ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

CORSO DI LAUREA MAGISTRALE IN

INGEGNERIA ELETTRONICA

TESI DI LAUREA
in
Lab Of Analog Circuit Design M

MODULAZIONE "SPREAD SPECTRUM" PER COMUNICAZIONI AD ULTRASUONI

CANDIDATO

Andrea Fato

RELATORE

Chiar.mo Prof. Luca De Marchi

CORRELATORE

Dott.ssa Federica Zonzini

Anno Accademico
2019/2020

Sessione *III*

PAROLE CHIAVE

1) **SHM**

2) **Onde di Lamb**

3) **CDMA**

4) **Spread Spectrum**

5) **Microcontrollori**

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

1 INTRODUZIONE

Da sempre, sono tanti i campi dell'ingegneria che si occupano di trovare soluzioni per automatizzare il monitoraggio di strutture quali ponti, edifici, monumenti ecc. Riuscire a individuare un danno nel momento in cui questo si viene a creare non solo permette di intervenire prima che esso degeneri, ma garantisce anche una maggiore sicurezza e un abbassamento dei costi.

Con il termine *Structural Health Monitoring* (SHM) si fa riferimento a un insieme di metodi atti a monitorare lo stato di salute di una struttura in modo automatico. Nella vasta gamma di soluzioni SHM, il presente progetto di tesi utilizza le onde ultrasoniche guidate, anche dette onde di Lamb. Nello specifico, è stato preso in considerazione come tali onde possano essere utilizzate anche a scopo di comunicazione fra sensori posti in locazioni differenti della medesima area da ispezionare. Il vantaggio di tale soluzione consta nel rendere il sistema di monitoraggio completamente autonomo da ulteriori strumentazioni esterne, spesso dipendenti da mezzi inaccessibili o cablaggio ingombrante. Questa analisi permette infatti di raccogliere dati sullo stato di salute dell'oggetto sfruttando i meccanismi di propagazione delle sue onde meccaniche.
Fra le tecniche di modulazione, la codifica ad accesso multiplo di canale (CDMA) risulta di particolare interesse in quanto consente di attuare, nello stesso intervallo di tempo e su tutto lo spettro, messaggi differenti inviati da trasmettitori diversi. Infatti, un problema comune della comunicazione su canali ad accesso multiplo è l'interferenza causata dalla presenza di più utenti che condividono lo stesso mezzo di propagazione. La codifica CDMA consente di rendere minime queste interferenze attraverso l'utilizzo di codici univoci di identificazione. Tale studio, applicato alle moderne tecniche SHM, può risultare molto importante per la corretta interpretazione dei dati rilevati.

In questo testo verranno trattati a livello teorico i principi alla base della comunicazione ad ultrasuoni per mezzo delle onde di Lamb, poi si entrerà più nello specifico nella tecnica CDMA scelta, con conseguente verifica della soluzione studiata per mezzo di simulazioni. Nel capitolo 2 verranno esposte le motivazioni alla base del monitoraggio strutturale, verrà poi discussa l'implementazione per mezzo delle onde di Lamb. Nel capitolo 3 si entrerà nel dettaglio della

comunicazione basata su tecniche CDMA: si parlerà dunque di diverse modulazioni di tipo *Spread Spectrum* (SS) e di algoritmi per la generazione di codici ortogonali di incapsulamento dell'informazione, dove cioè si tende ad azzerare l'interferenza multiutente... Il capitolo 4 sarà dedicato alla valutazione dei vari algoritmi tramite la loro implementazione e simulazione su MATLAB, mostrando la simulazione di un sistema completo di codifica e decodifica del segnale. Nel capitolo 5 verrà poi mostrata l'implementazione su microcontrollore.

2 IL MONITORAGGIO STRUTTURALE

2.1 FINALITÀ

Il monitoraggio strutturale si occupa della analisi e della prevenzione dei guasti che possono presentarsi nelle strutture, dall'ambito civile a quello industriale ed avionico. Si parla di ***Structural Health Monitoring*** (SHM).
Nell'SHM esistono diverse tecniche applicabili che hanno come scopo quello di individuare per tempo i danni che, per diversi motivi, si vengono a creare nelle strutture ingegneristiche. L'obiettivo è quello di rendere automatizzato tale processo di monitoraggio e di fare in modo che i controlli siano costanti anche quando la struttura sotto esame è in piena attività.
Non tutte le tecnologie esistenti, però, soddisfano l'obiettivo desiderato. Pertanto, ci si soffermerà principalmente sulla sola tecnologia basata sulle onde di Lamb, la quale è risultata essere una delle più promettenti per il raggiungimento dello scopo. Le onde di Lamb sono un tipo di onde acustiche scoperte nel 1917 dal matematico inglese Horace Lamb [6]. Negli ultimi decenni l'utilizzo di tali segnali ha trovato un risvolto pratico nel campo del monitoraggio strutturale. Infatti, grazie alla loro capacità di propagarsi per lunghe distanze con ridotta attenuazione e di riuscire ad individuare la presenza di danni di dimensioni millimetriche, oggi sono studiate come metodo di comunicazione non invasivo per il monitoraggio delle strutture ingegneristiche.
La propagazione delle onde di Lamb avviene tramite ultrasuoni. Le onde sfruttano le proprietà meccaniche del mezzo sottostante, pertanto non è necessaria strumentazione aggiuntiva. Dunque, i costi sono ridotti e l'analisi può essere fatta con strumentazione poco ingombrante, permettendo quindi l'applicazione in tempo reale. Una sfida, perciò, consiste nell'usare la tecnologia basata sulle onde di Lamb per rendere il sistema autonomo e intelligente.

L'importanza del monitoraggio strutturale è fondamentale, sono tanti, infatti, gli incidenti avvenuti negli ultimi anni e in quelli passati causa problemi dovuti a malfunzionamenti improvvisi. Si pensi per esempio al crollo del ponte Morandi avvenuto nell'agosto del 2018. Questo incidente provocò decine di morti e un danno economico non indifferente.
La procedura SHM fornisce uno strumento molto efficace per la prevenzione dei guasti e se usata in modo opportuno permette di evitare il verificarsi di gravi incidenti.

Un esempio pratico dell'utilità della ricerca in questo campo proviene dal settore militare. Poiché la nostra flotta di aerei militari invecchia, diventerà sempre più importante fornire dei dati sulla salute di una struttura come il telaio di un aeromobile o di un altro tipo di veicolo. La procedura SHM può fornire un modo per individuare la presenza di danni sulla struttura dell'aeromobile e di valutare il soddisfacimento o meno dei margini di sicurezza.
Nei velivoli più recenti realizzati con materiali compositi, sarà necessario fare monitoraggio strutturale per aiutare a prevedere un guasto catastrofico improvviso del composito, che può accadere quando i difetti del materiale si propagano rapidamente nella struttura.
Oggigiorno esistono soluzioni wireless basate sulle frequenze radio (RF). A differenza di altre tecnologie, questa non necessita di un numero elevato di cavi per la trasmissione dei dati. Una unità di elaborazione centrale si occupa di prelevare e processare i dati provenienti dai diversi sensori. In alternativa l'operazione di elaborazione può essere distribuita sui vari nodi sensori della rete, rendendo il sistema più tollerante nel caso uno o più dispositivi dovessero fallire.
Le soluzioni wireless, però, presentano anche diverse limitazioni.

1. **Sincronizzazione**: errori di sincronizzazione temporale fra i sensori possono causare imprecisioni nelle applicazioni SHM.
   Ogni nodo possiede un orologio interno che, inizialmente, non è sincronizzato con quello degli altri dispositivi della rete. Successivamente deve avvenire un processo per la corretta sincronizzazione, però l'oscillatore interno ai nodi è spesso soggetto a variazioni che causano fenomeni di jitter. Questi impattano negativamente sulla sincronizzazione.

2. **Scalabilità**: Poiché i sistemi SHM necessitano di una grande quantità di dati grezzi per misurare lo stato della struttura, è importante prendere la scalabilità del sistema in considerazione. Se il numero di nodi aumenta, il sistema deve inviare una quantità considerevole di dati via etere.

3. **Affidabilità**: Occorre tenere conto del fatto che le comunicazioni wireless possono essere inaffidabili. I sistemi wireless sono soggetti a perdita di pacchetti di dati. Molto spesso quando due nodi sono molto distanti fra loro il pacchetto potrebbe non raggiungere la destinazione. Inoltre, in ambienti indoor la perdita dei dati è maggiore causa la presenza di ostacoli che limitano la "visibilità" fra i nodi.

Per rispondere a tali limitazioni, all'interno del lavoro qui presente viene proposto l'utilizzo di tecniche basate sulle onde di Lamb come possibile alternativa [1] [13].

2.2 INTRODUZIONE ALLE ONDE DI LAMB E AL MONITORAGGIO STRUTTURALE SHM BASATO SU ULTRASUONI

Le onde di Lamb sono un tipo di onde guidate che vengono propagate nelle strutture laminari o a conchiglia. C'è stato un chiaro aumento di interesse nell'usare le onde di Lamb per il monitoraggio strutturale nel corso degli anni. Infatti, rispetto ad altre tecniche, la valutazione dei danni usando tali onde è in uno stadio di fiorente sviluppo.
Le onde di Lamb possono propagarsi per distanze relativamente grandi, anche in materiali con una alta percentuale di attenuazione, come i compositi polimerici. Tale fatto permette di coprire una larga area con pochi sensori. L'interazione delle onde di Lamb con un danno nella struttura può influenzare le proprietà di propagazione dell'onda causando un effetto di dispersione della stessa e la conversione dei suoi modi. Dunque, molta informazione è codificata nella dispersione delle onde di Lamb causata dal danno. Inoltre, diverse locazioni e/o gravità del danno causano fenomeni di dispersione unici.
Il meccanismo su cui si basa tale comunicazione ad ultrasuoni sfrutta il fatto che i danni presenti in una struttura possono indurre a delle modifiche nelle proprietà locali e globali della stessa. Questi cambiamenti sono inclusi nelle risposte dinamiche dei segnali catturati dalla struttura. L'obiettivo, dunque, consiste nel capire dai cambiamenti nelle caratteristiche del segnale la presenza o meno di un danno nel mezzo. Dunque, basandosi su un segnale di riferimento, relativo a una struttura priva di danni, all'insorgenza di un'eventuale cricca si noteranno delle differenze fra esso e il segnale rilevato.
I passi fondamentali per una corretta identificazione dei difetti usando le onde di Lamb sono i seguenti:

1. attivazione del segnale associato all'onda di Lamb usando un trasmettitore apposito e catturando l'onda dispersa dal danno tramite un sensore. La rete di sensori e attuatori viene costruita in accordo con la configurazione *pitch-catch* o *pulse-echo* (discusse successivamente);

2. estrazione e valutazione delle caratteristiche del segnale catturato con un apposito strumento di *signal processing*;

3. stabilire una relazione fra le caratteristiche del segnale estratto e i parametri del danno (presenza, locazione, gravità…) tramite un apposito modello;

4. estrarre i parametri di interesse basandosi sulla connessione stabilita nel passaggio 3.

Danni di dimensioni millimetriche possono essere identificati con buona accuratezza usando le onde di Lamb in un range di frequenze fra 1 e 10 MHz [3].

Inoltre, poca potenza è richiesta dal trasmettitore per l'identificazione del danno rispetto ad altri metodi.
Riassumendo, i vantaggi dell'utilizzare le onde di Lamb per la comunicazione nei sistemi SHM sono i seguenti:

1. capacità di ispezionare una larga area usando pochi trasduttori;
2. capacità di identificare eventuali danni nella struttura;
3. capacità di classificare vari tipi di danni usando differenti modi d'onda;
4. alta sensibilità ai danni e dunque elevata precisione nella loro identificazione;
5. possibilità di ispezionare strutture rivestite o isolate come tubi sottomarini o sotterranei;
6. possibilità di fare SHM in tempo reale, grazie alle poche risorse richieste dai dispositivi e al loro poco ingombro;
7. basso consumo e alta efficacia dei costi;

Nonostante ciò, siccome sono presenti allo stesso istante modi multipli dell'onda che si sovrappongono fra loro, l'analisi di un segnale di questo tipo non è semplice. Inoltre, le onde di Lamb sono inclini ad essere contaminate da una grande varietà di sorgenti d'interferenza fra cui il rumore ambientale, le variazioni della temperatura, l'interferenza dovuta a utenti multipli della rete ecc. Tutti questi fattori hanno portato lo studio delle onde di Lamb ad essere di interesse multidisciplinare. Ciò ha permesso lo sviluppo di moderni metodi di processamento del segnale, grazie ai quali molte problematiche relative all'alta suscettibilità delle onde alle interferenze esterne possono essere attenuate in modo considerevole. Proprio a tale proposito nel presente elaborato di tesi saranno trattate le tecniche principali per la corretta comunicazione fra i dispostivi della rete in presenza di interferenze.

