa sull'aggiunta alle informazioni da controllare di un breve codice di controllo. CRC non è in grado di correggere un eventuale errore e non è adatto per verificare la correttezza delle informazioni da un eventuale manomissione. Le informazioni di controllo aggiunte da CRC al messaggio iniziale rappresentano una divisione fra il cosiddetto polinomio generatore (specifico per ogni variante CRC) e il polinomio generato dal messaggio a cui viene sottratto il resto. Anche se di complessa spiegazione questa operazione può essere interamente eseguita in hardware mediante shift register e xor risultando molto efficiente. Il ricevitore eseguirà la divisione fra il polinomio generatore e il messaggio verificando che il resto sia nullo, se così non fosse il ricevitore è certo che sia presente almeno un errore nelle informazioni ricevute. [5]

Convolutional Coding

OFDM può utilizzare CC (Convolutional coding) per la determinazione e a differenza di CRC la correzione dell'errore. Il principio di funzionamento di questo algoritmo si basa sulla creazione di un diagramma a stati che permette di codificare non solo la sequenza di bit in ingresso ma anche la loro transizione di stato. Il rapporto fra quantità di bit in ingresso e quello in uscita è detto code rate e può variare a seconda delle circostanze, un code rate di $1/2$ ad esempio aggiunge 1 bit ogni bit in ingresso mentre con un rapporto $4/5$ viene aggiunto un bit ogni 4. Meno bit aggiunti si traduce in meno bit da inviare ma minore efficacia nella correzione d'errore [4]. Spesso in OFDM la tecnica di Convolutional coding viene utilizzata assieme ad altre tecniche di recupero errore più complesse come ad esempio Reed-Solomon in grado di recuperare ulteriormente informazioni danneggiate. E' bene puntualizzare che esiste un limite, matematicamente dimostrato, insuperabile alla quantità di informazioni trasferibili su un canale affetto da rumore, questo limite è detto di Shannon. [18]

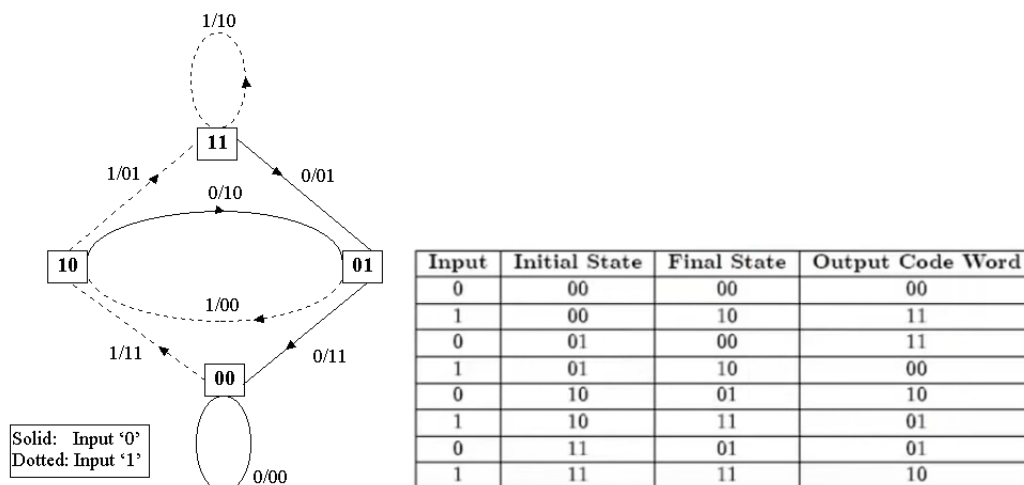


Figura 2.5: Diagramma a stati e tabella utilizzati per un implementazione di un codificatore CC con output di lunghezza doppia rispetto all'input.

• 2.1.5 Sincronizzazione in frequenza

Lo sfasamento in frequenza fra trasmettitore e ricevitore è un problema comune a tutte le telecomunicazioni. OFDM ne è particolarmente sensibile data la necessità di mantenere l'ortogonalità fra le sottoportanti, altrimenti si manifestano fenomeni non desiderati come l'ICI (Inter Carrier Interference) oppure sfasature del segnale modulato. Le cause di una cattiva sincronizzazione possono essere varie, una possibile origine è l'effetto Doppler causato dal movimento di un apparato rispetto all'altro durante la comunicazione. Per le trasmissioni wireless si verifica il cosiddetto multi-path, inoltre possono verificarsi imperfezioni sul clock (generatore di frequenza) fra gli apparati.

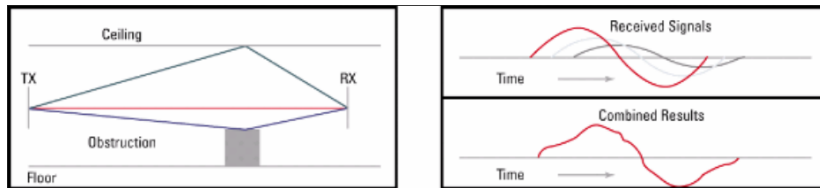


Figura 2.6: Multipath propagation [27]

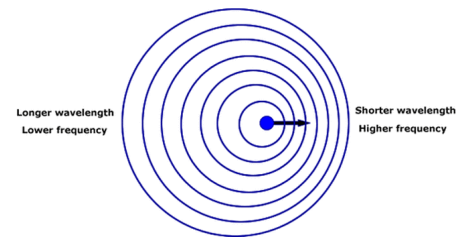


Figura 2.7: Effetto Doppler [10]

Tipologie di sottoportanti

Le sottoportanti in OFDM sono classificabili in tre tipologie. La maggior parte sono adibite al trasporto delle informazioni, le restanti sono utilizzate per l'invio di simboli pilota. Esiste un'ultima sottoportante particolare che non trasmette nulla (nemmeno la frequenza portante) situata a metà fra tutte le altre che serve al ricevitore per sapere l'esatto centro dell'intera banda.

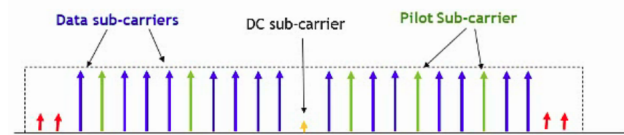


Figura 2.8: Tipologie sottopottanti OFDM [28]

Trasmissione Preambolo

L'invio di dati in OFDM è preceduto da informazioni che vengono utilizzate per la sincronizzazione dal ricevitore. Queste informazioni includono un preambolo e in alcune applicazioni un simbolo contenente alcuni parametri come il numero di simboli OFDM in arrivo.

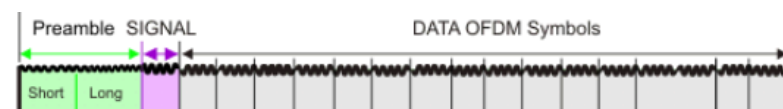


Figura 2.9: Sequenza invio OFDM [28]

Le sfasature in frequenza si dividono principalmente in due tipi, la prima è detta CFO (Carrier Frequency Offset) e rappresenta la sfasatura rispetto alle sottoportanti tra trasmettitore e ricevitore mentre la seconda è nota come SFO (Sampling Frequency Offset) e indica l'errore nella frequenza di campionamento. Il problema del CFO si manifesta in una sfasatura dei campioni ricevuti e viene corretto sincronizzando il ricevitore utilizzando il preabolo (la sua lunghezza

determina la precisione). L'SFO si manifesta in una sfasatura dei punti sulla costellazione ed è corretto utilizzando i simboli pilota sempre presenti in alcune specifiche sottoportanti. [13]

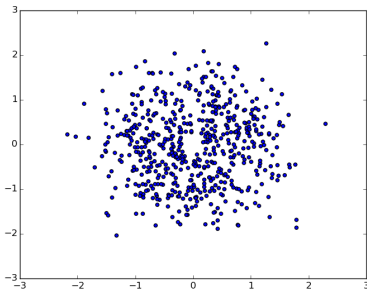


Figura 2.10: Costellazione con dati grezzi ricevuti 16QAM

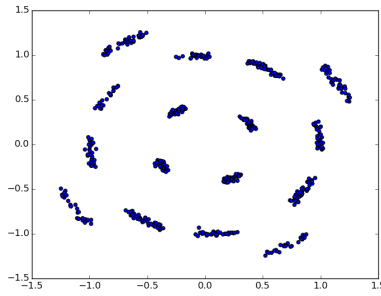


Figura 2.11: Corretta da CFO con preamble corto

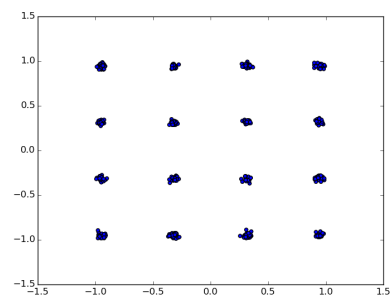


Figura 2.12: Corretta da CFO e SFO

[13]

