



UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA (CREMA)

Corso di Laurea in Sicurezza dei Sistemi e delle Reti Informatiche

Utilizzo di Software Defined Radio
per effettuare analisi di sicurezza su
protocolli di comunicazione wireless

RELATORE

Dott. Claudio A. Ardagna

CORRELATORE

Dott. Marco Anisetti

TESI DI LAUREA DI
Maurizio Agazzini
Matricola 875237

Anno Accademico 2016/2017

*Conosci tutti quelli che amo
La loro vita e la mia
Alcuni li hai visti arrivare
Altri andarsene via
Non è più baci sotto il portone
Non è più l'estasi del primo giorno
È una mano sugli occhi prima del sonno*

Prefazione

Sempre più sovente nel mondo della tecnologia vengono utilizzate tecnologie senza fili per la trasmissione di dati, sia nei dispositivi di largo consumo come i cellulari sia in quelli più di nicchia come i dispositivi IoT (Internet of Things). Fino a poco tempo fa le analisi di sicurezza su protocolli di comunicazione wireless erano molto complesse e dispendiose. Infatti, per poter effettuare tali attività, erano necessari strumenti costosi e conoscenze di elettrotecnica per la progettazione di hardware specifico che permettesse di comunicare utilizzando lo stesso layer di trasporto del dispositivo oggetto di analisi. Negli ultimi anni, tuttavia, sono stati sviluppati hardware “generici” che hanno la caratteristica di fornire un segnale “grezzo” a un computer che successivamente effettuerà a livello software tutte quelle operazioni che precedentemente venivano svolte tramite l’hardware (es. demodulazione, encoding). Questa tipologia di hardware è nota con il nome di Software Defined Radio (SDR). Oltre che per la progettazione e il collaudo di dispositivi radio, è anche possibile utilizzare le SDR per effettuare analisi di sicurezza su protocolli di comunicazione wireless.

Questo progetto si è concentrato sull’utilizzo delle SDR per l’analisi della sicurezza di protocolli wireless; il progetto mira ad individuare e approfondire i passi necessari per effettuare tale analisi.

Il lavoro è organizzato come segue: vengono presentati i principali concetti necessari per poter lavorare con le onde radio (Capitolo 2); successivamente (Capitolo 3) vengono esposti i possibili attacchi sui protocolli wireless. Vengono presentate le Software Defined Radio (Capitolo 4) e i tool che possono essere utilizzati per le analisi; infine (Capitolo 5) vengono presentati i casi di studio (case study) alla base del lavoro di analisi e vengono esposte alcune possibili contromisure (Capitolo 6).

Indice

1	Introduzione	1
2	Concetti base di segnali radio	3
2.1	Frequenze ISM e frequenze libere.....	4
2.2	Modulazione e demodulazione del segnale.....	6
2.2.1	Amplitude Shift Keying (ASK).....	6
2.2.2	ON OFF Keying (OOK)	6
2.2.3	Frequency Shift Keying (FSK).....	7
2.2.4	Phase Shift Keying (PSK)	7
2.3	Encoding.....	8
2.3.1	Non-Return-to-Zero (NRZ)	8
2.3.2	Pulse Width Modulation (PWM)	8
2.3.3	Pulse Interval and Width Modulation (PIWM).....	9
2.3.4	Manchester (o bi-phase)	9
2.4	Come è strutturato un segnale.....	10
2.5	Identificazione dei segnali mediante risorse pubbliche	11
3	Principali attacchi su protocolli wireless	13
3.1	Intercettazione	13
3.2	Spoofing	13
3.3	Replay attack	13
3.4	Jamming	14
3.5	Brute force	15
3.6	OpenSesame Attack	15
3.7	RollJam attack	16
3.8	Man In The Middle	17
4	Utilizzo delle Software Defined Radio per effettuare analisi di sicurezza ..	19
4.1	Software Defined Radio	19
4.2	Descrizione del materiale utilizzato.....	21
4.2.1	Terratec T Stick PLUS	21
4.2.2	HackRF One (https://greatscottgadgets.com/hackrf/)	21
4.2.3	Yard Stick One (https://greatscottgadgets.com/yardstickone/)....	21
4.3	Preparazione di un sistema per effettuare le analisi	22
4.4	Identificazione delle frequenze utilizzate mediante analizzatore di spettro	22
4.4.1	Memorizzazione del segnale ricevuto attraverso gqrx	25
4.5	Utilizzo del software GNU Radio (http://gnuradio.org/)	26

4.5.1	Salvataggio del segnale per successiva analisi	28
4.5.2	Replica di una trasmissione precedentemente ricevuta	29
4.6	Analisi del segnale mediante Inspectrum	30
4.6.1	Identificazione delle modulazioni mediante visualizzazione del segnale	31
4.6.2	Utilizzo di Inspectrum per l'identificazione della velocità di trasmissione	32
4.6.3	Identificazione del preambolo e delle sync word con Inspectrum.	33
4.7	Analisi del segnale mediante Universal Radio Hacker.....	34
4.8	Utilizzo di RFCAT per la ricezione e l'invio di dati.....	37
5	Case study analizzati	39
5.1	Telecomando per l'accensione di una presa elettrica	39
5.2	Sistema di sicurezza basato su protocollo X10	45
5.3	Sistema di chiusura centralizzata Fiat Stilo	51
5.4	Antifurto di nuova generazione	59
5.4.1	Telecomando di attivazione/disattivazione	60
5.4.2	Sensore magnetico	63
5.4.3	Sensore di movimento	64
5.4.4	Analisi del sistema anti-jamming	65
5.4.5	Analisi del protocollo di comunicazione.....	66
6	Discussione	69
6.1	Risultati.....	69
6.2	Possibili soluzioni ai problemi riscontrati.....	69
6.2.1	Intercettazione.....	69
6.2.2	Replay attack	70
6.2.3	Jamming.....	70
6.2.4	Brute force.....	70
6.2.5	OpenSesame	70
6.2.6	“RollJam” attack	71
6.2.7	Man In The Middle	71
6.3	Conclusioni	71
7	Bibliografia	72
8	Appendice A – Codice Sorgente	74
8.1	Codice per ricezione telecomando presa elettrica	74
8.2	Codice per ricezione telecomando X10	75
8.3	Codice per ricezione telecomando Fiat Stilo.....	77
8.4	Codice invio codice di sblocco sistema antifurto.....	78

8.5	Codice simulatore attacco brute force	79
8.6	Codice Jammer.....	79

1 Introduzione

In questo capitolo viene descritto lo scenario di partenza su cui si basa il presente lavoro di tesi: si cerca dunque di capire le motivazioni alla base della necessità di appositi strumenti per poter effettuare analisi sicurezza di protocolli wireless. Viene descritta infine l'organizzazione del lavoro.

Sempre più sovente nel mondo della tecnologia vengono utilizzate tecnologie senza fili per la trasmissione di dati, sia nei dispositivi di largo consumo come i cellulari, sia in quelli più di nicchia come i dispositivi IoT (Internet of Things). Le onde radio per loro natura sono di difficile contenimento, pertanto la superficie di attacco a tali dispositivi è molto più ampia. Inoltre molte di queste tecnologie vengono fornite senza nessuna specifica sui protocolli wireless utilizzati e neppure sulle misure di sicurezza adottate per garantire la riservatezza e l'integrità delle comunicazioni.

Fino a poco tempo fa le analisi di sicurezza su protocolli di comunicazione wireless erano molto complesse e dispendiose. Infatti, per poter effettuare tali attività, erano necessari strumenti costosi e conoscenze di elettrotecnica per la progettazione di hardware specifico che permettesse di comunicare utilizzando lo stesso layer di trasporto del dispositivo oggetto di analisi. Negli ultimi anni, tuttavia, sono stati sviluppati hardware "generici" che hanno la caratteristica di fornire un segnale "grezzo" a un chip programmabile o a un personal computer che successivamente effettuerà a livello software tutte quelle operazioni che precedentemente venivano svolte tramite l'hardware (es. demodulazione, encoding). Questa tipologia di hardware è nota con il nome di Software Defined Radio (SDR). Oltre che per la progettazione e il collaudo di dispositivi radio, è anche possibile utilizzare le SDR per effettuare analisi di sicurezza su protocolli di comunicazione wireless.

In questo lavoro è presentata una metodologia che permette di effettuare analisi su protocolli wireless mediante l'utilizzo di Software Defined Radio. Partendo dall'identificazione della frequenza utilizzata vengono poi effettuate alcune ricerche per individuare le principali caratteristiche della comunicazione per riuscire ad arrivare al livello applicativo del protocollo utilizzato, cercando, dove possibile, di analizzare anche questo.

Il lavoro inizia con un'introduzione dei principali concetti base sulle onde radio necessari per poter capire appieno gli strumenti che verranno utilizzati successivamente (Capitolo 2). Sono esposti i concetti di frequenza, ampiezza e lunghezza d'onda, i concetti di modulazione del segnale che vengono utilizzati per trasformare un'onda analogica in segnale digitale e gli encoding che servono per evitare errori durante le trasmissioni. Inoltre, vengono esposte le modalità di assegnazione delle frequenze e vengono fornite alcune risorse pubbliche che possono essere di grande aiuto quando viene analizzato un segnale. Vengono elencati e spiegati i principali attacchi che possono avvenire sui protocolli wireless, illustrandone anche la causa (Capitolo 3). Si parte da quelli più semplici ed intuitivi come l'intercettazione, e si arriva ad attacchi più specifici, come ad esempio il RollJam, che mediante vulnerabilità tecnologiche ed umane può

essere utilizzato per riuscire ad aprire le automobili. Viene descritto il funzionamento delle Software Defined Radio e i tools che possono venire utilizzati per effettuare l'analisi (Capitolo 4). Gli analizzatori di spettro che permettono di identificare le frequenze utilizzate durante la trasmissione, l'analisi visuale dell'onda che permette di identificare la modulazione e l'applicazione degli encoding, permetteranno di arrivare ai bit realmente trasmessi. Verranno poi affrontate le tematiche di memorizzazione e ritrasmissione dell'onda che permetteranno di effettuare l'attacco di replay attack. Verrà utilizzato il tool Universal Radio Hacker per la traduzione dell'onda in bit e infine verrà esposta la libreria RFCAT per ricevere e trasmettere onde radio in modo molto semplice mediante il linguaggio di programmazione Python. Verranno poi esposti i casi di studio (case study) analizzati durante il progetto, in particolare verranno esposti i risultati relativi a: una presa 220v wireless comandabile da uno specifico telecomando, un sistema di sicurezza con telecomando wireless basato su protocollo X10, una chiave di apertura di una Fiat Stilo e un sistema di antifurto di nuova generazione completamente wireless che include i telecomandi di sblocco, i sensori magnetici, i sensori di movimento e con un sensore anti-jamming che dovrebbe proteggere l'intero sistema da alcune tipologie di attacchi (Capitolo 5). Per concludere saranno esposti i risultati e ipotizzate delle possibili contromisure per rendere immuni, ove possibile, i protocolli wireless dagli attacchi identificati (Capitolo 6).

2 Concetti base di segnali radio

Con i termini segnali radio o onde radio si indicano un sottoinsieme di radiazioni elettromagnetiche. Le radiazioni elettromagnetiche costituiscono una combinazione di oscillazioni del campo elettrico e del campo magnetico che si propagano alla velocità della luce, nel vuoto, con un moto ondulatorio. Questa tipologia di onda è prodotta quando particelle caricate vengono accelerate.

Nella figura qui di seguito è riportato un esempio di onda elettromagnetica; l'asse X indica il tempo, B indica il campo magnetico ed E quello elettrico.

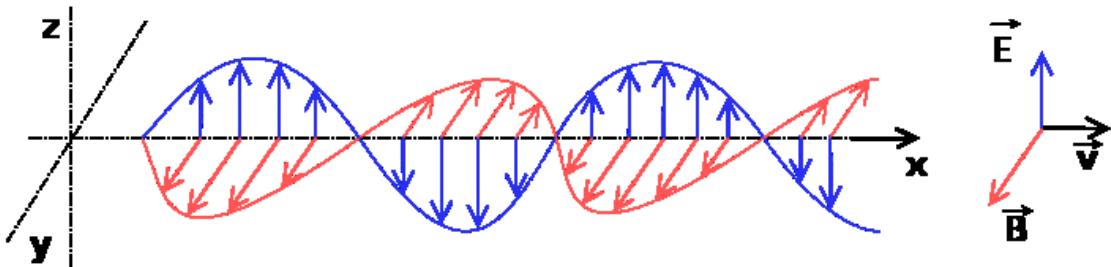


Figura 1. Esempio di onda elettromagnetica [1]

I primi passi in questo ambito furono effettuati nel 1800 da H.C. Oersted che scoprì il fenomeno dell'elettromagnetismo, e da M. Faraday, che successivamente scoprì l'induzione elettromagnetica [2]. J.C. Maxwell, partendo da questi risultati, formulò una teoria per cui l'elettricità, il magnetismo e la luce erano tutte manifestazioni del medesimo fenomeno, il campo elettromagnetico. Tale ipotesi fu teorizzata mediante quelle che attualmente vengono chiamate "equazioni di Maxwell" [3] che descrivono la propagazione nello spazio delle onde elettromagnetiche. Verso la fine del 1800 H. Hertz riuscì a creare un dispositivo in grado di produrre e rilevare le onde elettromagnetiche, confermando così la teoria di Maxwell.

Le onde elettromagnetiche possono essere classificate in base alla loro lunghezza nelle seguenti categorie: radio, microonde, infrarossi, visibili, ultravioletti, raggi X e raggi gamma.

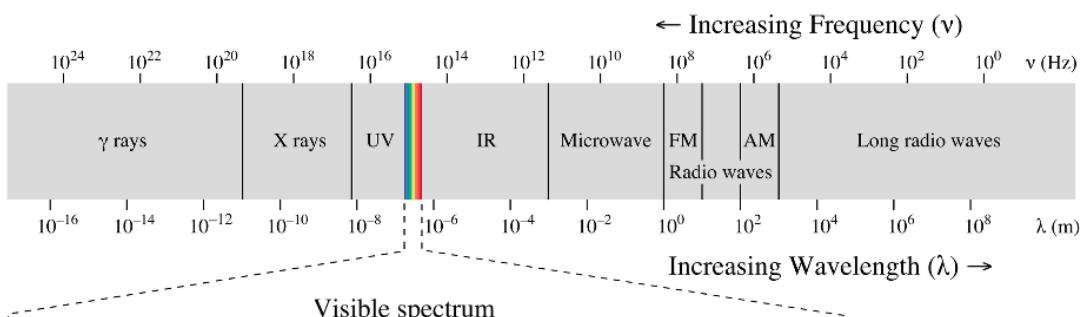


Figura 2. Tipologie e caratteristiche di onde elettromagnetiche [4]

Per poter lavorare con le onde elettromagnetiche è innanzitutto necessario definire alcuni concetti base: la frequenza, la lunghezza d'onda, l'ampiezza e la velocità.

- La **frequenza** è il numero di onde intere che attraversano un punto fisso in un dato arco di tempo. Generalmente viene misurata in Hertz (Hz): 1 Hz equivale a un'onda che effettua una singola transizione completa in 1 secondo.
- La **lunghezza d'onda** è la distanza misurata da un punto dell'onda al punto equivalente dell'onda successiva.
- L'**ampiezza** è la distanza tra il centro dell'onda e la sua parte più alta/bassa.
- La **velocità**, misurata normalmente in metri al secondo, è la relazione tra frequenza * lunghezza d'onda.

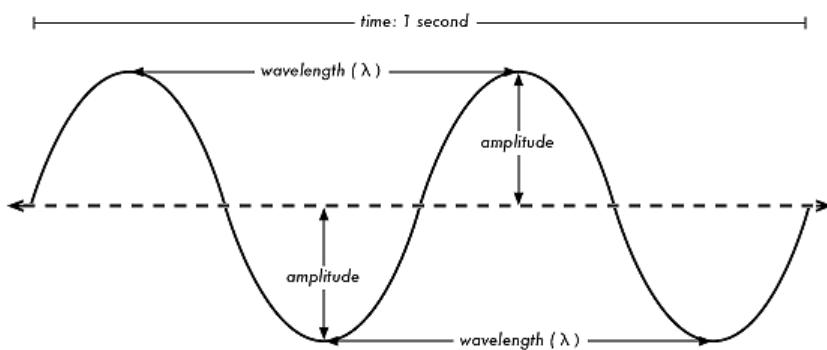


Figura 3. Informazioni su un'onda [5]

Ad esempio, la figura precedente illustra un'onda che effettua due transizioni complete in 1 secondo: pertanto essa ha una frequenza di 2 Hz.

Le misure normalmente utilizzate per le frequenze sono:

- Hz = hertz = 1 ciclo/secondo
- KHz = kilohertz = 1.000 cicli/secondo
- MHz = megahertz = 1.000.000 cicli/secondo
- GHz = gigahertz = 1.000.000.000 cicli/secondo

Ad esempio, nel caso in cui un'onda abbia una frequenza di 5 Hz (5 transizioni complete in 1 secondo) e abbia una lunghezza d'onda di 20 cm, la sua velocità sarà di 1 metro/secondo.

2.1 Frequenze ISM e frequenze libere

Per evitare la trasmissione sulle stesse frequenze da parte di più soggetti causando così delle interferenze, gli Stati hanno deciso di regolamentare l'utilizzo delle onde radio assegnando a determinati soggetti spazi di frequenze a loro riservati; solamente tali soggetti sono autorizzati alla trasmissione su quelle frequenze. In Italia viene rilasciato a livello nazionale un piano di ripartizione delle frequenze che definisce quali soggetti sono autorizzati a trasmettere e su quali frequenze [6]. Si definisce pertanto illegale la trasmissione di onde radio da parte di un soggetto su frequenze registrate ad altri soggetti.

Esistono frequenze definite ISM (Industrial Scientific Medical) per le quali non è richiesta l'autorizzazione per la trasmissione: queste frequenze possono essere utilizzate da chiunque, purché il fine sia quello definito a livello mondiale. Molti dei dispositivi di utilizzo quotidiano impiegano tali frequenze, come ad esempio badge, bluetooth, Wi-Fi e sistemi di allarme. Le frequenze ISM sono state definite con un accordo tra Stati a livello mondiale (anche se alcune variano a seconda dell'area).

Frequency range		Availability
6.765 MHz	6.795 MHz	Subject to local acceptance
13.553 MHz	13.567 MHz	Worldwide
26.957 MHz	27.283 MHz	Worldwide
40.66 MHz	40.7 MHz	Worldwide
433.05 MHz	434.79 MHz	only in Region 1, subject to local acceptance
902 MHz	928 MHz	Region 2 only (with some exceptions)
2.4 GHz	2.5 GHz	Worldwide
5.725 GHz	5.875 GHz	Worldwide
24 GHz	24.25 GHz	Worldwide
61 GHz	61.5 GHz	Subject to local acceptance
122 GHz	123 GHz	Subject to local acceptance
244 GHz	246 GHz	Subject to local acceptance

Tabella 1. Frequenze ISM [7]

In alcuni Stati sono presenti anche delle frequenze definite "libere" che possono essere utilizzate indiscriminatamente: ad esempio la frequenza di 315MHz non è ISM ma nell'area nordamericana è ritenuta libera [8].

Vengono riepilogate nella seguente figura le frequenze ISM e libere più utilizzate a livello mondiale.

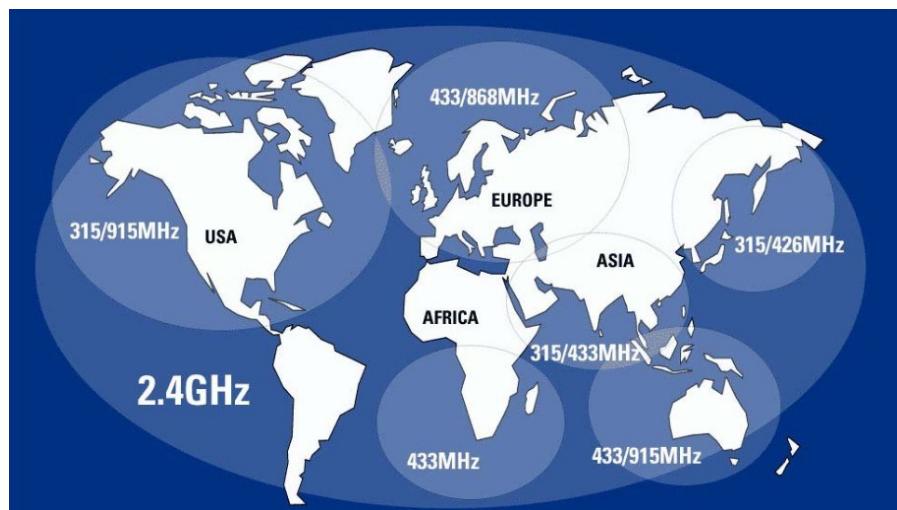


Figura 4. Frequenze utilizzabili a livello mondiale [9]

Esistono inoltre in alcuni Paesi delle frequenze il cui utilizzo è concesso ai soli radioamatori [10]. In Italia, per acquisire tale titolo, è necessario seguire un corso e sostenere un esame che permette di conseguire la "patente di operatore di stazione di radioamatore".

2.2 Modulazione e demodulazione del segnale

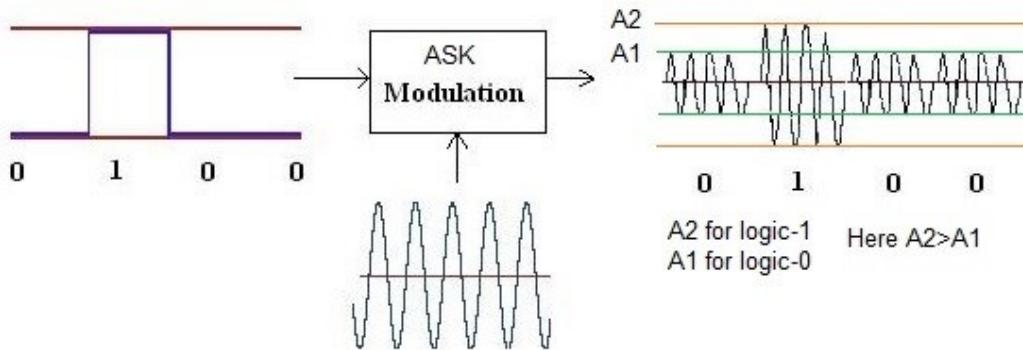
Le onde radio sono un mero mezzo di trasporto. Al fine di poterle utilizzare per trasportare dei dati digitali è necessario introdurre i concetti di modulazione e demodulazione che permettono di “tradurre” determinate caratteristiche dell’onda in un segnale digitale e viceversa.

Generalmente, il modulatore prende in input un’onda quadra e fornisce in output un’onda con determinate caratteristiche. Il demodulatore prende in input le caratteristiche dell’onda e fornisce in output una specifica serie di bit.

Esistono numerosi metodi di modulazione di un’onda, di seguito vengono esposti i più utilizzati [11].

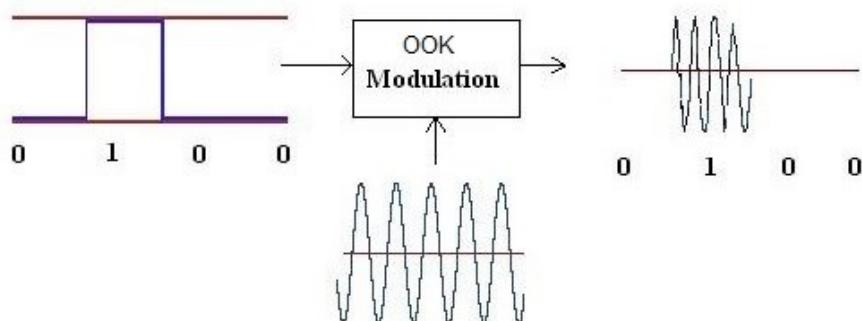
2.2.1 Amplitude Shift Keying (ASK)

Questa tipologia di modulazione è basata sull’ampiezza dell’onda. Viene definita una soglia minima per l’ampiezza: se l’onda supera questa soglia minima allora il dato trasportato sarà considerato un 1, altrimenti sarà uno 0.



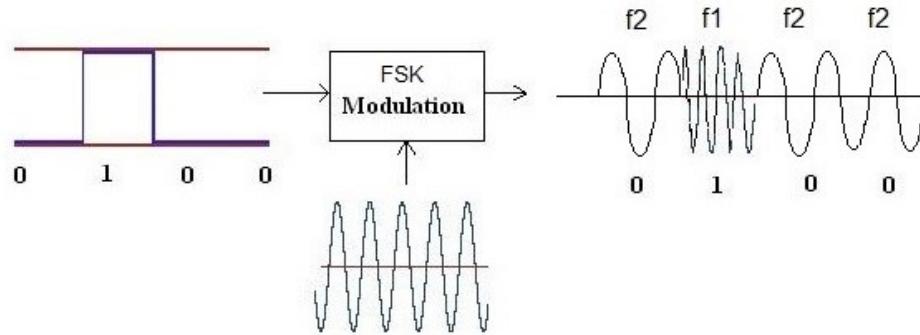
2.2.2 ON OFF Keying (OOK)

La modulazione OOK può essere ritenuta una variante dell’ASK. In questo caso i valori 0 e 1 sono determinati in base al fatto che l’onda sia o meno presente.



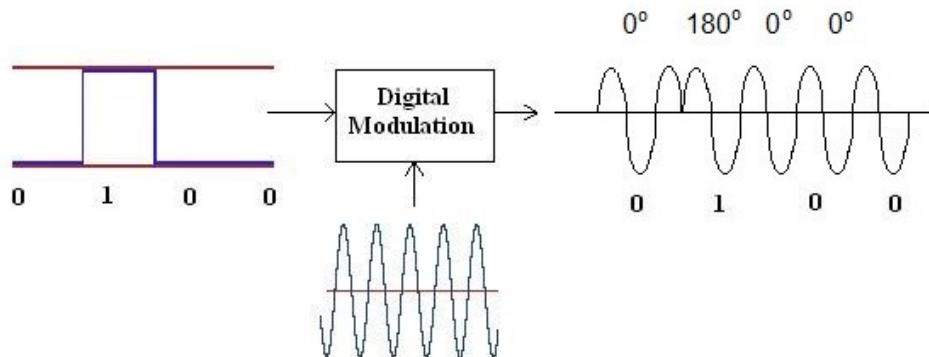
2.2.3 Frequency Shift Keying (FSK)

Nel caso di FSK l'ampiezza dell'onda non varia, ma è la frequenza dell'onda a variare e definire la conversione in bit. Anche in questo caso viene definita una frequenza minima e, se il valore della frequenza è inferiore alla soglia, allora il dato sarà 0, altrimenti 1.



2.2.4 Phase Shift Keying (PSK)

Nel caso di PSK la modulazione avviene cambiando la fase dell'onda che permette di identificare i bit settati a 1.

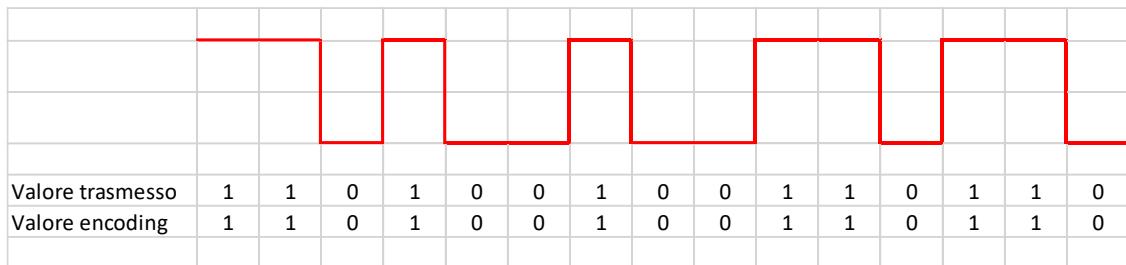


2.3 Encoding

Generalmente le onde radio dopo la modulazione verso un'onda quadra subiscono un'ulteriore trasformazione che porta ai dati reali. Questo avviene per compensare piccoli errori di trasmissione e di timing. Di seguito vengono esposti i meccanismi di **encoding** più comunemente usati.