Quando si analizza un segnale le domande che ci si pone sono le seguenti:

1. È presente un danno?
2. Dove si trova?
3. Quale è la sua dimensione?

Per l'identificazione dei danni basata su onde elastiche si considerano solitamente due possibili configurazioni, la ***pitch-catch*** e la ***pulse-echo***.

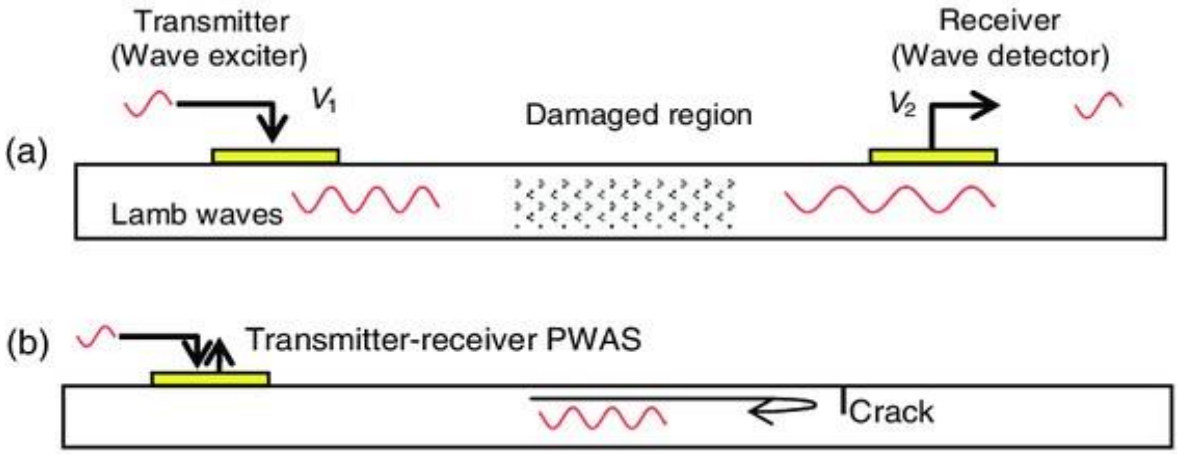

Figura 1 (a) Metodo pitch-catch (b) Metodo pulse-echo [2]

In *Figura 1* è fornita una rappresentazione grafica dell'implementazione dei due metodi di test. Nella configurazione *pitch-catch*, l'onda elastica viene attivata da una sorgente, il trasmettitore. Tale onda viaggia attraverso un oggetto e viene poi catturata da un sensore nella estremità opposta, il ricevitore. Nella configurazione *pulse-echo*, sia il trasmettitore che il ricevitore sono posizionati nello stesso lato e il ricevitore ottiene il segnale che "rimbalza" sull'oggetto da analizzare.
Queste onde possono fornire molte informazioni accumulate lungo il loro percorso, permettendo così di descrivere l'oggetto in questione.
Il set-up iniziale per un sistema SHM prevede per prima cosa la predisposizione di una rete di nodi sensori con una delle due configurazioni descritte sopra. I sensori agiranno dunque come trasmettitori e ricevitori inviando e ricevendo i segnali ad ultrasuoni. Nello scegliere le posizioni dei vari dispositivi occorre tenere conto della geometria della struttura sotto analisi. I sensori devono essere posti in modo che le interferenze fra di loro siano minime, è importante, dunque, fare delle prove per individuare la migliore disposizione dei nodi. Successivamente sarà necessario fare una raccolta di dati per creare una serie di dati di riferimento da utilizzare durante le analisi successive.

In conclusione, l'utilizzo della tecnologia basata sulla comunicazione ad ultrasuoni per mezzo delle onde di Lamb è motivata da quanto detto in precedenza e rispetto ad altre tecniche utilizzate nei processi SHM, quella delle onde di Lamb è sicuramente una delle più promettenti.
Per completezza, nella *Tabella 1* vengono riportate altre tecniche utilizzate per fare monitoraggio strutturale. Di ognuna di esse vengono descritte brevemente le principali caratteristiche; il meccanismo su cui si basano, le possibili applicazioni e gli svantaggi/limitazioni principali.
Nell'ultima riga si prende in considerazione anche il caso delle onde di Lamb.

Approach	Mechanism	Merits and applications	Demerits and limitations
Modal-data-based (eigen-frequency, mode shape and curvature, strain energy, flexibility, sensitivity, damping properties, *etc.*)	Based on the fact that presence of structural damage reduces structural stiffness, shifts eigen-frequencies, and changes frequency response function and mode shapes.	Simple and low cost; particularly effective for detecting large damage in large infrastructure or rotating machinery.	Insensitive to small damage or damage growth; difficult to excite high frequencies; need for a large number of measurement points; hypersensitive to boundary and environmental changes.
Electro-mechanical-impedance-based	Based on the fact that the composition of a system contributes a certain amount to its total electrical-mechanical impedance of the system, and presence of damage modifies the impedance in a high frequency range, normally higher than 30 kHz.	Low cost and simple for implementation; particularly effective for detecting defects in planar structures.	Unable to detect damage distant from sensors; not highly accurate; accurate for large damage only.
Static-parameter-based (displacement, strain, *etc.*)	Based on the observation that presence of damage causes changes in displacement and strain distribution in comparison with benchmark.	Locally sensitive to defects; simple and cost-effective.	Relatively insensitive to undersized damage or the evolution of deterioration.
Acoustic emission	Based on the fact that rapid release of strain energy generates transient waves, whereby presence or growth of damage can be evaluated by capturing damage-emitted acoustic waves.	Able to triangulate damage in different modalities including matrix crack, fibre fracture, delamination, microscopic deformation, welding flaw and corrosion; able to predict damage growth; surface mountable and good coverage.	Prone to contamination by environmental noise; complex signal; for locating damage only; passive method; high damping ratio of the wave, and therefore suitable for small structures only.
Elastic-wave-based (Lamb wave tomography, *etc.*)	Based on the fact that structural damage causes unique wave scattering phenomena and mode conversion, whereby quantitative evaluation of damage can be achieved by scrutinising the wave signals scattered by damage.	Cost-effective, fast and repeatable; able to inspect a large structure in a short time; sensitive to small damage; no need for motion of transducers; low energy consumption; able to detect both surface and internal damage.	Need for sophisticated signal processing due to complex appearance of wave signals, multiple wave modes available simultaneously; difficult to simulate wave propagation in complex structures; strong dependence on prior models or benchmark signals.

Tabella 1 Principali approcci per lo sviluppo di tecniche SHM [3]

2.3 LE ONDE DI LAMB

In un mezzo di propagazione illimitato ci sono due tipi di onde che si propagano, la ***compression wave*** (P-wave) e la ***shear wave*** (S-wave).
La presenza di mezzi finiti (ovvero non infinitamente estesi) influenza la propagazione dell'onda; infatti, quando l'onda incontra una discontinuità dovuta al mezzo, quali i bordi della struttura, si verifica un fenomeno di dispersione e di conversione dei modi. Nel 1885, la soluzione del problema del limite fu interpretata come un'onda di superficie, un terzo tipo di onda, che prende il nome di Rayleigh. Le onde di Rayleigh hanno una ampiezza che decresce rapidamente con la profondità, inoltre tali onde sono non-dispersive, ovvero la loro velocità non dipende dalla frequenza.
Negli anni successivi, un altro tipo di onde furono scoperte, le onde di Lamb. La loro formazione può essere considerata come una conseguenza della riflessione di P- e S-wave sulle superfici delle lastre e delle strutture sferiche. I risultati ottenuti sono una infinità di modi d'onda che si propagano, inoltre almeno due modi esistono ad una fissata frequenza [4].

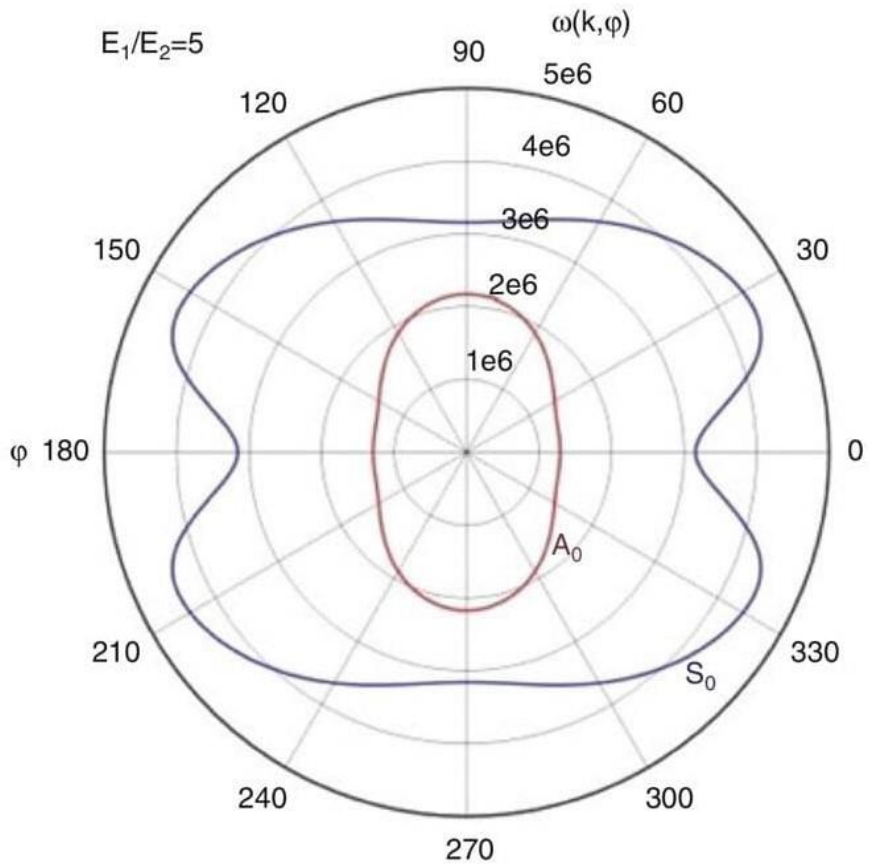

Figura 2 Dipendenza dalla direzione della velocità di fase dei due modi base per un materiale trasversalmente isotropo [4]

Le caratteristiche dispersive fanno sì che la velocità di fase dei modi simmetrici e antisimmetrici dipenda dalla frequenza. Al crescere di quest'ultima nuovi modi si aggiungono a quelli già presenti.