Proprietà e campi di utilizzo

OFDM grazie all'ortogonalità delle portanti permette di avere un'ottima efficienza sull'utilizzo della banda, inoltre la scelta (statica o dinamica) del tipo di modulazione da utilizzare in ogni sottoportante lo rende adatto sia in situazioni dove è presente un canale molto distorto che in casi in cui è desiderabile un elevato throughput. Grazie ai tempi di guardia fra i simboli trasmessi e al prefisso ciclico (Cyclic prefix), OFDM risulta robusto contro il problema della sovrapposizione sul segnale in ricezione di componenti provenienti da segnali riflessi (multipath propagation). OFDM richiede un'ottima sincronizzazione e risulta quindi sensibile all'effetto Doppler. OFDM soffre inoltre di un elevato PAPR (peak to average power ratio) causato dalla caratteristica di avere le sottoportanti in quadratura. L'ortogonalità garantisce di non avere sovrapposizioni sulla frequenza dove si trasmette un simbolo, ma ciò non si verifica nelle frequenze intermedie fra una sottoportante e l'adiacente. Accade così che in particolari circostanze le code dei simboli, su sottoportanti adiacenti, finiscano per sommarsi creando un picco di ampiezza molto superiore alla media. Questo fenomeno influisce nel dimensionamento degli apparati che devono essere scelti per non saturare il segnale ma allo stesso tempo che non siano sprecati funzionando sotto alla metà della potenza. [19]

- **ADSL** La connessione adsl avviene mediante doppioli lunghi anche qualche chilometro. Il doppiolo di rame presenta fisicamente una resistenza (prevedibile con la seconda legge di ohm) che è posizionata in serie al segnale, inoltre il doppiolo possiede un'induttanza. Quando è presente un segnale (corrente alternata) il doppiolo si comporta come un filtro attenuando, tale effetto si amplifica all'aumentare della distanza e della frequenza. OFDM viene utilizzato in questo campo proprio perché non risente molto della differenza di attenuazione fra le sottoportanti del canale trasmissivo.
- **Powerline** I dispositivi powerline utilizzano l'impianto elettrico come mezzo trasmissivo, viene utilizzato OFDM per la presenza di un canale molto variabile e soggetto a disturbi esterni non prevedibili.
- **Wlan, WiMAX** OFDM viene utilizzato in due fra i principali standard per la trasmissione di internet senza fili. Fornisce robustezza contro il problema della multipath propagation oltre ad un ottimo range di scelta sulle modulazioni. Si presta molto bene sia per situazioni di bassa qualità del canale sia in situazioni di stabilità dove si vuole ottenere un buon throughput.
- **Radio e televisione digitali** La televisione pubblica italiana, come molte di quelle europee, viene trasmessa secondo lo standard DVB-T (Digital Video Broadcasting-Terrestrial)

che sfrutta OFDM per inviare un flusso contenente i vari canali televisivi già compressi e provvisti di trame per la decodifica. La radio digitale DAB (Digital Audio Broadcasting) suddivide invece le stazioni in blocchi contenenti una decina di radio ognuno. Ogni blocco viene poi trasmesso utilizzando OFDM in bande diverse.

SDR (Software Defined Radio)

Tradizionalmente gli apparati per le telecomunicazioni vengono implementati in hardware, lo sviluppo risulta molto costoso quindi finisce per essere svolto da poche persone. Progettare in hardware richiede molto tempo ed il risultato è un sistema affidabile ma, con un compito specifico difficile da aggiornare o modificare una volta prodotto. Recentemente, con l'aumento della potenza di calcolo, è finalmente possibile svolgere con il software compiti precedentemente svolti da hardware specializzato. L'SDR è una scheda che contiene l'hardware aggiuntivo necessario ad un computer per poter ricevere e/o trasmettere informazioni. Un grande vantaggio dell'utilizzo di queste piattaforme è la possibilità di implementare la tecnica di trasmissione favorita con in aggiunta la possibilità di variare tutti i parametri tecnici (es. frequenza, larghezza di banda, frequenza di campionamento, ecc.). Un'applicazione interessante di questa tecnologia è la creazione di un sistema dinamico in grado di far variare la frequenza ed il metodo di trasmissione per adattarsi alla situazione presente nel migliore dei modi. Esistono diverse tipologie di SDR, i più economici costano appena una decina di euro e, seppure con qualche limitazione, riescono a ricevere fino a quasi 2GHz. Versioni più costose sono in grado anche di trasmettere contemporaneamente su un range di frequenza e sample rate più elevati. Da sottolineare che ogni diversa frequenza su cui si intende trasmettere-ricevere richiede una specifica antenna e che non esiste un'antenna generica. Il costo per una scheda SDR da laboratorio si aggira dai 500 ai 2000 euro ma è destinato a scendere visto che il suo vero valore, inclusa ricerca e progettazione, è stimato d'essere un quarto [25]. Gli SDR sono paragonabili alle schede audio, la differenza sta nell'essere in grado di effettuare molti più campionamenti (2-50Mhz contro 40-200khz).

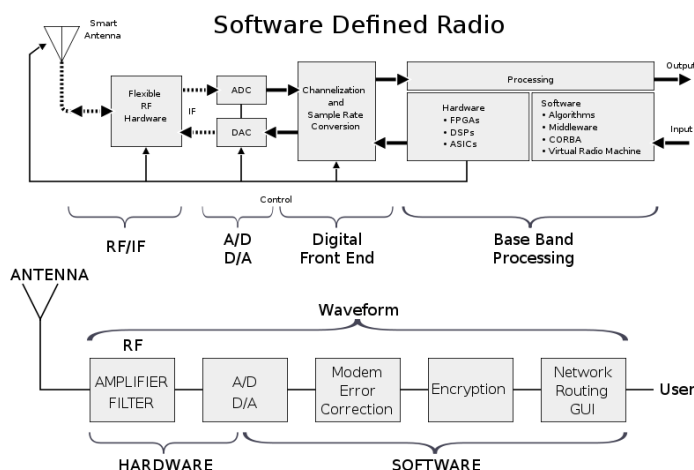


Figura 3.1: Diagramma blocchi funzionamento SDR [8]

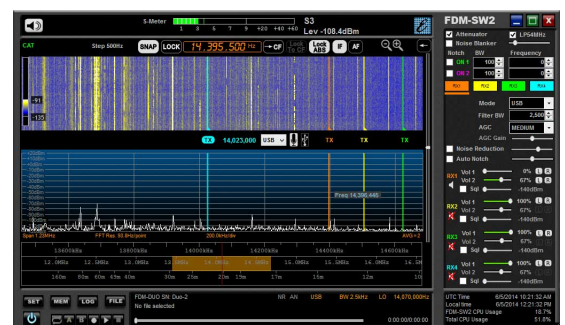


Figura 3.2: Software generico per l'analisi dello spettro [26]

- **3.0.1 USRP (Universal Software Radio Peripheral)**

URSP è una tipologia di SDR venduta dal marchio Ettus Research pensata per essere accessibile a tutti. Alcune versioni contengono un processore su cui può essere caricato del software per funzionare in autonomia. Sono disponibili anche modelli con una scheda ethernet integrata per il controllo remoto attraverso una rete locale o pubblica. Gli USRP permettono la trasmissione e la ricezione contemporaneamente e dispongono di un ampio range di frequenze che varia in relazione al modello ma è di gran lunga più elevato rispetto agli sdr-rtl. Le schede USRP sono utilizzabili con il driver UHD, disponibile sottoforma di due blocchi: uno per la ricezione ed uno per la trasmissione, pronti per essere integrati nel proprio flusso su Gnuradio.

Ettus USRP-B210

Questo modello del costo di 1100 euro in dotazione ai laboratori dell' università permette di variare la frequenza da 70MHz a 6GHz e raggiungere una frequenza di campionamento di 56MHz. Supporta MIMO in full-duplex

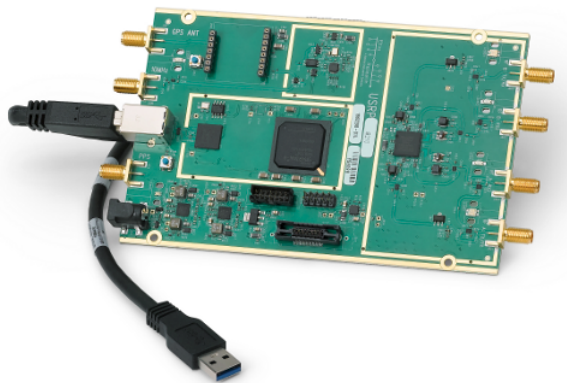


Figura 3.3: USRP modello B210[29]

- **3.0.2 RTL-SDR**

Questa particolare tipologia di SDR sono i più economici presenti sul mercato, vengono venduti come decoder per lo standard DVB-T della televisione digitale e per DAB e FM della radio. Utilizzando driver alternativi è possibile ricevere un flusso di campionamenti dalla scheda. Questa tipologia di SDR è solo in grado di ricevere il segnale, inoltre presenta limitazioni sia sulla frequenza di ricezione (minore di 2GHz) che sulla frequenza di campionamento (minore di 2MHz per una lettura affidabile, fino a 3Mhz).