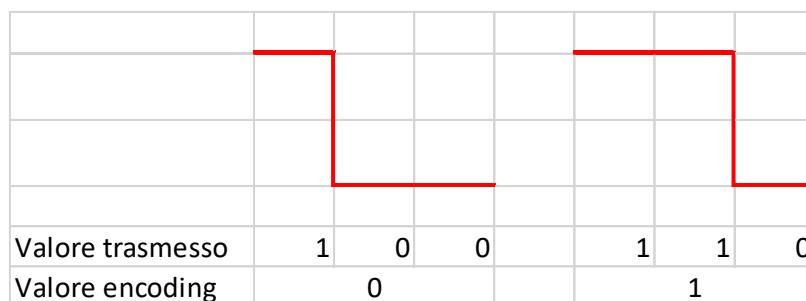
2.3.1 Non-Return-to-Zero (NRZ)

L'NRZ è il meccanismo di encoding più semplice: quando l'onda ha segnale maggiore di zero allora è da decodificarsi con un bit a 1, in alternativa a 0. Tale encoding viene utilizzato solo nelle trasmissioni radio più semplici perché non ha meccanismi di controllo sui dati trasmessi e inoltre non consente di distinguere un momento di non trasmissione rispetto a una trasmissione di soli 0.

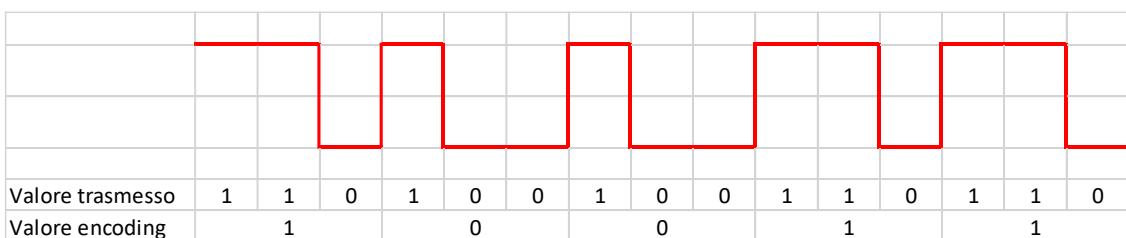


2.3.2 Pulse Width Modulation (PWM)

L'encoding PWM è tra i più utilizzati e si basa sulla lunghezza dell'impulso per stabilire se questo è uno 0 o un 1, simile a un segnale morse. Generalmente per processare i dati trasmessi tramite tale encoding si suddivide l'onda in parti uguali. Ogni segnale sarà composto da 3 sezioni come segue:



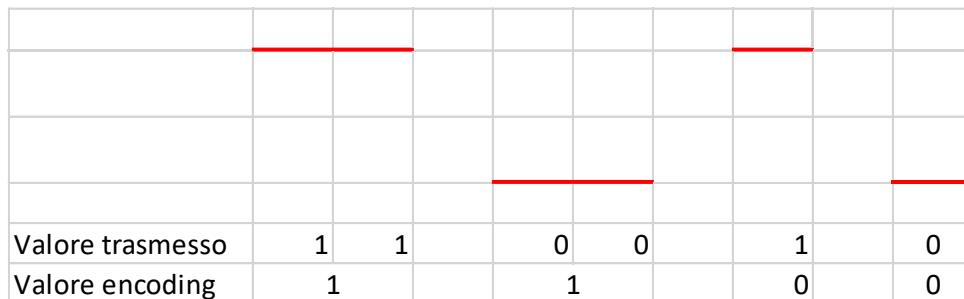
Di seguito viene riportato un esempio di trasmissione e la sua relativa traduzione in bit:



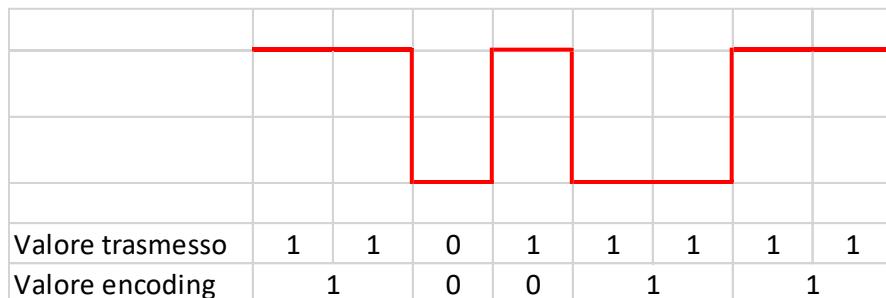
Generalmente tale encoding è semplice da identificare perché presenta un *pattern* ben preciso: il primo dato trasmesso è sempre 1 e può essere seguito o da 10 o da 00.

2.3.3 Pulse Interval and Width Modulation (PIWM)

Il PIWM è un'estensione del PWM. Esso si basa sullo stesso concetto del PWM, ma il meccanismo di lunghezza è esteso anche durante l'assenza di impulso.



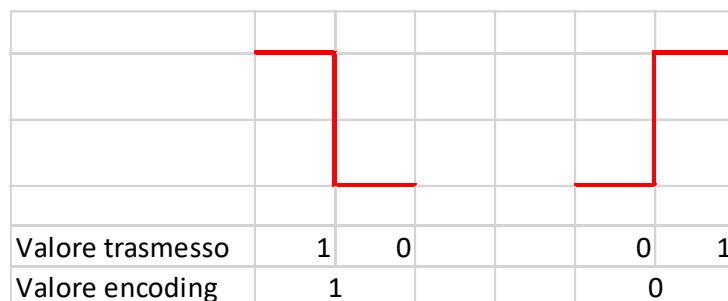
Di seguito viene riportato un esempio di trasmissione dati:



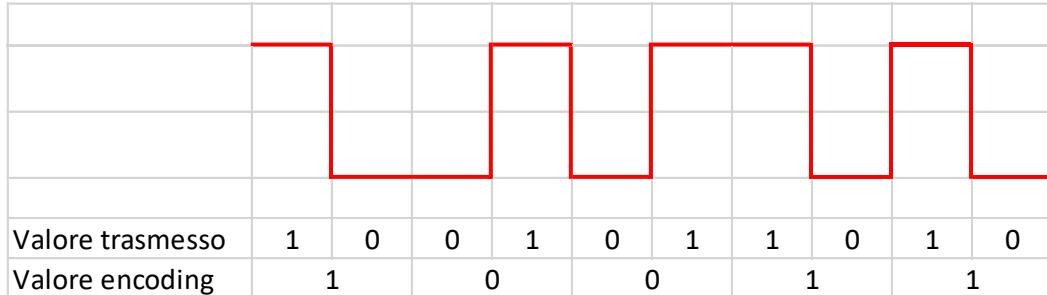
Generalmente è possibile identificare tale encoding se si nota che non vi è correlazione tra i segnali positivi e negativi; tuttavia è possibile anche che venga confuso con l'encoding NRZ.

2.3.4 Manchester (o bi-phase)

L'encoding Manchester si basa sul *phase shifting*, cioè sulla transizione del segnale da 0 a 1 o viceversa a intervalli regolari: tale transizione identifica il dato trasmesso. Generalmente se la transizione è da 1 a 0 il dato finale sarà 1, mentre se si passa da 0 a 1 allora sarà 0.



Di seguito viene riportato un esempio di dato trasmesso mediante tale encoding:



L'encoding Manchester è semplice da identificare perché in nessun caso produrrà più di due valori uguali consecutivi.

2.4 Come è strutturato un segnale

Generalmente i segnali radio vengono tutti strutturati in un modo simile perché tutti gli hardware necessari per la ricezione e la trasmissione devono risolvere problemi simili. Prima di trasmettere i dati reali possono venire trasmessi dei dati di sincronizzazione, in modo che la stazione ricevente possa identificare con semplicità la comunicazione. È possibile riassumere la struttura di un segnale come segue:

- **preamble**, pattern fisso che permette di identificare l'inizio della comunicazione.
- **sync word**, che permette di identificare una specifica trasmissione (esempio, uguale per tutti i dispositivi di un certo tipo).
- **dati**, che contengono effettivamente il dato di trasmissione.
- **segnale di fine**, che permette di identificare la fine della trasmissione.

A causa dei disturbi che una trasmissione wireless può subire è ampiamente diffuso l'utilizzo di trasmissioni multiple per comunicare un singolo dato. Esse vengono replicate un certo numero di volte nel tempo, in modo che anche se una di queste fosse corrotta dalle interferenze le altre potrebbero comunque riuscire a trasmettere il dato correttamente.

2.5 Identificazione dei segnali mediante risorse pubbliche

Tra i problemi frequenti si segnalano l'identificazione e l'interpretazione dei segnali radio, che spesso possono costituire un passaggio complesso. Esistono tuttavia alcune risorse potenzialmente utili.

Tra queste si cita la Signal Identification Guide (<https://www.sigidwiki.com>), che contiene la descrizione di numerosi segnali radio, le frequenze utilizzate, l'encoding, lo scopo, etc.

Signal Name	Description	Frequency	Mode	Modulation	Bandwidth	Location	Sample Audio	Waterfall image
2G CDMA (IS-95)	2G CDMA, also known as IS-95, is a cellular standard.	850 MHz	AM	QPSK	1.228 MHz			
3G WCDMA	CDMA2000, known primarily as 3G mobile, is a family of 3G data protocols used to send voice, text and signaling data to smart phones and other wireless devices.	824 MHz — 2,000 MHz	RAW, AM	QAM, QPSK, CDMA	1.25 MHz — 3.9 MHz			
4G LTE Network	Long Term Evolution Network. Also known as 4G LTE Data and Evolved Universal Terrestrial Radio Access (E-UTRA). Data service for wireless consumer devices.	700 MHz — 2,200 MHz	RAW	OFDM, PSK, QAM	5 MHz — 20 MHz	Worldwide		

Figura 5. Esempio di informazioni presenti all'interno del sito Signal Identification Guide

Negli Stati Uniti esiste inoltre una legge che richiede che tutti i dispositivi venduti che effettuano trasmissioni wireless siano validati dalla FCC (Federal Communications Commission), che certifica il rispetto delle normative americane. Nell'ambito del processo di validazione la FCC effettuerà alcune prove tecniche oltre a richiedere al produttore le specifiche dei dispositivi, compresi i documenti tecnici che dettagliano le onde radio utilizzate. Al termine di questo processo tali documenti vengono resi pubblici e consultabili da chiunque sul sito dell'ente americano.

Volendo analizzare un dispositivo (anche se acquistato in Italia) è possibile che questo riporti sulla confezione o sul case l'FCC ID, l'identificativo univoco assegnato allo specifico prodotto.



Figura 6. Esempio di FCC ID identificabile nel case dell'Apple iPhone

Grazie a tale identificativo è possibile utilizzare il database della FCC per accedere ai documenti relativi al dispositivo. Spesso all'interno di questi documenti sono citate le frequenze utilizzate, la modulazione e le ulteriori caratteristiche di interesse.

I siti di riferimento sono:

- <https://www.fcc.gov> (sito ufficiale della FCC)
- <https://fcc.io/> (sito che permette di effettuare ricerche semplici sul database FCC)

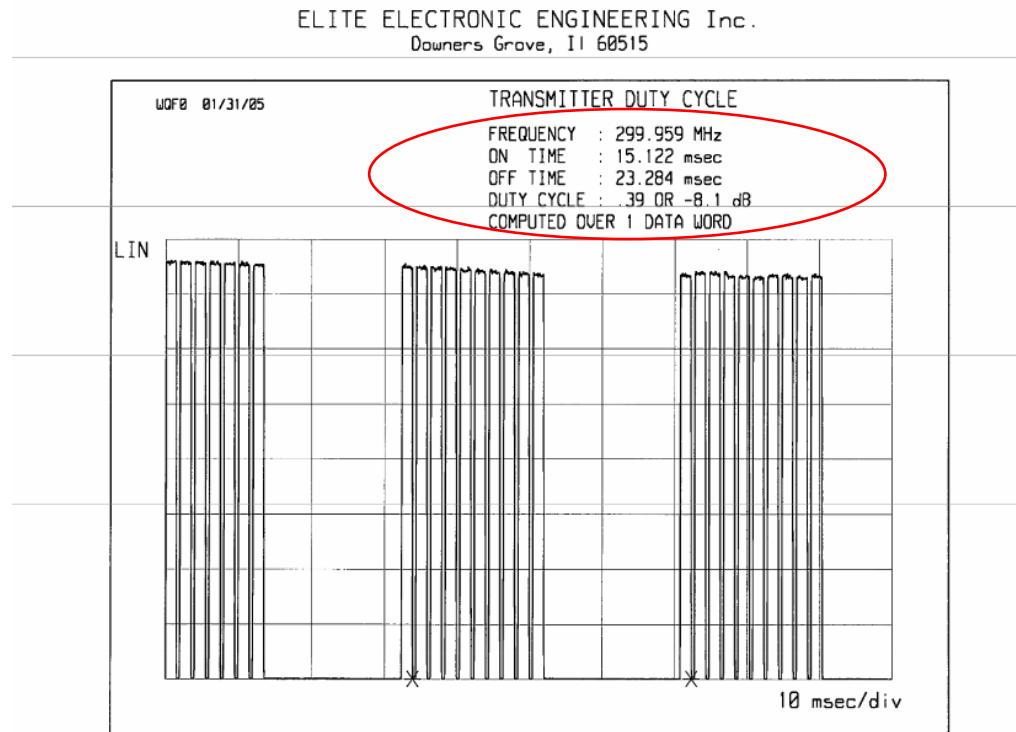


Figura 7. Alcune informazioni reperibili per il dispositivo con FCC ID HBW2392

3 Principali attacchi su protocolli wireless

In questo capitolo verranno illustrati i principali attacchi che si possono effettuare su protocolli di comunicazione wireless [12] [13].

3.1 Intercettazione

L'intercettazione delle comunicazioni wireless è uno degli attacchi più semplici da eseguire su tecnologie wireless poiché, per i meccanismi di funzionamento delle onde radio, risulta molto difficile controllarne la diffusione. Attraverso particolari antenne è possibile ricevere un segnale radio debole anche a molti chilometri di distanza. È pertanto importante capire e tenere conto di tale problematica mentre si progetta un apparato che deve comunicare tramite onde radio.

Negli anni sono nate alcune aziende che hanno promosso vari metodi per contenere le onde radio; tuttavia, risulta molto difficile applicare in completa sicurezza tali soluzioni.

3.2 Spoofing

A causa del fatto che nei protocolli che utilizzano onde radio non vi è una connessione fisica tra le due parti che partecipano alla comunicazione, è molto difficile identificare chi invia effettivamente un segnale. Ciò permette a chiunque di generare un segnale che sarà interpretato dalla stazione ricevente (utilizzando la corretta frequenza e modulazione). Pertanto non vi sono modi, a livello di trasporto, per limitare la possibilità di inserimento di una trasmissione non veritiera o malevola.

Se il protocollo radio risulta debole e non protetto a livello applicativo sarà quindi possibile creare delle comunicazioni che la ricevente interpreterà come valide.

Esistono alcune tecniche pensate per identificare se l'interlocutore con cui avviene una comunicazione cambia e/o viene sostituito, attualmente viene utilizzato ad esempio l'effetto doppler [14] per identificare se la sorgente di comunicazione cambia posizione durante la comunicazione. Tali tecniche di identificazione sono comunque molto complesse e costose e pertanto solo i dispositivi di alto profilo ne fanno uso.

3.3 Replay attack

Il *replay attack* è un attacco molto semplice che consiste nel replicare un segnale precedentemente ricevuto. L'attacco pertanto è effettuato in due fasi: nella prima fase è necessario riuscire a ricevere (intervettazione) il segnale radio a cui si è interessati, nella seconda invece si trasmette nuovamente il segnale precedentemente ricevuto.

Tali operazioni possono essere effettuate anche senza essere a conoscenza delle modalità di funzionamento del protocollo. Per poter effettuare tale attacco è quindi sufficiente conoscere la frequenza utilizzata per le trasmissioni.

Generalmente i protocolli suscettibili a tale tipologia di attacco sono i più semplici; per evitare tale problematica è sufficiente implementare un contatore di messaggi o in alternativa utilizzare un canale cifrato che verifichi la presenza di un *timestamp* di sincronizzazione.

3.4 Jamming

Il *jamming* è una tecnica molto semplice che permette di disturbare e rendere più difficile la comunicazione tra più soggetti che utilizzano onde radio. L'attacco è basato sull'invio di un segnale sulla frequenza di trasmissione utilizzata da altri soggetti, in modo che le comunicazioni vengano disturbate. Esistono principalmente due tipologie di jamming:

- Dumb jamming, in cui l'invio del segnale radio da parte di un agente di minaccia è costante sul canale di comunicazione e tutte le trasmissioni risultano completamente "corrotte". Se la potenza del segnale generato è abbastanza forte nessun utente potrà più utilizzare le frequenze disturbate.
- Smart jamming, nel quale l'invio di onde è effettuato solamente a intervalli definiti. In questo modo l'agente di minaccia può corrompere i singoli messaggi trasmessi, potendo così selezionare quali sono le specifiche trasmissioni che vuole disturbare.

Il dumb jamming risulta essere molto più semplice da realizzare, tuttavia è anche il più semplice da individuare. Infatti, è facile accorgersi quando sulle frequenze desiderate qualcuno trasmette in modo costante.

Lo smart jamming, invece, è molto più difficile da individuare. Infatti, effettuando piccole trasmissioni di disturbo, i messaggi vengono corrotti senza che il ricevente possa essere in grado di discernere se i disturbi nella trasmissione siano causati da interferenze naturali o, al contrario, da jamming.

Generalmente per mitigare il jamming vengono implementati alcuni meccanismi di cambio costante di frequenza (*frequency hopping*) che rendono più difficile ma non impossibile l'esecuzione di tale attacco.

3.5 Brute force

Nel caso in cui il protocollo wireless utilizzi un codice presente all'interno della trasmissione per autenticare il mittente, è possibile effettuare un attacco di tipo *brute force* alla ricerca del codice corretto inviando tutte le possibili combinazioni.

Generalmente è possibile classificare tale attacco in due differenti tipologie:

- online: si inviano tutti i codici possibili alla ricevente trasmettendo effettivamente il dato
- offline: si esegue l'analisi di alcune trasmissioni precedentemente registrate per identificare come avviene la generazione dei codici e poi trasmettere solo quando è stato identificato il modo corretto

Si fa notare come nel caso del brute force online sia necessario rispettare le tempistiche dalla comunicazione e inoltre non sia possibile parallelizzare tale attacco; infatti due o più sistemi che trasmettono codici genererebbero delle interferenze tra di loro non permettendo la ricezione del codice corretto. Questi due fattori rallentano notevolmente un attacco di tipo brute force online, rendendolo pressoché inutile anche nel caso in cui lo spazio delle chiavi sia relativamente piccolo.

3.6 OpenSesame Attack

L'attacco denominato OpenSesame è stato implementato e dimostrato da Samy Kamkar [15] nel 2015. Tale attacco velocizza il brute force di alcune tipologie di protocolli basati su codice statico, ma potenzialmente è possibile applicarlo anche su quelli a codice dinamico. Per poter eseguire tale attacco è tuttavia necessario che la trasmissione avvenga in un modo specifico.

L'attacco è sfruttabile solo ed esclusivamente su protocolli unidirezionali che non hanno un preambolo e/o una sync word ed effettuano una trasmissione costante dei codici di accesso. Nello specifico Kamkar ha rilevato che le riceventi utilizzavano degli *shift register* [16] per poter interpretare correttamente la sequenza di bit ricevuti, identificando il codice corretto al di là della sequenza di bit ricevuti. Naturalmente, per il loro funzionamento, in tali sistemi non può esistere un meccanismo anti brute force.

Si illustra di seguito un esempio del funzionamento di una ricevente che dovrebbe ricevere 12 bit nel caso in cui siano stati inviati 16 bit. In rosso sono riportati i bit che vengono interpretati per la comparazione del codice.

N° comparazione	Bit presi in considerazione
1	1001100111011011
2	1001100111011011
3	1001100111011011
4	1001100111011011
5	1001100111011011

Sostanzialmente inviando una sequenza di 16 bit vengono comparate 5 differenti chiavi a 12 bit. Kamkar, una volta analizzato tale metodo di identificazione del codice, ha ricercato la possibilità di creare specifiche sequenze di bit che potessero velocizzare un attacco di tipo brute force. Kamkar ha sfruttato un algoritmo per la generazione di sequenze di De Bruijn [17] che, unitamente al meccanismo di shift register utilizzato dalla ricevente, permette di velocizzare notevolmente l'attacco di tipo brute force.

Brute forcing a 3-bit code

```
1 bit      10 bits    20 bits    30 bits    40 bits    50 bits
bit 012345678901234567890123456789012345678901234567890
      000---001---010---101---011---111---110---100--- <-- 48 bits
```

OpenSesame Attack

```
First, remove the wait periods (reduces 50%):
1 bit      10 bits    20 bits    30 bits    40 bits    50 bits
bit 012345678901234567890123456789012345678901234567890
      000001010101011111110100 <-- 24 bits

By overlapping (De Bruijn), we reduce another ~62%:
000 011
001 111
010 110
101 100
0001011100 <-- 10 bits!
```

Figura 8. Esempio esplicativo delle sequenze di De Bruijn utilizzate nell'attacco OpenSesame

Sulla base dell'analisi effettuata da Kamkar, questa tipologia di attacco unitamente ad alcuni piccoli accorgimenti, permette di effettuare un attacco brute force su una chiave da 12 bit in circa 8 secondi, contro i 29 minuti necessari per esplorare l'intero spazio delle chiavi con un attacco standard.

3.7 RollJam attack

La tecnica chiamata *RollJam* è utilizzata per attaccare sistemi *rolling code* non basati sul tempo, ad esempio quelli che spesso vengono utilizzati nelle auto.

I sistemi a rolling code classici si basano su un seme condiviso e un contatore. Attraverso specifiche funzioni matematiche generano dei sottocodici a partire dal seme iniziale. Ogni codice è indipendente dal precedente e dal successivo, pertanto anche se un agente di minaccia riuscisse a intercettarne una serie non sarebbe in grado di generare i successivi. Per evitare replay attack, quando un codice viene ricevuto invalida automaticamente tutti i precedenti.

L'attacco è basato su un insieme di problematiche, alcune tecnologiche e altre umane. Vediamole nel dettaglio illustrandole con l'esempio di un telecomando per l'apertura di un'automobile.

In primis è necessario specificare che generalmente l'hardware che riceve il segnale non è impostato su una singola frequenza precisa, ma riceve una serie di frequenze adiacenti. Questo avviene perché i circuiti di trasmissione dei telecomandi possono presentare delle imprecisioni e pertanto se la ricevente utilizzasse una frequenza troppo specifica non tutti i codici arriverebbero correttamente.

Il primo step dell'attacco è effettuare un jamming del segnale, cioè trasmettere un segnale continuo sulle frequenze limitrofe a quelle utilizzate dal telecomando, in modo che la ricevente non riesca a decodificare il segnale del telecomando legittimo. L'attaccante, utilizzando hardware specifico al contrario della ricevente, può impostare dei filtri più fini e pertanto ricevere il segnale escludendo le frequenze su cui sta avvenendo il jamming.

A questo punto si attende che il telecomando trasmetta e si riceve il primo codice. L'utente, non avendo nessun feedback, molto probabilmente schiacerà nuovamente il pulsante inviando così il secondo codice. Ricevuto anche il secondo codice il jamming viene interrotto e immediatamente si invia il primo codice dei due ricevuti.

Così facendo l'utente riceverà il feedback corretto (ad esempio chiusura della macchina), tuttavia l'attaccante sarà ancora in possesso di un codice valido che potrebbe replicare, ad esempio per aprire nuovamente l'auto.

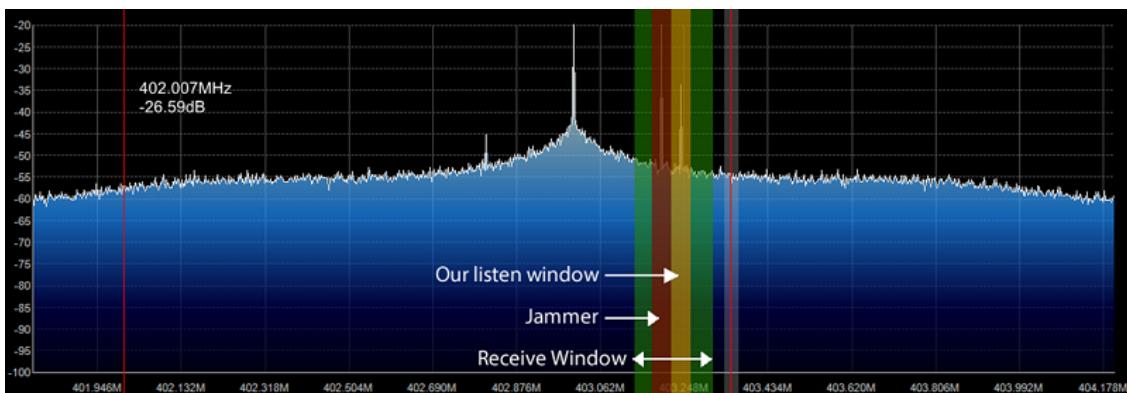


Figura 9. Le trasmissioni durante un attacco di tipo RollJam [18]

3.8 Man In The Middle

Nel caso di protocolli radio basati sulla prossimità (es. auto che si aprono automaticamente quando il proprietario si trova in un determinato raggio d'azione) è stata dimostrata la possibilità di eseguire attacchi Man In The Middle per riuscire a trasportare a distanza il segnale. Visto che generalmente i protocolli utilizzati sono ritenuti sicuri si sfrutta la caratteristica di questi dispositivi che sono sempre in "ricezione" e "trasmissione" automaticamente.

L'idea è quella di utilizzare due dispositivi differenti, uno dei quali deve stare vicino alla chiave dell'auto e l'altro vicino all'auto stessa. Mediante Internet questi due dispositivi comunicheranno tra di loro ed "inoltreranno" il segnale ricevuto uno all'altro.

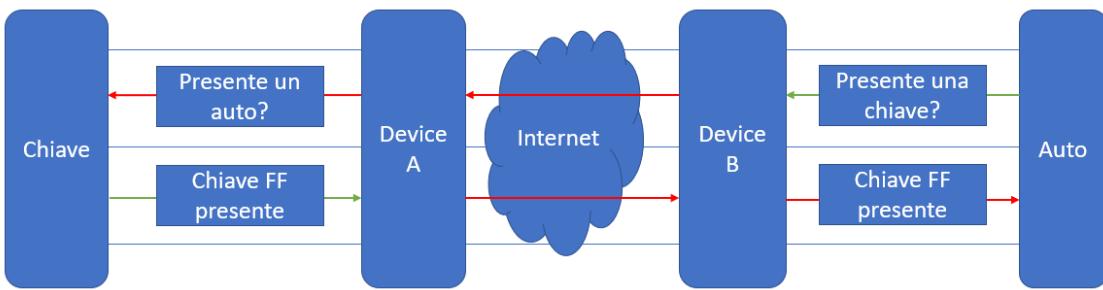


Figura 10. Esempio di attacco MITM su un telecomando dell'automobile

La principale problematica nella realizzazione di tale attacco è costituita dai tempi molto stretti che i protocolli devono mantenere, pertanto non è sempre possibile inoltrare il segnale attraverso Internet nei tempi necessari per poi fornire la risposta. Alcune modifiche a tale attacco utilizzano reti wireless ad alta velocità per poter rispettare le tempistiche necessarie.

4 Utilizzo delle Software Defined Radio per effettuare analisi di sicurezza

4.1 Software Defined Radio

Con il termine Software Defined Radio si intendono degli hardware pensati per la ricezione e/o trasmissione di onde radio con la caratteristica di essere generici e non specifici per un singolo protocollo. Agli inizi degli anni '90 Joseph Mitola ha ipotizzato e coniato il termine SDR, descrivendo un hardware composto principalmente da due componenti, un'antenna (che permette la ricezione del segnale) e un convertitore analogico digitale (che permette di tradurre l'onda in segnali digitali). Tutte le successive componenti e configurazioni potevano essere gestite attraverso un processore riprogrammabile. Le SDR hanno subito una enorme diffusione quando alcuni ricercatori hanno scoperto che alcune pennette USB molto economiche e utilizzate per ricevere i canali televisivi del digitale terrestre contenevano al loro interno un chip in grado di ricevere e processare un numero molto elevato di frequenze. Da quel momento in poi sono nati numerosi progetti che permettono di utilizzare le SDR per gli ambiti più disparati; è possibile ricevere dati del traffico aereo [19], immagini satellitari [20] e molto altro ancora.

Le SDR si limitano quindi a trasferire il segnale grezzo a un processore riprogrammabile; sovente vengono utilizzate per inviare il segnale ad un personal computer che poi si occuperà di effettuare le operazioni di demodulazione, encoding, etc. In questo modo non sarà più necessario costruire apposito hardware per la trasmissione o la ricezione del segnale, ma sarà possibile costruire tale logica a livello software [21].