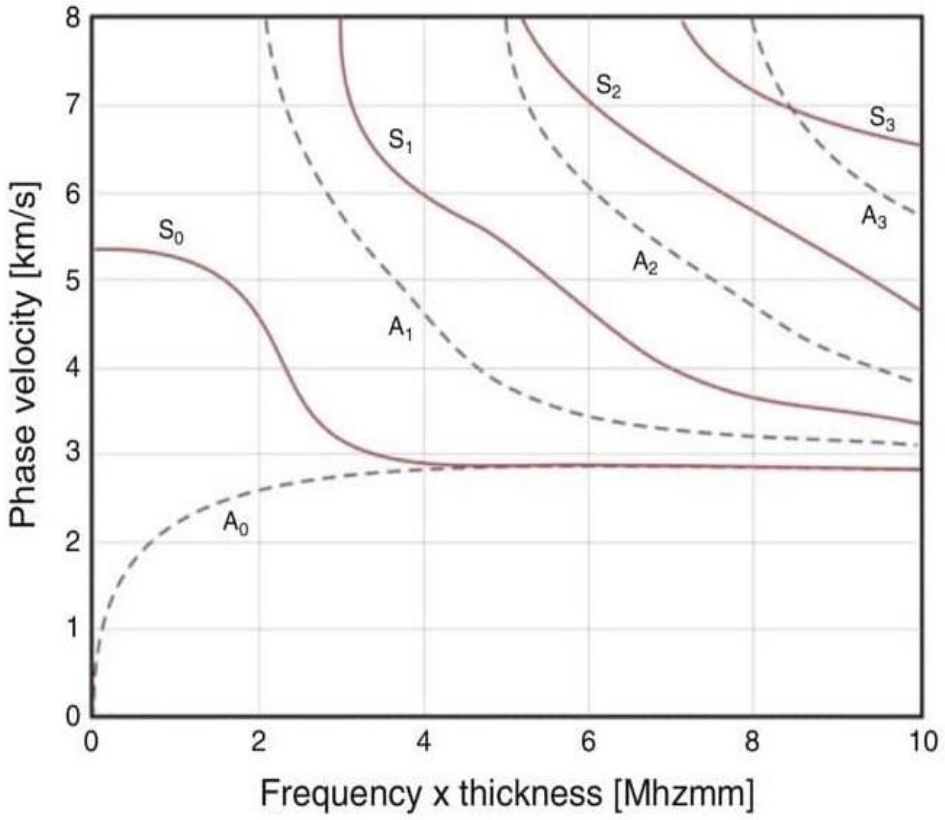

Figura 3 Diagramma di dispersione per un materiale isotropo (alluminio) [4]

Un approccio alla propagazione delle onde guidate, consiste nel cercare soluzioni sinusoidali all'equazione delle onde elastiche lineari, con opportune condizioni al contorno date dalla geometria. Le equazioni di Lamb possono essere derivate considerando una piastra solida avente estensione infinita nelle direzioni x e y e spessore d nella direzione z. Le soluzioni, ottenute mediante la risoluzione di un problema agli autovalori per spostamenti lungo x e z, sono nella forma:

$$\xi = A_x f_x(z) e^{i(\omega t - k\alpha)} \quad (1)$$

$$\zeta = A_z f_z(z) e^{i(\omega t - k\alpha)} \quad (2)$$

Caratterizzate da lunghezza d'onda $2\pi \,/\, k$ e frequenza $\omega \,/\, 2\pi$.
Lo spostamento è una funzione solo di x, z, t; non c'è, dunque, spostamento nella direzione y.
La condizione fisica al contorno per le superfici libere della piastra è che la componente di stress nella direzione z in $z = \pm\frac{d}{2}$ sia zero [6].
Applicando queste due condizioni alle soluzioni sopra formalizzate dell'equazione delle onde, è possibile trovare una coppia di equazioni caratteristiche. Queste sono:

$$\frac{\tanh(\frac{\beta d}{2})}{\tanh(\frac{\alpha d}{2})} = \frac{4\alpha\beta k^2}{(k^2+\beta^2)^2} \quad (3)$$

per i modi simmetrici e divengono, per i modi antisimmetrici,

$$\frac{\tanh(\frac{\beta d}{2})}{\tanh(\frac{\alpha d}{2})} = \frac{(k^2+\beta^2)^2}{4\alpha\beta k^2} \qquad (4)$$

con

$$\alpha^2 = k^2 - \frac{\omega^2}{c_l^2} \qquad \text{e} \qquad \beta^2 = k^2 - \frac{\omega^2}{c_t^2}$$

c_l e c_t sono le velocità dell'onda longitudinale e dell'onda trasversale.
Tramite metodi numerici è inoltre possibile trovare la velocità di fase $c_p = \omega/k$ e la velocità di gruppo $c_g = d\omega/dk$ come funzioni di d/λ o fd, dove λ è la lunghezza d'onda e k il numero d'onda.
Come nelle onde di Rayleigh che si propagano lungo superfici libere, il movimento delle particelle nelle onde di Lamb è ellittico con le sue componenti x e z che dipendono dalla profondità nella piastra.

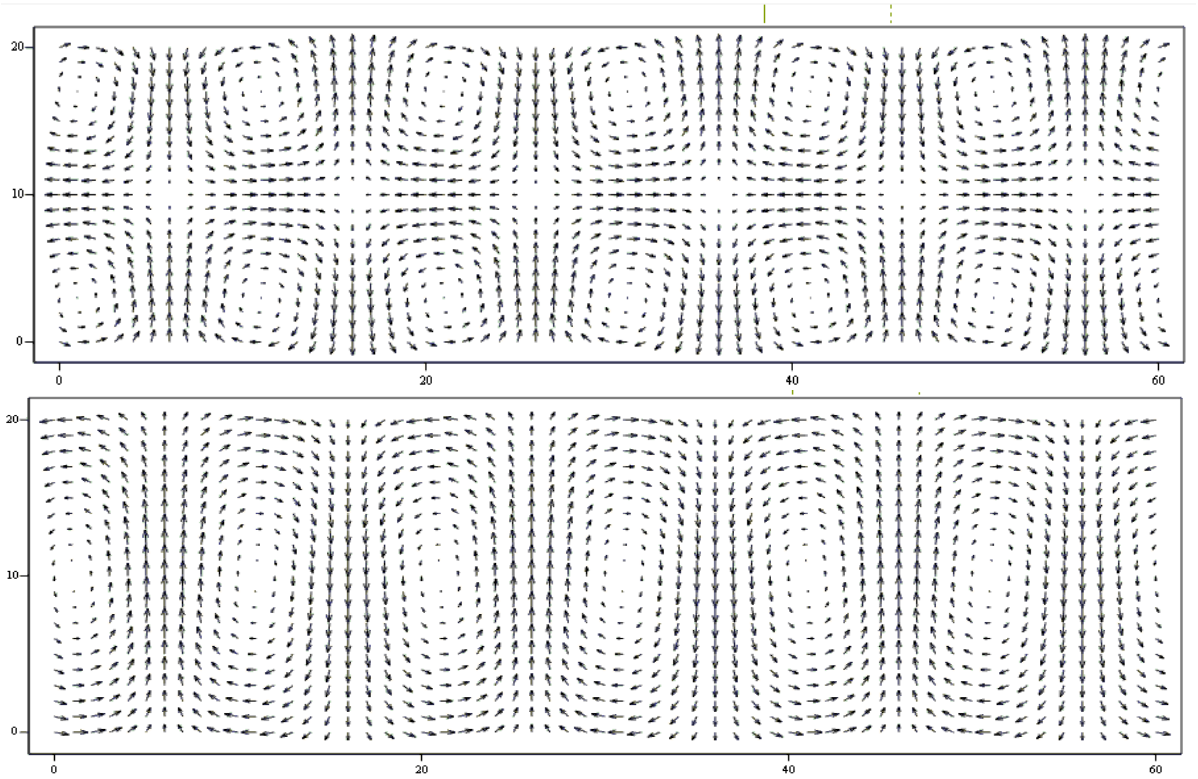

Figura 4 In alto il moto delle particelle nel modo simmetrico, in basso il moto delle particelle nel modo antisimmetrico [5]

Come già accennato, le onde di Lamb mostrano una dispersione della velocità; cioè, la loro velocità di propagazione c dipende dalla frequenza (o lunghezza d'onda), nonché dalle costanti elastiche e dalla densità del materiale. Il parametro chiave per lo studio del comportamento dell'onda nella piastra è il rapporto fra

lo spessore d e la lunghezza d'onda λ. Questo parametro definisce la rigidità della lastra e dunque la velocità dell'onda.

Dal momento che per ogni onda c = f λ, si ottiene un altro parametro chiamato *prodotto frequenza-spessore*:

$$fd = \frac{dc}{\lambda}$$

Le forme d'onda sperimentali osservate nelle piastre possono essere comprese mediante interpretazione con riferimento alle curve di dispersione. Un esempio di tali grafici è stato mostrato in *Figura 3*. Il grafico presenta il prodotto fd nell'asse x e la velocità di fase dell'onda nell'asse y.

Un'altra osservazione va fatta per il modo simmetrico e antisimmetrico di ordine zero, i quali hanno una caratteristica che li distingue da quelli di ordine superiore. Infatti, questi modi hanno "frequenze nascenti" pari a zero. Quindi sono gli unici modi che esistono sull'intero spettro delle frequenze, da zero a frequenze indefinitamente alte.
Il modo simmetrico viene spesso designato S0, mentre il modo antisimmetrico viene spesso designato A0. Questi due modi sono i più importanti perché (a) esistono a tutte le frequenze e (b) nella maggior parte delle situazioni pratiche trasportano più energia rispetto ai modi di ordine superiore.
Nella gamma delle basse frequenze (cioè quando la lunghezza d'onda è maggiore dello spessore della lastra) questi modi sono spesso chiamati ***extensional mode*** e ***flexural mode***, dove il primo fa riferito al modo simmetrico e il secondo al modo antisimmetrico. Questi termini descrivono la natura del movimento e le rigidità elastiche che governano le velocità di propagazione. Per il modo simmetrico, estensionale, il moto ellittico delle particelle è principalmente nel piano della piastra, mentre per il modo antisimmetrico, flessionale, è perpendicolare al piano della piastra. Queste caratteristiche cambiano alle frequenze più alte.
Negli esperimenti che consentono di eccitare e rilevare sia il modo estensionale che quello flessionale, il modo estensionale appare spesso come un precursore a velocità più alta e con minore ampiezza del modo flessionale. Il modo flessionale è il più facile da eccitare dei due e spesso trasporta la maggior parte dell'energia [6].

2.4 APPLICAZIONE DELLE ONDE DI LAMB IN AMBITO SHM

Il rilevamento dei danni con le onde di Lamb si basa sull'osservazione di alterazioni nella forma d'onda del segnale dovute a cambiamenti nel mezzo di propagazione. I danni possono essere considerati come dei disturbi in un corpo elastico altrimenti omogeneo che provocano riflessioni, rifrazioni e conversioni dei modi delle onde.

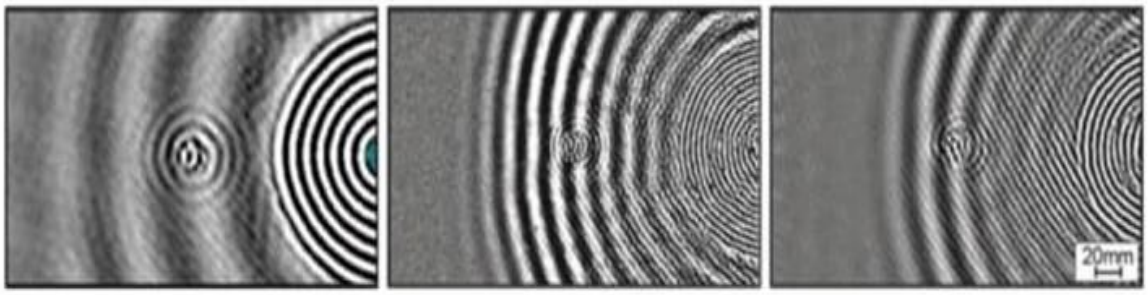

Figura 5 Conversione di modo con riduzione dello spessore [4]

L'immagine sopra, *Figura 5*, mostra un esempio tipico di un campo d'onda eccitato da destra e che si propaga verso sinistra in un materiale di polimero rinforzato con fibra di carbonio (CFRP). Il campo d'onda consiste in un veloce modo S0 e in un più lento modo A0. Al centro delle tre immagini è presente una deformazione che riduce localmente lo spessore della piastra. Il modo S0 arriva per primo in corrispondenza della regione con spessore ridotto e viene parzialmente convertito in un modo A0, mentre il modo antisimmetrico iniziale ancora non è arrivato al danno. Il modo convertito e il suo tempo di volo verso una posizione specifica possono essere misurati da un sensore apposito. Successivamente, una elaborazione del segnale può fornire informazioni riguardanti lo stato della struttura comparando i dati ottenuti con le misurazioni di base. Tramite la procedura descritta e l'utilizzo di tecniche di *signal processing* è possibile, dunque, caratterizzare i danni presenti nella struttura.
Una limitazione di questa tecnica, tuttavia, è che impedisce il rilevamento di difetti intrinseci. Oltre a ciò, diverse indagini hanno osservato che la propagazione dell'onda è spesso alterata solo quando la lunghezza d'onda del segnale è inferiore alle dimensioni del difetto.
Alcuni dei cambiamenti comuni osservati nella forma d'onda del segnale includono; ritardo nel tempo di arrivo o nel tempo di volo (ToF) di un modo, presenza di nuove riflessioni, attenuazione locale o fenomeni di conversione del modo.