Figura 3.4: RTL-SDR venduto come ricevitore DVB-T, DAB e FM[24]

Gnuradio

GNU Radio è una piattaforma gratis e open-source per lo sviluppo di codice finalizzato all'implementazione di software radio. Lo sviluppo può essere eseguito sia utilizzando schede hardware esterne oppure simulando lo scenario virtualmente. Il progetto GnuRadio è nato nel 2001 con l'idea di portare il concetto di free-software anche nel mondo delle Software Defined Radio visto che in precedenza questo settore era dominato solo da software proprietari. Attualmente è molto utilizzato sia in ambito accademico che commerciale e negli ultimi anni anche nel mondo hobbistico. Lo sviluppo in GnuRadio consiste nella disposizione di una serie di blocchi ciascuno dei quali svolge un'operazione ben precisa. Il collegamento fra i blocchi è monodirezionale e rappresenta il flusso delle informazioni dalla sorgente al pozzo finale.

Il programma ufficiale dispone di molti moduli per l'elaborazione delle informazioni e l'interazione esterna, tuttavia è possibile creare un proprio blocco da inserire nel flusso con il linguaggio python oppure C++. Per creare un blocco personalizzato Gnuradio mette a disposizione pyBOMBS il cui compito è quello di preparare l'ambiente per lo sviluppo e l'integrazione del blocco scaricando e configurando le librerie necessarie senza che l'utente debba preoccuparsene. Per la creazione di tutti i file necessari per il funzionamento di un blocco viene fornita l'utilità `gr_modtool` che attraverso riga di comando guida l'utente sulla configurazione dei parametri necessari. Un modulo Gnuradio deve possedere delle caratteristiche standard stabilite per il corretto funzionamento. Ogni blocco deve specificare che tipologia di valori aspetta in ingresso ed in uscita (Int, Float, Complex, ecc.), quanti ingressi e uscite fornirà, la lista dei parametri da richiedere all'utente per l'utilizzo e il rapporto fra il numero di campioni in ingresso e quello in uscita. I blocchi dunque devono avere un rapporto costante fra input e output rappresentato da una costante. Il tool `gr_modtool` permette la creazione (se lo sviluppatore lo specifica) di un file python per il testing (Quality assurance). Eseguendo il file di test viene simulato un semplice diagramma come se fosse stato creato nell'ambiente grafico, questo diagramma è personalizzabile e permette di aggiungere tutti i blocchi necessari. Solitamente è sufficiente avere un diagramma composto da tre blocchi: il primo per fornire le informazioni al blocco sotto test, il blocco stesso ed infine un blocco che ritorna i risultati ottenuti permettendo al codice di confrontarli con quelli desiderati. Una volta completato lo sviluppo è possibile compilare il blocco per renderlo utilizzabile all'interno dell'ambiente Gnuradio.

Crittografia RSA

L'algoritmo RSA (Rivest-Shamir-Adleman) è un algoritmo per la crittazione ampiamente utilizzato che basa il suo funzionamento sulla difficoltà di fattorizzare un numero generato moltiplicando due numeri primi grandi. La base matematica dell'algoritmo venne pubblicata nel 1976 da due matematici Diffie e Hellman, famosi per aver inventato l'algoritmo Diffie-Hellman, utilizzato ancora oggi per instaurare una crittografia a chiave simmetrica ma senza la trasmissione della chiave. L'algoritmo RSA venne pubblicato nel 1977 tuttavia era già stato segretamente documentato da un matematico militare britannico qualche anno prima ma venne mantenuta la notizia segreta fino al 1997.

Caratteristiche

– 5.1.1 Codifica asimmetrica con doppia chiave

Le due chiavi sono dette privata e pubblica e vengono generate dallo stesso dispositivo, poi viene pubblicata solo quella pubblica. RSA è un algoritmo a chiave asimmetrica che utilizza quindi due chiavi distinte per la procedura di codifica e decodifica, a differenza degli algoritmi a chiave privata condivisa in RSA. La chiave privata è posseduta solo da uno dei due soggetti della comunicazione rendendone più difficile l'ottenimento da parte di un eventuale attaccante.[22]

– 5.1.2 Algoritmo unico

RSA utilizza lo stesso algoritmo per la codifica e la decodifica delle informazioni.[22]

– 5.1.3 Chiavi intercambiabili

E' possibile utilizzare le chiavi nell'ordine preferito, ad esempio durante l'invio di un messaggio il trasmettitore lo codificherà con la chiave pubblica del ricevitore mentre per le firme digitali il trasmettitore cripterà un hash del messaggio con la propria chiave privata permettendo al ricevitore di verificare l'identità.[22]

– 5.1.4 Procedimento creazione chiave monodirezionale

E' computazionalmente semplice generare la chiave pubblica partendo da quella privata mentre il contrario è proibitivo. Non è matematicamente dimostrata l'impossibilità della scoperta di un algoritmo che renda la procedura inversa efficiente.[22]

Algoritmo

– 5.2.1 Generazione delle chiavi

Il primo passo consiste nel generare la chiave privata e quella pubblica. Tutta la seguente procedura viene effettuata solo su un dispositivo.

- * vengono scelti due numeri primi molto grandi con lunghezza simile e ne viene eseguita il prodotto $n = p * q$, n sarà il modulo utilizzato nell' algoritmo di codifica/decodifica
- * viene calcolato $f(n) = (q - 1) * (p - 1)$
- * viene scelto un numero e tale che $1 < e < f(n)$ e che $MCD(e, f(n)) = 1$
- * viene calcolato $d = e^{-1} \text{ mod } f(n)$ utilizzando l'algoritmo di euclide

La chiave pubblica sarà formata dalla coppia (e,n) mentre quella privata (d,n). [23]

– 5.2.2 Cifratura

La cifratura verrà eseguita dal mittente utilizzando la chiave pubblica resa nota dal destinatario (e,n) calcolando $C = M^e \text{ mod } n$. [23]

– 5.2.3 Decifratura

La decifratura verrà eseguita dal destinatario con la propria chiave privata (d,n) decodificando il messaggio $M = C^d \text{ mod } n$. [23]

OFDM in Gnuradio

La prima parte del lavoro svolto è consistita nell'implementazione di OFDM nell'ambiente GnuRadio. Il funzionamento di OFDM necessita di vari meccanismi complessi come l'assegnazione di informazioni alle sottoportanti, le correzioni in frequenza, l'equalizzazione e la correzione d'errore come precedentemente spiegato nella parte teorica. Lo svolgimento di queste operazioni sono rese possibili dalla presenza sia di blocchi generici utili ad esempio per la correzione d'errore che di blocchi disponibili specificamente per l'implementazione di OFDM. Per una comunicazione standard OFDM in Gnuradio non è necessaria la scrittura di algoritmi, il lavoro consiste nel collegamento e nella configurazione dei parametri al fine di fare comunicare tutto nella maniera corretta. La comunicazione è divisa in due parti, una per la trasmissione che ha il compito di generare campioni per il driver dell'usrp-sdr ed una per la ricezione che partendo dai campionamenti effettuati dall'rtl-sdr decodifica le informazioni. Di seguito verranno documentate parte per parte tutte le fasi necessarie. E' stato scelto di trasferire un file audio in formato wma perchè non effettuando compressioni non viene influenzato dalla mancanza di campionamenti. L'unico problema si verifica eventualmente quando i campioni persi fanno parte della piccola sezione riservata ai metadati. Un pacchetto perso risulterà in circa mezzo millisecondo di audio mancante. La frequenza scelta per la trasmissione è 915Mhz che risulta disponibile per uso civile in Italia [1], la scelta non ha potuto comprendere 2.4ghz e 5ghz visto che l'Sdr economico utilizzato in ricezione arriva a massimo 2Ghz.

Trasmittitore

La trasmissione è composta da quattro sezioni principali connesse fra loro.