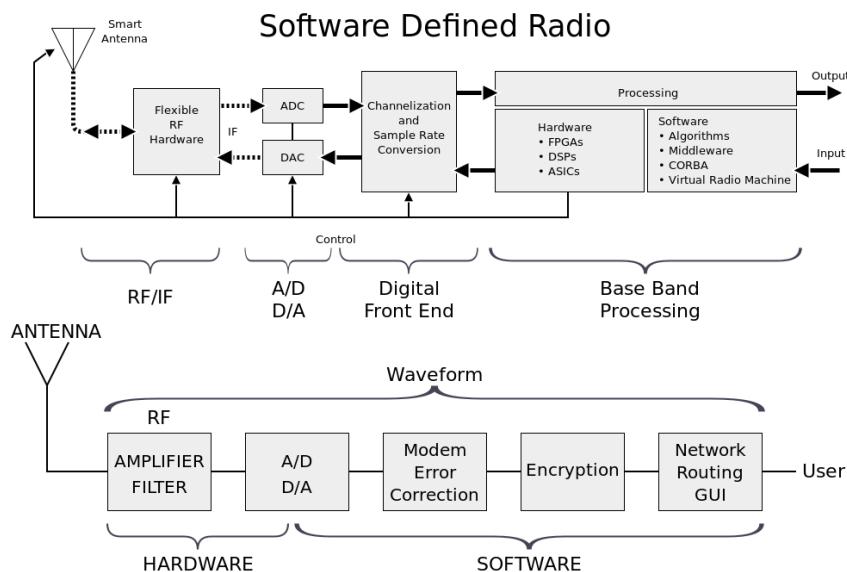


Figura 11. Normale schema di una SDR [22]

È possibile definire la seguente categorizzazione delle SDR:

- totale assenza di processore: il segnale viene inoltrato direttamente al computer in modalità “grezza”.
- presenza di processore programmabile: che permette di gestire direttamente dall’hardware alcune operazioni (es. modulazione, encoding, etc).

La prima tipologia viene generalmente utilizzata durante la fase di definizione e testing dei protocolli; infatti, in questo modo, non è più necessario creare l’apposito hardware per verificare le trasmissioni. La seconda tipologia è stata creata nel tempo soprattutto per poter abbassare i costi di produzione di alcuni hardware; infatti, a livello di ottimizzazione di scala, è più conveniente produrre un chip che effettua N differenti operazioni piuttosto che M chipset prodotti con una tiratura inferiore. Per tale ragione, oggi vengono utilizzati sovente dei chipset generici che possono essere riprogrammati. Un esempio è rappresentato dal chip CC-1111 della Texas Instruments che offre la possibilità di comunicare con le principali modulazioni ed encoding sulle frequenze al di sotto del Ghz e risulta essere utilizzato in numerosi prodotti.

Si segnala inoltre come siano presenti tali differenze perché non è detto che la comunicazione tra SDR e PC sia abbastanza veloce. Le velocità necessarie per poter trasferire le informazioni relative all’onda e la generazione della risposta potrebbero essere superiori a quelle definite nel protocollo, pertanto in alcuni casi è necessario prevedere dei processori specifici all’interno dell’hardware che si occupino di effettuare tali operazioni.

Negli ultimi anni sono nate numerose SDR con caratteristiche differenti; vengono elencate di seguito le principali:

- <https://myriadrf.org/projects/limesdr/>
- <https://www.nuand.com/>
- <https://greatscottgadgets.com/hackrf/>
- <http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html>
- <https://airspy.com/>

Nei successivi capitoli verrà presentata una metodologia che permette di identificare le principali caratteristiche dell’onda radio e, utilizzando i tool selezionati, sarà possibile riuscire a raggiungere il livello applicativo per poi identificare quali possono essere le sue vulnerabilità.

4.2 Descrizione del materiale utilizzato

Il materiale utilizzato durante lo studio è stato selezionato tra i differenti hardware presenti nel mercato sulla base delle loro caratteristiche tecniche e di un'analisi costi/benefici, allo scopo di scegliere quelli acquistabili anche con un budget ridotto e pertanto accessibili a chiunque.

4.2.1 Terratec T Stick PLUS

La Terratec T Stick nasce come device USB per ricevere la TV digitale terrestre (DVB-T) sul computer. Dopo alcuni anni dal suo rilascio sul mercato e la presenza della prima documentazione sulle SDR è emerso che il chip (RTL+E4000) utilizzato all'interno della scheda Terratec poteva essere utilizzato per ricevere un *range* di frequenze molto più ampio.

Il costo di questo dispositivo è molto ridotto (si parla di circa 25€) e le frequenze supportate sono molteplici: 52 MHz - 2200 MHz.

Tale dispositivo è utilizzabile in sola ricezione, infatti non è in grado di trasmettere segnali radio. Pertanto, esso può essere utilizzato per una prima analisi, ma non può essere utilizzato per attacchi attivi (es. replay attack, etc.).

4.2.2 HackRF One (<https://greatscottgadgets.com/hackrf/>)

HackRF One è una SDR open source progettata da Michael Ossmann, uno dei ricercatori di sicurezza più attivi nell'ambito trasmissioni wireless. Il progetto è nato per abbassare il costo (circa 300€) di una SDR che supporti un range di frequenze molto ampio e sia in grado di ricevere e di trasmettere segnali radio. Le sue principali caratteristiche sono le seguenti:

- 1 MHz - 6 GHz
- Half-duplex
- Grande velocità di campionamento
- Compatibilità con GNU Radio e con i principali software Open Source

4.2.3 Yard Stick One (<https://greatscottgadgets.com/yardstickone/>)

Yard Stick One nasce per operare sulle frequenze sotto il GHz, in particolare su tutte quelle frequenze ISM e free normalmente utilizzate dalla maggior parte dei dispositivi di largo consumo. Ha la caratteristica di avere un chip programmabile al suo interno che permette di eseguire alcune operazioni in autonomia senza dover passare il segnale grezzo al computer. Un'altra importante caratteristica è la compatibilità con rfcat (<https://bitbucket.org/atlas0fd00m/rfcat>), una libreria Python per ricevere e trasmettere dati tramite SDR. Le frequenze supportate sono: 300-348 MHz, 391-464 MHz, e 782-928 MHz.

4.3 Preparazione di un sistema per effettuare le analisi

Le SDR possono essere utilizzate su qualsiasi sistema operativo. Durante lo svolgimento di questa tesi è stato deciso di utilizzare la distribuzione Linux Ubuntu per la velocità di installazione del software necessario e la presenza di pacchetti già opportunamente compilati.

In particolare, è stato deciso di utilizzare la versione 16.04, che risulta essere l'ultima versione LTS (Long Term Support) distribuita, in modo da avere un supporto nel tempo garantito in caso di necessità.

È stata aggiornata l'intera distribuzione con l'ultimo software disponibile, tramite i seguenti comandi:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

4.4 Identificazione delle frequenze utilizzate mediante analizzatore di spettro

La prima problematica da risolvere per effettuare un'analisi del protocollo wireless è l'identificazione delle frequenze su cui il dispositivo di interesse sta operando. A tale scopo è possibile utilizzare le SDR come analizzatori di spettro. In questo modo è possibile visualizzare in tempo reale su quali frequenze vengono effettuate delle trasmissioni.

Esistono numerosi software con tale funzione; di seguito vengono riportati i più noti:

- gqrx <http://gqrx.dk/>
- osmocom_fft <https://osmocom.org/projects/sdr/wiki/GrOsmoSDR>
- QSpectrumAnalyzer <https://github.com/xmikos/qspectrumanalyzer>
- rfcat (utilizzando l'opzione -s) <https://bitbucket.org/atlas0fd00m/rfcat>
- Cubic SDR <http://cubicsdr.com/>

Tutti i software elencati funzionano in modo simile. Si è deciso di utilizzare “gqrx” perché risulta essere il più diffuso e pertanto è possibile reperire facilmente documentazione su Internet. In questa sezione è stato utilizzato l’hardware “HackRF One” che supporta il maggior numero di frequenze tra i tre a disposizione.

Sono stati inseriti all’interno del sistema dei repository gestiti direttamente dagli sviluppatori del software di interesse in modo da ottenere le ultime versioni disponibili.

```
$ sudo apt-get purge --auto-remove gqrx
$ sudo apt-get purge --auto-remove gqrx-sdr
$ sudo apt-get purge --auto-remove libgnuradio*
$ sudo add-apt-repository -y ppa:gqrx/gqrx-sdr
$ sudo add-apt-repository -y ppa:bladerf/bladerf
$ sudo add-apt-repository -y ppa:ettusresearch/uhd
```

```
$ sudo add-apt-repository -y ppa:myriadrf/gnuradiox
$ sudo add-apt-repository -y ppa:myriadrf/drivers
$ sudo apt-get update
$ sudo apt-get install gqrx-sdr
```

È stato inoltre installato il software “volk_profile” per ottimizzare l’hardware utilizzato sulle librerie GNU Radio sul quale è basato l’analizzatore di spettro.

```
$ sudo apt-get install libvolk1-bin volk_profile
```

Al primo avvio si presenterà la schermata di configurazione che permette di inserire le principali caratteristiche dell’hardware che si sta utilizzando. Nel nostro caso si seleziona come device “HackRF One”.

Una volta avviato, il software si presenta in una situazione simile a quella di seguito riportata; è possibile utilizzare il pulsante in alto a sinistra per iniziare l’analisi.

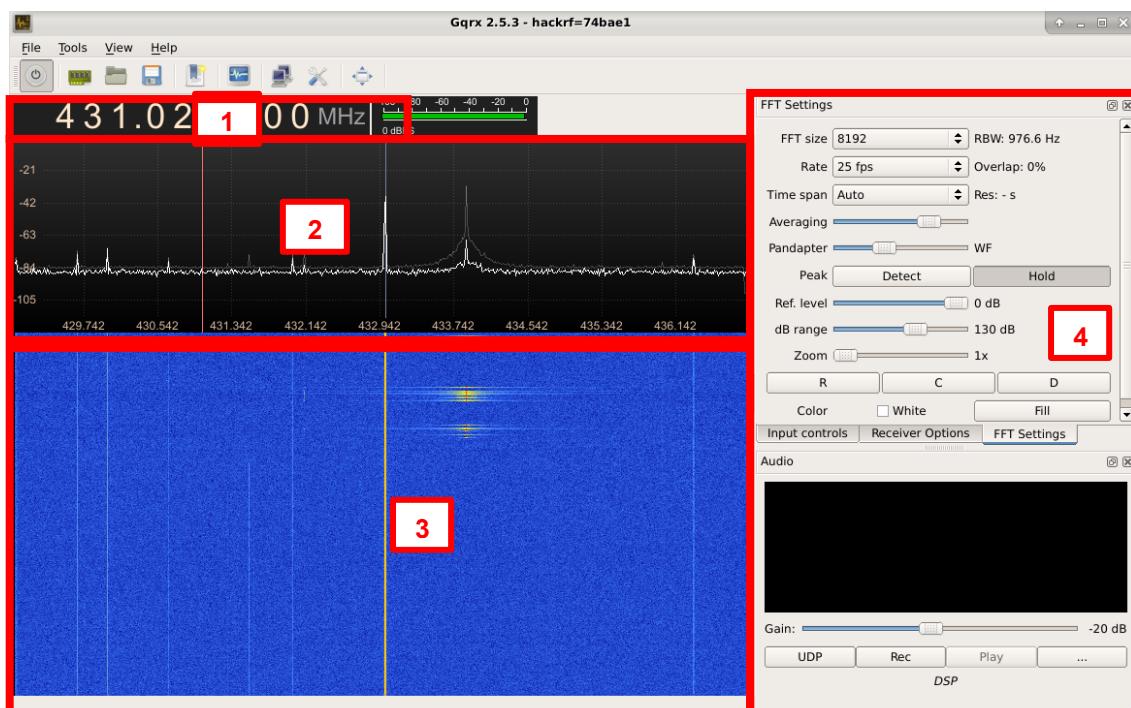


Figura 12. Gqrx mentre effettua l’analisi dello spettro.

Le aree principali dell’interfaccia sono le seguenti:

1. Area di selezione per la frequenza centrale e conseguentemente anche per lo spazio di frequenze visualizzate.
2. Visualizzazione in tempo reale dello spettro delle onde radio (costituita da una linea con tante piccole increspature).
3. Diagramma waterfall che rappresenterà le trasmissioni nel tempo: si avrà un colore blu quando la trasmissione non è presente e dal giallo al rosso quando è presente, a seconda della potenza.
4. Area di configurazione del tool.

I picchi nella seconda area identificano le trasmissioni che attualmente avvengono nel raggio di ricezione. Ogni trasmissione determina un picco rispetto alle increspature che identificano il “rumore di fondo” naturale, che non può essere mai 0 e quindi completamente piatto.

Questo permette di identificare se il sistema che si vuole analizzare è nel range di frequenze che si stanno monitorando o meno. Più si avvicina il dispositivo che trasmette all'antenna più il picco sarà alto; in questo modo si può anche determinare se la trasmissione avviene dal dispositivo che si vuole analizzare oppure se è associata ad un altro dispositivo che trasmette nella stessa area. Se il dispositivo non trasmette nelle frequenze che si osservano bisogna cambiare frequenza fino a quando non si riesce a identificarlo, mediante la sezione 1.

È inoltre possibile effettuare uno zoom andando sull'asse delle ascisse o delle ordinate e utilizzando la rotella del mouse. Tra le varie funzionalità utili del software è anche possibile utilizzare “Peak Hold” e “Peak Detect” che permettono, rispettivamente, di salvare il valore massimo di picco nel grafico (visualizzato in grigio) ed identificare le possibili trasmissioni mediante dei pallini automaticamente inseriti nel grafico.

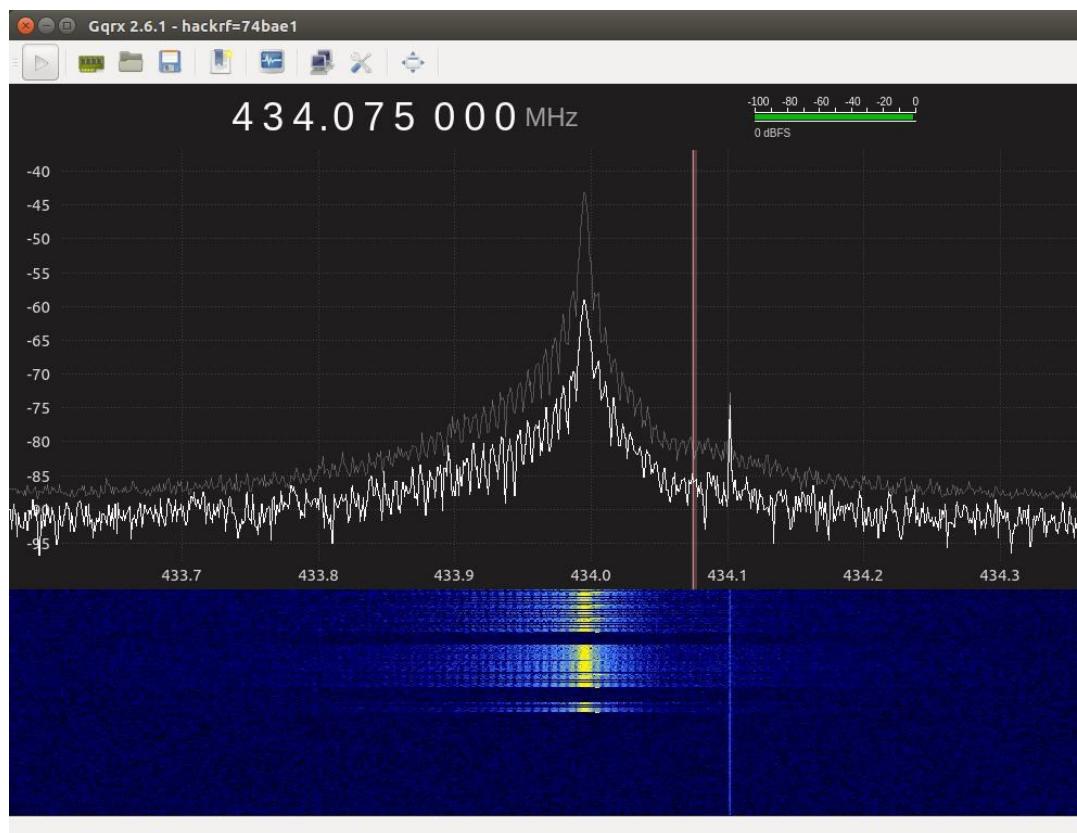


Figura 13. Esempio di picchi di trasmissione

Ad esempio, nell'immagine riportata, si può determinare in base ai picchi che esiste una trasmissione attorno ai 434.0Mhz e un'altra attorno ai 434.1Mhz (molto più debole della precedente). Spostandosi con il mouse sui picchi è possibile visualizzare con esattezza la frequenza di trasmissione. Inoltre, grazie al grafico waterfall, si può anche monitorare nel tempo come avviene la trasmissione, se questa è costante o meno.

All'interno della Figura 12 è possibile notare un picco centrale che coincide con la frequenza impostata. Questo picco non è realmente una trasmissione, ma un errore causato dalla maggior parte delle SDR; è quindi importante che nel caso si utilizzi l'analizzatore di spettro su un dispositivo la cui frequenza è conosciuta, sia impostata una frequenza vicina e non quella con esattezza, altrimenti la ricezione non sarà ottimale perché questo picco andrà in sovrapposizione con l'onda che si vuole analizzare.

Nella release 2017.02.1 del firmware per HackRF è stata rilasciata una nuova funzionalità “`hackrf_sweep`”, che secondo lo sviluppatore, permette di effettuare un'analisi dello spettro su 8Ghz contemporaneamente [23].

4.4.1 Memorizzazione del segnale ricevuto attraverso gqrx

Sono presenti numerosi strumenti che permettono la memorizzazione del segnale per una successiva analisi e/o replica; tuttavia, in questa fase, è stato utilizzato l'analizzatore di spettro “gqrx” per effettuare tale registrazione.

Selezionando dal menu “Record and play IQ data” è possibile accedere a una specifica sezione che permetterà di registrare i segnali. Tutte le impostazioni saranno le stesse già configurate.

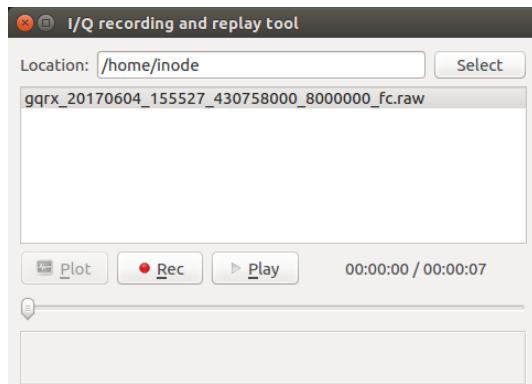


Figura 14. Schermata di registrazione e play di segnali

4.5 Utilizzo del software GNU Radio (<http://gnuradio.org/>)

GNU Radio è uno dei software più utilizzati in ambito SDR. Esso permette di generare rapidamente mediante un'interfaccia grafica un codice Python in grado di gestire l'onda radio a cui siamo interessati. Si può installare il software mediante i comandi:

```
$ sudo apt-get update
$ sudo apt-get install gnuradio
```

E richiamarlo mediante:

```
$ gnuradio-companion
```

L'interfaccia del software è la seguente:

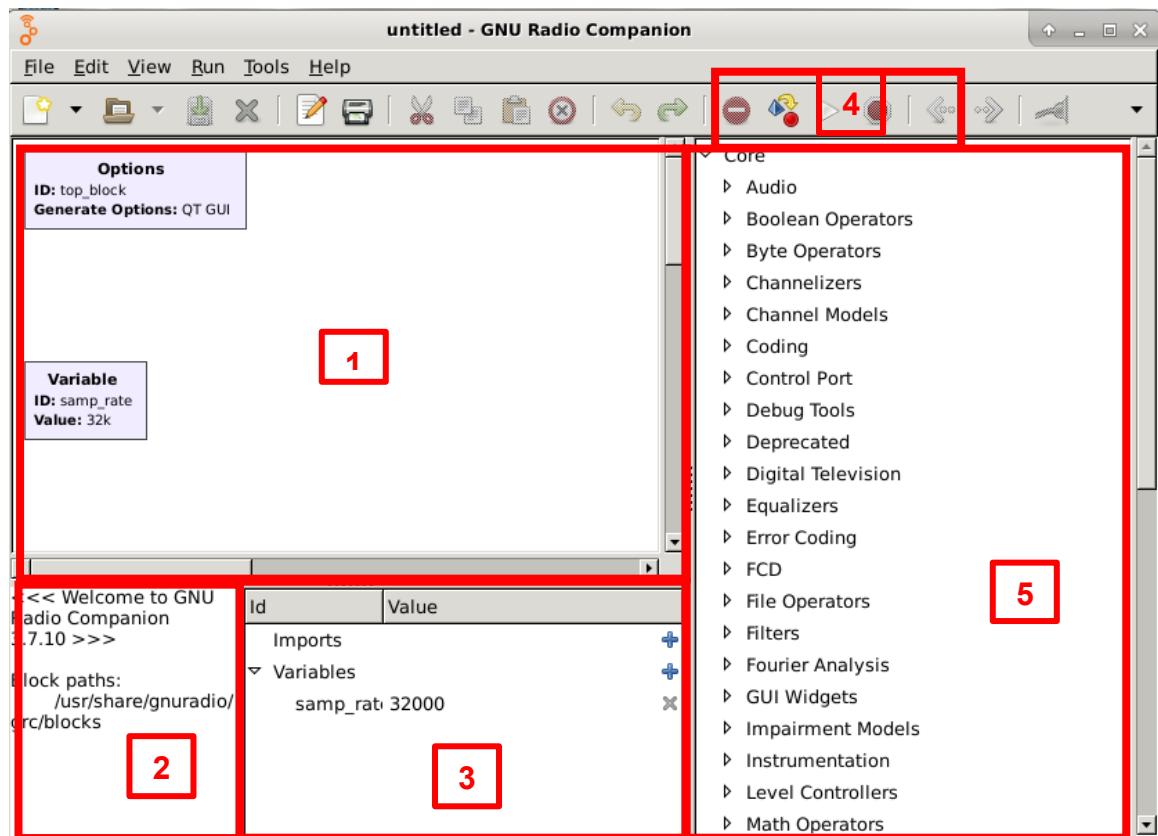


Figura 15. GnuRadio-companion

Dove:

1. Area di definizione dello script Python che verrà prodotto
2. Area di standard output del programma
3. Informazioni relative all'oggetto selezionato
4. Toolbar di esecuzione
5. Oggetti che possono essere aggiunti

È possibile trascinare gli oggetti che si trovano nell'area 5 all'interno dell'area 1, e collegandoli tra loro si può definire un workflow che verrà poi eseguito

successivamente all'avvio dello script. Per un tutorial sull'utilizzo base di questo strumento si rimanda a:

http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_GRC

Come primo esempio è stato utilizzato l'HackRF One per ricevere un segnale su una frequenza specifica e visualizzare come output due componenti differenti, un analizzatore di spettro e l'onda generata per le specifiche frequenze ricevute.

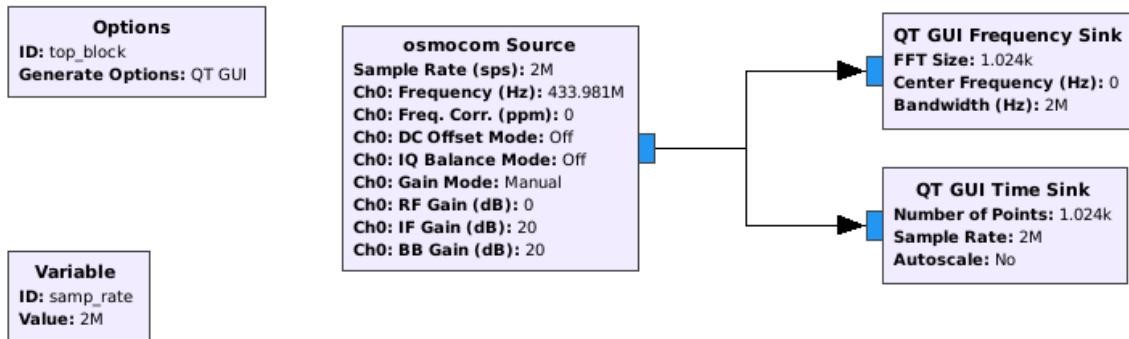


Figura 16. 001_base.grc

Questo semplice schema permette di ricevere l'input della HackRF One e portarlo in visualizzazione di due differenti oggetti; il primo visualizzerà l'onda in base alla potenza ricevuta nel tempo e il secondo quello dell'ampiezza. Sarà così possibile vedere sia i dati che precedentemente visualizzati con l'analizzatore di spettro, sia l'onda ricevuta. In questo esempio specifico è stata impostata la frequenza di 433.981Mhz e una velocità di campionamento di 2 milioni di rilevazioni al secondo.

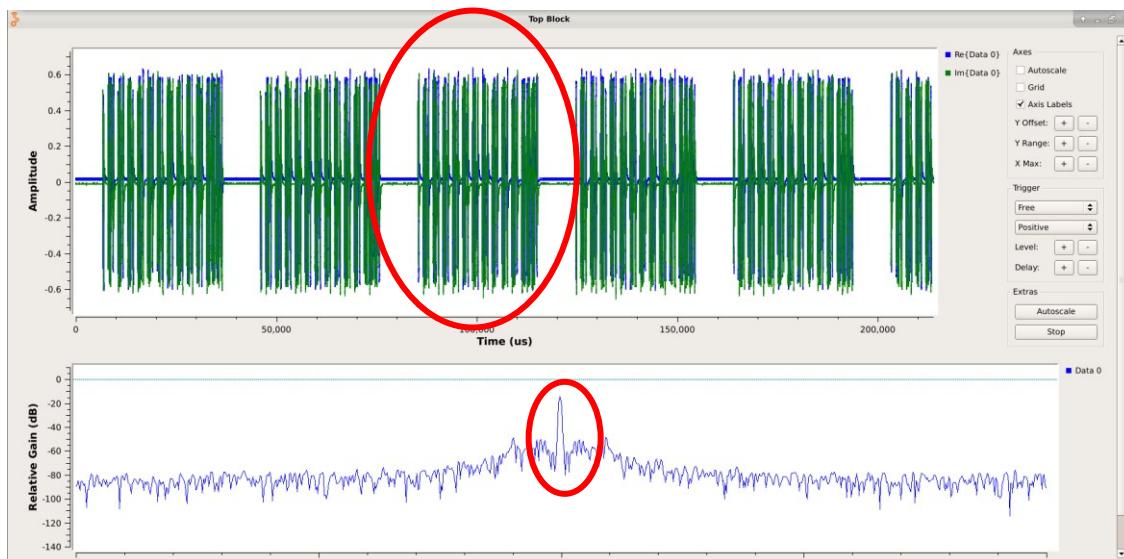


Figura 17. Output di 001_base.grc su altro device

4.5.1 Salvataggio del segnale per successiva analisi

Il passo successivo è salvare il segnale radio su file, in modo da poter eseguire successive analisi puntuali.

Attraverso questo script viene definito nuovamente come input l'HackRF One, tenendo il grafico di ricezione del segnale e aggiungendo anche un oggetto che permette di salvare su disco il segnale radio, in particolare nel file "/tmp/01.cap".

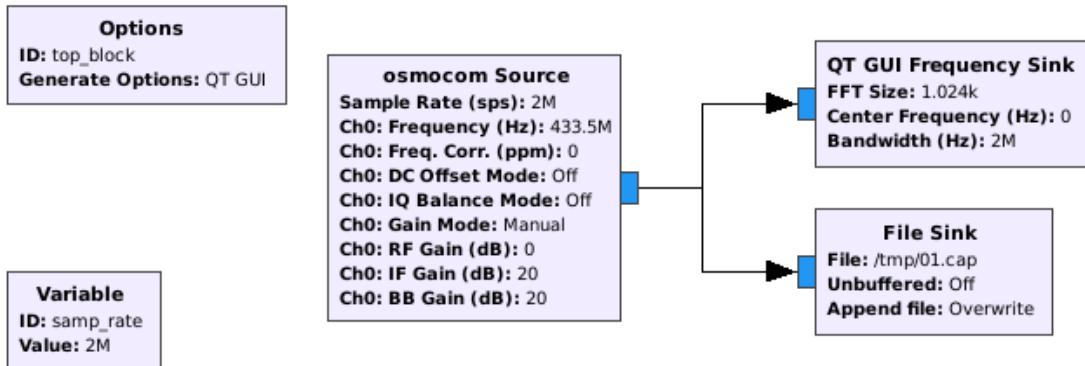


Figura 18. 002_save_to_file.grc

Si sottolinea come il salvataggio di tali dati con un sample rate alto (attualmente impostato a 2M/s) potrebbe generare un file di output molto grande.