Un ruolo fondamentale nell'applicazione delle onde di Lamb lo ha la strumentazione elettronica utilizzata. I dispositivi elettronici devono avere piccole dimensioni e consumare poca potenza. Questi requisiti sono necessari in quanto si vuole riuscire a fare delle analisi in tempo reale. Per tale motivo dispositivi troppo ingombranti non sono adatti.
Nelle applicazioni SHM, i trasduttori piezoelettrici (PZT) vengono spesso utilizzati insieme alle onde di Lamb. Essi possono essere implementati in modo permanente sul materiale da monitorare e sono relativamente discreti (spesso circa 0,5 mm di spessore e pochi millimetri di larghezza). In aggiunta a questo, uno dei principali vantaggi dei sensori PZT è la loro capacità sia di generare che di rilevare le onde di Lamb, consentendone l'uso in varie reti e configurazioni (fra cui la pitch-catch e la pulse-echo).
I materiali piezoelettrici hanno delle proprietà che consentono la conversione fra

energia meccanica ed energia elettrica. Quando il materiale piezoelettrico viene sottoposto ad una tensione alternata può produrre una vibrazione meccanica oscillatoria. Viceversa, un'espansione e una contrazione oscillatoria del materiale producono una tensione alternata ai terminali. Questo fenomeno è chiamato *effetto piezoelettrico*.
Esistono diversi tipi di sensori utilizzati per il monitoraggio strutturale. In *Figura 6* ne vengono mostrati alcuni.

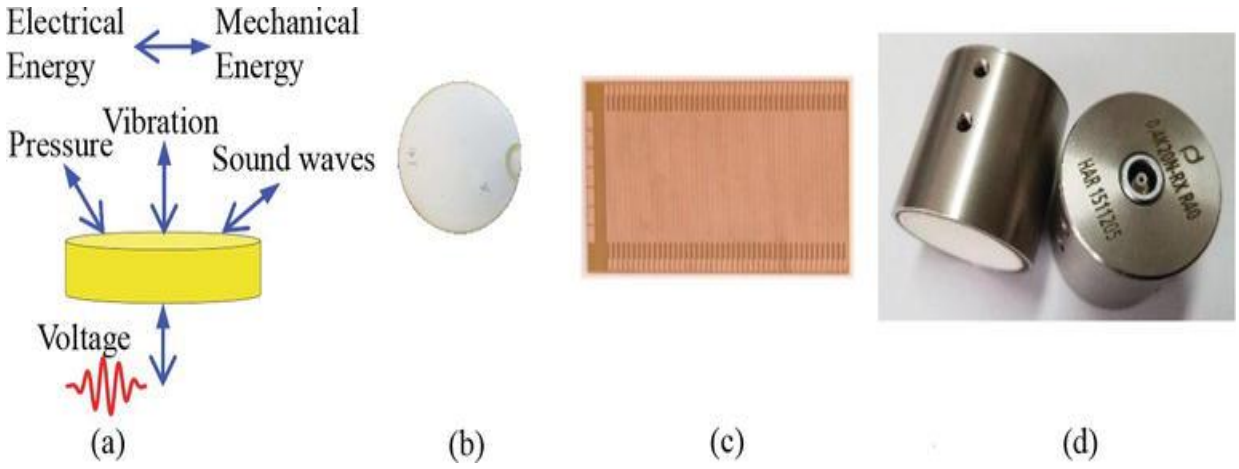

Figura 6 (a) Effetto piezoelettrico; (b) PWAT; (c) Trasduttore interdigitale; (d) Trasduttori accoppiati ad aria [14]

I **Piezoelectric Wafer Active Transducers** (PWAT) hanno una semplice forma rotonda o rettangolare. Tipicamente questi trasduttori hanno gli elettrodi sulla superfice inferiore e superiore. Tramite l'effetto piezoelettrico precedentemente descritto, questi dispositivi applicano e rilevano i segnali delle onde di Lamb tramite il contatto diretto con il piano sottostante.
I **Trasduttori Interdigitali** (IDT) hanno elettrodi a forma di pettine progettati con ceramiche piezoelettriche tradizionali oppure con i più recenti materiali piezoelettrici MFC (Micro-Fiber Composite) e PVDF (Poly Vinylidene Fluoride). Attraverso la regolazione dello spazio tra gli elettrodi interdigitali adiacenti, gli IDT sono in grado di generare onde di Lamb con lunghezza d'onda specifica. Inoltre, questi trasduttori possono essere di varia forma per far fronte anche a superfici curve in modo da riuscire a rilevare al meglio i segnali.
Un altro tipo sono i **trasduttori accoppiati ad aria**. Quest'ultimi sono spesso utilizzati per il rilevamento e la attivazione del segnale ad ultrasuoni senza avere contatti diretti con la superficie, in questo modo si limita al minimo la contaminazione dei dati rilevati.
Inoltre, con lo sviluppo della microelettronica, nuove tecnologie miniaturizzate sono state rese possibili. I trasduttori basati su queste tecnologie hanno diversi vantaggi rispetto ai trasduttori ultrasonici convenzionali, tra cui le minori dimensioni, il basso consumo energetico e la possibilità di creare array mono- e bi-dimensionali [14].

3 COMUNICAZIONE GW IN AMBIENTI MULTI-USER

3.1 INTRODUZIONE

Quando più utenti sono simultaneamente collegati, possono dar luogo ad interferenza a causa del fatto che il mezzo di trasmissione del segnale è condiviso. L'acronimo ***Code Division Multiple Access*** (CDMA) indica un insieme di tecniche che consentono l'accesso multiplo al canale minimizzando il più possibile l'interferenza causata dal principio di sovrapposizione. Per farlo si utilizzano dei codici distinti che identificano in modo univoco ogni trasmettitore che ha accesso al mezzo. Questo consente ai diversi utenti di condividere la stessa banda di frequenze e nello stesso intervallo temporale.
Oltre alla CDMA esistono altri metodi impiegati nella comunicazione multiutente. Questi ulteriori metodi sono il ***Time Division Multiplexing*** (TDM) e il ***Frequency Division Multiplexing*** (FDM). La TDM si basa sulla suddivisione della comunicazione in slot temporali, ovvero ad ogni utente viene assegnato uno specifico slot temporale entro il quale può trasmettere il proprio segnale. L'unione di più slot forma un *frame*. In questo modo più utenti possono trasmettere sullo stesso mezzo senza subire interferenza, per farlo è però necessaria la sincronizzazione fra i dispositivi della rete. Nel caso della FDM la suddivisione avviene in frequenza. Ad ogni utente viene assegnata una banda in frequenze differente nella quale trasmettere il proprio segnale. Anche in questo caso si riesce idealmente ad annullare l'interferenza fra i diversi utenti della rete, ma a differenza del TDM non è necessaria alcuna sincronizzazione.
La CDMA, però, risulta avere diversi vantaggi rispetto alle altre due tecniche, fra cui principalmente il maggior data-rate e l'alta flessibilità. Inoltre, come nel caso della FDM, non è necessaria la sincronizzazione.

I metodi CDMA impiegano la tecnologia ***Spread Spectrum*** (SS). Verranno trattati due tipi di modulazione SS, la *Frequency Hopping Spread Spectrum* (FHSS) e la *Direct Sequence Spread Spectrum* (DSSS), con maggiore enfasi per quest'ultima in quanto utilizzata nel progetto in questione.
La generazione del codice impiegato dalle tecniche SS prevede l'utilizzo di algoritmi. Ognuno di essi permette la generazione di più codici aventi proprietà dipendenti dalla procedura utilizzata per la loro creazione. Dunque, molto importante sarà la valutazione e la scelta dell'algoritmo da impiegare a seconda delle necessità dell'applicazione. I diversi algoritmi saranno dunque simulati e

messi a confronto per poi sceglierne uno per l'implementazione finale su micro-controllore.

3.2 SPREAD SPECTRUM

Nella modulazione SS, il segnale viene modulato usando una sequenza di impulsi detta *spreading code* o *spreading sequence*. Tale sequenza viene generata in maniera pseudo-casuale. L'effetto della modulazione è di incrementare la banda del segnale da trasmettere. Dalla parte del ricevitore, invece, la sequenza è utilizzata per de-modulare il segnale e dunque riportare la sua banda alle dimensioni di partenza.
Lo "spreading" del segnale mediante le tecniche SS ha diversi vantaggi, fra i quali:

- immunità da vari tipi di rumore e dalla distorsione causata dal multipath;
- occultamento della informazione del segnale causata dalla modulazione stessa;
- possibilità per diversi utenti di utilizzare indipendentemente la stessa banda di frequenze con poca interferenza [8];

Le due tecniche SS considerate sono la FHSS e la DSSS.

3.2.1 Frequency Hopping Spread Spectrum

Nella ***Frequency Hopping Spread Spectrum*** (FHSS) la banda di frequenze disponibile viene divisa in N canali a banda stretta. L'asse dei tempi si considera suddiviso in segmenti di lunghezza fissata, dove ogni segmento viene chiamato *hop time*. Ad ogni *hop time* il segnale viene trasmesso su un canale differente. Ognuno di questi canali viene identificato in modo univoco così che si possa definire un percorso, inteso come sequenza di canali, del segnale. Una sequenza pseudo-casuale, pn_t, generata dal modulatore, è usata insieme ad una modulazione FSK per traslare la frequenza della portante in modo apparentemente casuale all'*hoping rate* R_h. Dunque, è la sequenza pn_t a definire la successione di canali per i quali passerà il segnale. Infatti, ad ogni *hop time* il generatore pn fornisce una sequenza di n chip, chiamata *frequency word* (FW), la quale determina una delle 2^n *frequency positions* indicate con f_{hi}. Il trasmettitore e il ricevitore seguono lo stesso pattern definito dalla sequenza pn_t, per tale motivo il codice deve essere noto ad entrambi. Trasmettitori diversi utilizzano un codice differente in modo tale da essere distinguibili al ricevitore.
Considerando una media su molti salti, lo spettro del segnale FHSS complessivo

può arrivare ad occupare una banda molto ampia, si ha, dunque, un effetto di "spreading" del segnale nelle frequenze.

Si possono considerare due diverse implementazioni della tecnica FHSS, la ***slow hopping*** e la ***fast hopping***. Con la *slow hopping* uno o più simboli vengono trasmessi sulla stessa frequenza portante, mentre con la *fast hopping* un simbolo viene trasmesso su più portanti [8]. Di seguito viene mostrata una rappresentazione delle due implementazioni della FHSS.