– 6.1.1 Formazione pacchetti dal flusso in ingresso

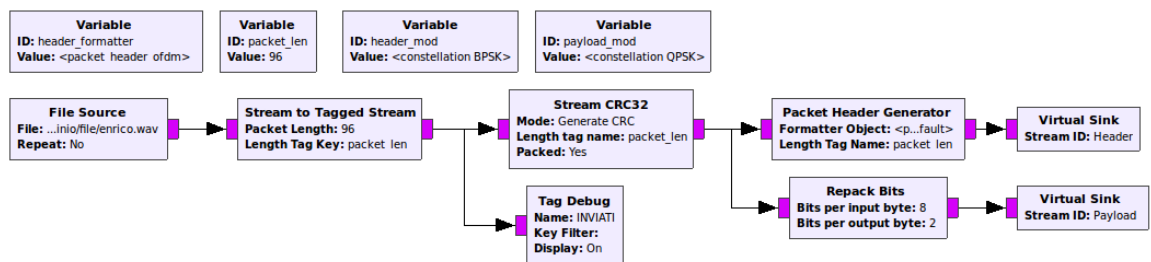


Figura 6.1:

Le informazioni vengono lette da file e fornite dal blocco "File Source" 8bit alla volta, questi byte vengono aggregati in pacchetti da 96 tramite l'aggiunta di un tag. A questo punto il blocco "CRC32" appende il codice di controllo per permettere al ricevitore di verificare la presenza di errori. Le informazioni vengono processate sia dal blocco "Repack bits" che ha il compito di generare quattro nuovi byte ognuno dei quali con soli due bit dell'input (la costellazione qpsk permette l'invio di 2 bit alla volta quindi i 6 restanti lasciati a zero e verranno ignorati) che dal blocco "Packet Header generator". Quest'ultimo ha il compito di generare gli header dei pacchetti che permettono al ricevitore di riconoscere l'inizio di una sequenza di simboli OFDM. Gnuradio mette a disposizione degli oggetti "costellation" che contengono informazioni utili relative alle caratteristiche delle costellazioni, ad esempio il numero di bit significativi per la costellazione oppure la mappatura fra bit e punti delle costellazioni (Ad esempio l' oggetto digital.constellation_qpsk viene utilizzato nel modulo

”Repack bits” chiamando la funzione `bits_per_symbol()` che restituisce 2).

Le informazioni vengono poi trasferite a due ”Virtual Sink” che fungono solamente da collegamento logico verso i ”Virtual Source”, il loro scopo è quello di rendere la composizione più leggibile.

– 6.1.2 Codifica informazioni in Simboli

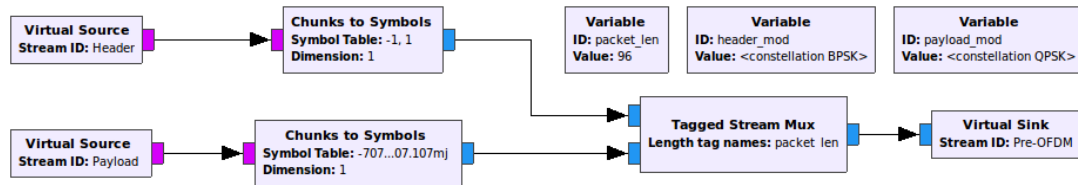


Figura 6.2:

Questa sezione del trasmettitore riceve due flussi dalla precedente contenenti i bit divisi fra header e payload. Il blocco ”Chunks to Symbols” convertono i bit (che ricordiamo sono già nel formato 1 bit significativo per byte per l’ header e 2 bit significativi per byte per il payload) in numeri complessi della rispettiva costellazione.

La mappatura fra bit significativi e numeri complessi viene fornita dall’oggetto di appoggio disponibile per le varie modulazioni fornito da gnuradio descritto brevemente nel punto precedente chiamando rispettivamente le funzioni `payload_mod.points()` e `header_mod.points()`. A questo punto è necessario unire in un solo flusso (mantenendo la divisione fedele ai blocchi iniziali) i punti delle costellazioni ottenuti ricalcolando il tag della lunghezza, questo lavoro viene eseguito dal blocco ”Tagged Stream Mux”.

– 6.1.3 Allocazione simboli sulle sottoportanti e creazione campioni per l’SDR

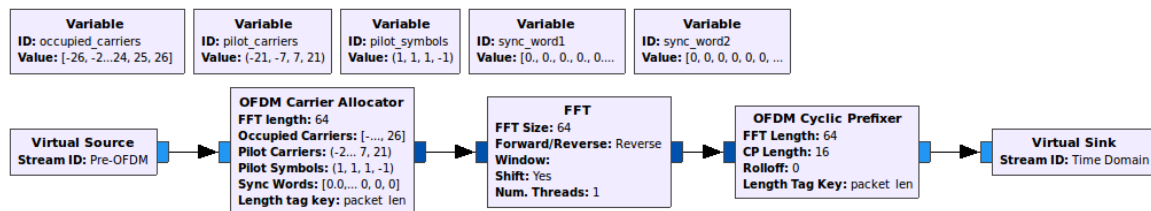


Figura 6.3:

L’obiettivo di questa sezione è quello di allocare sulle sottoportanti di OFDM le informazioni e creare i campioni da spedire nel dominio del tempo. Il blocco ”OFDM Carrier Allocator” si occupa di allocare i punti delle costellazioni e i simboli pilota (`pilot_symbols`) rispettivamente alle sottoportanti definite come `occupied_carriers` e `pilot_carriers`. Inoltre aggiunge all’inizio di ogni blocco trasmesso i `sync_words` utilizzati dal ricevitore per sincronizzazione ed equalizzazione.

L’output rappresenta un simbolo OFDM pronto per essere portato nel dominio del tempo dal blocco ”FFT” che esegue appunto la trasformata di fourier veloce. Il blocco ”OFDM Cyclic Prefixer” aggiunge il prefisso ciclico spiegato in precedenza.

– 6.1.4 Passaggio dati all'SDR

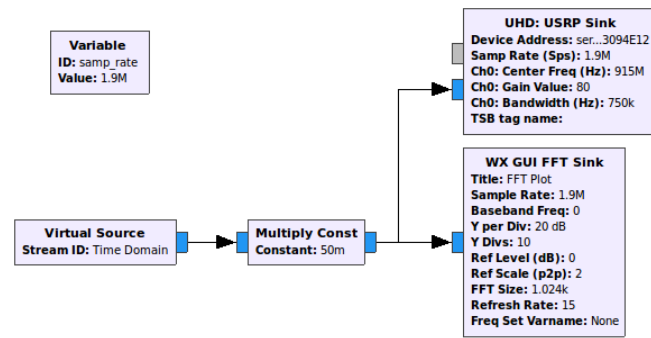


Figura 6.4:

L'ultima operazione consiste nel passare il simbolo OFDM al driver dell'SDR configurandolo con i parametri necessari per la trasmissione. Il blocco "WX GUI FFT Sink" serve per avere un feedback sulla trasmissione mostrando un grafico sul dominio delle frequenze (Riportato nella sezione "ortogonalità delle sottoportanti OFDM").

Ricevitore

La ricezione è composta da tre sezioni principali connesse fra loro.

– 6.2.1 Ricezione informazioni, sincronizzazione

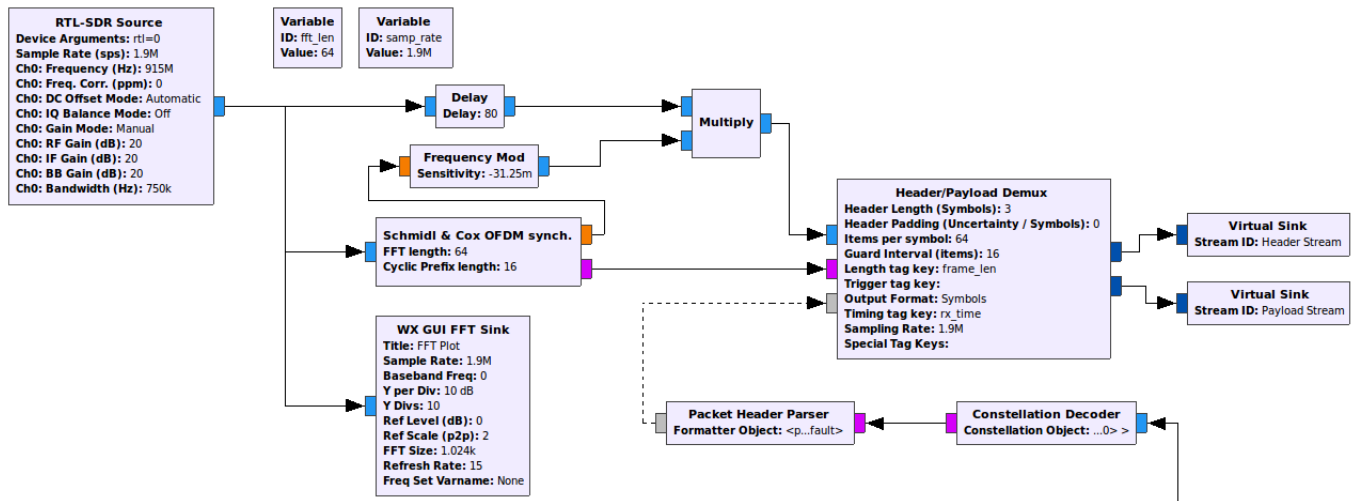


Figura 6.5:

Questa prima sezione del ricevitore ha il compito di leggere i campionamenti forniti dal driver per l'rtl-sdr e ottenere le informazioni nuovamente divise fra header e payload. Il blocco centrale di questa sezione è senza dubbio "Header/payload Demux" che ha proprio questo compito strettamente necessario per poi poter essere analizzati in seguito. I campionamenti sono forniti in forma complessa (I/Q) e rappresentano l'ampiezza del segnale in quell'istante ma anche l'andamento della funzione che lo ha generato. Il processo di ricezione ha inizio quando il blocco "Schmidl & Cox OFDM sync" individua l'inizio della trasmissione inviando un segnale di comando (trigger) verso il demuxer. Il suo funzionamento è molto complesso e permette assieme al blocco "Frequency mod" la sincronizzazione necessaria per l'inizio della lettura. Il blocco "Header/payload Demux" inoltra sulla prima uscita (Header Stream) i primi 3 simboli OFDM (composti da 64 campioni ognuno) e rimane in attesa fino

a che non gli vengono fornite le informazioni contenute nell'header. L'analisi dei campioni contenenti le informazioni sul payload verrà approfondita nella sezione successiva, per ora è importante notare che "Header/payload Demux" riceve queste informazioni sul terzo ingresso. A questo punto il demuxer grazie alle informazioni dell' header può inoltrare il numero corretto di campioni corrispondenti ai simboli OFDM sulla seconda uscita (Payload Stream).

– 6.2.2 Equalizzazione e ottenimento simboli

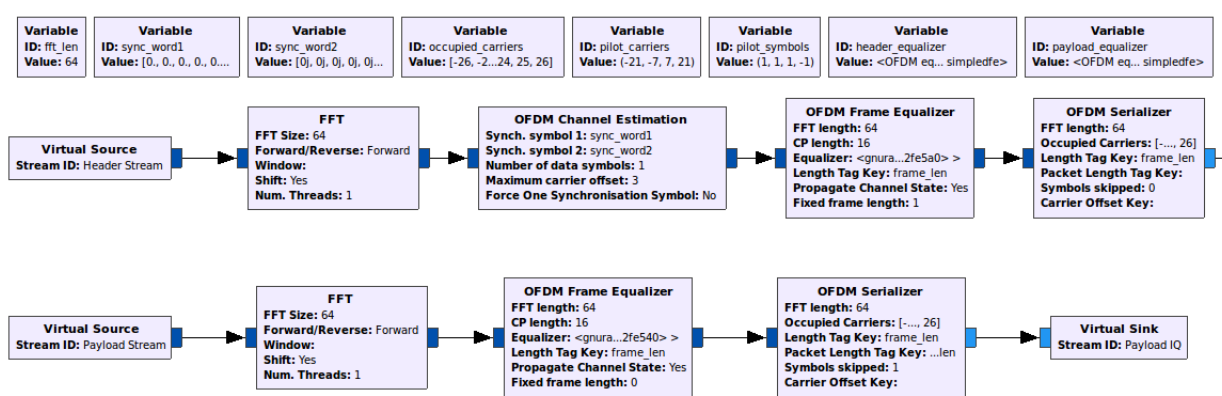


Figura 6.6:

L'obiettivo di questa sezione è quello di ottenere dai campionamenti ricevuti le costellazioni delle sottoportanti OFDM. La procedura consiste nell'applicazione della trasformata di Fourier veloce per passare al dominio delle frequenze. Nella decodifica dell' header è presente il blocco "Channel Estimator" che sfruttando i primi due sync_words ha lo scopo di ottenere informazioni di partenza sulle caratteristiche di sfasatura CFO (Carrier Frequency Offset) e di attenuazione canale. Il blocco successivo "OFDM Frame Equalizer" utilizza queste informazioni per effettuare la prima equalizzazione, le informazioni sulle caratteristiche del canale vengono poi aggiornate alla ricezione di ogni simbolo OFDM grazie ai simboli pilota contenuti nelle apposite sottoportanti. L'ultima operazione della sezione viene svolta dal blocco "OFDM Serializer" e consiste nell'invertire il lavoro svolto dall'allocatore nella fase di trasmissione al fine di ottenere un flusso contenente solo i punti delle costellazioni che contengono informazioni in modo ordinato e raggruppati secondo pacchetto di trasmissione (mediante l'aggiunta di un tag).

– 6.2.3 Decodifica payload e scrittura file

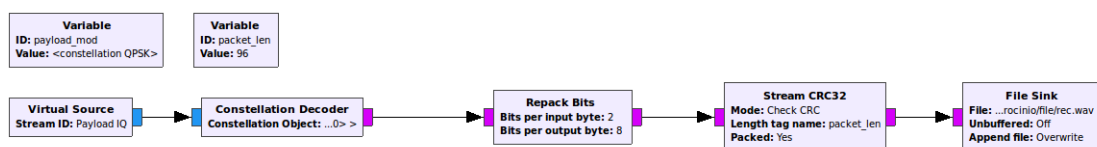


Figura 6.7:

Il blocco "Constellation Decoder" effettua la decodifica dei numeri complessi ricevuti. Le informazioni ottenute dalla decodifica di una singola costellazione in qpsk sono rappresentabili in 2 bit, è dunque necessario l'utilizzo del blocco "Repack bits" che preleva solo i due bit significativi di quattro decodifiche successive per completare un byte. Sono ancora contenute informazioni relative all'appartenenza ad un blocco coerenti con quelle iniziali in fase di trasmissione, "Stream CRC32" esegue la verifica pacchetto per pacchetto scartandolo se contenente errori. In conclusione il blocco "File Sink" si occupa della scrittura su file completando il processo di ricezione.

Testing

Caratteristiche generali del sistema sotto test

Sono stati effettuati alcuni test per osservare il comportamento in un ambiente reale. Le misurazioni sono state effettuate trasferendo un file audio contenente 1.9MB ad una distanza di due metri. La frequenza per la trasmissione è 915Mhz con un bandwidth di 750khz. L'alternativa sarebbe stata 433Mhz ma un breve test non ha rilevato sostanziali differenze nè sulla quantità dei dati ricevuti nè sulle risorse impiegate dai due computer.

Un altro dato interessante è che la quantità di cpu utilizzata per la parte di trasmissione, confrontata con quella per la ricezione, è risultata essere maggiore del 30% (il confronto è stato fatto rapportando la percentuale di cpu usata sui due computer in relazione al benchmark sul singolo core).

Parametri specifici

Nell'eseguire le misurazioni per i grafici G6.1 e G6.2 la trasmissione impiegava 9.2 secondi, il throughput era dunque di 1.6Mb/s con frequenza di campionamento di 1.8Msps. L'andamento del grafico G6.1 presenta una perdita di informazioni maggiore nella trasmissione anche se la qualità del segnale dovrebbe essere migliore. Questo comportamento può essere associato a due diversi fattori entrambi legati all'aumento della potenza in trasmissione: il multipath propagation e il precedentemente discusso problema del PAPR.

La determinazione della qualità del suono ricevuto (grafico G6.2) è stata effettuata facendo ascoltare i vari file ricevuti in maniera non ordinata a due persone distinte chiedendo di giudicarli da 0 a 5, considerando 5 la qualità del file originale e 1 la soglia minima sotto la quale risulta impossibile la comprensione. E' possibile osservare l'effetto della perdita dei pacchetti sull'audio ricevuto in Figura 6.8.

I grafici G6.5 e G6.4 sono utili per capire come la frequenza dei campionamenti giochi un ruolo fondamentale nel sistema. La sezione blu rappresenta il range dentro il quale si riescono a ricevere informazioni. Per valori di sample-rate minori di 1.7Msps il sistema non è in grado di ricevere informazioni. Questo è dovuto al limite matematico che impone di avere campionamenti in numero almeno doppio rispetto alla larghezza di banda utilizzata. Valori superiori di 2Msps non sono raggiungibili con l'SDR-RTL che compromette subito la comunicazione.

Durante l'esecuzione dei test, variando il sample-rate, è stato notato che superato il 104% di utilizzo della cpu sul computer in trasmissione avveniva un rallentamento dell'esecuzione del flusso in Gnuradio non generando tutti i campioni.