Per tale ragione è buona norma utilizzare dei filtri, che permettono di selezionare il range di frequenze a cui siamo realmente interessati e non salvare il resto dello spettro. In questo modo il file che verrà generato sarà molto più piccolo rispetto al precedente.

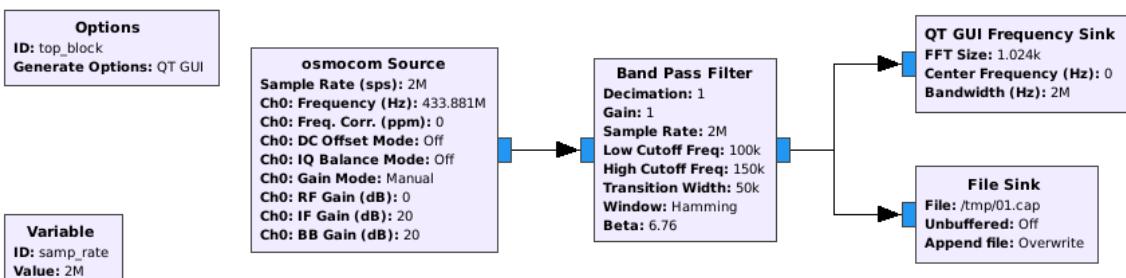


Figura 19. 003_save_to_file_filter.grc

Inoltre, tale configurazione faciliterà anche durante la trasmissione; non verranno infatti trasmessi dati su altre frequenze.

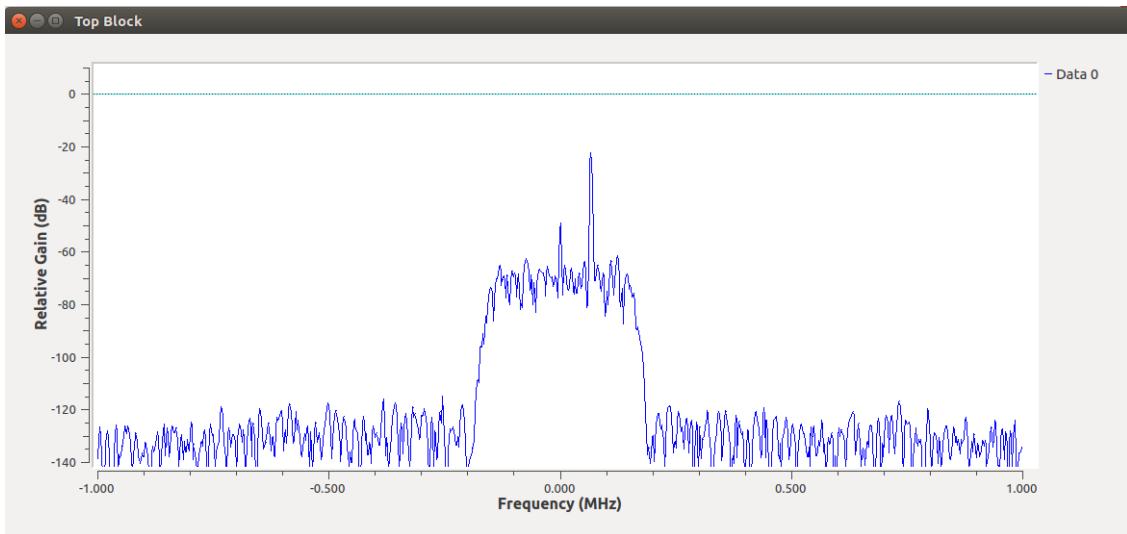


Figura 20. Esempio di registrazione con filtro

4.5.2 Replica di una trasmissione precedentemente ricevuta

Uno degli attacchi più semplici da effettuare sulle onde radio è il “replay attack”. Una volta che il segnale è stato salvato su disco è possibile ritrasmetterlo quante volte si vuole. Se la trasmissione ricevuta si basa su codici statici e non variabili nel tempo la trasmissione che si effettuerà sarà identica a quella originale, se non per la potenza del segnale che sarà certamente differente.

Viene riportato di seguito lo schema che permette di inoltrare il segnale precedentemente ricevuto.

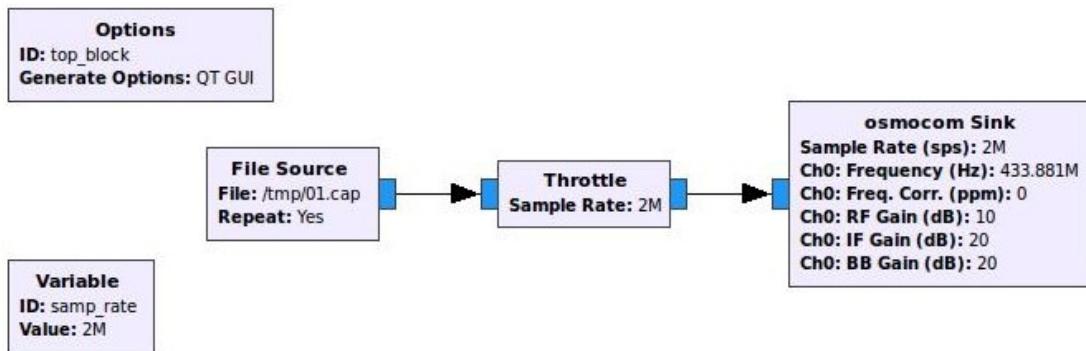


Figura 21. 004_replay.grc

Come è possibile vedere lo schema parte dal file di input che precedentemente era stato salvato, passa il suo output a una funzionalità di “Throttle” che permette di limitare la lettura del file alla velocità a cui è stato acquisito e successivamente viene inviato alla scheda HackRF One. La frequenza che bisogna impostare è la stessa che è stata utilizzata durante la registrazione, infatti il file non contiene informazioni sulla frequenza assoluta utilizzata dalle onde, ma solo quella relativa rispetto al centro delle frequenze.

Lo schema illustrato ritrasmette le onde ricevute nello stesso identico modo. Questo vuol dire che se l'onda ricevuta è tenue, perché ad esempio il trasmettitore era distante, è possibile che una volta ritrasmessa non sia in grado di arrivare correttamente alla stazione ricevente. Per questa ragione in alcuni

casi è necessario inserire un ulteriore elemento che si occupa di “amplificare” i dati dell’onda, un moltiplicatore. Nello schema successivo viene inserito un moltiplicatore che creerà un’onda 6 volte più ampia.

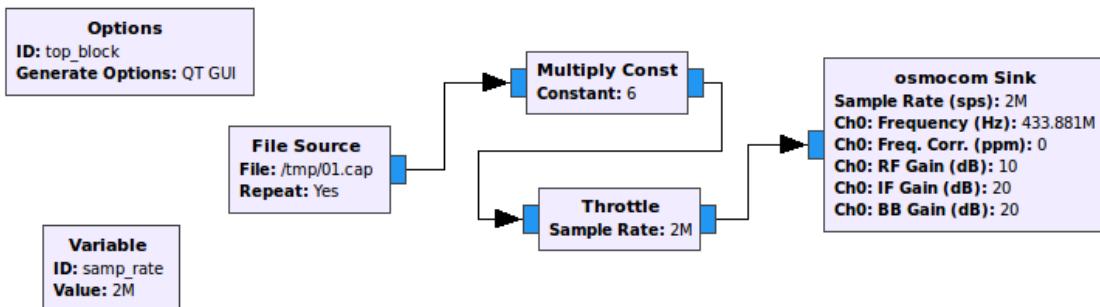


Figura 22. 005_replay_multiply.grc

Il software GNU Radio è estremamente potente e fornisce una serie di componenti già preimpostati che possono essere utilizzati. Inoltre, come è stato visto, è possibile generare a partire da uno schema il relativo codice Python, che successivamente potrà essere utilizzato in qualsiasi programma.

4.6 Analisi del segnale mediante Inspectum

Inspectum è uno strumento che permette di visualizzare il segnale precedentemente registrato in modo da effettuare le prime analisi per cercare di identificare la modulazione utilizzata. Per prima cosa è necessario installare le librerie liquid-dsp (<http://liquidsdr.org/>).

```

$ sudo apt-get install automake
$ git clone https://github.com/jgaeddert/liquid-dsp
$ cd liquid-dsp
$ ./bootstrap.sh
$ ./configure
$ make
$ sudo make install

```

E successivamente il software stesso.

```

$ git clone https://github.com/miek/inspectum
$ cd inspectum
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install

```

Una volta che la compilazione è giunta al termine è possibile aprire uno dei file precedentemente registrati con qqrx o con GnuRadio. L’interfaccia dello strumento apparirà come segue.

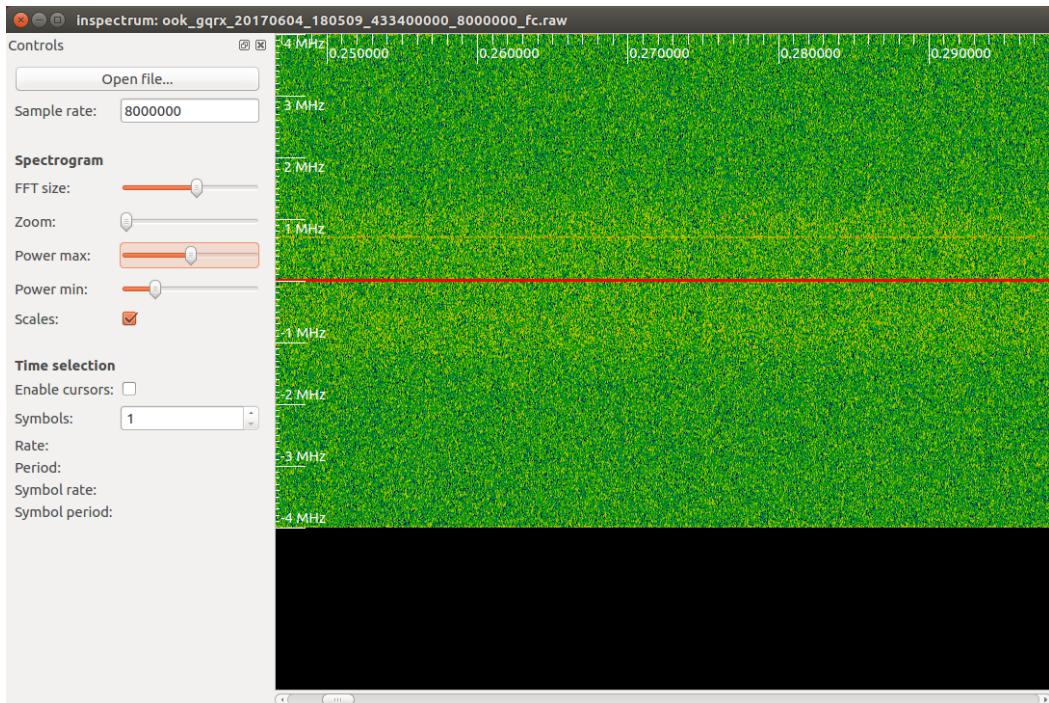


Figura 23. Inspectrum appena caricata una registrazione

Nella parte destra dello schermo è presente lo stesso diagramma waterfall visualizzato dall'analizzatore di spettro, ma con la caratteristica di essere navigabile: è possibile quindi muoversi nell'asse del tempo.

Nella parte sinistra sono presenti invece le impostazioni dello strumento: in particolare è possibile utilizzare le opzioni “Power max” e “Power min” per eliminare il rumore di fondo registrato. Nel momento in cui si raggiunge un buon bilanciamento di entrambi i valori si dovrebbe avere una situazione come quella riportata in Figura 24.

4.6.1 Identificazione delle modulazioni mediante visualizzazione del segnale

Attraverso questo strumento è possibile cercare di capire visivamente la tipologia di modulazione utilizzata a seconda delle caratteristiche del segnale. Vengono riportati di seguito alcuni esempi. Si ricorda che il segnale presente sulla frequenza centrale risulta essere un errore causato dall'hardware delle SDR.

Nel caso di modulazione OOK si possono trovare delle linee che sono presenti o non presenti a seconda dell'attuale stato di trasmissione.

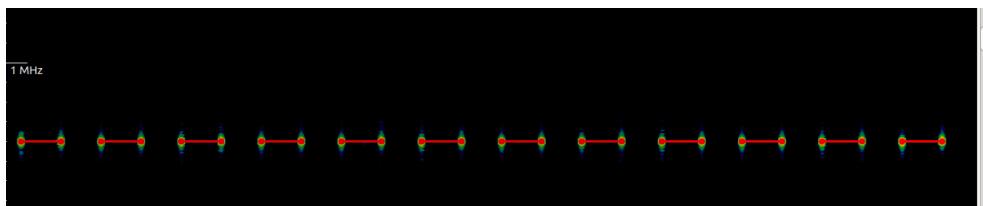


Figura 24. Esempio di modulazione ASK/OOK

Nel caso di modulazione ASK invece è possibile trovare una linea continua di cui cambia l'intensità, per cui alcune sue parti potrebbero essere rosse e altre arancioni.

Nel caso invece di modulazione FSK è possibile trovare che la trasmissione avviene su frequenze differenti e pertanto su linee diverse.

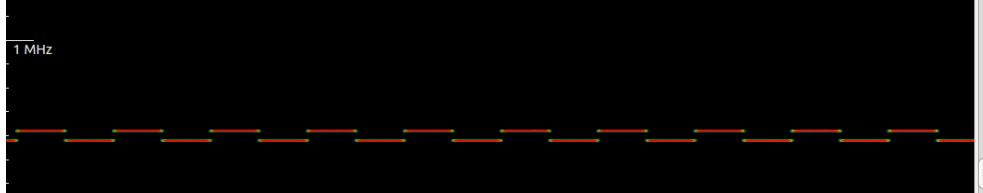


Figura 25. Esempio di modulazione FSK

Infine, nel caso di modulazione GFSK, è possibile trovare una linea ondulata ma continua nel tempo.

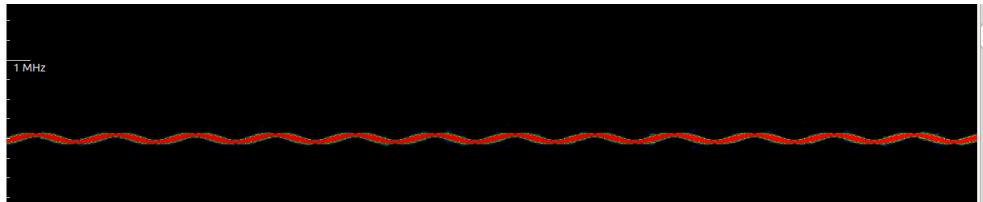


Figura 26. Esempio di modulazione GFSK

4.6.2 Utilizzo di Inspectrum per l'identificazione della velocità di trasmissione

Selezionando l'opzione “Enable cursor” viene automaticamente inserita all'interno del waterfall un'area che è possibile allargare e rimpicciolire a piacimento. Tale “finestra” permette di misurare il tempo rapportato all'onda che si analizza.

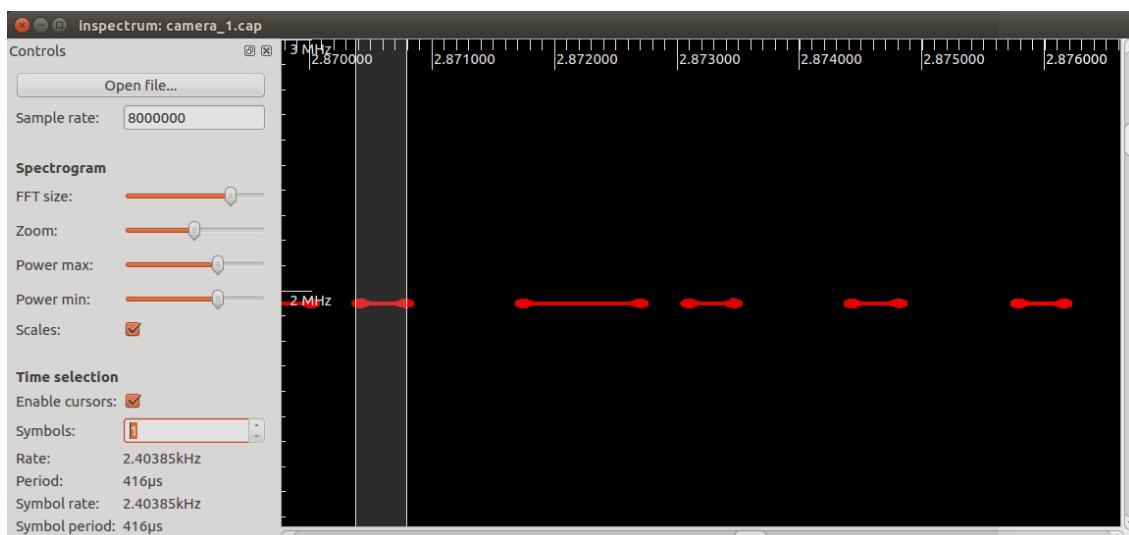


Figura 27. Inserimento dei cursori in Inspectrum

Bisogna identificare lo spazio più “corto” presente nel segnale, successivamente aumentando il numero di simboli bisogna allargare o restringere la finestra in modo che i vari simboli rispecchino il segnale che è in analisi.

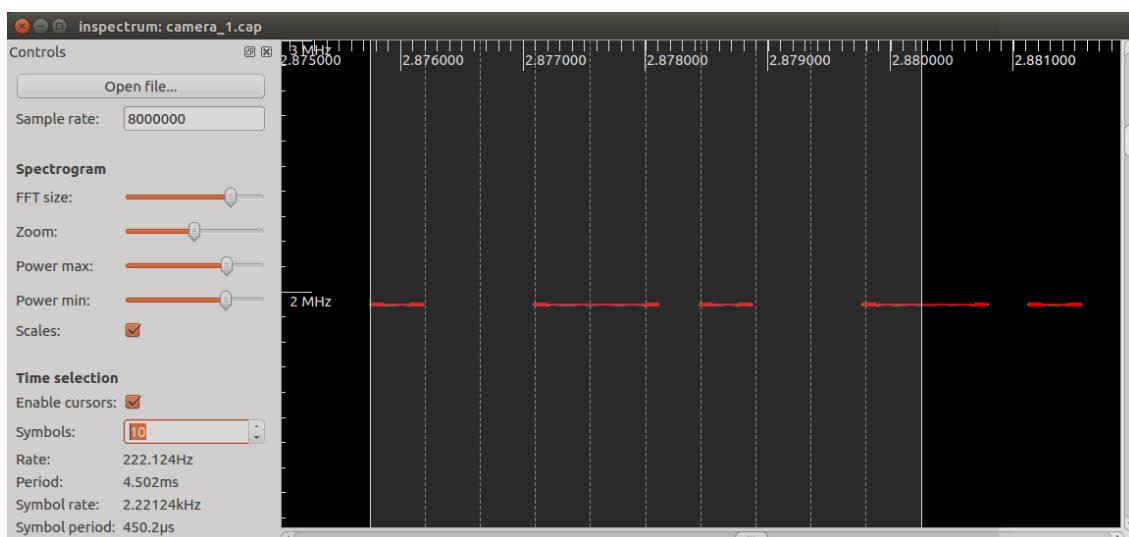


Figura 28. Utilizzo dei cursori per identificare la velocità di trasmissione

Una volta eseguita tale operazione il valore di “Symbol period” sarà il valore di partenza; infatti, generalmente, nei protocolli wireless tutti i segnali trasmessi sono un multiplo della trasmissione più corta. Il dato può essere rappresentato in “ns” (nanosecondi), “μs” (microsecondi) o “ms” (millisecondi). “Symbol rate” indicherà invece quanti bits vengono trasmessi al secondo, in questo caso circa 2221.

Il metodo descritto per l’identificazione della velocità di trasmissione può essere utilizzabile solo ed esclusivamente nel caso in cui la modulazione permetta di visualizzare una differenza nel grafico waterfall, è infatti difficile utilizzare tale metodo per identificare segnali PSK o simili.

4.6.3 Identificazione del preambolo e delle sync word con Inspectrum

Utilizzando le funzionalità descritte e in particolare la funzionalità di “zoom” è possibile cercare di identificare visualmente se esiste un preambolo nella comunicazione analizzata. Si ricorda che un preambolo generalmente è una sequenza di 0 e 1 consecutivi che permette al ricevente di sincronizzarsi sui tempi che verranno utilizzati e identificare l’inizio di una comunicazione. Allo stesso modo sarà anche possibile cercare di identificare se nella comunicazione esiste anche una sync word, che sarà la stessa per tutte le comunicazioni effettuate dallo stesso dispositivo.

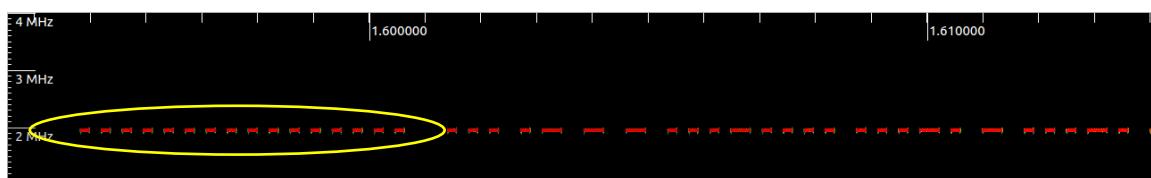


Figura 29. Esempio di preambolo

4.7 Analisi del segnale mediante Universal Radio Hacker

Un altro strumento molto utile nella prima fase delle analisi dei segnali è Universal Radio Hacker (<https://github.com/jopohl/urh>). Anche in questo caso per prima cosa è necessario installare il tool sul sistema.

```
sudo apt-get install python3-numpy python3-psutil python3-zmq
python3-pyqt5 g++ libpython3-dev python3-pip
sudo pip3 install urh
```

Universal Radio Hacker si divide in 3 differenti aree:

- **Interpretation** viene utilizzata per analizzare il segnale grezzo
- **Analysis** viene utilizzata per interpretare il segnale
- **Generator** viene utilizzata per inviare un nuovo segnale attraverso una SDR

Una volta aperto un segnale precedentemente registrato URH si presenterà nel seguente modo:

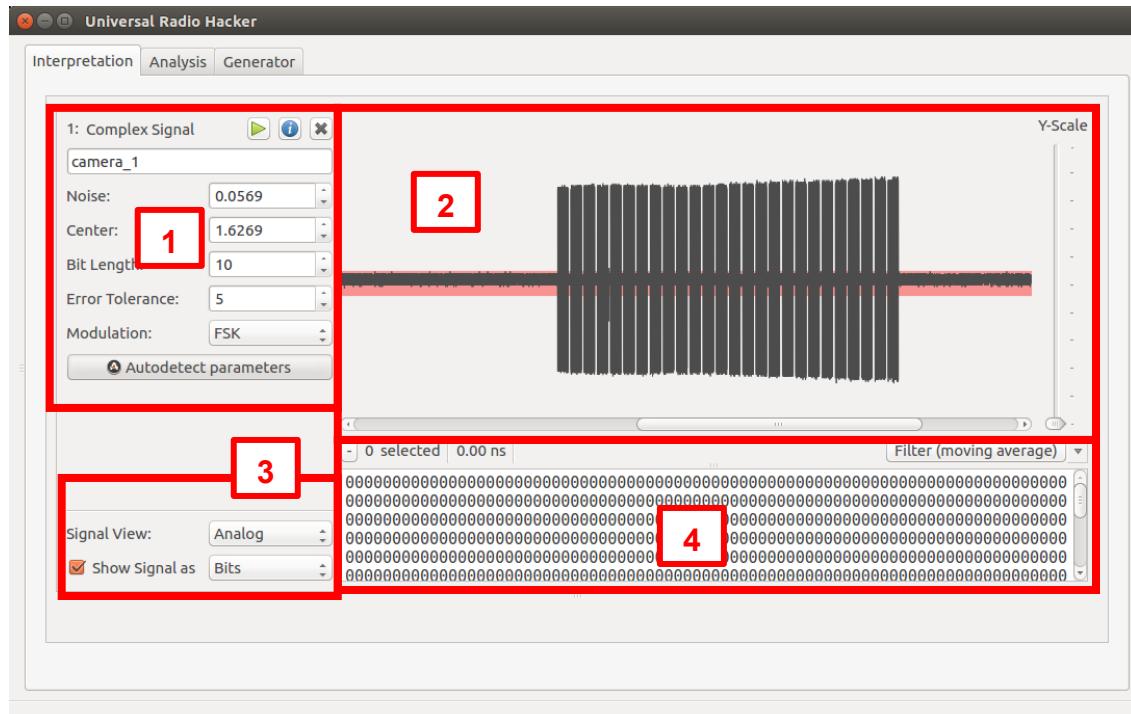


Figura 30. Universal Radio Hacker

Le sezioni sono le seguenti:

1. La configurazione base del tool per il segnale analizzato
2. L'onda che viene visualizzata con le impostazioni selezionate
3. Informazioni sulla visualizzazione dell'onda e dei dati presenti nella sezione successiva
4. Interpretazione dei dati presenti all'interno dell'onda

Il tool supporta le seguenti modulazioni: ASK PSK FSK, che possono essere utilizzate per convertire i dati. È possibile reperire maggiori dettagli sulle modalità di utilizzo del tool nel canale YouTube dello sviluppatore [24]. È inoltre possibile

visualizzare l'onda una volta che è avvenuta la modulazione; questo aiuta ad identificare se la modulazione selezionata è quella corretta o meno.



Figura 31. Visualizzazione dell'onda demodulata

Selezionando i differenti bit nell'area 4 viene automaticamente visualizzata nell'onda la stessa selezione che porta alla creazione degli stessi.

Dopo aver impostato i dati corretti è possibile passare alla modalità “Analysis”.

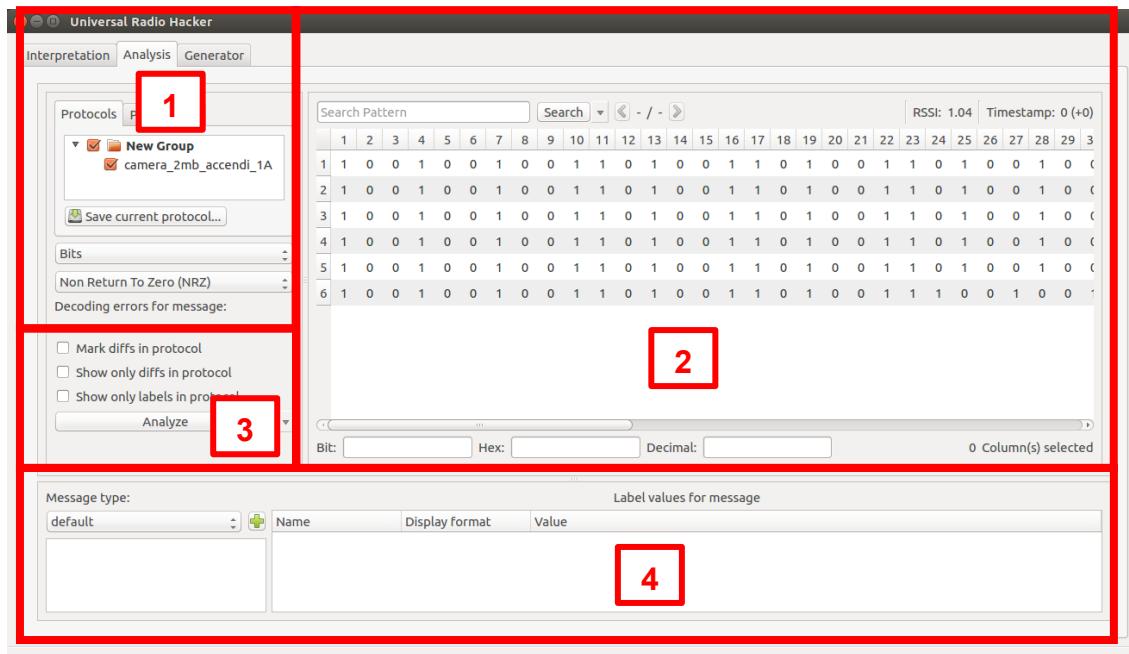


Figura 32. Area per l'analisi

Le sezioni sono le seguenti:

1. Informazioni relative alla tipologia di encoding impostato. È possibile impostare differenti tipologie di encoding che il programma già conosce o in alternativa è possibile crearne di nuove. Esiste anche la possibilità di chiamare un programma esterno che effettui il decoding e l'encoding del segnale.
 2. Area di visualizzazione delle comunicazioni avvenute: in quest'area i dati vengono automaticamente modificati a seconda della modulazione e delle altre opzioni selezionate nelle altre sezioni.
 3. Sezione di analisi che permette di visualizzare le differenze nelle differenti comunicazioni avvenute e permette di selezionare alcune modalità di

visualizzazione, ad esempio la sola visualizzazione dei bit cambiati tra le differenti comunicazioni.