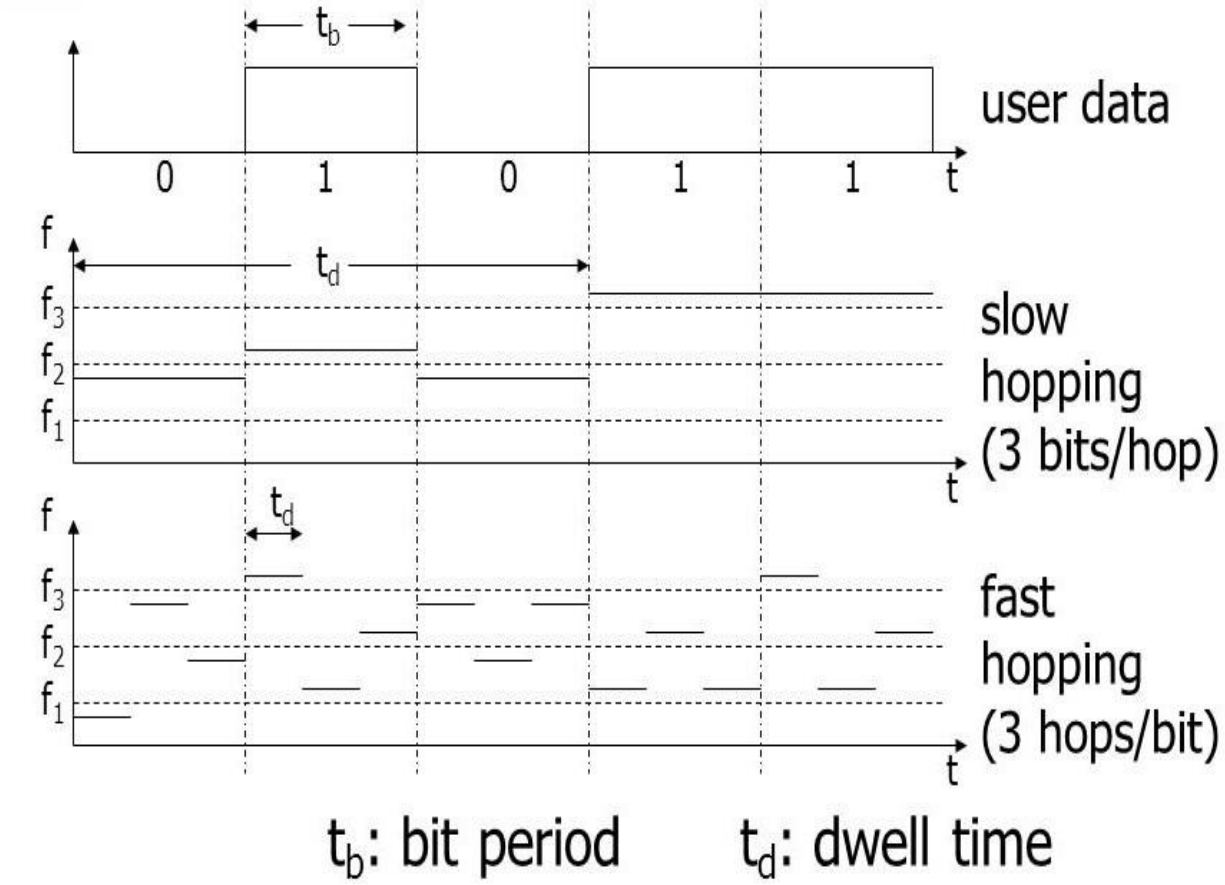

Figura 7 In alto la FHSS con slow hopping, in basso la FHSS con fast hopping [7]

3.2.2 Direct Sequence Spread Spectrum

Un'altra tecnica SS è la ***Direct Sequence Spread Spectrum*** (DSSS). In questo caso ogni bit del segnale originale viene modulato con una sequenza di chip, detta *spreading code*. Ogni bit del segnale è moltiplicato per tale sequenza.
Dal punto di vista spettrale, tale operazione crea un effetto di distribuzione della banda del segnale nelle frequenze. In corrispondenza all'allargamento della banda del segnale, la densità spettrale diminuisce e tale effetto è inversamente proporzionale al numero di chip utilizzati per la moltiplicazione con il singolo bit.

Anche in questo caso per ogni trasmettitore il codice generato è univoco e si ricava attraverso un algoritmo che andrà a definire le proprietà del codice stesso. Il ricevitore userà la stessa sequenza per de-modulare il segnale ricevuto. Per farlo moltiplicherà la sequenza di chip per il segnale modulato DSSS.

Esistono due varianti possibili per la modulazione DSSS, una utilizza lo ***short code*** e l'altra la ***long code***. Nel caso dello *short code,* la sequenza pseudo-casuale è lunga quanto il numero di campioni del singolo bit. Pertanto, ogni bit viene moltiplicato per lo stesso codice. Nel caso della *long code,* invece, la sequenza è più lunga del singolo bit e in questo caso bit diversi potrebbero essere moltiplicati per un diverso codice. In questo lavoro si è scelto di adottare la *short code.*

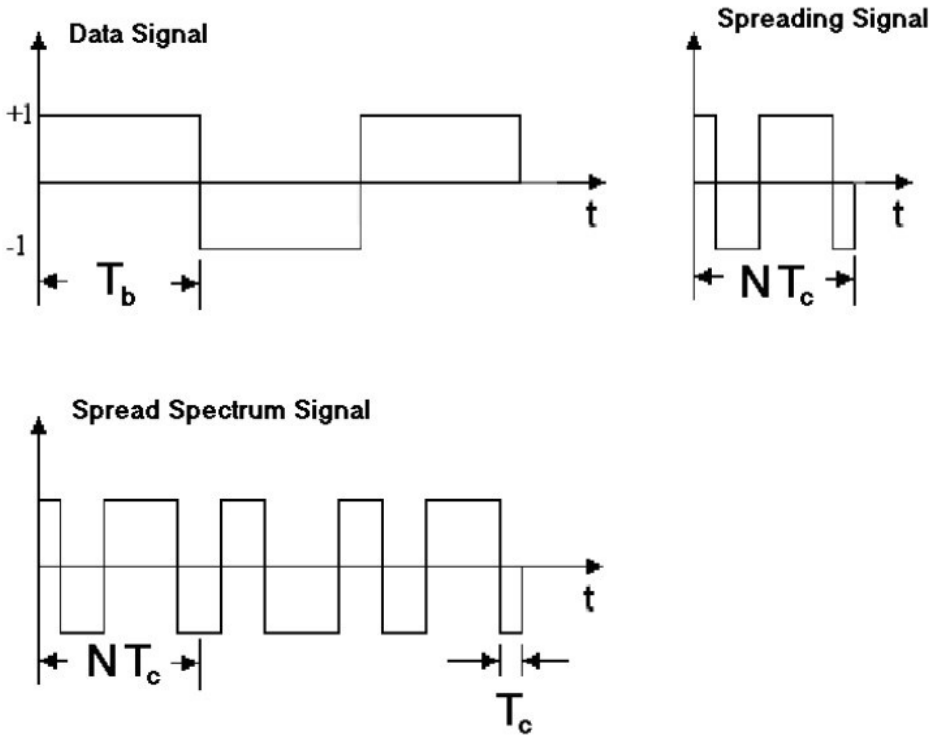

Figura 8 Modulazione DSSS con short code [7]

Considerando la *Figura 8*, dato N il numero di chip, T_c la durata del singolo chip e T_b la durata del bit, si ottiene $T_b = NT_c$.
N e T_c possono essere scelti in diversi modi, l'importante è che il loro prodotto sia sempre pari a T_b. Inoltre, Il numero di chip, N, incide sulla ampiezza dello spettro del segnale trasmesso; dunque, più tale numero sarà alto più lo spettro risulterà piatto. In generale, si consideri un segnale d_t di partenza e una sequenza pn_t. Dato T_s, il tempo di simbolo del segnale, si può scrivere la frequenza di simbolo come $R_s = 1/T_s$. Mentre dato T_c, il tempo di chip, si ha che la frequenza di chip è $R_c = 1/T_c$. Moltiplicando il segnale d per la sequenza pn_t si ottiene il segnale modulato tx_b.

$$tx_b = d_t . pn_t$$

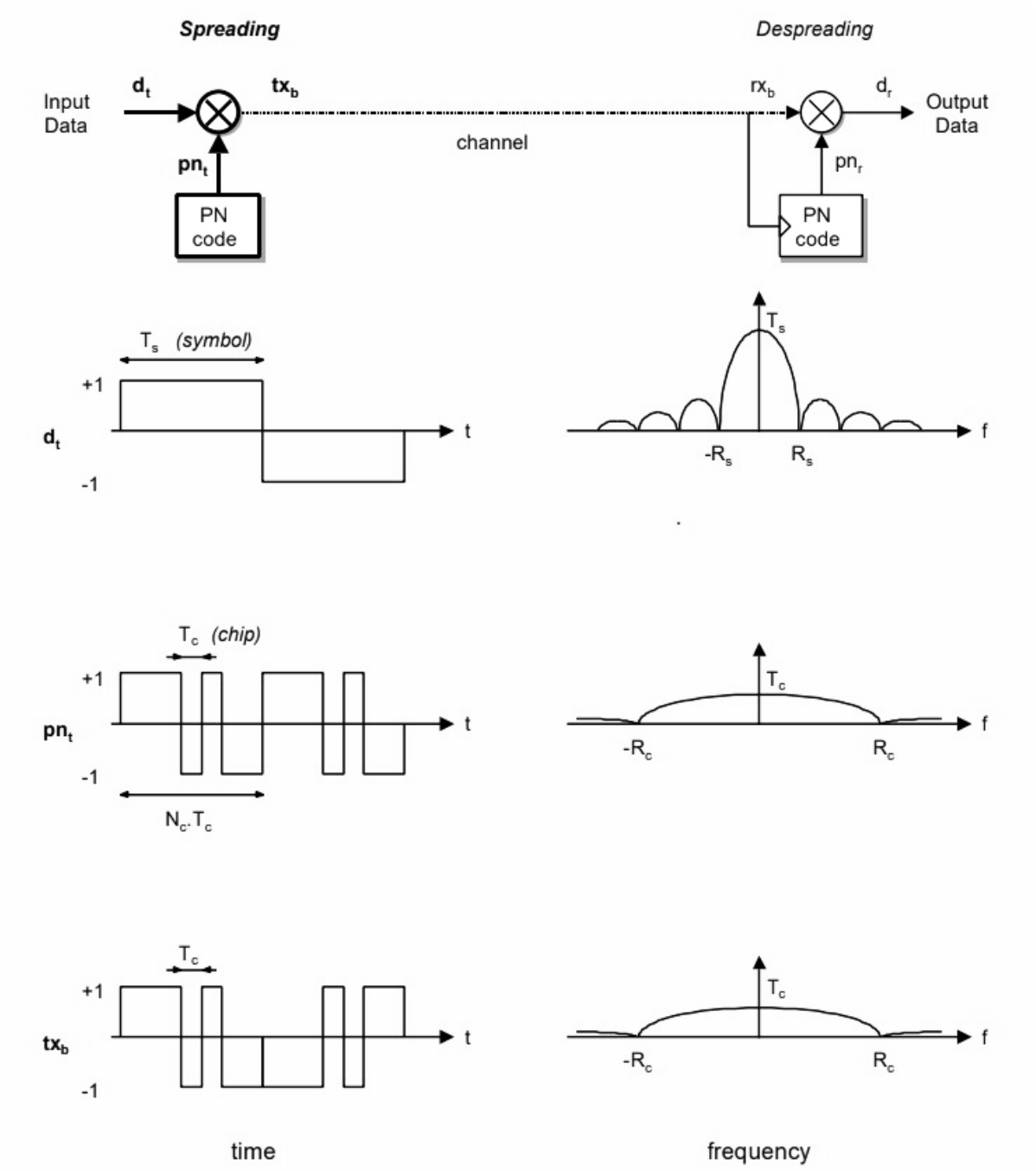

Figura 9 Rappresentazione nel tempo e nella frequenza del segnale modulato DSSS [8]

Il ***processing gain*** è un parametro che indica il rapporto tra la larghezza di banda diffusa (BW_{ss}) (per via dell'effetto di spreading) e la larghezza di banda non diffusa (BW_{info}). In questo caso

$$G_p = \frac{BW_{ss}}{BW_{info}} = \frac{R_c}{R_s} = \frac{T_s}{T_c} = N$$

ovvero il numero di chip, N, definisce il guadagno in larghezza di banda.
Inoltre, l'ampiezza dello spettro del segnale modulato è

$$T_c = \frac{T_s}{N}$$

Dunque, è chiaro che N risulta essere un parametro fondamentale nella modulazione DSSS.

Siccome la modulazione si basa sull'utilizzo di *spreading codes* diversi per ogni trasmettitore, risulta importante che tali codici abbiano buone proprietà di ortogonalità (ovvero che i codici abbiano una funzione di cross-correlazione idealmente nulla) in modo tale da ridurre al minimo le interferenze fra dispositivi che trasmettano simultaneamente nel mezzo.
Essendo la modulazione DSSS più efficiente in termini di costi e di potenza rispetto alla FHSS [16], nei paragrafi e capitoli successivi si farà riferimento alla sola modulazione DSSS, verranno pertanto valutati diversi algoritmi per la generazione degli *spreading codes*, ognuno con diverse caratteristiche.