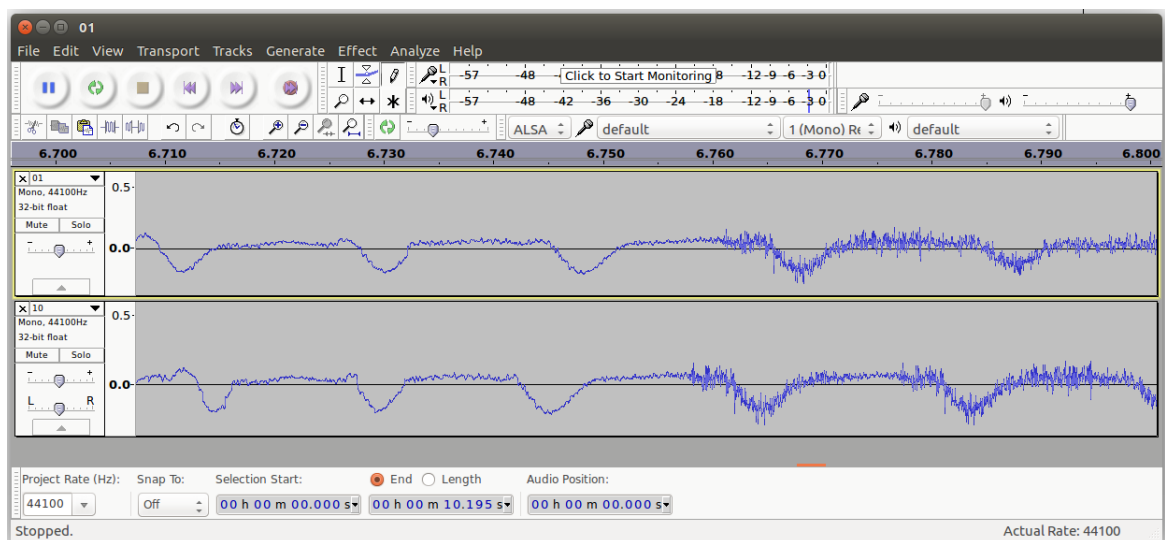
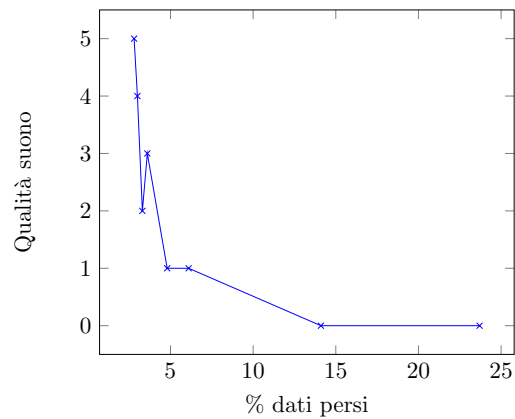
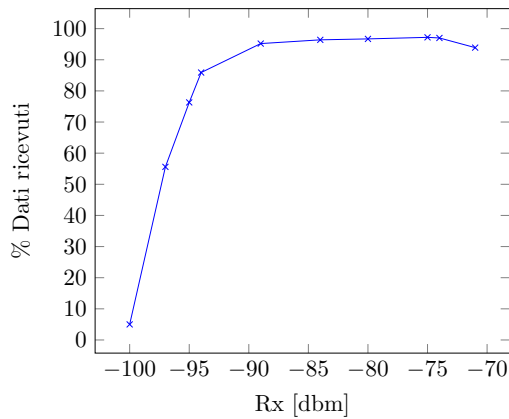
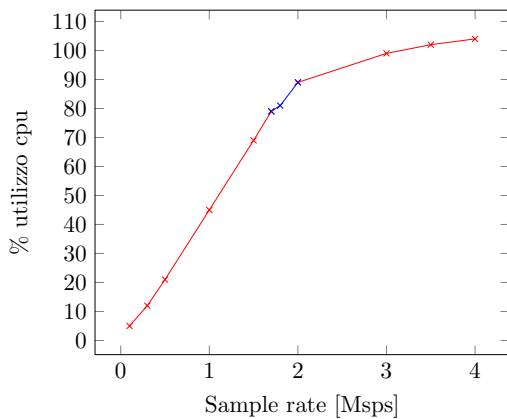


Figura 6.8: Confronto fra l'audio originale e quello ricevuto con perdita del 21%

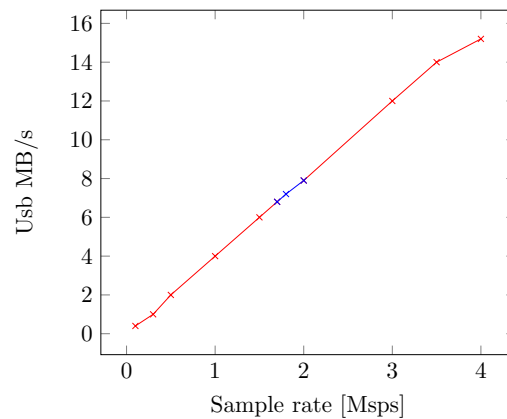
G6.1 Dati ricevuti in base alla potenza del segnale ricevuto G6.2 Qualità suono in relazione alla quantità di dati



G6.3 Utilizzo cpu all' aumento del sample-rate



G6.4 Utilizzo Usb all' aumento del sample-rate



Implementazione Rsa in Gnuradio

La seconda parte del lavoro svolto è stata quella di valutare la possibilità dell'aggiunta di una crittografia RSA al sistema di trasmissione OFDM.

Il primo passo è stato sviluppare un modulo che codificasse le informazioni passando poi all'implementazione di un secondo blocco per la decodifica.

Le chiavi utilizzate non sono assolutamente sicure, sono state scelte piccole solo per rendere facile la comprensione e seguire il funzionamento. In un applicazione reale sono necessarie chiavi di almeno 1024bit.

Sviluppo dei moduli

Lo sviluppo dei moduli è diviso in tre parti fondamentali descritte in seguito, per la creazione dei file di supporto e l'importazione delle librerie necessarie Gnuradio mette a disposizione un tool chiamato gr_moodtool utilizzabile da riga di comando.

– 7.1.1 Definizione proprietà moduli per l'ambiente grafico

In Figura 7.1 sono mostrati i file contenenti la definizione dei parametri necessari per l'interazione con l'ambiente. I blocchi sono costituiti da un ingresso per ricevere il flusso dati dal programma "Sink" ed un uscita per permettere la continuazione "Source", inoltre

si possono chiedere parametri aggiuntivi che devono essere forniti dall'utente nel formato corretto.

Al codificatore è richiesto sapere la coppia di valori che costituiscono la chiave pubblica del decodificatore mentre per la decriptazione è sufficiente la chiave privata.