4. Area di impostazione dei protocolli, in cui è possibile definire la struttura della comunicazione, in modo che il programma possa visualizzare come avviene la comunicazione ed effettuare una colorazione delle differenti aree a seconda della specifica definita.

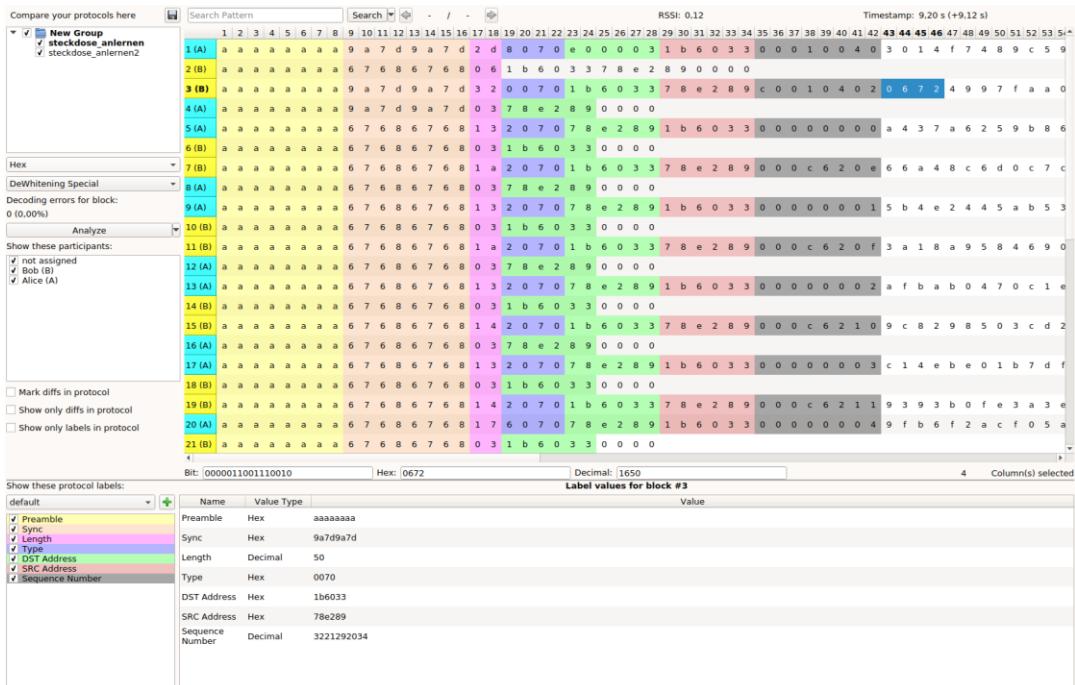


Figura 33. Esempio di decodifica del protocollo applicativo

4.8 Utilizzo di RFCAT per la ricezione e l'invio di dati

Dopo aver reperito le principali informazioni sul segnale e sulla sua struttura, si possono utilizzare queste librerie Python appositamente create per ricevere e trasmettere dati specifici. Le librerie sono compatibili con i principali apparati basati sul chipset CC1111 che viene utilizzato per le trasmissioni sub Ghz. Utilizzando questi hardware ci si trova in una situazione differente rispetto all'utilizzo di SDR come la HackRF One, infatti questa soluzione non invia il segnale grezzo al PC ma istruisce un chip hardware generico su come interpretare il segnale radio e solo dopo l'elaborazione trasmette il risultato. Questa tipologia di approccio naturalmente può essere limitante, infatti saranno utilizzabili solo ed esclusivamente le funzionalità implementate all'interno del chipset.

Le librerie forniscono le principali attività implementate dal chipset; tuttavia, per una lista completa delle funzionalità e delle impostazioni, è necessario riferirsi al datasheet del chipset stesso che è reperibile al seguente indirizzo:

- <http://www.ti.com/lit/ds/symlink/cc1110-cc1111.pdf>

Le librerie permetteranno di modificare le impostazioni presenti nel chipset che, una volta svolte alcune operazioni di ricezione e/o trasmissione, si occuperà di trasmettere il risultato sul canale di comunicazione USB.

L'utilizzo di tale librerie unite all'hardware YardStickOne rende particolarmente semplice creare degli appositi script Python per ricevere e trasmettere il segnale; tuttavia è necessario specificare una serie di parametri che devono essere impostati con precisione e sono il risultato delle precedenti analisi. Senza questi dati è impossibile la ricezione e la trasmissione delle informazioni.

Tutte le YardStickOne vengono fornite con un firmware già precaricato che sarà formato da due parti, un bootloader e un firmware vero e proprio. Il primo fornisce oltre alle funzionalità di boot del firmware anche delle funzionalità di riprogrammazione del chipset stesso in modo che sia possibile caricare un nuovo firmware sul chipset senza necessità di ulteriore hardware specifico.

È possibile reperire i sorgenti dell'ultimo firmware disponibile al seguente indirizzo <https://github.com/atlas0fd00m/rfcat> e successivamente è possibile compilarlo ed inviarlo al dispositivo mediante i seguenti comandi:

```
git clone https://github.com/atlas0fd00m/rfcat
cd rfcat
sudo cp etc/udev/rules.d/20-rfcat.rules /etc/udev/rules.d
sudo udevadm control-reload-rules
sudo apt-get install python-usb
sudo python setup.py install
```

Di seguito vengono illustrati i principali passi per poter utilizzare le librerie in questione, tutti gli script python forniti assieme a tale tesi hanno una struttura simile.

Per prima cosa è necessario importare la libreria che permetterà di comunicare con il dispositivo USB.

```
import rflib
```

Si definisce un oggetto che utilizza le librerie.

```
d = rflib.RfCat()
```

Si imposta la modulazione da utilizzare, le modulazioni supportate sono le seguenti:

- MOD_2FSK
- MOD_GFSK
- MOD_ASK_OOK
- MOD_MSK
- MANCHESTER, che può essere utilizzato in combinazione con le precedenti.

Ad esempio, è possibile impostare la modalità ASK/OOK utilizzando:

```
d.setMdmModulation(rflib.MOD_ASK_OOK)
```

si imposta la frequenza su cui lavorare

```
d.setFreq(433948000)
```

si imposta la velocità di trasmissione e/o ricezione come numero di bit al secondo

```
d.setMdmDRate(2238)
```

si imposta la presenza di un numero di preamboli e viene impostata una sync word

```
d.setPktPQT(1)
d.setMdmSyncWord(0b101010101010)
d.setMdmSyncMode(rflib.SYNCM_CARRIER_16_of_16)
```

e per finire si riceve o si trasmette un dato

```
y, t = d.RFrecv(timeout=30000)
d.RFxmit(rf_data,repeatt=1)
```

5 Case study analizzati

5.1 Telecomando per l'accensione di una presa elettrica

Il primo dispositivo analizzato è una presa elettrica 220v comandata da un telecomando. Il telecomando è in grado di comandare contemporaneamente tre di queste prese ed inoltre è possibile impostarlo su differenti frequenze utilizzando uno switch a posizioni situato nel retro.

In primis è stata identificata la frequenza utilizzata dal telecomando utilizzando l'analizzatore di spettro.

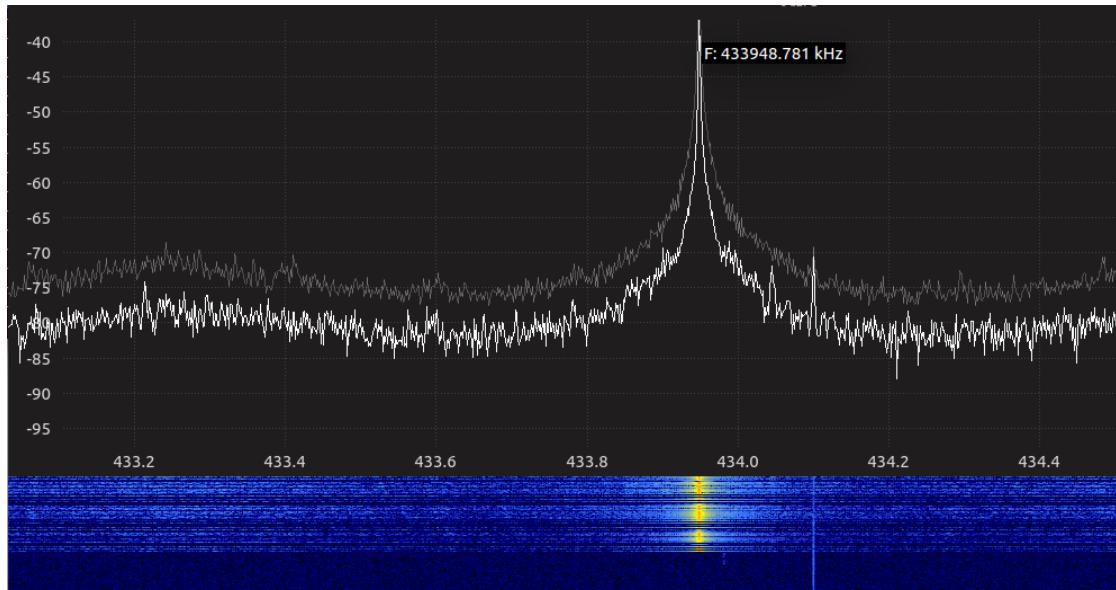


Figura 34. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 433.948 Mhz, che differisce parzialmente da quella riportata nel retro del dispositivo che è di 433.92 Mhz.

Viene registrato il segnale e si utilizza Inspectrum per analizzare la struttura del segnale.

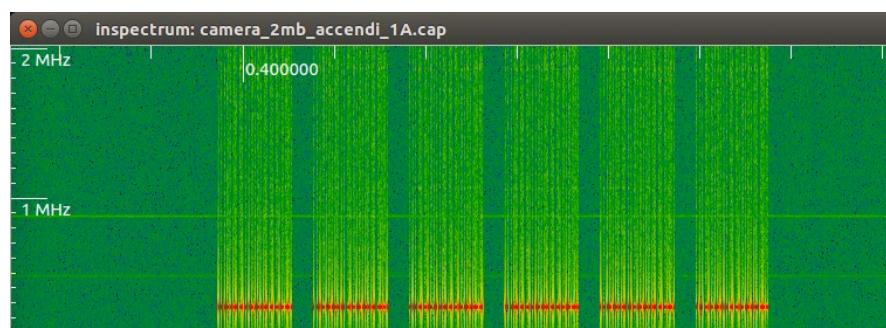


Figura 35. Comunicazione completa

Si nota la presenza di 6 aree di trasmissione, utilizzando i controlli relativi alla potenza del segnale e allo zoom si analizza una singola area.

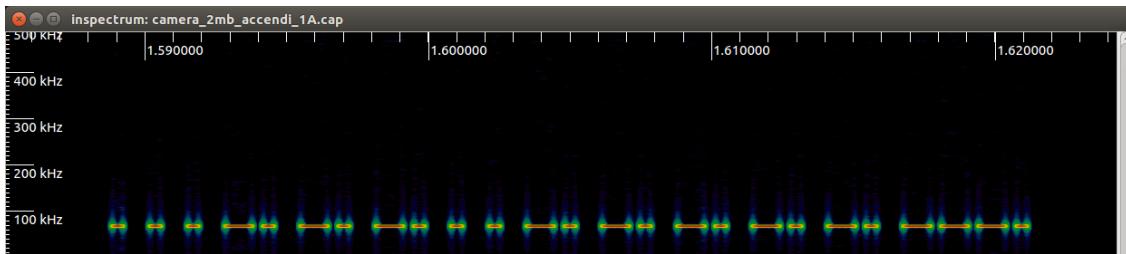


Figura 36. Zoom su singola trasmissione

È possibile vedere come:

- non vi sia una variazione della frequenza, che esclude quindi la variazione della modulazione FSK
- non vi è una differenza sostanziale tra la potenza di trasmissione, ma sembra che vi sia un codice basato su una modulazione "trasmetto" o "non trasmetto". Questa analisi permette di affermare che la modulazione utilizzata è OOK.
- sono presenti 25 segnali differenti, intermezzati da assenza di trasmissione.
- i segnali sono o lunghi o corti e questo vale anche per la parte di assenza di trasmissione.
- non sembrano essere presenti preamboli e/o sync word all'inizio del segnale.
- le 6 aree di trasmissione sopra citate sono la ripetizione sempre dello stesso segnale.

Utilizzando uno zoom ulteriore è possibile utilizzare lo strumento cursors per misurare le caratteristiche di ogni singola trasmissione.

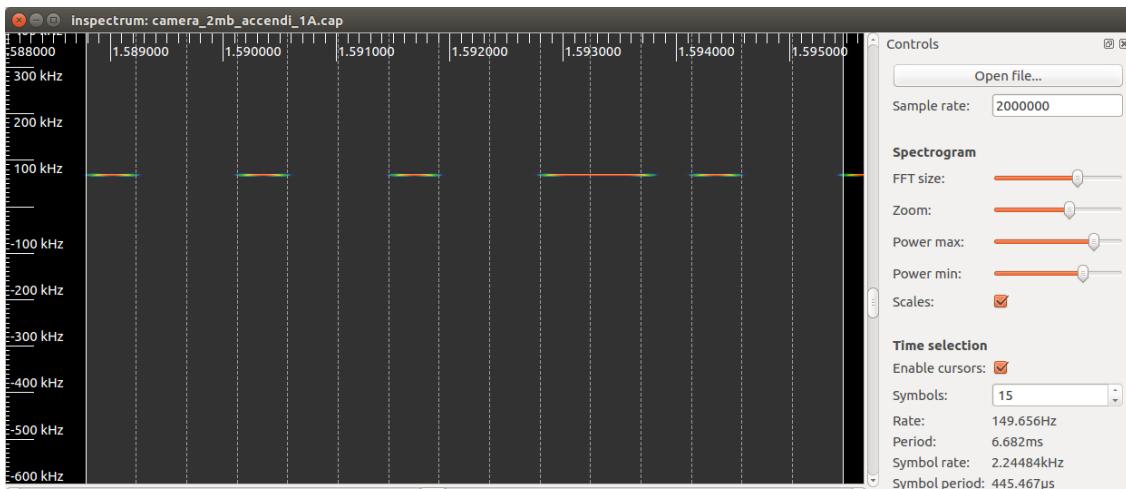


Figura 37. Misurazione dei tempi del segnale

Risulta che ogni singolo simbolo ha una durata di 445us per segnali corti e 990us per quelli lunghi.

$$\text{Numero di trasmissioni massime in } 1s = \frac{1s}{445\mu s} = 2247$$

Visto che la registrazione è stata effettuata a 2M campionamenti/secondo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{2247} = 890$$

Si utilizza il tool URH impostando i dati rilevati che permetterà di “tradurre” in modo veloce il segnale in bits.

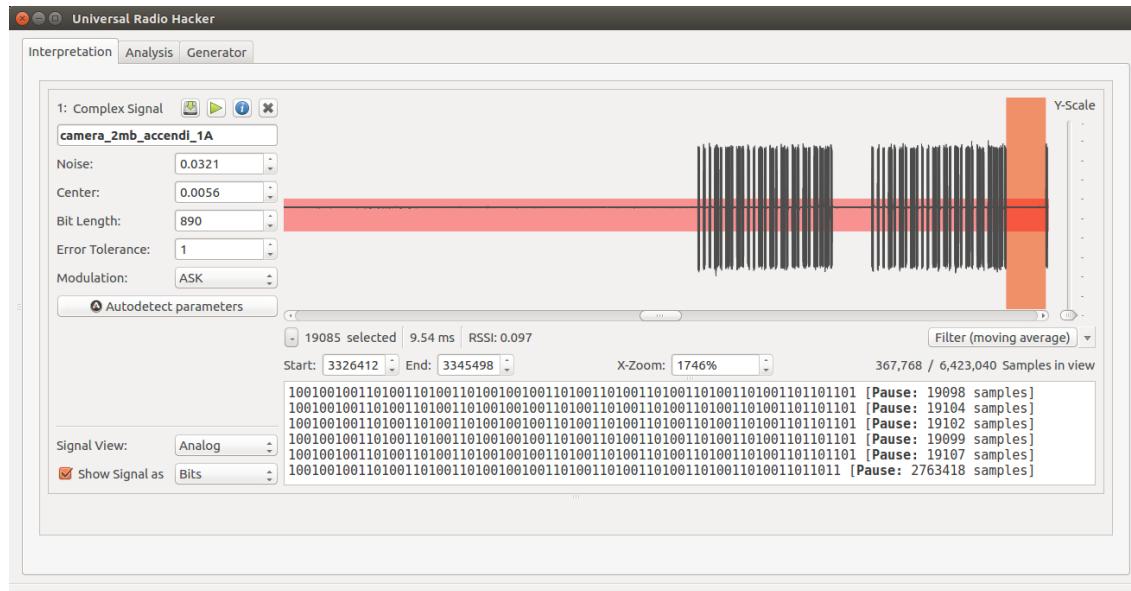


Figura 38. Traduzione del segnale in bits

Il software identifica 6 sequenze di dati identiche, ognuna formata da 73 bits. Si segnala come il tool automaticamente separi le sequenze a seconda del numero di spazi vuoti tra una sequenza e l'altra, ciò nonostante il software non segnalera gli 0 nell'area in basso.

Analizzando la sequenza è possibile notare che è possibile suddividere la sequenza a gruppi di 3, rispettando le seguenti regole:

- il primo bit sarà sempre 1
 - l'ultimo bit sarà sempre 0

L'unica eccezione è l'ultimo bit dell'intera trasmissione, che viene segnato come da solo, in questo caso ci sono 2 ipotesi:

- è un bit di fine trasmissione e quindi non è da tenere in considerazione
 - il software commette un errore non interpretando i 2 bit successivi (entrambi a 0)

Il rispetto delle regole sopra citate a gruppi di 3 suggerisce che l'encoding dei dati utilizzati sia PWM, una sequenza formata da 100 sarà ritenuto un bit 0 e una sequenza formata da 110 sarà un bit a 1.

Utilizzando le specifiche funzionalità del software è possibile creare un apposito encoding in grado di decodificare il dato grezzo. Il modo più semplice per effettuare questo encoding è quello di prendere in considerazione solo il bit centrale, tralasciando gli altri 2 bit.

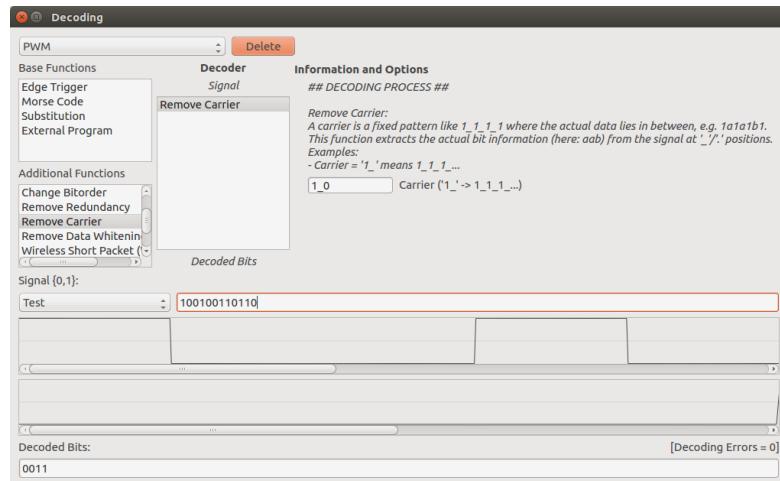


Figura 39. Impostazione dell'encoding

Una volta effettuata questa trasformazione è possibile lavorare sul dato reale, cioè 24 bit (se si ipotizza che l'ultimo sia un bit di fine trasmissione).

Una volta identificati questi dati sono stati ricevuti i differenti tasti del telecomando variando anche il selettori presenti sul retro. Vengono segnalati in rosso le differenze rispetto alla prima trasmissione.

Comando	Bit ricevuti
Accendi 1°	000101010001010101010111
Spegni 1°	000101010001010101010100
Accendi 2°	000101010100010101010111
Spegni 2°	000101010100010101010100
Accendi 3°	000101010101000101010111
Spegni 3°	000101010101000101010100
Accendi 1B	010001010001010101010111
Accendi 1C	010100010001010101010111
Accendi 1D	010101000001010101010111

È possibile notare inoltre una strana anomalia nei dati; in tutte le trasmissioni al di fuori degli ultimi 3 bit è possibile identificare un pattern in cui il primo valore è sempre 0 (evidenziati in verde nell'ultima riga). È possibile che tale comportamento sia un caso o alternativamente è anche possibile che il primo bit venga sempre utilizzato come meccanismo di controllo per la trasmissione.

Possiamo quindi riassumere il segnale ricevuto come segue:

- WWWWWWWWWWWXXXXXXYYYYYYYYZZ

Dove:

- W, è l'identificativo del device (A,B,C,D)
- X, è l'identificativo del pulsante premuto (1,2,3)
- Y, risulta essere una parte statica e non variata nel nostro caso
- Z, sono i bit relativi al segnale di accensione (11) e spegnimento (00)

Intercettazione

Il protocollo applicativo non presenta alcuna misura atta alla limitazione dell'intercettazione e dell'interpretazione dei dati trasmessi, pertanto tale attacco è di semplice applicazione.

Spoofing

Una volta individuato il protocollo applicativo e capito a cosa servono i differenti bit è facile creare una trasmissione con un preciso scopo.

Replay attack

Il codice trasmesso è fisso, pertanto la ritrasmissione dello stesso dato genererà lo stesso evento. In questo caso è anche possibile ritrasmettere il dato senza capirne il contenuto.

Jamming

Essendo un hardware molto economico non è stata implementata nessuna protezione rispetto ad attacchi di tipo jamming, pertanto il device non è in grado di rilevarli e/o di prevenirli.

Brute force

Nel caso in cui si volesse effettuare un brute force sui bit che identificano il devices (WWWWWWWWWWWW) e quelli che attualmente sono fissi (YYYYYYYY) ma che potrebbero cambiare ad esempio nel caso di differenti marche, ci si trova di fronte al caso di 16 bit con 2^{16} possibili combinazioni. Nel caso in cui invece si scoprissse che il primo bit della coppia è sempre a 0 allora si parlerebbe di 2^8 possibili combinazioni.

Viene stimato un attacco di questo tipo tenendo conto di trasmettere solo una volta la sequenza e di aspettare 1ms tra una trasmissione e l'altra.

$$\begin{aligned} \text{N° massimo di trasmissioni/s} &= \frac{1s}{((445\mu s * 3 * 24) + 1ms)} = \sim 30 \\ \text{Tempo necessario su } 2^{16} &= \frac{2^{16}}{30} = 2184s = \sim 36 \text{ minuti} \\ \text{Tempo necessario su } 2^8 &= \frac{2^8}{30} = \sim 8s \end{aligned}$$

Si riepilogano di seguito le caratteristiche del segnale analizzato.
Si riepiloga in una tabella riassuntiva la suscettibilità ad attacchi.

Caratteristica	Dato
Frequenza	433.948 Mhz
Segnale minimo	445us
Symbol rate	2247

Modulazione	ASK/OOK
Encoding	PWM

Si riepilogano di seguito le caratteristiche del segnale analizzato.

Si riepiloga in una tabella riassuntiva la suscettibilità ad attacchi.

Attacco	Vulnerabilità	Note
Intercettazione	Sì	La frequenza è fissa e il dato anche
Spoofing	Sì	
Replay attack	Sì	Il dato è sempre uguale
Jamming	Sì	Possibile
Brute force	Sì	Possibile
OpenSesame	No	Il protocollo non usa registry shifting
RollJam	N/A	Inutile a causa di attacchi più semplici
MITM	N/A	Inutile a causa di attacchi più semplici

Nel corso della tesi sono stati sviluppati appositi script per:

- ricezione dei dati del telecomando
- invio dei dati
- jamming del segnale

5.2 Sistema di sicurezza basato su protocollo X10

È stato analizzato un telecomando di sicurezza utilizzato su un antifurto basato sul protocollo X10, molto diffuso negli USA. Il modello analizzato è basato sui modelli reperibili in Europa, pertanto le frequenze saranno sicuramente differenti rispetto a quelle reperibili in America. Il modello di telecomando è KR21E e presenta 4 pulsanti principali: un pulsante per disattivare totalmente l'allarme, uno per attivare completamente l'allarme, uno per attivare solo il perimetrale e uno per accendere/spegnere la luce.



In primis si identifica la frequenza utilizzata dal telecomando utilizzando l'analizzatore di spettro.

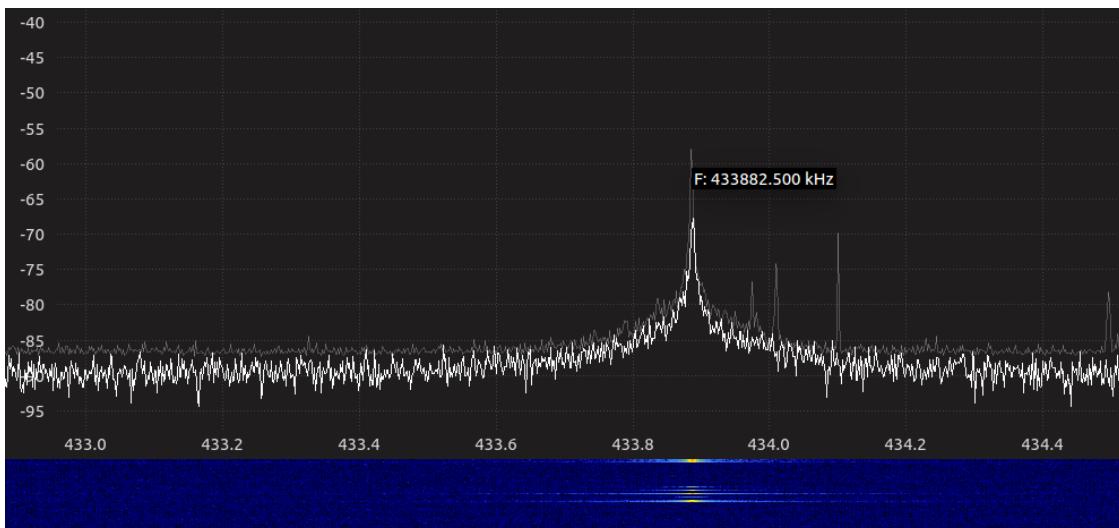


Figura 40. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 433.882 Mhz, effettuando lo stesso procedimento su differenti telecomandi è stata riscontrata che la frequenza utilizzata differisce tra un dispositivo e l'altro.

Viene registrato il segnale e si utilizza Inspectrum per analizzare la struttura del segnale.

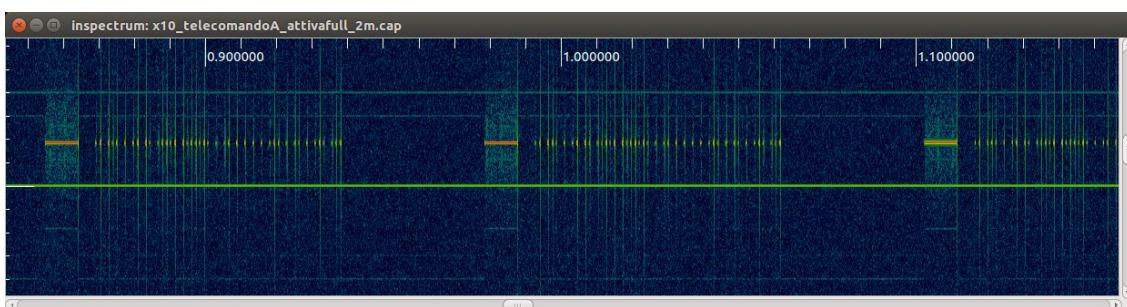


Figura 41. Comunicazione per le prime 3 ripetizioni

Si nota la presenza di 5 aree di trasmissione; utilizzando i controlli relativi alla potenza del segnale e allo zoom viene ora analizzata una singola area.