3.3 PERFORMANCE DELLA MODULAZIONE DSSS IN PRESENZA DI INTERFERENZE

Si considera un segnale trasmesso tx_b ottenuto con una modulazione DSSS che consisteva nel moltiplicare il segnale di partenza d_t per una sequenza pn_t.

$$tx_b = d_t . pn_t$$

Il segnale ricevuto dal lato ricevitore sarà il seguente

$$rx_b = tx_b + i$$

dove i è un termine che indica l'interferenza.
A questo punto, il ricevitore tenta di ricostruire il segnale di partenza sfruttando lo stesso codice utilizzato dal trasmettitore per effettuare la modulazione DSSS

$$d_r = rx_b . pn_t = d_t . pn_t . pn_t + i . pn_t$$

Si considera la sequenza pn_t ottenuta da un generatore pseudo-casuale e con simboli che possono assumere i valori [-1, +1]. Di conseguenza d_r può essere riscritto come

$$d_r = d_t + i . pn_t$$

In tale scrittura si osserva che l'interferenza viene moltiplicata per la sequenza pn_t. Nel caso di una interferenza con banda stretta in frequenza, tale fatto causa un allargamento della sua banda e una conseguente diminuzione della densità

spettrale associata. Il segnale d_t, invece, è quello di partenza prima di applicare la modulazione DSSS, perciò la sua banda sarà più stretta rispetto a quella della interferenza moltiplicata per pn_t.

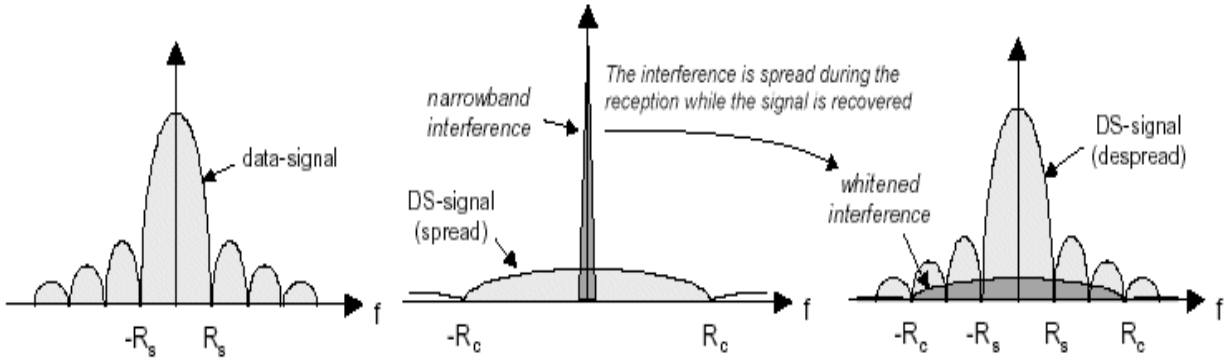

Figura 10 Effetto della modulazione e demodulazione DSSS sugli spettri del segnale e della interferenza [9]

Pertanto, con l'utilizzo di un apposito filtro passa-basso (LPF) è possibile eliminare gran parte della interferenza.
L'effetto di riduzione della interferenza è proporzionale al *processing gain* G_p. Infatti, solo $1/G_p$ della potenza del rumore iniziale è rimasta nella banda R_s del segnale.

Un altro caso è quello in cui l'interferenza è generata da un altro utente che effettua sul suo segnale una modulazione DSSS utilizzando un codice di *spreading* differente. Tali tipi di interferenza sono dette ***Multiple Access Interferences*** (MAI).
In questo caso, per via dello *spreading*, l'interferenza sarà a banda larga nelle frequenze e la sua densità spettrale di potenza sarà bassa.
Si può dunque scrivere

$$d_r = rx_b . pn_{t1} = d_{t1} . pn_{t1} . pn_{t1} + d_{t2} . pn_{t2} . pn_{t1}$$

Da cui risulta

$$d_r = d_{t1} + d_{t2} . pn_{t2} . pn_{t1}$$

Solo il segnale di interesse per il ricevitore viene de-modulato correttamente. Il segnale interferente viene moltiplicato per un codice che non corrisponde a quello utilizzato dal suo trasmettitore, dunque il suo spettro di potenza rimarrà diffuso, mentre quello del segnale di interesse tornerà ad essere quello di partenza.
Nella pratica non si riuscirà mai ad annullare completamente l'interferenza, in quanto i codici utilizzati non saranno mai esattamente ortogonali fra

loro. Ciò crea un limite al massimo numero di utenti che possono trasmettere simultaneamente nello stesso mezzo di propagazione.

Un altro problema tipico dei canali wireless è la presenza di percorsi multipli dal trasmettitore verso il ricevitore che il segnale può percorrere. Tale fatto è spesso dovuto a fenomeni di riflessione. L'effetto di questi percorsi multipli risulta in una fluttuazione del segnale ricevuto dal sensore (fading). Si parla di ***multipath channels***.

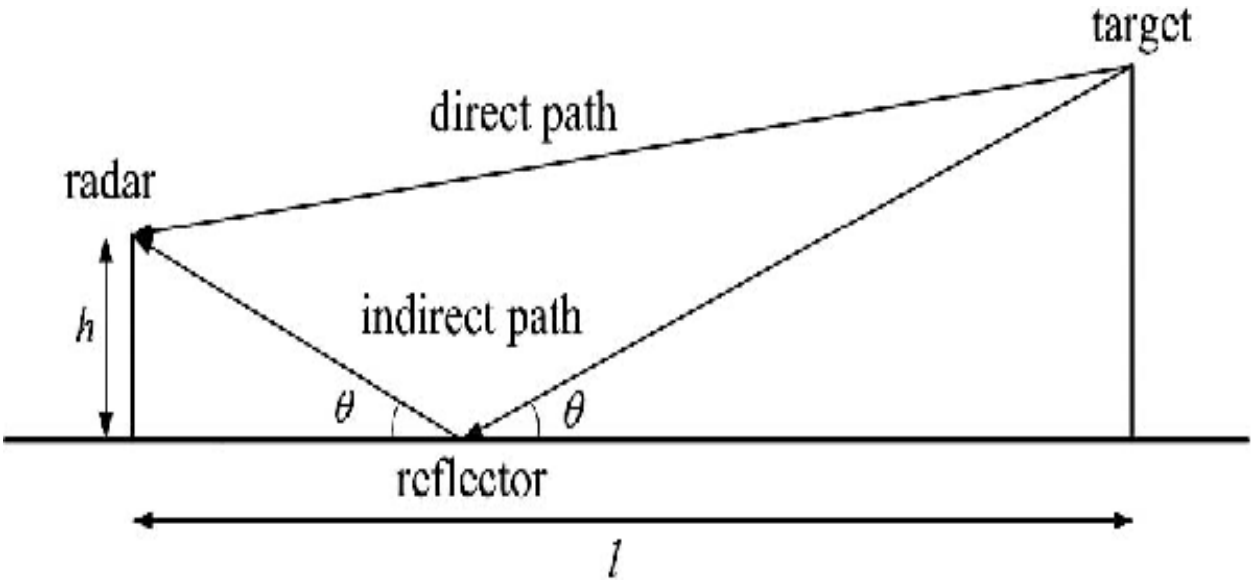

Figura 11 Esempio di multipath in un sistema wireless [10]

Ogni percorso ha la propria attenuazione e il proprio ritardo temporale. La cosa fondamentale è mantenere il percorso diretto e rigettare tutti gli altri. Se si considera il caso di due soli percorsi, uno diretto e uno non-diretto ritardato di un fattore τ rispetto al primo, si possono fare alcune considerazione. Sia rx_d il segnale ricevuto dal percorso diretto e rx_r il segnale ricevuto dal percorso non-diretto, si può scrivere il segnale ottenuto dal sensore come una somma dei due più un contributo di rumore dato dal canale.
Dunque, nel caso di una modulazione DSSS-BPSK risulta

$$rx = rx_d + rx_r + n = Ad_t(t)pn(t)\cos(\omega_0 t) + \alpha Ad_t(t-\tau)pn(t-\tau)\cos(\omega_0 t + \theta) + n(t)$$

dove:

- τ è il ritardo temporale del percorso non-diretto rispetto al percorso diretto;

- θ è un fattore di sfasamento fra la portante nel percorso diretto e quella nel percorso non-diretto;

- α è un fattore di attenuazione del percorso secondario;

Per un ricevitore sincronizzato al segnale del percorso diretto, la de-modulazione viene effettuata moltiplicando il segnale ricevuto per pn(t).
pn(t) ha una funzione di autocorrelazione per cui vale la seguente proprietà

$$pn(t)pn(t) = 1$$

$$pn(t)pn(t-\tau) < 1$$

Essenzialmente, i segnali provenienti da percorsi diversi da quello diretto sono sfasati di uno o più periodi di chip, dunque sono non correlati. L'interferenza dovuta al multipath viene, perciò, attenuata e idealmente resa nulla [8].

3.4 ALGORITMI PER LA GENERAZIONE DEI CODICI DI SPREADING

Nella modulazione DSSS un ruolo fondamentale lo hanno i codici utilizzati per fare lo spreading del segnale. Esistono diversi algoritmi per generare le sequenze, ognuno dei quali ha i suoi vantaggi e svantaggi.
Un primo metodo consiste nel generare una ***sequenza pn*** con simboli appartenenti a [-1, +1]. Tale sequenza può essere generata in modo molto semplice, per esempio sfruttando la funzione *rand()* di MATLAB.

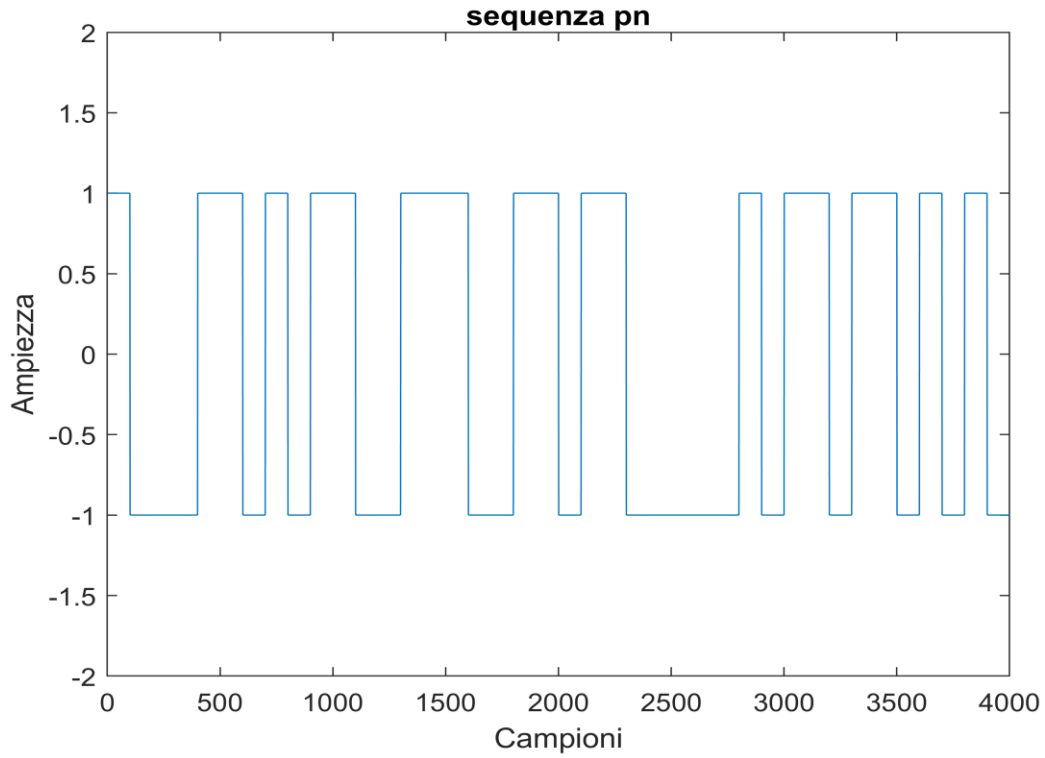

Figura 12 Porzione della sequenza pn generata con MATLAB

L'autocorrelazione della sequenza pn ha proprietà molto simili a quelle del rumore bianco.
Di seguito sono elencate le principali proprietà di una sequenza pn ideale.

1) ***Proprietà di equilibrio***

In ogni periodo della sequenza, il numero di 1 differisce dal numero di -1 al più per un digit (per Nc dispari).

$$\sum pn = +1$$

2) ***Distribuzione run-length***

Una corsa (*run*) è una sequenza di un singolo tipo di simbolo. Fra le corse di 1 e -1 in ogni periodo si ha che circa una metà delle corse di ogni tipo sono di lunghezza 1, circa un quarto di lunghezza 2, un ottavo di lunghezza 3 e così via.