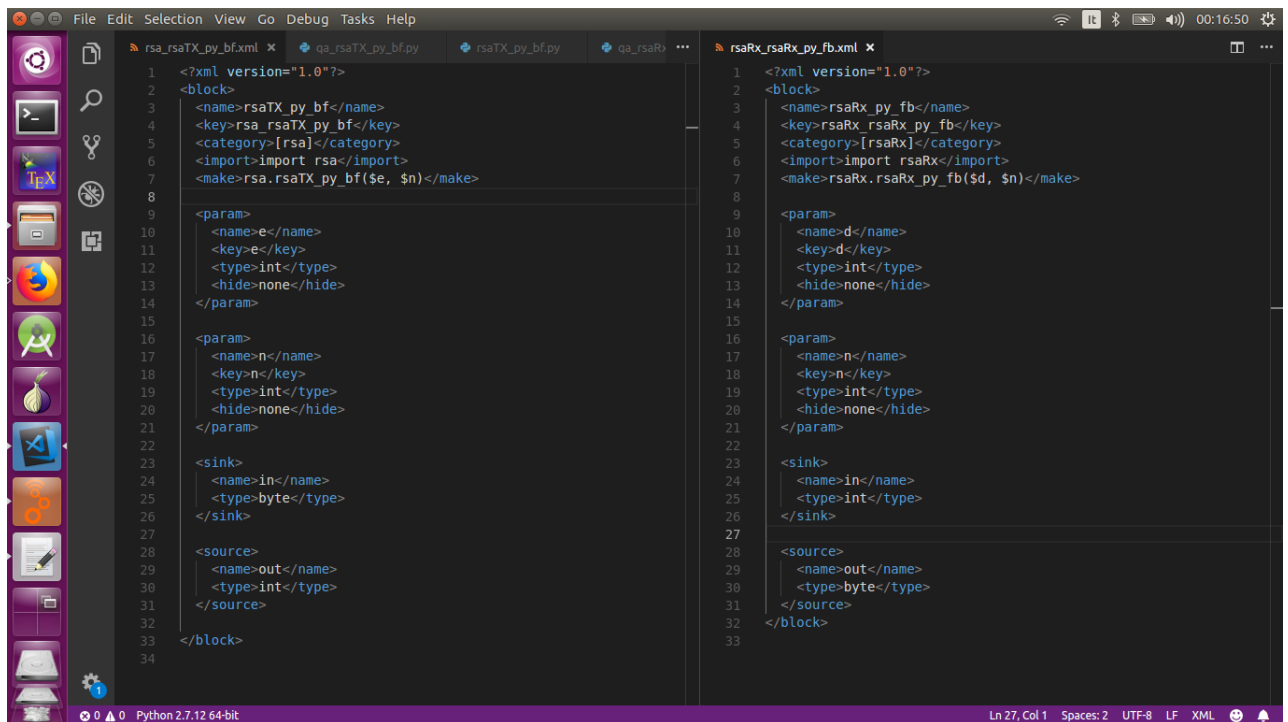


Figura 7.1: File contenenti la definizione dei parametri nell' ambiente grafico

– 7.1.2 Algoritmo per cifratura e decifratura

La codifica e la decodifica rsa si effettua utilizzando lo stesso algoritmo e fornendo la chiave corretta. Nell'implementazione di RSA è importante non effettuare subito l'elevamento a potenza in quanto può originare numeri enormi, l'approccio migliore consiste nel moltiplicare la base n volte calcolando il modulo ad ogni passaggio. Il collegamento logico fra i blocchi all'interno dell'ambiente gnradio al momento dell'esecuzione viene tradotto in un file python il cui compito è quello di gestire il flusso delle informazioni istanziando le classi dei blocchi necessari e successivamente chiamare il loro metodo "work" presente in ogni modulo. Lo sviluppo del proprio algoritmo deve essere implementato proprio all'interno della funzione work. Le informazioni vengono trasferite attraverso un vettore di vettori. Il primo indice rappresenta la numerazione dell'ingresso o dell'uscita mentre il secondo permette di gestire l'arrivo di più informazioni alla volta. Il blocco codificatore riceve 8 bit alla volta che vengono processati in python con il complemento a due. Il risultato della cifrazione non

```
def work(self, input_items, output_items):
    in0 = input_items[0]
    out = output_items[0]
    for j in range(0, len(in0)):
        init = int(0)
        init = long(in0[j])
        if long(in0[j]) < 0:
            init = ((init)%self.n_public_key) + 256
        res = init
        for count in range(1, self.e_public_key):
            res = (res*init)%self.n_public_key
        print ("messaggio codificato (tx): " + str(res) + " da: " + str(in0[j]+256 if in0[j] < 0 else in0[j]))
        out[j] = int(res)
    return len(output_items[0])
```

Figura 7.2: Algoritmo per il blocco di codifica

risulta più rappresentabile con 8bit e per la nostra analisi di fattibilità effettuata con chiavi deboli è stato sufficiente l'utilizzo di una variabile intera. L'applicazione in uno scenario reale implicherebbe la risoluzione di problematiche discusse nella sezione delle conclusioni. Il blocco Decodificatore riceve le informazioni codificate e le riporta nel formato originale.

```
def work(self, input_items, output_items):
    in0 = input_items[0]
    out = output_items[0]

    for i in range(0, len(in0)):
        resi = long(in0[i])%self.n
        res = resi
        for count in range(1, self.d):
            res = (res*(in0[i]%self.n))%self.n
        print ("messaggio decodificato (rx): " + str(res) + " da " + str(resi))
        out[i] = int(res)
    return len(output_items[0])
```

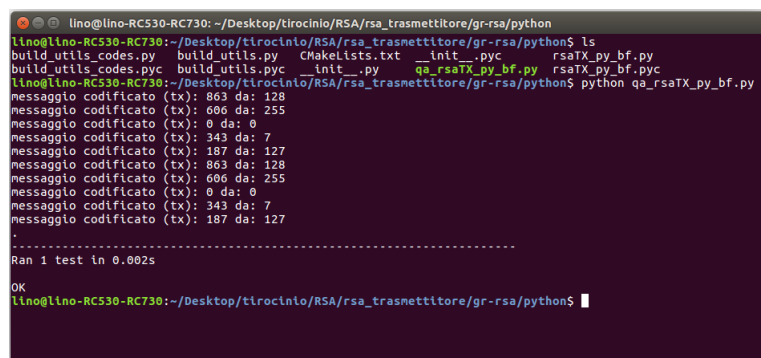
Figura 7.3: Algoritmo per il blocco di decodifica

– 7.1.3 Codice di test

Gnuradio permette la creazione di casi di test come spiegato nella sezione di introduzione teorica. Il testing è particolarmente utile in scenari come quello presente dove un eventuale errore nell'algoritmo verrebbe propagato ai blocchi successivi senza produrre segnalazioni. Le funzioni di testing sono utili anche durante lo sviluppo in quanto permettono di provare il codice scritto senza dover compilare tutto il modulo per renderlo disponibile nell'ambiente Gnuradio. Il testing viene effettuato istanziando la classe del blocco in analisi collegandolo ad una sorgente dati che si occuperà di passare i dati specificati ed un pozzo per riceverli una volta processati. Vengono inoltre impostati i parametri richiesti dal modulo (nel nostro caso riguardano i due valori di cui sono composte le chiavi). Infine vengono confrontati i risultati utilizzando delle funzioni apposite dette "assert" che restituiscono un messaggio di errore se falliscono. Nel testing del blocco di codifica i valori scelti (128,255,0,7,127) coprono tutti i casi limite del complemento a due.

```
def test_001_t (self):
    src_data = (128,255,0,7,127)
    expected_result = (863,606,0,343,187)
    src = blocks.vector_source_b (src_data)
    mult = rsaTX_py_bf (3,943)
    snk = blocks.vector_sink_i ()
    self.tb.connect (src, mult)
    self.tb.connect (mult, snk)
    self.tb.run ()
    result_data = snk.data ()
    self.assertEqual(expected_result, result_data)
```

Figura 7.4: Codice di test per il blocco di codifica



```
lino@lino-RC530-RC730: ~/Desktop/tirocinio/RSA/rsa_trasmettitore/gr-rsa/python$ ls
build_utils_codes.py  build_utils.py  CMakeLists.txt  __init__.pyc  rsaTX_py_bf.py
build_utils_codes.pyc  build_utils.pyc  __init__.py  qa_rsaTX_py_bf.py  rsaTX_py_bf.pyc
lino@lino-RC530-RC730:~/Desktop/tirocinio/RSA/rsa_trasmettitore/gr-rsa/python$ python qa_rsaTX_py_bf.py
messaggio codificato (tx): 863 da: 128
messaggio codificato (tx): 606 da: 255
messaggio codificato (tx): 0 da: 0
messaggio codificato (tx): 343 da: 7
messaggio codificato (tx): 187 da: 127
messaggio codificato (tx): 863 da: 128
messaggio codificato (tx): 606 da: 255
messaggio codificato (tx): 0 da: 0
messaggio codificato (tx): 343 da: 7
messaggio codificato (tx): 187 da: 127
.....
Ran 1 test in 0.002s
OK
lino@lino-RC530-RC730:~/Desktop/tirocinio/RSA/rsa_trasmettitore/gr-rsa/python$
```

Figura 7.5: Esecuzione codice di test

Importazione in Gruradio

Il passo finale consiste nel compilare i moduli e farli funzionare nell'ambiente grafico Gnuradio. Nella console inclusa, in basso nell'interfaccia utente, vengono stampati dei feedback generati dai moduli durante il loro lavoro. L'esempio riportato trasferisce un testo cifrandolo. Questo modello non è tuttavia ancora applicabile ad una comunicazione reale per le motivazioni discusse nelle conclusioni.