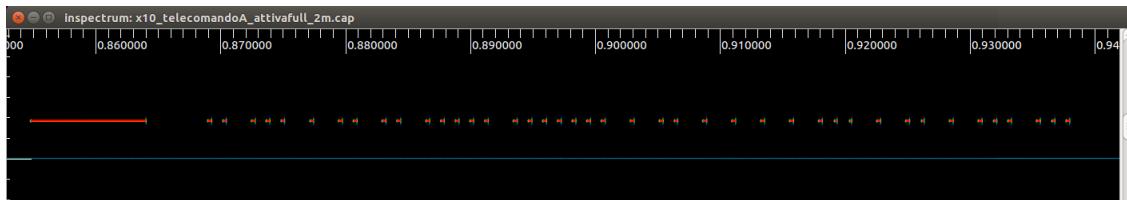


Figura 42. Zoom su singola trasmissione

È possibile vedere come:

- non vi sia una variazione della frequenza, che esclude quindi la variazione della modulazione FSK
- non vi è una differenza sostanziale tra la potenza di trasmissione, ma sembra che sia un codice basato su una modulazione basata sul “trasmetto” o “non trasmetto”. Questa analisi permette di affermare che la modulazione utilizzata è OOK.
- è presente un preambolo che consiste nella trasmissione continua per circa 9ms che identifica l'inizio della trasmissione
- tutti i segnali a eccezione del preambolo hanno lunghezza uguale, la lunghezza che cambia è quella della non trasmissione.

Utilizzando uno zoom ulteriore è possibile utilizzare lo strumento cursors per misurare le caratteristiche di ogni singola trasmissione.

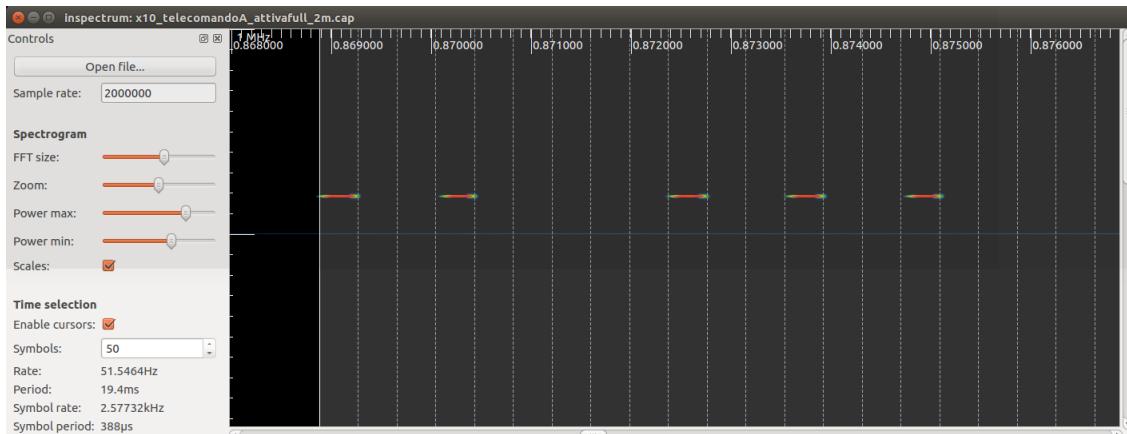


Figura 43. Misurazione dei tempi del segnale

Risulta che ogni singolo simbolo abbia una durata di 388us e anche gli spazi vuoti siano un multiplo dello stesso tempo

$$\text{Numero di trasmissioni massime in } 1s = \frac{1s}{388\mu s} = 2577$$

Essendo che la registrazione è stata effettuata a 2M campionamenti/secondo questo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{2577} = 776$$

È possibile utilizzare il tool URH che, impostando i dati rilevati, permette di tradurre in modo veloce il segnale in bits.

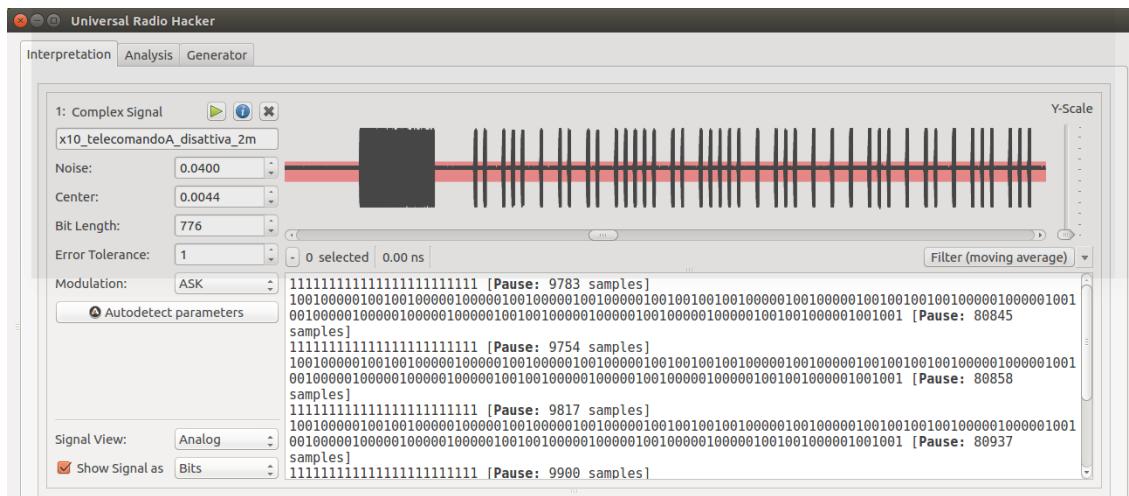


Figura 44. Traduzione del segnale in bits

La trasmissione è un susseguirsi di due sequenze, “100” e “10000”. Effettuando alcune ricerche su Internet non è stato identificato nessun encoding che funzioni in questo modo, tuttavia è stato individuato come tutte le comunicazioni avvenute, anche utilizzando telecomandi differenti, abbiano esattamente 42 sequenze.

È stato ipotizzato che i bit potessero essere rappresentati in modo differente rispetto ai classici encoding, ipotizzando di convertire gli "10000" in 1 e i "100" in 0.

Per effettuare i test è stato sviluppato un piccolo script python che effettua questo decoding, in modo da poterlo utilizzare dentro URH.

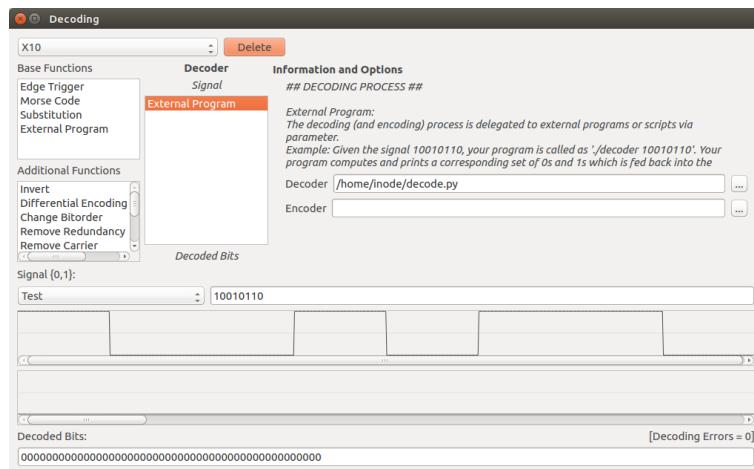


Figura 45. Impostazione dell'encoding

Una volta effettuata questa trasformazione è stato possibile lavorare sul dato reale, cioè 42 bit. Per facilitare alcune conversioni in esadecimale lo script che effettua il decoding del segnale restituisce sempre 48 bit, anche se realmente quelli trasmessi sono solo 42.

Comando	Bit ricevuti
A - Disattivazione	101000011010111010000110011100100000000000
A - Full	101000011010111000000110111100100000000000
A - Perimetro	10100001101011100000110111100100000000000
A - Luce (on)	1010000110101110010001101011100100000000000
A - Luce (off)	101000011010111010001100011100100000000000
B - Disattivazione	00001110000000001000011001110011010001000

Effettuando alcuni controlli sugli ID segnalati sul sistema di antifurto siamo in grado di identificare parzialmente i dati presenti all'interno della comunicazione. Ulteriori analisi hanno permesso di dedurre che gli ulteriori bytes presenti sono dei blocchi di controllo basati su degli XOR algebrici.

E' possibile quindi riassumere il segnale ricevuto come segue:

- **UUUUUUUU**VVVVVVVVWWWWWWWWWWXXXXXXYYYYYYZZ

Dove:

- **U** è il primo byte relativo all'identificativo del telecomando
- **V** è lo xor tra il byte precedente e 0x0F
- **W** è l'identificativo della funzionalità richiesta
- **X** è lo xor tra il byte precedente e 0xFF
- **Y** è il secondo byte relativo all'identificativo del telecomando
- **Z** è il terzo byte relativo all'identificativo del telecomando

Si fa notare come **Z** è formato da soli 2 bits, essendo la comunicazione totale da 42 bits.

W risulta avere il seguente valore a seconda del tasto premuto:

- 0x86 nel caso di disattivazione dell'allarme
- 0x06 nel caso di attivazione completa dell'allarme
- 0x0E nel caso dell'attivazione dell'allarme perimetrale
- 0x46 nel caso dell'utilizzo del tasto luce
- 0xC6 nel caso dell'utilizzo del tasto luce (alla seconda pressione)

Intercettazione

Il protocollo applicativo non presenta alcuna misura atta alla limitazione dell'intercettazione e all'interpretazione dei dati trasmessi, pertanto tale attacco è di semplice applicazione.

Spoofing

Una volta individuato il protocollo applicativo e capito a cosa servono i differenti bit è facile creare una trasmissione con un preciso scopo.

Replay attack

Il codice trasmesso è fisso, pertanto la ritrasmissione dello stesso dato genererà lo stesso evento. In questo caso è anche possibile ritrasmettere il dato senza capirne il contenuto.

Jamming

Non sono stati rilevati meccanismi atti a implementare una protezione rispetto ad attacchi di tipo jamming, pertanto il device non è in grado di rilevarli e/o di prevenirli.

Brute Force

Nel caso in cui si volesse effettuare un brute force sui bit che identificano il telecomando (**U**, **Y** e **Z**) vuol dire effettuare l'attacco verso 18 bit con 2^{18} possibili combinazioni. Viene stimato ora un attacco di questo tipo tenendo conto di trasmettere solo una volta la sequenza e di aspettare 1ms tra una trasmissione e l'altra.

È importante notare come la comunicazione che avviene cambia di lunghezza a seconda dei bit presenti; infatti, la trasmissione dei bit impostati a 1 (2328us) richiede il doppio del tempo rispetto alla trasmissione di uno 0 (1164us).

Stimando che per esplorare l'intero spazio di chiavi la quantità di 1 e di 0 sarà identica nel tempo è possibile definire i seguenti tempi.

$$N^{\circ} \text{ massimo di trasmissioni/s} = \frac{1s}{((1746\mu s * 42) + 1ms)} = \sim 13$$

$$\text{Tempo necessario su } 2^{18} = \frac{2^{18}}{13} = 20165s = \sim 5,6 \text{ ore}$$

Si riepilogano di seguito le caratteristiche del segnale analizzato.

Caratteristica	Dato
Frequenza	433.882 Mhz
Segnale minimo	388us
Symbol rate	776
Modulazione	ASK/OOK
Encoding	Proprietario

Si riepiloga in una tabella riassuntiva la suscettibilità ad attacchi.

Attacco	Vulnerabilità	Note
Intercettazione	Sì	La frequenza è fissa e il dato anche
Spoofing	Sì	
Replay attack	Sì	Il dato è sempre uguale

Jamming	Sì	Possibile
Brute force	Sì	Possibile
OpenSesame	No	Il protocollo non usa registry shifting
RollJam	N/A	Inutile a causa di attacchi più semplici
MITM	N/A	Inutile a causa di attacchi più semplici

Nel corso della tesi sono stati programmati appositi script per:

- ricezione dei dati del telecomando
- invio dei dati arbitrari
- jamming del segnale

5.3 Sistema di chiusura centralizzata Fiat Stilo

È stato analizzato un telecomando di apertura remota dell'automobile Fiat Stilo del 2005. L'analisi si è concentrata sul protocollo di comunicazione che avviene quando vengono premuti i pulsanti di apertura o chiusura veicolo. Non è stato analizzato invece il comportamento del chip di prossimità presente all'interno della chiave e utilizzato durante la verifica dell'autenticità della stessa durante la fase di accensione del veicolo.

In primis è stata identificata la frequenza utilizzata dal telecomando utilizzando l'analizzatore di spettro.

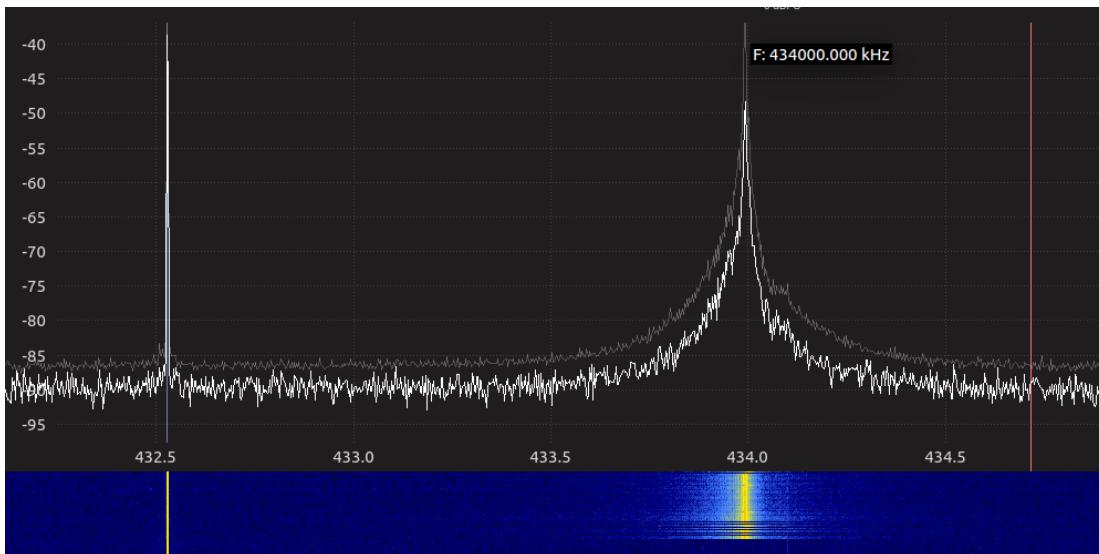


Figura 46. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 434.000 Mhz, l'utilizzo di una frequenza così precisa mette in luce come probabilmente l'hardware presente all'interno del dispositivo sia di qualità, infatti generalmente è difficile avere una tale precisione nella selezione della frequenza.

Viene registrato il segnale e si utilizza Inspectrum per analizzare la struttura del segnale.

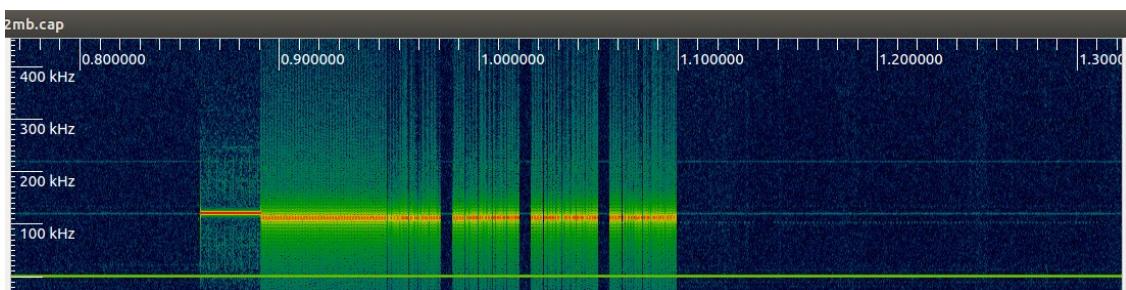


Figura 47. Segnale nel suo complesso

In questo caso è possibile notare le seguenti caratteristiche:

- all'inizio della trasmissione vi è una trasmissione continua per un periodo di tempo di 30ms su una frequenza differente da quella utilizzata per poi trasmettere i dati.
- Esiste una prima trasmissione “anomala” e successivamente 3 trasmissioni che vengono effettuate a distanza fissa.
- Ogni trasmissione è caratterizzata da una prima sezione, un periodo di stop e successivamente da una trasmissione più lunga.

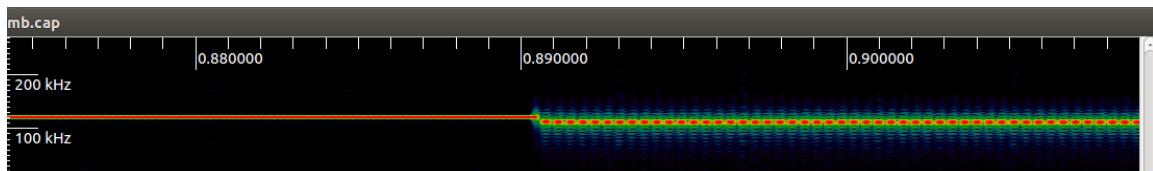


Figura 48. Differenza di frequenza sulla trasmissione iniziale

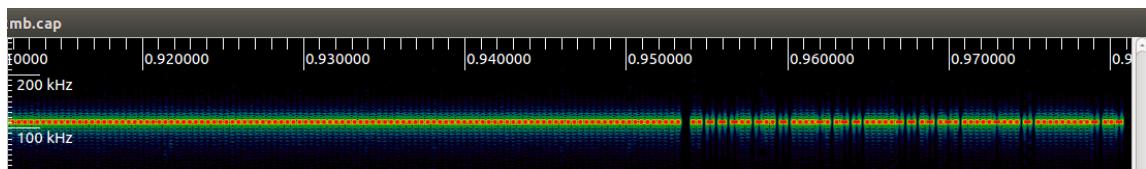


Figura 49. Anomalia nella prima trasmissione

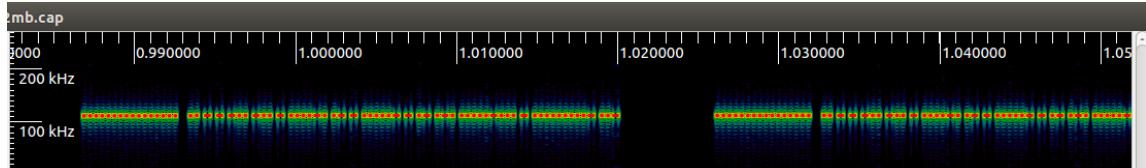


Figura 50. Successive trasmissioni

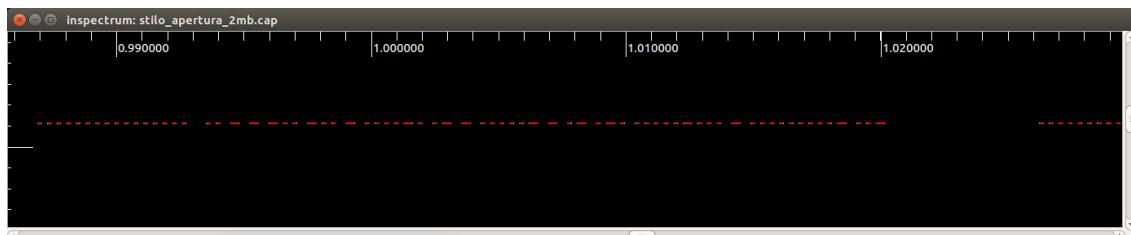


Figura 51. Zoom su singola trasmissione

È possibile vedere come:

- non vi sia una variazione della frequenza, che esclude quindi la variazione della modulazione FSK, la variazione all'inizio della sequenza può essere ritenuta un segnale di start.
- non vi è una differenza sostanziale tra la potenza di trasmissione, ma sembra che sia un codice basato su una modulazione basata sul “trasmetto” o “non trasmetto”. Questa analisi permette di affermare che la modulazione utilizzata è OOK.
- tutte le trasmissioni sono precedute da un preambolo, da un periodo di pausa e poi dal dato reale
- i segnali sono o lunghi o corti e questo vale anche per la parte di assenza di trasmissione.
- i dati presenti nelle 4 trasmissioni sono la ripetizione sempre dello stesso segnale.

Utilizzando uno zoom ulteriore è possibile utilizzare lo strumento cursors per misurare le caratteristiche di ogni singola trasmissione.

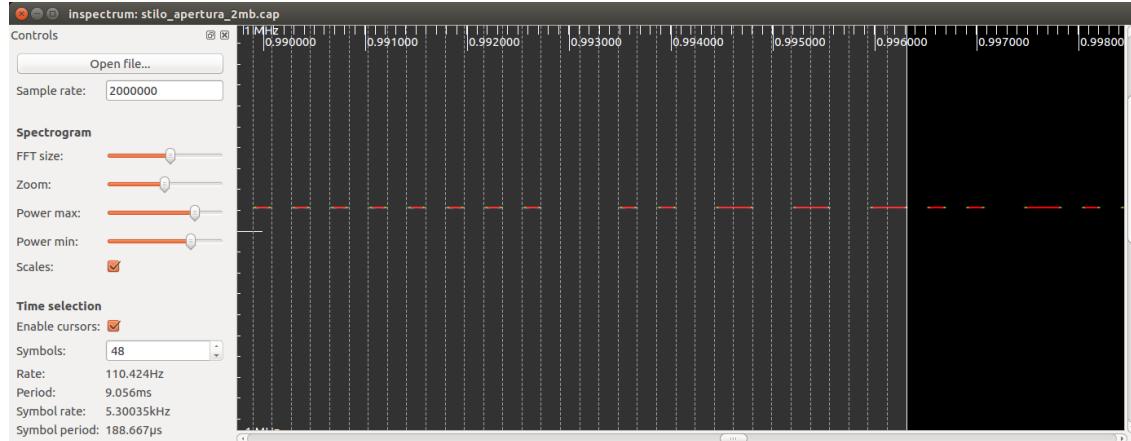


Figura 52. Misurazione dei tempi del segnale

Risulta che ogni singolo simbolo abbia una durata di 118us per segnali corti e 236us per quelli lunghi.

$$\text{Numero di trasmissioni massime in } 1\text{s} = \frac{1\text{s}}{118\mu\text{s}} = 5320$$

Visto che la registrazione è stata effettuata a 2M campionamenti/secondo questo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{2247} = 375$$

È possibile utilizzare ora il tool URH impostando i dati rilevati permette di tradurre in modo veloce il segnale in bits.

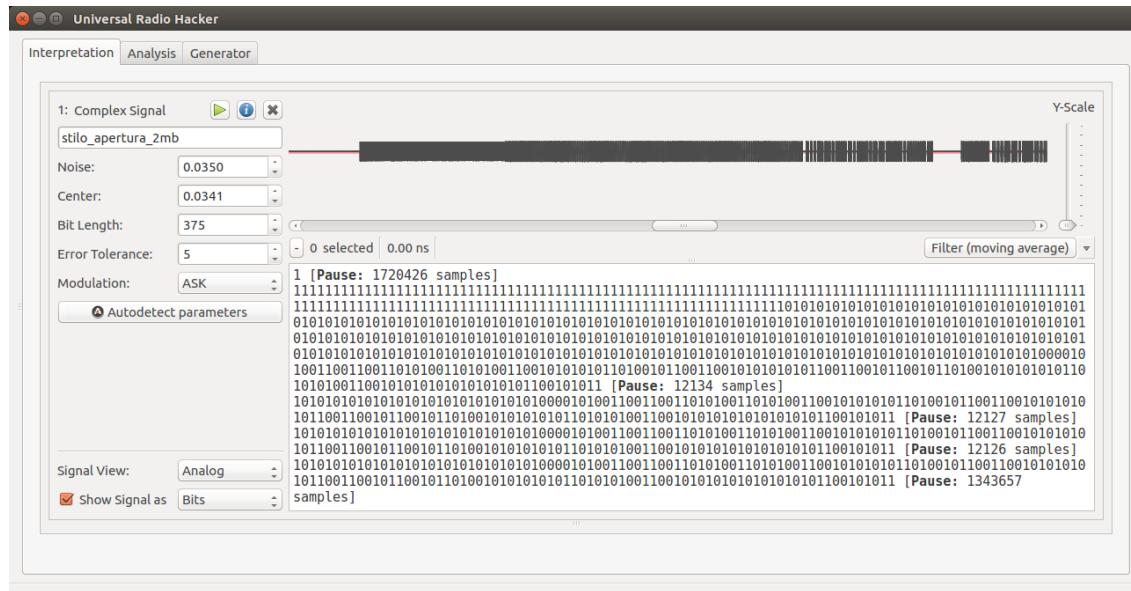


Figura 53. Traduzione del segnale in bits

Il software a questo punto identifica una serie di frequenze differenti. Si riepilogano di seguito le caratteristiche:

- la prima parte risulta essere il segnale costante per un totale di 30.36ms, è quel segnale che è stato visto precedentemente su una frequenza differente
- una pausa di 790us
- la parte successiva è formata da 331 simboli per un totale di 62.58ms
- una pausa di 790us
- 141 bit per un totale di 26.64ms
- una pausa di circa 6ms
- un nuovo preambolo questa volta formato da 31 simboli per un totale di 5.83ms
- una pausa di 790us
- 141 bit per un totale di 26.64ms
- la ripetizione delle ultime 4 cose per altre 2 volte

Prendendo in considerazione solo l'area di dati e analizzando i bit è possibile accorgersi che dividendo il flusso a coppie di bit queste coppie sono sempre formate da un 1 e uno 0, ma da due simboli dello stesso tipo. Questo permette di identificare che l'encoding utilizzato è molto probabilmente il Manchester.

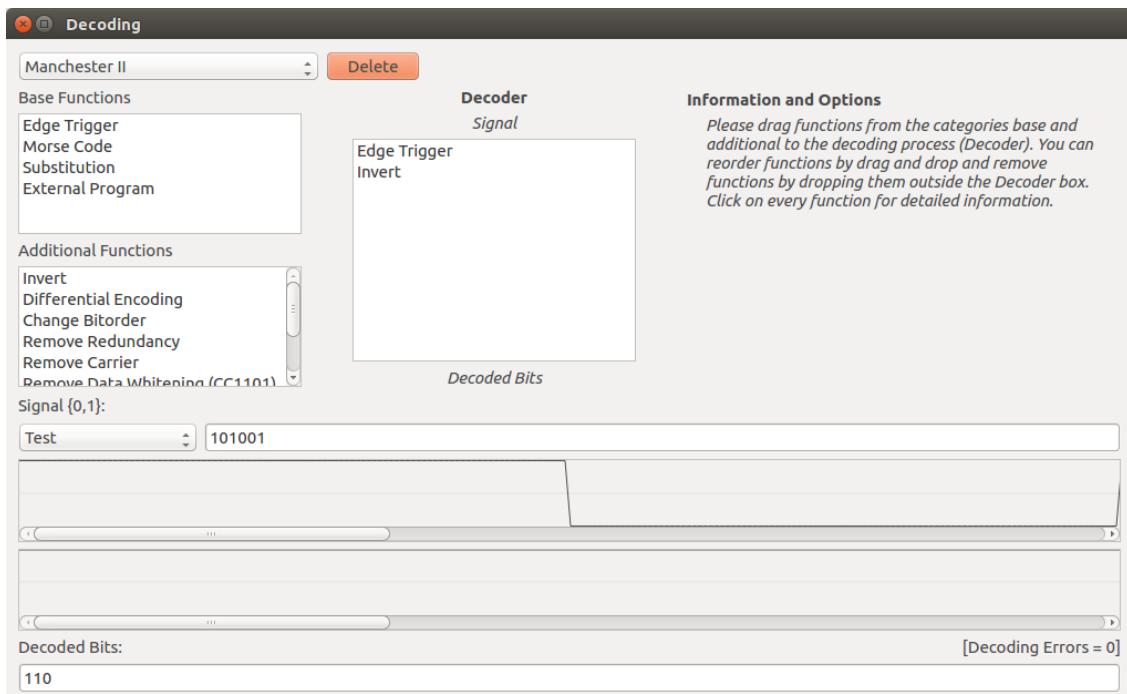


Figura 54. Impostazione dell'encoding

Una volta effettuata questa trasformazione è possibile lavorare sul dato reale, cioè 71 bits. Sono stati eseguiti quindi vari test con due differenti telecomandi.

Comando	Bit ricevuti
Telecomando A	1000010111001101000011010111011101010010011000000111101000000000001111
Telecomando A	1000010111001101000011010111011101010010011000000111101000000000001111
Telecomando A	101111101011000111100000010100100010100100110000001111010000000000011001
Telecomando A	101111101011000111100000010100100010100100110000001111010000000000011001
Telecomando A	11011101100111101000001110001101010010011000000111101000000000011111
Telecomando A	100100100001000100000011011011001011011011100011110010000000000010111
Telecomando B	11101110000001100110010001101101101100001101101101110001111001000000000010111
Telecomando B	1111011001011001110110110000110110110111000111100100000000000111
Telecomando B	100100111100000100101000010011000011011011100011110010000000000010001

Telecomando B	1000100101111100001010011111000110110111000111100100000000000000101
Telecomando B	100001011100110100001101011101110101001001100000111101000000000001111
Telecomando B	100001011100110100001101011101110101001001100000111101000000000001111
Telecomando B	10111110101100011110000010100100101001001100000111101000000000011001

L'utilizzo di differenti tasti presenti sul telecomando non varia in modo sensibile ed è possibile riassumerla come segue:

- 1 bit sempre uguale
- 32 bit che cambiano ogni comunicazione
- 37 bit fissi che variano a seconda del telecomando utilizzato
- 4 bit che variano ogni trasmissione
- 1 bit sempre uguale

Dopo aver effettuato vari test e senza essere giunti all'identificazione di come avviene applicativamente la comunicazione, si deciso di aprire uno dei telecomandi per cercare maggiori informazioni.