3) ***Autocorrelazione***

La funzione di autocorrelazione, come già accennato, è molto simile a quella del rumore bianco.
La funzione di autocorrelazione per la sequenza pn è definita nel seguente modo

$$R_a(\tau) = \int_{-N_c T_c/2}^{N_c T_c/2} pn(t).pn(t+\tau)dt$$

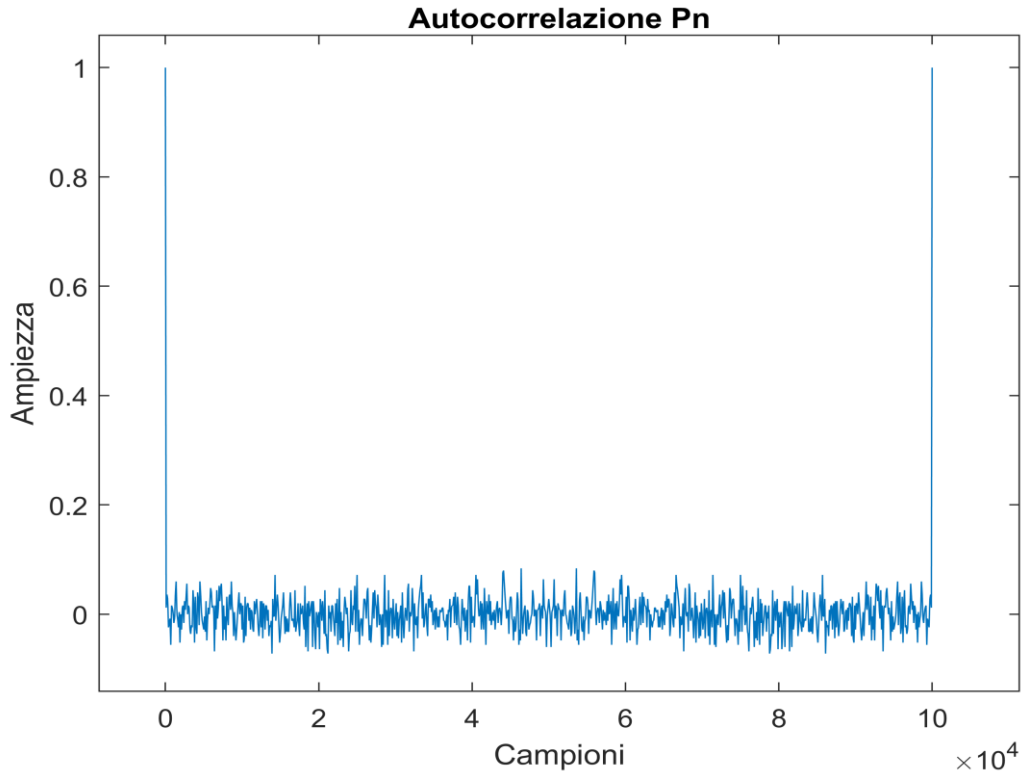

Figura 13 Funzione di autocorrelazione normalizzata di una sequenza pn

Si osserva essere una funzione periodica. I picchi massimi dell'autocorrelazione sono in corrispondenza della esatta sincronizzazione della sequenza pn con sé stessa.

4 Spettro in frequenza

Causa la natura periodica della sequenza pn, lo spettro nelle frequenze ha linee spettrali che diventano sempre più vicine fra loro più si incrementa la lunghezza della sequenza, N_c.

5) Cross-correlazione

La cross-correlazione descrive l'interferenza fra due codici pn differenti.

$$R_c(\tau) = \int_{-N_cT_c/2}^{N_cT_c/2} pn_i(t).pn_j(t+\tau)dt$$

Quando la cross-correlazione $R_c(\tau)$ è nulla per ogni valore di τ, i codici sono detti ortogonali. Quando i codici di diversi utenti sono ortogonali non c'è interferenza dopo il *de-spreading* e la comunicazione di ogni utente è protetta.
Nella pratica non si avranno mai codici esattamente ortogonali fra loro.

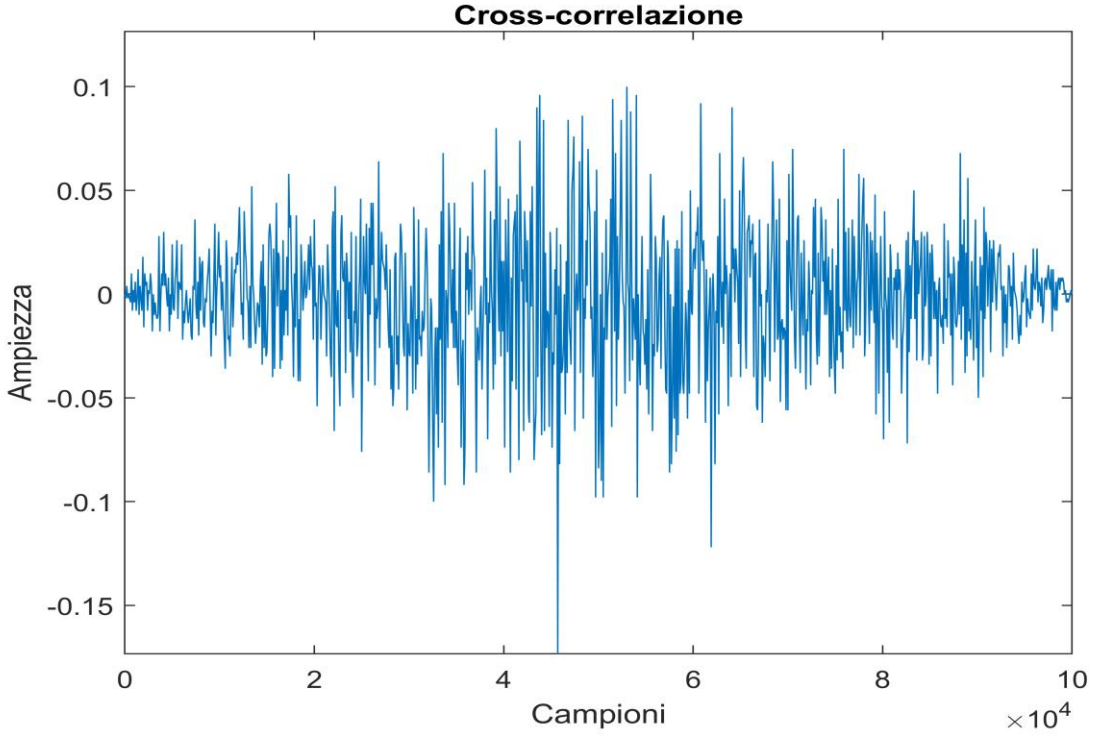

Figura 14 Cross-correlazione normalizzata fra due sequenze pn

Un altro metodo consiste nel generare una ***M-sequence***, ovvero una sequenza pn che soddisfa le citate proprietà e che, inoltre, è di lunghezza massima.
La M-sequence viene generata tramite l'utilizzo di un ***Linear Feedback Shift Register*** (LFSR) costituito da L flip-flop. Un LFSR è un registro a scorrimento il cui bit di ingresso è una funzione lineare del suo stato precedente. Questa funzione lineare spesso è un xor fra i bit salvati in alcuni flip-flop del registro. Il valore iniziale è detto seme e di solito viene scelto in modo tale che almeno un suo elemento sia diverso da zero.

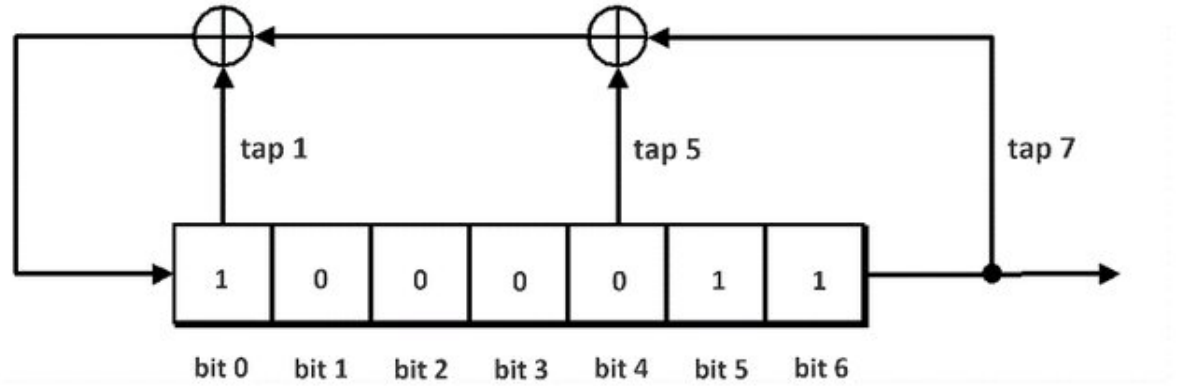

Figura 15 Rappresentazione di un LFSR [11]

Poiché il funzionamento del registro è deterministico, il flusso di valori prodotto da esso è completamente determinato dal suo stato corrente (o precedente). Allo stesso modo, poiché il registro ha un numero finito di stati possibili (cioè combinazioni di valori all'interno del registro), alla fine entrerà in un ciclo periodico. Nonostante ciò, il registro LFSR può avere un ciclo di lunghezza relativamente ampia e i bit possono apparire all'interno del ciclo come se fossero casuali. Dunque, tale registro a scorrimento può generare una sequenza pseudo-casuale pn simile a quella descritta precedentemente. Con la giusta scelta dei tappi da inserire nella retroazione, è possibile rendere tale sequenza pn generata una M-sequence, ovvero una sequenza avente lunghezza pari al numero di stati distinti del registro LFSR.
Considerando l'immagine in *Figura 15*, si può scrivere:

$$\begin{cases} bit0(n+1) = bit0(n)\ xor\ bit4(n)\ xor\ bit6(n) \\ bit1(n+1) = bit0(n) \\ bit2(n+1) = bit1(n) \\ bit3(n+1) = bit2(n) \\ bit4(n+1) = bit3(n) \\ bit5(n+1) = bit4(n) \\ bit6(n+1) = bit5(n) \end{cases}$$

Il bit in posizione sei corrisponde all'uscita del registro a scorrimento. I valori che assume ad ogni stato creano la sequenza casuale. Inoltre, poiché i registri LFSR di questo tipo passano attraverso ogni possibile stato binario (ad eccezione del vettore nullo), la sequenza in uscita si ripeterà periodicamente.
Lo stato "nullo" è uno stato dal quale il registro a scorrimento non sarebbe in grado di uscire, per tale motivo è importante non inizializzare mai il seme a tale valore.
Considerando un LFSR con L flip-flop, le proprietà della M-sequence generata sono le seguenti:

1) <u>Periodo M-sequence</u>

La sequenza ha un periodo di lunghezza pari al numero di possibili stati distinti assumibili dal registro a scorrimento:

$$Periodo = N_{stati} = 2^L - 1$$

2) Proprietà di equilibrio

In un singolo periodo di una M-sequence la somma di tutti gli elementi, con 0/1 mappati in -1/1, fa +1. Ovvero nella sequenza ci saranno 2^{L-1} "uni" e $2^{L-1} - 1$ "zeri".

3) Distribuzione run length

Per ogni periodo della M-sequence, metà delle corse (di tutti gli 1 o gli 0) ha lunghezza 1, un-quarto ha lunghezza 2, un-ottavo ha lunghezza 3, ecc.

4) Autocorrelazione

La funzione di autocorrelazione per la M-sequence è idealmente -1 per ogni valore di ritardo τ, eccetto per valori fra [-1, 1], in cui la correlazione varia linearmente da -1 a $2^L - 1 = N_c$. Dove N_c indica il numero di chip della sequenza. Dunque, il picco di autocorrelazione sale con l'incrementarsi di N_c e approssima sempre più la funzione di autocorrelazione del rumore bianco.