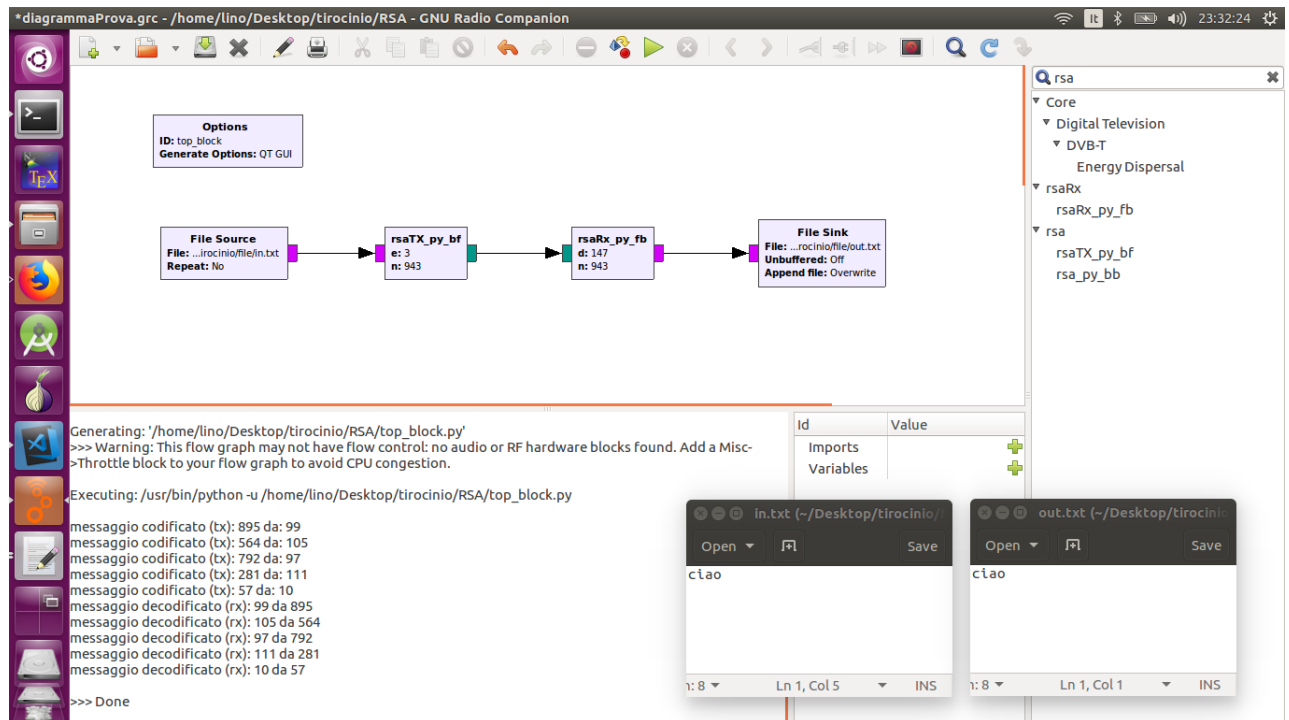


Figura 7.6: Codice di test per il blocco di codifica

Conclusioni

- Uno degli obiettivi del lavoro svolto è quello di analizzare il comportamento di un SDR economico in un utilizzo reale. Come prevedibile è stato necessario sfruttarlo al massimo delle potenzialità per ottenere una ricezione affidabile. La limitazione posta sulla frequenza di campionamento è appena sufficiente per superare il limite teorico sotto il quale non è possibile operare (Grafico 6.3). Durante il progresso del lavoro svolto è stato utilizzato anche un Ettus b210 per la ricezione (con le stesse variabili impostate nel sistema) che ha dimostrato di avere una notevole maggiore sensibilità e qualità dei campionamenti rispetto all'SDR-RTL. Queste piattaforme economiche sono comunque definitivamente in grado di operare anche in ambienti con tecniche di trasmissione avanzate garantendo una buona ricezione del segnale.
- Dai test effettuati, facendo ascoltare i campioni audio ricevuti, è emersa un interessante caratteristica sulla percezione umana della qualità di un suono. Infatti per piccole quantità di campioni persi (massimo 2-3%) la qualità viene ancora considerata buona mentre non appena si supera la soglia del 4% viene rapidamente alterata a tal punto che sopra al 10% non è possibile la comprensione delle parole. Questo risultato è perfettamente in linea con altri esperimenti riguardanti il dataloss consentito per uno streaming. [7]
- L'implementazione di OFDM in Gnuradio ha evidenziato un limite per quanto riguarda il sample-rate ottenibile. Per valori superiori a 4Msps la potenza di calcolo del processore sul singolo core fa da collo di bottiglia. Gnuradio dichiara di avere un sistema automatico in grado di distribuire il lavoro su più core quando è possibile. Una possibile spiegazione alla limitazione riscontrata potrebbe essere semplicemente una decisione presa da questo algoritmo automatico.
- Lo sviluppo dei blocchi per la crittografia RSA e la loro simulazione nell'ambiente Gnuradio hanno permesso di capire quali sono le problematiche da affrontare al fine di rendere questo sistema efficace in un utilizzo reale. Anzitutto le chiavi devono essere composte da 1024 - 2048 bit per non permettere ad un eventuale attaccante l'utilizzo di un attacco a forza bruta. La crittografia va fatta sull'intero blocco da 96byte scrivendo un algoritmo che faccia uso di una struttura dati per la memorizzazione dei byte necessari. Sarebbe possibile a tale scopo approfondire e utilizzare la predisposizione presente in Gnuradio per spedire vettori di byte. Nei sistemi di comunicazione moderni la codifica delle informazioni viene spesso implementata in una forma ibrida che consiste in uno scambio iniziale criptato con un algoritmo a chiave pubblica (ad esempio RSA) per la condivisione di una chiave simmetrica utilizzata poi per codificare le informazioni successive.
- Il costo ridotto degli SDR-RTL e la quantità di tutorial a disposizione per imparare a ricevere qualche tipo di segnale permetterà a mio avviso un fenomeno simile a quello avvenuto nel campo dell'elettronica dopo l'introduzione di Arduino. Esso infatti si distingueva dai microcontrollori disponibili fino ad allora proprio per le medesime caratteristiche. Gli SDR-RTL hanno tutte le potenzialità necessarie per permettere lo stesso passo avanti, avvicinando giovani menti ancora inesperte a questo mondo interessante.

Bibliografia

- [1] Allocazione-frequenze. <http://www.mise.gov.it/index.php/it/comunicazioni/radio/pnrf-piano-nazionale-di-ripartizione-delle-frequenze>.
- [2] Ask. https://it.wikipedia.org/wiki/Amplitude-shift_keying.
- [3] Constellation. <https://aerospaceresearch.net/?p=825>.
- [4] Convolutional-coding. <http://www.tdm.uni-oldenburg.de/2004/Material/faltung.htm>.
- [5] Crc. https://it.wikipedia.org/wiki/Cyclic_redundancy_check.
- [6] cyclic-prefix. http://www.sharetechnote.com/html/Communication_OFDM.html.
- [7] dataloss. https://en.wikipedia.org/wiki/Packet_loss.
- [8] Diagramma-sdr. https://it.wikipedia.org/wiki/Software_defined_radio#/media/File:SDR_et_WF.svg.
- [9] Dpsk. https://www.tutorialspoint.com/digital_communication/digital_communication_differential_phase_shift_keying.htm.
- [10] Effetto-doppler. <https://soundwavesreillymckennaaly.weebly.com/doppler-effect.html>.
- [11] Efficienza-spettrale. https://it.wikipedia.org/wiki/Efficienza_spettrale.
- [12] Equalizzazione. <https://pdfs.semanticscholar.org/d8f6/7c4e4a42164bed115af4610d22adc78afec3.pdf>.
- [13] Frequency-offset-ofdm. https://openofdm.readthedocs.io/en/latest/freq_offset.html.
- [14] Fsk. <http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-FSK.html>.
- [15] Modulazioni digitali. <http://www.digitaler-bos-funk.de/digital/digital.htm>.
- [16] Ofdm. <http://www.revolutionwifi.net/revolutionwifi/2015/3/how-ofdm-subcarriers-work>.
- [17] Ofdm. https://en.wikipedia.org/wiki/Orthogonal_frequency-division_multiplexing#Orthogonality.
- [18] Ofdm. https://en.wikipedia.org/wiki/Orthogonal_frequency-division_multiplexing#Guard_interval_for_elimination_of_intersymbol_interference.
- [19] Papr. <http://www.techplayon.com/papr-peak-average-power-ratio-matters-power-amplifier/>.
- [20] Qam. <https://www.radio-electronics.com/info/rf-technology-design/quadrature-amplitude-modulation-qam/what-is-qam-tutorial.php>.
- [21] Qpsk. <http://tingegneria.altervista.org/88/>.
- [22] Rsa. <https://retiavanzate.eu/>.
- [23] Rsa. <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-9900/rsa/testo.htm>.
- [24] Rtl-immagine. <https://shyamjos.com/assets/img/noaa/rtlsdr-dongle.jpg>.
- [25] Sdr-più-redditizio-della-droga. <https://zeptobars.com/en/read/AD9361-SDR-Analog-Devices-DAC-ADC-65nm>.
- [26] Sdr-software. https://www.wimo.com/elad-fdm-duo-sdr-transceiver_e.html.
- [27] Sottoportanti-ofdm. <https://www.webnews.it/2008/11/24/tecnica-ofdm/>.
- [28] Subcarriers. <http://www.teletopix.org/4g-lte/lte-sub-carrier/>.
- [29] Usrp-b210. <https://www.ettus.com/product/details/UB210-KIT>.