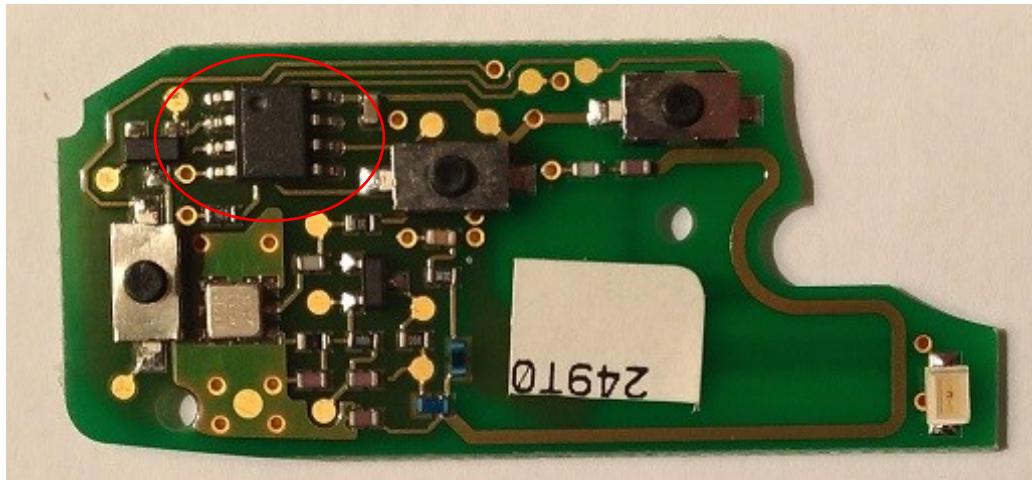


Figura 55. Scheda elettronica all'interno del telecomando

La scheda elettronica presente all'interno del telecomando contiene numerosi componenti; tuttavia è presente un solo chip. Effettuando la lettura del codice presente sopra il chip (HCS412) e ricercando il relativo datasheet [25] è stato possibile identificarlo come "KEELOQ Code Hopping Encoder and Transponder".

L'analisi del documento di specifica del chipset ha permesso di reperire maggiori informazioni sulla comunicazione, che sono riepilogate di seguito:

- la comunicazione inizia con un bit di inizio comunicazione e uno di fine comunicazione
- 32 bit una comunicazione cifrata con una chiave condivisa che contiene varie informazioni, tra cui il pulsante schiacciato e un numero generato da un seme condiviso che tiene conto anche del numero di volte che è stato schiacciato il pulsante.
- 32 bit che identificano il seriale del telecomando
- 1 bit che segnala il caso in cui la batteria sia in esaurimento
- 2 bit di CRC

- 2 bit che permettono di identificare le comunicazioni sequenziali

Applicando le informazioni reperite ai dati acquisiti è quindi possibile definire il protocollo utilizzato come segue:

-  XYZ

Dove:

- **V** parte del codice cifrata che contiene il seme
 - **W** codice seriale del telecomando e 1 bit che permette di identificare la batteria bassa
 - **X** CRC dei dati precedenti
 - **Y** Codice che permette di identificare due trasmissioni successive
 - **Z** bit di start e fine trasmissione

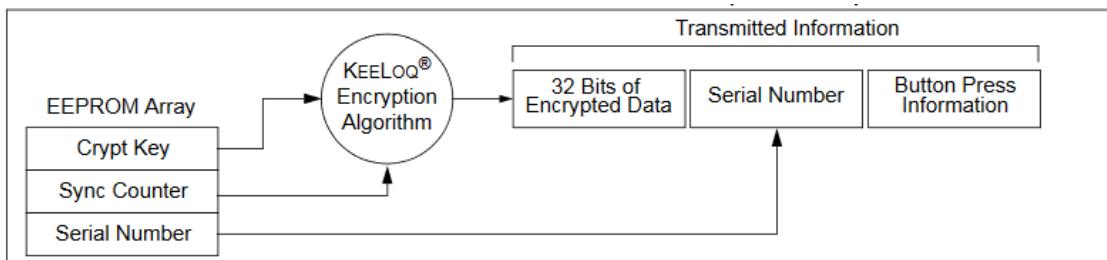


Figura 56. Schema per la generazione dei 32 bit cifrati

and $CRC[1]_{n+1} = CRC[0]_n \oplus Di_n$
 with $CRC[0]_{n+1} = (CRC[0]_n \oplus Di_n) \oplus CRC[1]_n$

Figura 57. Meccanismo di CRC

Intercettazione

Anche se i devices possono essere soggetti a intercettazione, i dati trasmessi risultano essere cifrati e di difficile interpretazione.

Spoofing

I meccanismi di cifratura e generazione del codice basato sul seme non permettono di creare richieste arbitrarie senza essere a conoscenza delle chiavi utilizzate.

Replay attack

All'interno dei dati cifrati esiste un codice che viene generato a partire da un seme condiviso e che cambia a seconda del numero di generazioni avvenute. L'auto memorizza l'ultimo codice utilizzato e invalida tutti i possibili codici precedenti. Effettuando un replay attack il pacchetto applicativo risulterà essere

corretto, ma tuttavia l'auto sarà in grado di rilevare che il codice è già stato utilizzato.

Jamming

Non sono stati rilevati meccanismi atti ad implementare una protezione rispetto ad attacchi di tipo jamming, pertanto il device non è in grado di rilevarli e/o di prevenirli.

Brute Force

Avendo già intercettato il seriale del telecomando è possibile effettuare un brute force sulla parte cifrata anche senza conoscerne il contenuto per un totale 32 bit con 2^{32} possibili combinazioni.

Viene stimato ora un attacco di questo tipo tenendo conto di trasmettere solo una volta la sequenza e di aspettare 1ms tra una trasmissione e l'altra.

$$N^{\circ} \text{ massimo di trasmissioni}/s = \frac{1s}{((118\mu s * 2 * 71) + 1ms)} = \sim 56$$

$$\text{Tempo necessario su } 2^{32} = \frac{2^{32}}{56} = 76695844s = \sim 2.5 \text{ anni}$$

Dati i tempi necessari per poter effettuare tale attacco è da ritenere una strada non percorribile. Un'alternativa può essere quella di effettuare un'intercettazione di un codice e successivamente cercare di identificare la chiave condivisa e il seme utilizzato dal telecomando. Esistono alcuni studi svolti mediante l'uso di GPU (Graphical Processing Unit) che dimostrano come tale strada sia possibile attraverso un hardware relativamente economico [26].

Roll Jam

È stata analizzata la possibilità di effettuare un attacco "RollJam", utilizzando una vulnerabilità tecnologica unita ad un comportamento umano.

Sono state utilizzate due differenti SDR per eseguire l'attacco con successo. La prima effettua un jamming delle frequenze vicine a quelle del telecomando; così facendo l'auto non riuscirà a riconoscere il codice inviato e in modo quasi automatico il proprietario premerà nuovamente il tasto del telecomando inviando un secondo codice.

La seconda SDR sarà in attesa di questi codici e una volta ricevuto il secondo codice interromperà il jamming e si invierà il primo codice ricevuto. In questo modo il secondo codice sarà valido e potrà essere utilizzato successivamente.

Man In The Middle

Anche se tecnicamente percorribile, tale attacco non può essere utilizzato in questo dispositivo perché è necessario premere un pulsante e pertanto non può essere sfruttato. Caso differente se invece il dispositivo fosse di prossimità.

Si riepilogano di seguito le caratteristiche del segnale analizzato.

Caratteristica	Dato
Frequenza	434.000 Mhz
Segnale minimo	118us
Symbol rate	5300
Modulazione	ASK/OOK
Encoding	Manchester

Si riepiloga in una tabella riassuntiva la suscettibilità ad attacchi.

Attacco	Vulnerabilità	Note
Intercettazione	No	I dati sono parzialmente cifrati
Spoofing	No	I dati sono parzialmente cifrati
Replay attack	No	Un contatore limita la possibilità di attacco
Jamming	Sì	Possibile
Brute force	Sì	Possibile ma su tempi molto lunghi (anni)
OpenSesame	No	Il protocollo non usa registry shifting
RollJam	Sì	
MITM	No	

Nel corso della tesi sono stati programmati appositi script per:

- ricezione dei dati del telecomando
- invio dei dati
- jamming del segnale
- attacco RollJam

5.4 Antifurto di nuova generazione

Come ultimo case study è stato analizzato un sistema di antifurto completamente wireless con le seguenti caratteristiche:

- telecomandi di attivazione/disattivazione
- sensore magnetico wireless
- sensore di movimento wireless

È stato deciso di analizzare questo dispositivo perché ha la caratteristica di essere dotato di un sensore “anti jamming” che dovrebbe identificare questa tipologie di attacchi.



Figura 58. Sistema analizzato

Vengono riportate di seguito le informazioni fornite dal produttore relative all’antijamming: “dispositivo Anti Jamming, anche detto Antiaccecamento, sistema ideato ad hoc per le nostre centrali antifurto che rileva la saturazione del canale radio e quindi genera un allarme manomissione qualora un malintenzionato si avvalga di strumenti chiamati Jammer”. Il dispositivo viene collegato fisicamente alla centralina e dovrebbe essere in grado di rilevare quindi tale attacco.



5.4.1 Telecomando di attivazione/disattivazione

In primis si identifica la frequenza utilizzata dal telecomando mediante l'analizzatore di spettro.

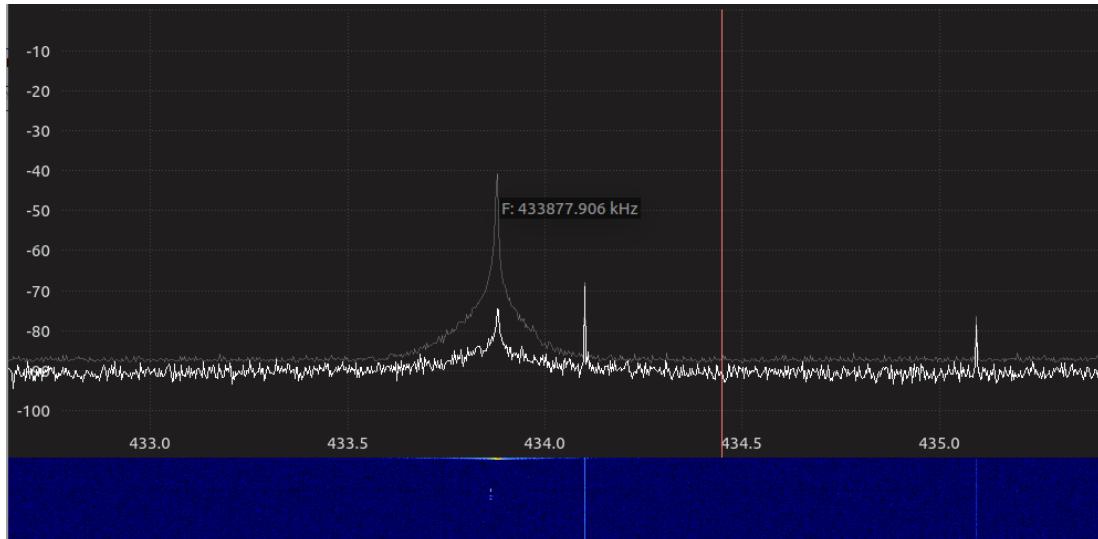


Figura 59. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 433.877 Mhz. Viene registrato il segnale e si utilizza Inspectum per analizzare la struttura del segnale.

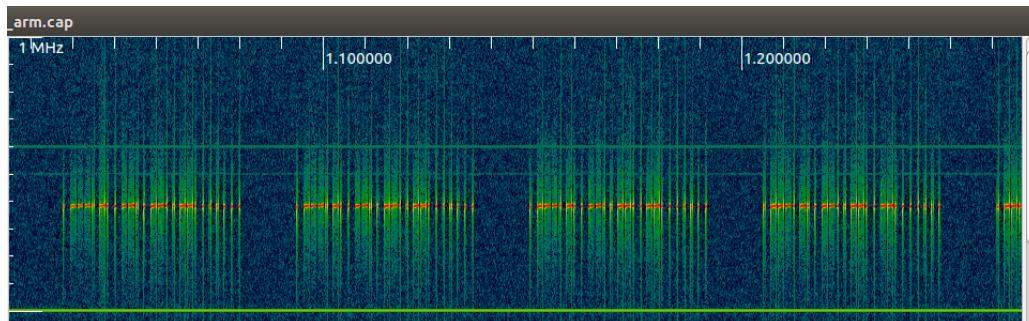


Figura 60. Comunicazione completa

Si nota la presenza di 5 aree di trasmissione; utilizzando i controlli relativi alla potenza del segnale ed allo zoom si analizza ora una singola area.

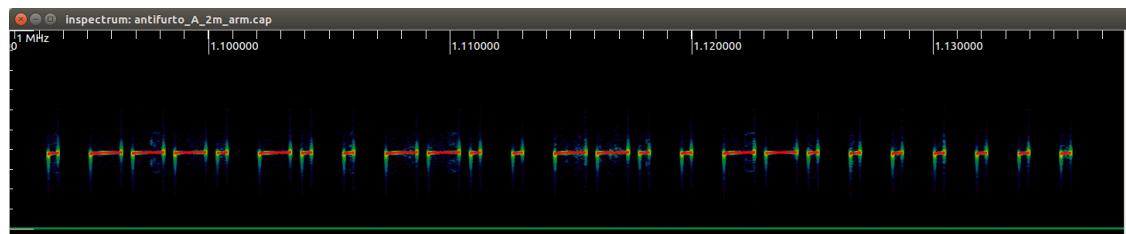


Figura 61. Zoom su singola trasmissione

È possibile vedere come:

- non vi sia una variazione della frequenza, che esclude quindi la variazione della modulazione FSK

- non vi è una differenza sostanziale tra la potenza di trasmissione, ma sembra che sia un codice basato su una modulazione basata sul “trasmetto” o “non trasmetto”. Questa analisi permette di affermare che la modulazione utilizzata è OOK.
- sono presenti 25 segnali differenti, intermezzati da assenza di trasmissione.
- i segnali sono o lunghi o corti e questo vale anche per la parte di assenza di trasmissione.
- non sembrano essere presenti preamboli e/o sync word all'inizio del segnale.
- le 5 aree di trasmissione sopra citate sono la ripetizione sempre dello stesso segnale.

Utilizzando uno zoom ulteriore è possibile utilizzare lo strumento cursors per misurare le caratteristiche di ogni singola trasmissione.

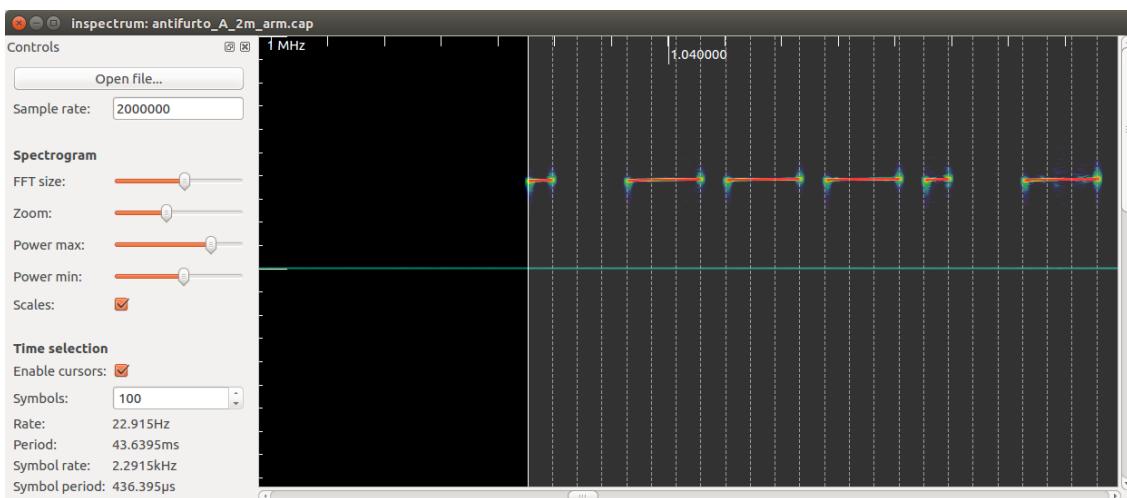


Figura 62. Misurazione dei tempi del segnale

Risulta che ogni singolo simbolo abbia una durata di 437us tra trasmissione e non trasmissione.

$$\text{Numero di trasmissioni massime in } 1s = \frac{1s}{437\mu s} = 2288$$

Visto che la registrazione è stata effettuata a 2M campionamenti/secondo questo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{2247} = 874$$

Utilizzando ora il tool URH impostando i dati rilevati è possibile di tradurre in modo veloce il segnale in bits.



Figura 63. Traduzione del segnale in bits

Il software identifica 5 sequenze di dati identiche, ognuna formata da 100 bits.

Analizzando la sequenza è possibile notare che è possibile suddividere la sequenza a gruppi di 4, rispettando le seguenti regole:

- il primo bit sarà sempre 1
- i due bit centrali possono essere due 1 o due 0
- l'ultimo bit sarà sempre 0

L'unica eccezione è l'ultimo bit, che viene segnato come da solo, in questo caso ci sono 2 ipotesi:

- è un bit di fine trasmissione e quindi non è da tenere in considerazione
- il software commette un errore non interpretando i 3 bit successivi (entrambi a 0)

Il rispetto delle regole sopra citate a gruppi di 4 suggerisce che l'encoding dei dati utilizzati sia una variante di PWM, una sequenza formata da 1000 sarà ritenuto un bit 0 e una sequenza formata da 1110 sarà un bit a 1.

Utilizzando le specifiche funzionalità del software è possibile creare un apposito encoding che sia in grado di decodificare il dato grezzo. Il modo più semplice di effettuare questo encoding è quello rimuovere il primo bit e l'ultimo del gruppo di quattro e poi rimuovere i simboli duplicati.

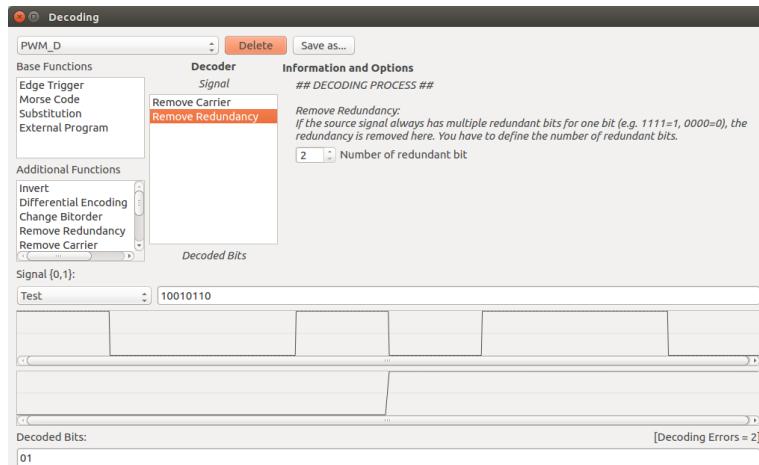


Figura 64. Impostazione dell'encoding

Una volta effettuata questa trasformazione è possibile lavorare sul dato reale, cioè 24 bit (se si ipotizza che l'ultimo sia un bit di fine trasmissione).

5.4.2 Sensore magnetico

È stata identificata la frequenza anche per il sensore magnetico.

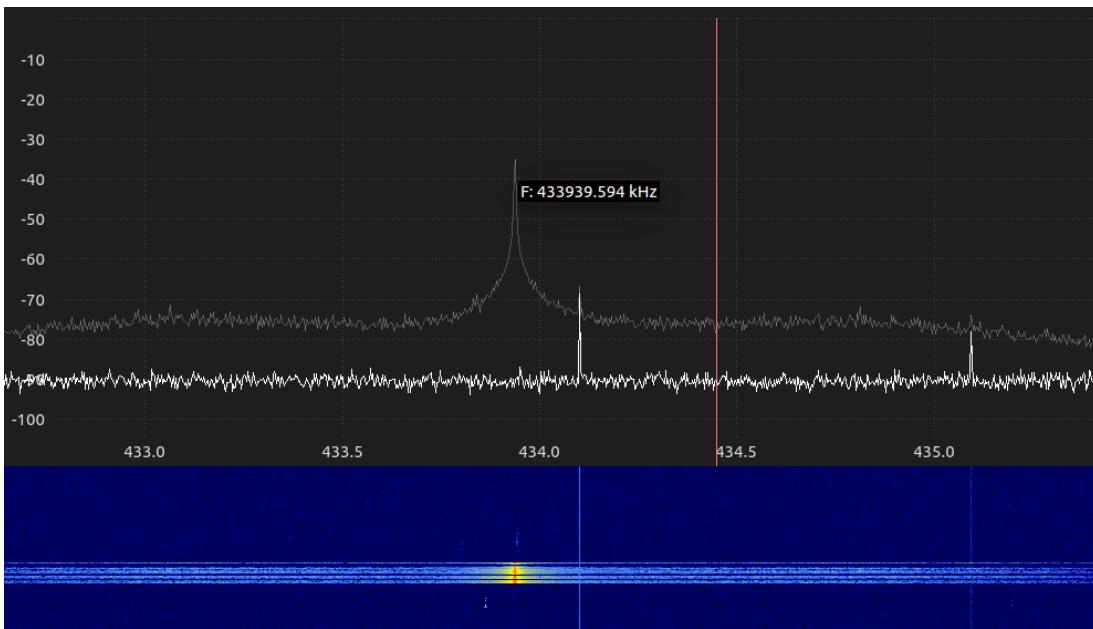


Figura 65. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 433.939 MHz.

Effettuando le stesse operazioni per identificare la trasmissione, risulta che la trasmissione avviene con tempistiche differenti, in questo caso il tempo minimo è 471us

$$\text{Numero di trasmissioni massime in } 1s = \frac{1s}{471\mu s} = 2123$$

Visto che la registrazione è stata effettuata a 2M campionamenti/secondo questo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{2123} = 942$$

La tipologia di trasmissione e di dati contenuti risulta però identica a quella del telecomando.

5.4.3 Sensore di movimento

È stata identificata la frequenza anche per il sensore di movimento.

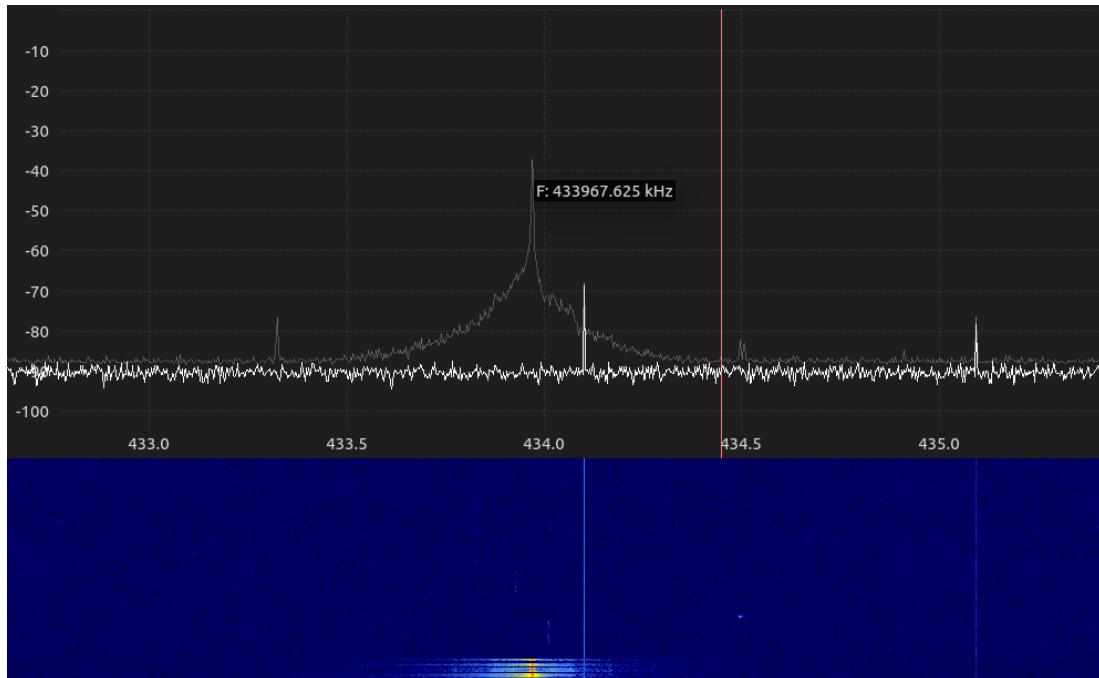


Figura 66. Frequenza identificata mediante l'analizzatore di spettro

La frequenza utilizzata risulta essere di 433.967 Mhz.

Effettuando le stesse operazioni per identificare la trasmissione risulta che la trasmissione avviene con tempistiche differenti, in questo caso il tempo minimo è 514us.

$$\text{Numero di trasmissioni massime in } 1s = \frac{1s}{514\mu s} = 1945$$

Visto che la registrazione è stata effettuata a 2M campionamenti/secondo questo vuol dire che:

$$\text{Numero di samples necessari per una transizione} = \frac{2000000}{1945} = 1028$$

La tipologia di trasmissione e di dati contenuti risulta però identica a quella del telecomando.

5.4.4 Analisi del sistema anti-jamming

Il sistema di allarme viene fornito con un apposito apparato anti jamming che se collegato alla centralina fa scattare un allarme quando un jamming delle frequenze viene rilevato. A livello hardware è possibile configurare in due modi il sensore attraverso uno switch fisico. Nella prima modalità il jamming viene segnalato dopo 4 secondi, nel secondo invece dopo 8 secondi.

Dopo aver configurato il dispositivo sono state effettuate alcune prove di jamming, mediante RFCAT è stata effettuata una trasmissione continua sulle frequenze utilizzate, tuttavia il dispositivo anti jamming non segnalava nulla alla centralina. Durante tale periodo però i telecomandi dell'antifurto non funzionavano e neppure i vari sensori magnetici e di movimento, pertanto il jamming veniva effettuato correttamente.

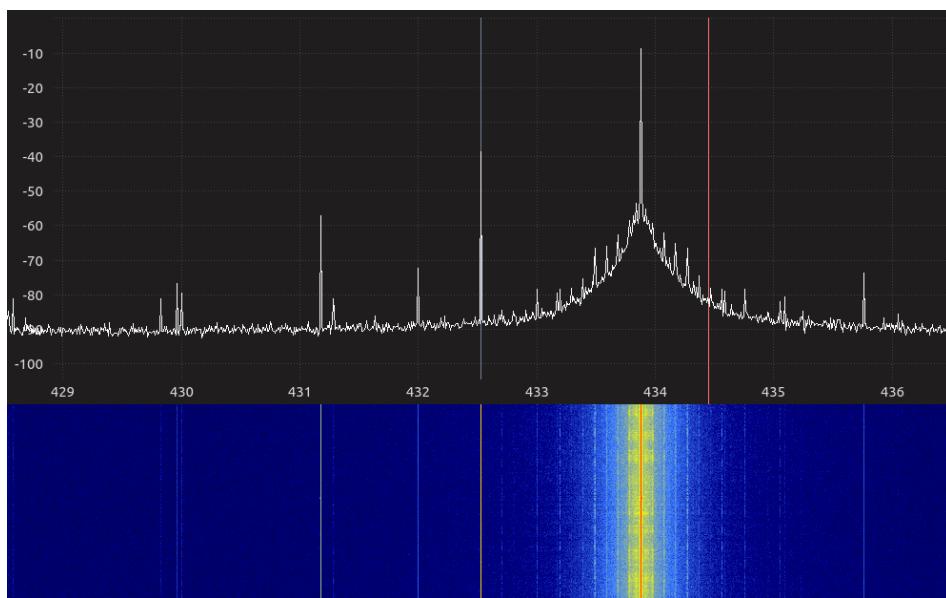


Figura 67. Dumb jamming del segnale

Un'analisi più approfondita ha permesso di identificare la modalità di funzionamento di tale apparato: l'allarme viene generato nel caso in cui vi sia una trasmissione che rispetta il protocollo di comunicazione per il numero di secondi impostato. Risulta pertanto che tale dispositivo non è un reale sistema anti jamming, ma più che altro risulta essere un identificatore di trasmissioni compatibili con la centralina. Anche se non voluto dai produttori, tale dispositivo risulta essere un meccanismo anti brute force; infatti, il brute force effettua un numero di tentativi molto lungo e pertanto farebbe scattare il meccanismo interno.

5.4.5 Analisi del protocollo di comunicazione

Come è possibile vedere, le tre diverse tipologie di dispositivi utilizzano differenti frequenze e timing per la trasmissione; tuttavia, le caratteristiche non variano e i segnali trasmessi sono comparabili.

Dai test effettuati la centralina è in grado di ricevere indistintamente i comandi su tutte queste frequenze e anche su timing differenti. Vengono riportati di seguito un'analisi dei dati che sono trasmessi dai dispositivi dopo il decoding.

Comando	Bit ricevuti
Telecomando A – Arm	011101001100110011000000
Telecomando A – Disarm	011101001100110000001100
Telecomando A – Home arm	01110100110011000000000011
Telecomando A – Alarm	011101001100110000110000
Telecomando B – Arm	011101011101001111000000
Telecomando B – Disarm	011101011101001100001100
Telecomando B – Home arm	01110101110100110000000011
Telecomando B – Alarm	011101011101001100110000
Sensore magnetico	111101000100110101010000
Sensore movimento	0011101110101010101100

È possibile quindi ipotizzare che il segnale ricevuto segue il seguente protocollo:

- **XXXXXYYYYYYYYYYYYZZZZZZZ**

Dove:

- **X**, è la tipologia di dispositivo (telecomando, sensore porta, sensore movimento)
- **Y**, l'id del dispositivo
- **Z**, l'identificativo del comando inviato

Avendo un subset limitato di dispositivi è possibile che il protocollo sia parzialmente differente.

Sono stati inoltre eseguiti una serie di test per verificare i bit che non vengono presi in considerazione nel caso di sensori magnetici o di movimento. Vengono riportati di seguito in rosso quelli che vengono ignorati.

Comando	Bit ricevuti
Sensore magnetico	111101000100110101010000
Sensore movimento	0011101110101010101100

Intercettazione

Il protocollo applicativo non presenta alcuna misura atta alla limitazione dell'intercettazione e l'interpretazione dei dati trasmessi, pertanto tale attacco è di semplice applicazione.

Spoofing

Una volta individuato il protocollo applicativo e capito a cosa servono i differenti bit è facile creare una trasmissione con un preciso scopo.

Replay attack

Il codice trasmesso è fisso, pertanto la ritrasmissione dello stesso dato genererà lo stesso evento. In questo caso è anche possibile ritrasmettere il dato senza capirne il contenuto.

Jamming

Il dispositivo è venduto assieme a un sistema anti jamming; tuttavia, i test effettuati hanno evidenziato come tale sistema non sia in grado di rilevare neppure i jamming più semplici.

Brute force

Nel caso in cui si volesse effettuare un brute force sui bit che si ipotizzano identifichino il dispositivo di sblocco (**YYYYYYYYYYYYYY**) ci si troverà di fronte al caso di 12 bit con 2^{12} possibili combinazioni.

Viene stimato ora un attacco di questo tipo tenendo conto di trasmettere solo una volta la sequenza alla velocità utilizzata dal dispositivo originale (quindi 128 segnali tra i dati e le pause).

$$\begin{aligned} N^{\circ} \text{ massimo di} \frac{\text{trasmissioni}}{\text{s}} &= \frac{1\text{s}}{437\mu\text{s} * 128} = \sim 18 \\ \text{Tempo necessario su } 2^{12} &= \frac{2^{12}}{18} = 227\text{s} = \sim 4 \text{ minuti} \end{aligned}$$

Visto che la ricevente è in grado di poter ricevere da differenti dispositivi comunicazioni che utilizzano velocità diverse tra di loro sono stati effettuati alcuni test per verificare fino a che punto il dispositivo riuscisse a leggere correttamente i dati aumentando la velocità di trasmissione, in modo poi da velocizzare l'attacco. Le analisi hanno permesso di identificare che la trasmissione minima è per essere sempre identificata è di 232us.

$$\begin{aligned} N^{\circ} \text{ massimo di} \frac{\text{trasmissioni}}{\text{s}} &= \frac{1\text{s}}{232\mu\text{s} * 128} = \sim 33,5 \\ \text{Tempo necessario su } 2^{12} &= \frac{2^{12}}{33,5} = 122\text{s} = \sim 2 \text{ minuti} \end{aligned}$$

Su quest'ultima ipotesi sono state effettuate ulteriori prove con il dispositivo anti jamming (o anti brute force) per verificare se è comunque possibile effettuare tale attacco. È stato riscontrato che per non fare scattare l'allarme è necessario attendere 0.8 secondi dopo poco meno di 4 secondi di trasmissione. Effettuando nuovamente i calcoli con il sistema di protezione attivo risultano necessari i tempi seguenti.

$$\text{Tempo necessario su } 2^{12} = \frac{2^{12}}{33,5} + \frac{\frac{2^{12}}{33,5}}{4} * 0,8 = \sim 146s = \sim 2,5 \text{ minuti}$$

Si riepilogano di seguito le caratteristiche del segnale analizzato.

Caratteristica	Dato
Frequenza	433.877Mhz, 433.939Mhz, 433.967Mhz
Segnale minimo	437us, 471us, 514us
Symbol rate	2288, 2123, 1945
Modulazione	ASK/OOK
Encoding	PWM modificato

Si riepiloga in una tabella riassuntiva la suscettibilità ad attacchi.

Attacco	Vulnerabilità	Note
Intercettazione	Sì	La frequenza è fissa e il dato anche
Spoofing	Sì	
Replay attack	Sì	Il dato è sempre uguale
Jamming	Sì	Possibile
Brute force	Sì	Possibile anche nel caso di sensore di protezione
OpenSesame	No	Il protocollo non usa registry shifting
RollJam	N/A	Inutile a causa di attacchi più semplici
MITM	N/A	Inutile a causa di attacchi più semplici

Nel corso della tesi sono stati programmati appositi script per:

- ricezione dei dati del telecomando
- invio dei dati
- jamming del segnale
- brute force anche con sistema anti jamming

6 Discussione

6.1 Risultati

L'analisi di differenti dispositivi che utilizzano onde radio ha portato all'identificazione di numerosi problemi di sicurezza nei protocolli utilizzati. Si ipotizza che spesso le scelte relative alla sicurezza delle comunicazioni siano state prese senza aver prima effettuato un'analisi che possa metterne in luce i rischi reali.

Anche se le onde radio vengono ormai largamente utilizzate, l'analisi ha evidenziato come la sicurezza delle comunicazioni non venga quasi presa in considerazione e anche quei dispositivi che dovrebbero avere un'elevata sicurezza sono comunque suscettibili ad attacchi noti. Inoltre, il sistema anti jamming analizzato che non funziona correttamente, evidenzia come i produttori di tali dispositivi non abbiano adeguate competenze per la corretta progettazione delle contromisure.

Vengono riepilogati di seguito a quali attacchi sono suscettibili i vari dispositivi analizzati.

	Presa 220v	Antifurto X10	Telecomando Fiat	Antifurto nuova generazione
Intercettazione	Sì	Sì	No	Sì
Spoofing	Sì	Sì	No	Sì
Replay attack	Sì	Sì	No	Sì
Jamming	Sì	Sì	Sì	Sì
Brute force	Sì	Sì	Sì	Sì
OpenSesame	N/A	N/A	N/A	N/A
RollJam	N/A	N/A	Sì	N/A
MITM	Sì	Sì	Sì	Sì

6.2 Possibili soluzioni ai problemi riscontrati

Si segnala che la soluzione di alcune delle problematiche affrontate potrebbe comportare un aumento dei costi di produzione dell'hardware, che potrebbe non essere accettabile per i dispositivi a basso costo. Di seguito vengono prese in esame le principali tipologie di attacchi riscontrate e descritte e si ipotizzano possibili soluzioni.

6.2.1 Intercettazione

Per la natura stessa della comunicazione non è possibile risolvere completamente questa problematica; è tuttavia possibile mitigare il rischio che questa comporta mediante l'utilizzo della cifratura del traffico. Si segnala inoltre che esiste anche la possibilità di utilizzare tecniche avanzate di progettazione degli edifici e/o vernici speciali che permettono di limitare la fuoriuscita di onde radio da edifici di una particolare importanza.

6.2.2 Replay attack

È necessario utilizzare un contatore temporale o incrementale nei messaggi che vengono scambiati a livello wireless in modo da poter sempre identificare se il messaggio è frutto di una ritrasmissione da parte di un attaccante.

6.2.3 Jamming

Anche se il dumb jamming è di semplice individuazione, un jamming intelligente diventa pressoché impossibile da rilevare. Inoltre, anche nel caso cui si riesca a individuare un attacco jamming in corso, non è comunque possibile utilizzare il canale di comunicazione. L'unico modo per mitigare questa tipologia di attacchi è utilizzare tecniche di *frequency hopping*, che tuttavia non consentono di risolvere completamente il problema.

Esistono inoltre alcune tecniche pensate per determinare se l'interlocutore con cui avviene una comunicazione cambia e/o viene sostituito: a tale scopo viene utilizzato ad esempio l'effetto doppler [14] che consente di rilevare se la sorgente di comunicazione cambia locazione durante la comunicazione. Tali tecniche sono comunque molto complesse e costose pertanto solo strumenti di alto profilo ne fanno uso.

6.2.4 Brute force

Esistono differenti tecniche per poter bloccare gli attacchi di tipo brute force. In primis è possibile implementare un meccanismo di blocco automatico dopo un numero di tentativi errati. Purtroppo, questa soluzione nel caso di onde radio risulta essere di difficile implementazione: la trasmissione, infatti, potrebbe essere disturbata o altre trasmissioni potrebbero essere considerate tentativi di attacco e pertanto il dispositivo potrebbe creare un disservizio a sé stesso nel caso di blocco di tutti i tentativi di accesso (inclusi quelli legittimi).

Utilizzare un protocollo applicativo che implementi una cifratura con una chiave di lunghezza adeguata unitamente ad una trasmissione di una certa lunghezza può risolvere il problema alla radice; infatti il tempo necessario per effettuare l'attacco diventerebbe impraticabile, soprattutto a causa del fatto che tale attacco non è parallelizzabile.

È inoltre consigliabile l'utilizzo di un protocollo bidirezionale in cui vi sia uno scambio *challenge-response*, in modo da limitare anche i replay attack.

6.2.5 OpenSesame

L'attacco OpenSesame è di semplice soluzione perché si basa sull'utilizzo di *registry shifting* per l'identificazione di sequenze all'interno di una trasmissione continua. Inserendo un preambolo e/o una sync word prima di ogni trasmissione questo attacco viene automaticamente bloccato.

6.2.6 “RollJam” attack

È possibile limitare l’attacco di tipo “RollJam” utilizzando un protocollo bidirezionale che effettua uno scambio challenge-response che permette al dispositivo di identificare se la trasmissione è legittima o meno. È anche possibile utilizzare un sistema di mutua autenticazione basato su certificati e/o chiave condivisa che permetta di irrobustire ulteriormente la comunicazione tra i dispositivi.

6.2.7 Man In The Middle

L’attacco Man In The Middle senza aver necessità di accedere ai dati presenti nella comunicazione è quello di più difficile risoluzione. Infatti, inoltrando la comunicazione attraverso un differente canale al dispositivo, questo risponderà sempre in modo corretto. È possibile utilizzare il frequency hopping con una sequenza basata su algoritmo con un segreto condiviso in modo che all’attaccante risulti difficile seguire la comunicazione. È anche possibile utilizzare una frequenza molto elevata con dei tempi di risposta molto stretti: in questo modo l’attaccante potrebbe non riuscire a comunicare al dispositivo la richiesta e ricevere la risposta nei tempi dettati dal protocollo.

6.3 Conclusioni

Le attività svolte hanno permesso di analizzare differenti dispositivi, sono stati identificate le caratteristiche delle comunicazioni senza fili utilizzate e i protocolli applicativi. Anche se sono stati analizzati dispositivi di differenti tipologie, tra cui alcuni che dovrebbero avere per la loro natura un alto grado di sicurezza, generalmente è stata riscontrata una bassa attenzione sulla sicurezza delle trasmissioni e tutti i dispositivi sono risultati suscettibili ad attacchi noti.

In un prossimo futuro la tecnologia senza fili sarà sempre più utilizzata. Già tuttora molti dispositivi sono in grado di comunicare tra di loro e questo diventerà la normalità tra pochi anni. Proprio per tale ragione è molto importante che la sicurezza di queste comunicazioni sia garantita in modo che questi dispositivi non possano venire attaccati.

L’utilizzo delle Software Defined Radio è risultato un ottimo strumento per poter eseguire analisi di sicurezza su tale comunicazioni, permettendo di riscontrare le problematiche principali sui protocolli applicativi utilizzati.

Il lavoro di tesi lascia spazio a sviluppi futuri nell’ambito dell’analisi di protocolli a livello applicativo. In particolare, l’analisi può essere espansa e migliorata con ulteriori studi focalizzati sulla verifica della robustezza delle comunicazioni cifrate che avvengono tra differenti dispositivi in modo da garantirne anche la robustezza in caso di attacchi più complessi e strutturati.

7 Bibliografia

- [1] «Onda elettromagnetica,» [Online]. Available: https://en.wikipedia.org/wiki/Electromagnetic_radiation#/media/File:Onde_electromagnetique.svg.
- [2] P. Williams, Michael Faraday: A Biography, Simon and Schuster, 1971.
- [3] K. Yee, «Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media,» *IEEE Transactions on antennas and propagation*, vol. 14.3, pp. 302-307, 1966.
- [4] «Spettro elettromagnetico,» [Online]. Available: https://en.wikipedia.org/wiki/Electromagnetic_radiation#/media/File:EM_spectrumrevised.png.
- [5] «What is a Wave?,» [Online]. Available: http://www.vias.org/wirelessnetw/wndw_04_02_01.html.
- [6] «Piano ripartizione frequenze 2015,» [Online]. Available: <http://www.mise.gov.it/index.php/it/comunicazioni/radio/pnrf-piano-nazionale-di-ripartizione-delle-frequenze>.
- [7] «Frequenze ISM,» [Online]. Available: https://en.wikipedia.org/wiki/ISM_band.
- [8] «Code of Federal Regulations, Title 47, Part 15 (47 CFR 15),» [Online]. Available: https://en.wikipedia.org/wiki/Title_47_CFR_Part_15.
- [9] «Developing a New Wireless Hardware Product? – Here's Your Most Important Decision,» [Online]. Available: <http://predictabledesigns.com/most-important-decision-when-creating-wireless-product/>.
- [10] «Frequenze radioamatori,» [Online]. Available: https://it.wikipedia.org/wiki/Bande_radioamatoriali.
- [11] «Home of RF and Wireless Vendors and Resources,» [Online]. Available: <http://www.rfwireless-world.com/Terminology/ASK-vs-FSK-vs-PSK.html>.
- [12] «A Survey on Wireless Security: Technical Challenges, Recent Advances and Future Trends,» [Online]. Available: <http://k-elektronik.org/~byteskrew/paper/A%20Survey%20on%20Wireless%20Security.pdf>.
- [13] «A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks,» [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/0909/0909.0576.pdf>.
- [14] «Deception jamming method based on micro-Doppler effect for vehicle target,» [Online]. Available: <http://ieeexplore.ieee.org/document/7491492/>.
- [15] «Open Sesame Project,» [Online]. Available: <https://samy.pl/opensesame/>.
- [16] S. W. Golomb, SHIFT REGISTER SEQUENCES: Secure and Limited-Access Code Generators, Efficiency Code Generators, Prescribed Property Generators, Mathematical Models., 1982.

- [17] T. Helleseth, «De Bruijn Sequence,» *Encyclopedia of Cryptography and Security*, pp. 315-316, 2011.
- [18] «Bypassing Rolling Code Systems,» [Online]. Available: <http://andrewmohawk.com/2016/02/05/bypassing-rolling-code-systems/>.
- [19] «Aircraft Tracking with Mode S: Modez & Aviation Mapper,» [Online]. Available: <http://spench.net/drupal/research/mode-s>.
- [20] «Receiving NOAA Weather Satellite Images,» [Online]. Available: <https://www rtl-sdr com/rtl-sdr-tutorial-receiving-noaa-weather-satellite-images/>.
- [21] «Software Defined Radio: Basic Principles and Applications,» [Online]. Available: <http://www scielo org co/pdf/rfing/v24n38/v24n38a07 pdf>.
- [22] «Wikipedia - Software Defined Radio,» [Online]. Available: https://it wikipedia org/wiki/Software_defined_radio#/media/File:SDR_et_WF svg.
- [23] «Hackrf 2017.02.1 release notes,» [Online]. Available: <https://github com/mossmann/hackrf/releases/tag/v2017.02.1>.
- [24] «Canale Youtube di Johannes Pohl,» [Online]. Available: <https://www youtube com/channel/UCqIWuCQfX00XHFiwTENI79A>.
- [25] «HCS412 Datasheet,» [Online]. Available: <http://ww1 microchip com/downloads/en/DeviceDoc/41099D pdf>.
- [26] «High speed cipher cracking: the case of Keeloq on CUDA,» [Online]. Available: <http://home deib polimi it/barenghi/lib/exe/fetch.php?media=parma2012 pdf>.

8 Appendice A – Codice Sorgente

Vengono di seguito proposti gli estratti di codice Python più significativi utilizzati durante lo svolgimento del progetto.

8.1 Codice per ricezione telecomando presa elettrica

```
1. import rflib
2. import bitstring
3. import signal
4. import time
5. import sys
6. import binascii
7.
8. def decode_pwm(s):
9.
10.    final = ""
11.
12.    for i in range(0, len(s), 3):
13.        if s[i:i+3] == "100":
14.            final = final + '0'
15.        elif s[i:i+3] == "110":
16.            final = final + '1'
17.        else:
18.            break
19.
20.    return final
21.
22. def encode_pwm(s,padding = 0):
23.
24.    final = ""
25.
26.    for i in range(0, len(s)):
27.        if s[i] == '0':
28.            final = final + "100"
29.        else:
30.            final = final + "110"
31.
32.    if padding != 0:
33.        final = final.ljust(padding, '0')
34.
35.    return final
36.
37.
38. d = rflib.RfCat()
39.
40. d.setMdmModulation(rflib.MOD_ASK_OOK)
41. d.setFreq(433948000)
42. d.setMdmDRate(2238)
43. d.setPktPQT(0)
44. d.makePktFLEN(10)
45. d.setMdmSyncMode(rflib.SYNCH_CARRIER_16_of_16)
46. d.setRFRegister(rflib.AGCCTRL0, 0x91)
47. d.setRFRegister(rflib.AGCCTRL0, 0x91)
48. d.setRFRegister(rflib.AGCCTRL1, 0x00)
49. d.setRFRegister(rflib.AGCCTRL2, 0x07)
50. d.setMdmSyncWord(0b0)
51.
52.
53. d.setMdmDeviatn(3000)
54. d.setMdmChanBW(101562.5)
55.
56. print d.reprRadioConfig()
```

```
57.  
58.  
59. while True:  
60.  
61.     try:  
62.  
63.         y, t = d.RFrecv(timeout=30000)  
64.  
65.         sampleString=y.encode('hex')  
66.  
67.         print bin(int(y.encode('hex'), 16))[2:]  
68.  
69.         decoded_data = decode_pwm(str(bin(int(y.encode('hex'),16))[2:]))  
70.  
71.         print "Received: " + y.encode('hex') + ' (' + bin(int(y.encode('hex'),  
16))[2:] + ') - PWMDATA: ' + str(int(decoded_data,2)) + ' (' + decoded_data +')  
72.  
73.  
74.     except KeyboardInterrupt:  
75.         break  
76.     except rflib.chipcon_usb.ChipconUsbTimeoutException:  
77.         break  
78.  
79. d.setModeIDLE()
```

8.2 Codice per ricezione telecomando X10

```
1. import rflib  
2. import bitstring  
3. import signal  
4.  
5. import time  
6. import sys  
7. import binascii  
8.  
9.  
10. def decode_pwm_s(s, padding=0):  
11.  
12.     final = ""  
13.  
14.     decoded_n = -1  
15.  
16.     for i in s:  
17.  
18.         decoded_n = decoded_n + 1  
19.  
20.         if s.startswith("100000"):  
21.             final = final + '1'  
22.             s = s[6:]  
23.         elif s.startswith("100"):  
24.             final = final + '0'  
25.             s = s[3:]  
26.         else:  
27.             break  
28.  
29.         if padding != 0:  
30.             final = final.ljust(padding, '0')  
31.  
32.     return final, decoded_n  
33.  
34. def encode_pwm_s(s,padding = 0):  
35.  
36.     final = ""
```

```
37.
38.     for i in range(0, len(s)):
39.         if s[i] == '1':
40.             final = final + "100000"
41.         else:
42.             final = final + "100"
43.
44.     if padding != 0:
45.         final = final.ljust(padding, '0')
46.
47.     return final
48.
49.
50.
51. d = rflib.RfCat()
52.
53. d.setMdmModulation(rflib.MOD_ASK_OOK)
54. d.setFreq(433884000)
55. d.setMdmDRate(2574)
56. d.setPktPQT(0)
57.
58. d.setRFRegister(rflib.TEST2, 0x81)
59. d.setRFRegister(rflib.TEST1, 0x35)
60. d.setRFRegister(rflib.AGCCTRL0, 0x91)
61. d.setRFRegister(rflib.AGCCTRL0, 0x91)
62. d.setRFRegister(rflib.AGCCTRL1, 0x00)
63. d.setRFRegister(rflib.AGCCTRL2, 0x07)
64.
65. d.setMdmSyncWord(0b1111111111111111)
66. d.setMdmSyncMode(rflib.SYNCM_16_of_16)
67.
68.
69. while True:
70.
71.     try:
72.
73.         # Riceve un segnale entro 30 secondi
74.         y, t = d.RFrecv(blocksize=30,timeout=30000)
75.
76.         segnale = str(bin(int(y.encode('hex')), 16))[2:]
77.
78.         segnale_ricevuto = segnale
79.
80.         init = "00000100"
81.
82.         if init not in segnale:
83.             continue
84.
85.         segnale = segnale[segnale.find(init)+len(init):]
86.
87.         segnale = "100" + segnale
88.
89.         decoded_data, num = decode_pwm_s(segnale,padding=48)
90.
91.         if num > 3:
92.
93.             print "Segnale ricevuto: " + segnale_ricevuto + " len " + str(len(segnale_ricevuto))
94.             print "Decoded hex: "+ hex(int(decoded_data,2)) + " len (" + str(num) + ") [" + str(bin(int(decoded_data,2))) + "]"
95.
96.
97.         except KeyboardInterrupt:
98.             break
99.         except rflib.chipcon_usb.ChipconUsbTimeoutException:
100.            break
```

```
101.  
102. d.setModeIDLE()
```

8.3 Codice per ricezione telecomando Fiat Stilo

```
1. import rflib  
2. import bitstring  
3. import signal  
4. import time  
5. import sys  
6. import binascii  
7. from termcolor import colored  
8. from operator import xor  
9.  
10. def decode_manchester(s):  
11.     final = ""  
12.  
13.     for i in range(0, len(s), 2):  
14.         if s[i:i+2] == "10":  
15.             final = final + '1'  
16.         elif s[i:i+2] == "01":  
17.             final = final + '0'  
18.         else:  
19.             break  
20.  
21.     return final  
22.  
23.  
24. def keeloc_crc(s):  
25.     crc = [0, 0]  
26.  
27.     for i in range (0,65):  
28.         if s[i] == '1':  
29.             cur = 1  
30.         else:  
31.             cur = 0  
32.  
33.         cur_crc1 = crc[1]  
34.  
35.         crc[1] = crc[0] ^ cur  
36.         crc[0] = (crc[0] ^ cur) ^ cur_crc1  
37.  
38.     return str(crc[0]),str(crc[1])  
39.  
40. d = rflib.RfCat()  
41.  
42. d.setMdmModulation(rflib.MOD_ASK_OOK)  
43. d.setFreq(433946000)  
44. d.setMdmDRate(5300)  
45. d.setPktPQT(0)  
46.  
47. d.setMdmSyncMode(rflib.SYNCH_CARRIER_16_of_16)  
48. d.setMdmSyncWord(0b0101010101010000)  
49.  
50. d.makePktFLEN(100)  
51.  
52. while True:  
53.  
54.     try:  
55.         y, t = d.RFrecv(timeout=30000)  
56.  
57.         sampleString=y.encode('hex')  
58.  
59.         # Split received data
```

```

60.     rec1 = y[:26-8]
61.     rec2 = y[26:26*2-8]
62.     rec3 = y[26*2:26*3-8]
63.     rec3 = y[26*3:26*4-8]
64.
65.     decoded = decode_manchester(str(bin(int(rec1.encode('hex'), 16))[2:]))
66.
67.     print "Received 1: " + rec1.encode('hex') + " " + format(int(decoded,2)
68. , 'x') + ' (' + decoded + ')'
69.     decoded_cleaned = decoded[1:-5]
70.
71.     crc0, crc1 = keeloq_crc(decoded_cleaned)
72.
73.     print colored('Hopping code : ', 'yellow'), colored( decoded[0:31] , 'g
reen')
74.     print colored('Serial number : ', 'yellow'), colored( decoded[32:65] , 'g
reen')
75.     print colored('Low battery : ', 'yellow'), colored( decoded[65] , 'gre
en')
76.
77.     if crc0 == decoded[66] and crc1 == decoded[67]:
78.         print colored('CRC : ', 'yellow'), colored( decoded[66:68]
], 'green'), colored("(OK)", 'green')
79.     else:
80.         print colored('CRC : ', 'yellow'), colored( decoded[66:68]
], 'green'), colored("(KO)", 'red')
81.
82.     print colored('Queue counter : ', 'yellow'), colored( decoded[67:69] , 'g
reen')
83.
84.     except KeyboardInterrupt:
85.         break
86.     except rflib.chipcon_usb.ChipconUsbTimeoutException:
87.         break
88.
89. d.setModeIDLE()

```

8.4 Codice invio codice di sblocco sistema antifurto

```

1. import rflib
2. import bitstring
3. import signal
4. import time
5.
6. d = rflib.RfCat()
7.
8. code = "001111011101010101011100"
9.
10. code = ''.join(['1000' if b == '0' else '1110' for b in code])
11. code = code + '1'
12. code = code.ljust(128, '0')
13.
14. rf_data = bitstring.BitArray(bin=code).tobytes()
15.
16. d.setMdmModulation(rflib.MODASK_0OK)
17. d.setFreq(433882000)
18.
19. d.setMdmDRate(2322)
20. d.setPktPQT(0)
21. d.setMdmSyncMode(0)
22. d.setMaxPower()
23.

```

```

24. d.RFxmit(rf_data, repeat=5)
25.
26. d.setModeIDLE()

```

8.5 Codice simulatore attacco brute force

```

1. import rflib
2. import bitstring
3. import signal
4. import time
5. import datetime
6. from time import sleep
7.
8. d = rflib.RfCat()
9.
10. code = "01110100111110000001100"
11. code = ''.join(['1000' if b == '0' else '1110' for b in code1])
12. code = code1 + '1'
13. code = code1.ljust(128, '0')
14.
15. rf_data = bitstring.BitArray(bin=code).tobytes()
16.
17. d.setMdmModulation(rflib.MOD_ASK_OOK)
18. d.setFreq(433882000)
19. d.setMdmDRate(4300)
20. d.setPktPQT(0)
21. d.setMdmSyncMode(0)
22. d.setMaxPower()
23.
24. rep = 115
25. sleep_time=2
26.
27. for i in range(0,4096,115):
28.
29.     a = datetime.datetime.now()
30.     d.RFxmit(rf_data,repeat=rep)
31.     b = datetime.datetime.now()
32.
33.     print (b - a).microseconds
34.
35.
36.     sleep(sleep_time)
37.
38. d.setModeIDLE()

```

8.6 Codice Jammer

```

1. import rflib
2. import bitstring
3. import signal
4. import time
5.
6. d = rflib.RfCat()
7.
8. rf_data = bitstring.BitArray(bin='11111111111111111111111111111111').tobytes()
9.
10.
11. d.setMdmModulation(rflib.MOD_ASK_OOK)
12. d.setFreq(433882000)
13. d.setMdmDRate(2)
14. d.setPktPQT(0)
15. d.setMdmSyncMode(0)
16.

```

```
17. d.setMaxPower()  
18. d.RFxmit(rf_data, repeat=2000)  
19. d.setModeIDLE()
```