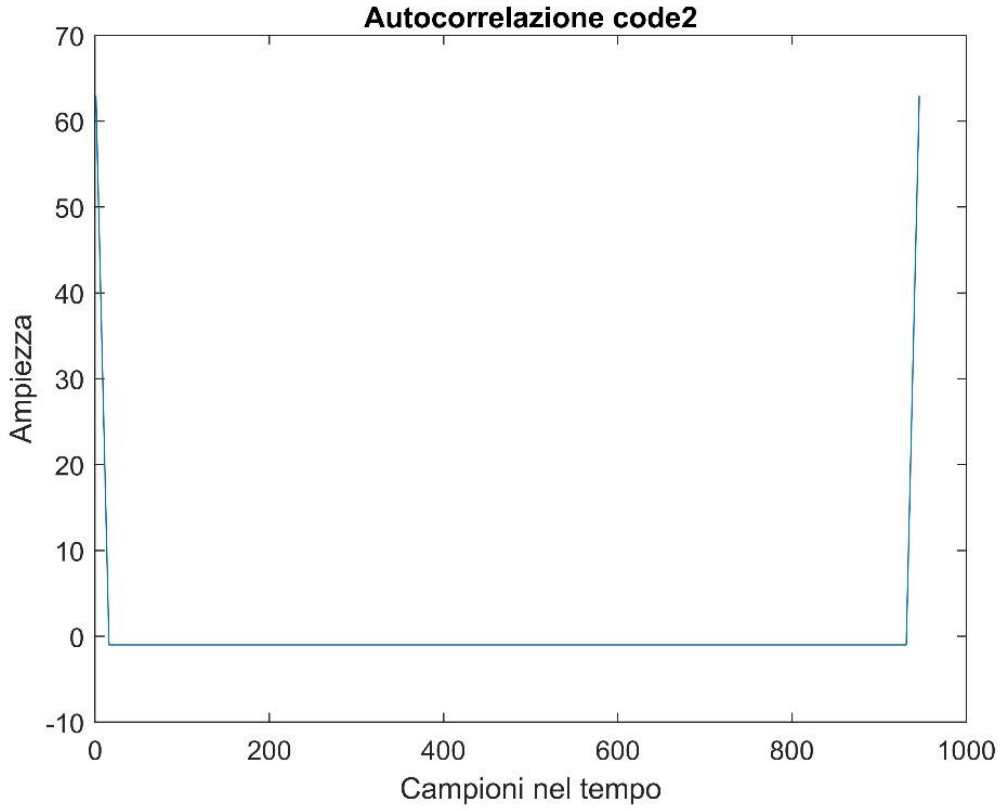

Figura 16 Autocorrelazione di una M-sequence lunga 63 chip generata con un registro LFSR

4) cross-correlazione

La cross-correlazione fra due M-sequence differenti non ha un buon comportamento come l'autocorrelazione. A seconda dell'algoritmo utilizzato per generare la M-sequence, ci possono essere diverse caratteristiche. Nello specifico gli algoritmi di Gold e di Kasami fanno in modo che i picchi della funzione di cross-correlazione siano contenuti entro certi valori limite.

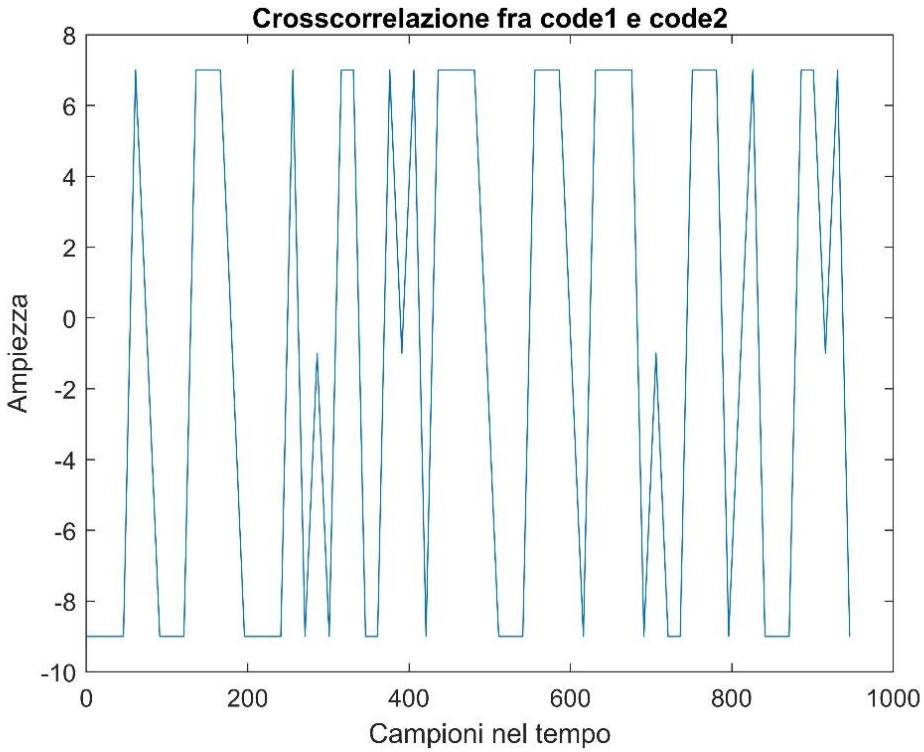

Figura 17 Cross-correlazione fra due M-sequences differenti ottenute con l'algoritmo di Kasami

Per la generazione di M-sequences vengono usati degli appositi algoritmi. Il resto del capitolo presenterà alcuni di questi metodi e ne valuterà i vantaggi e gli svantaggi.

ALGORITMO DI GOLD

Il primo metodo è quello chiamato ***Gold code***, o ***Gold sequence***, dal nome del ricercatore Robert Gold. Ciò che Robert Gold ha scoperto alla fine degli anni '60 è stato un approccio per generare un numero elevato di sequenze pseudo-casuali con bassi valori di correlazione reciproca e di lunghezza uguale.
Uno dei problemi delle M-sequences è la caratteristica della loro cross-correlazione, la quale non presenta buone proprietà come nel caso della autocorrelazione. Invece, con i codici generati utilizzando il metodo di Gold, la cross-correlazione ha picchi controllati. Ciò permette di avere bassa interferenza fra gli utenti scegliendo opportunamente i parametri per la generazione delle sequenze con Gold.
Per la generazione dei codici con l'algoritmo di Gold occorre seguire alcuni passaggi fondamentali. Per prima cosa occorre generare due M-sequence di partenza

con l'utilizzo di due registri LFSR. Scelta la lunghezza L dei registri, per la generazione delle M-sequences occorre scegliere opportunamente i tappi da mettere in retroazione, essi prendono il nome di *preferred pair*.
I due registri produrranno, dunque, due M-sequences. Di quest'ultime sarà fatto l'xor fra gli elementi, ciò produrrà una nuova M-sequence. Successivamente ad ogni passaggio si effettua un ritardo su una delle due M-squences iniziali e si ripete l'operazione di xor sugli elementi. Si procede allo stesso modo per $2^L - 1$ passaggi per un totale di $2^L + 1$ M-sequences generate, considerando le due iniziali.

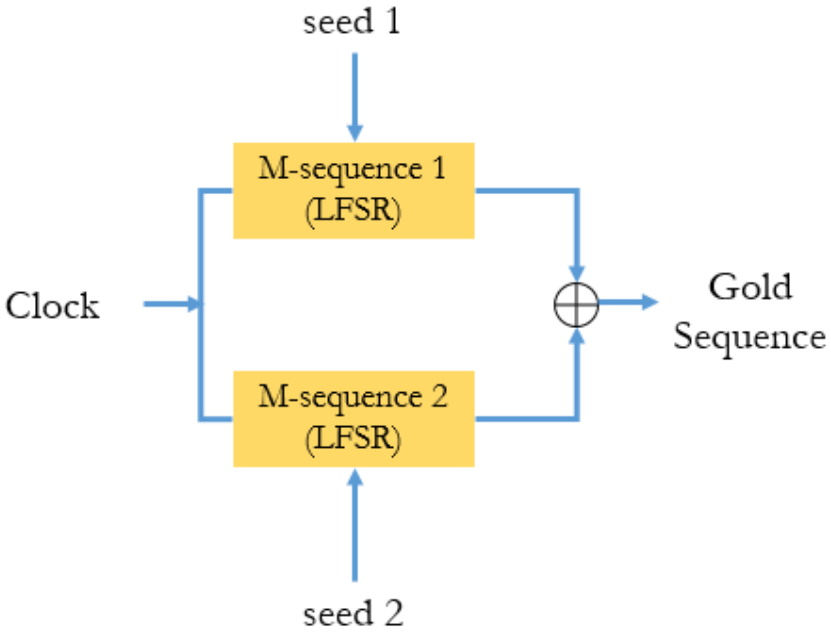

Figura 18 Schema a blocchi per la generazione delle sequenze di Gold [12]

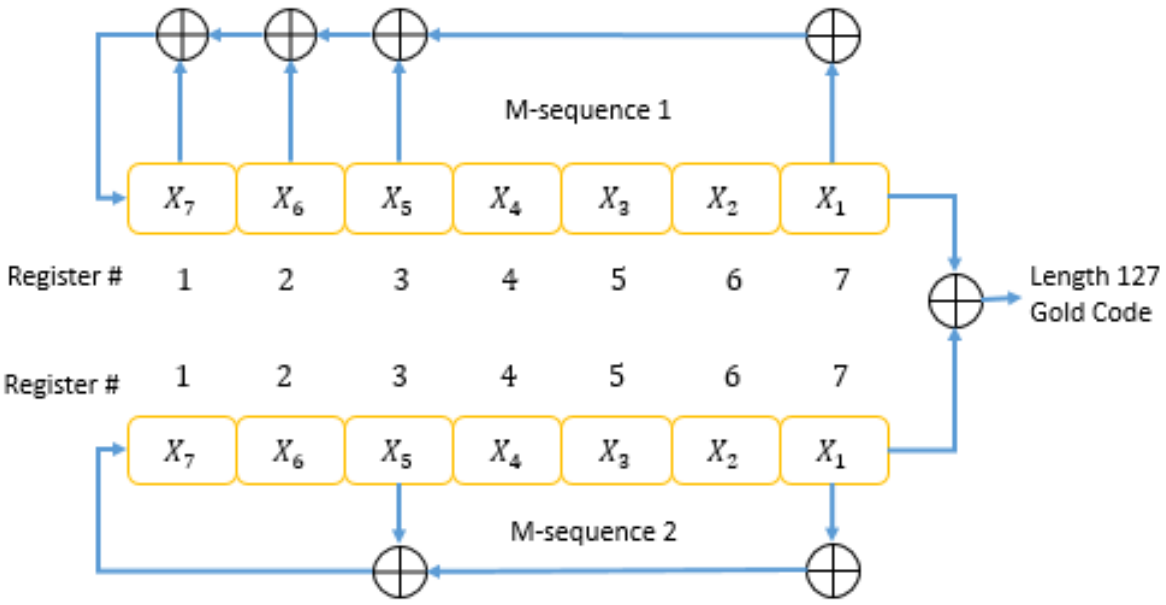

Figura 19 Schema a blocchi per la generazione delle sequenze di Gold con dettaglio dei registri LSFR [12]

Di seguito viene riportata una tabella per la scelta delle *preferred pairs*

L	$N_c=2^L-1$	preferred pairs of m-sequences	3-value crosscorrelations			bound
5	31	[5,3] [5,4,3,2]	7	-1	-9	-29%
6	63	[6,1] [6,5,2,1]	15	-1	-17	-27%
7	127	[7,3] [7,3,2,1] [7,3,2,1] [7,5,4,3,2,1]	15	-1	-17	-13%
8	255	[8,7,6,5,2,1] [8,7,6,1]	31	-1	-17	+12%
9	511	[9,4] [9,6,4,3] [9,6,4,3] [9,8,4,1]	31	-1	-33	-6%
10	1023	[10,9,8,7,6,5,4,3] [10,9,7,6,4,1] [10,8,7,6,5,4,3,1] [10,9,7,6,4,1] [10,8,5,1] [10,7,6,4,2,1]	63	-1	-65	-6%
11	2047	[11,2] [11,8,5,2] [11,8,5,2] [11,10,3,2]	63	-1	-65	-3%

Tabella 2 Preferred pairs delle M-sequences [8]

Dalla *Tabella 2* si possono osservare le proprietà caratteristiche delle M-sequences generate con il metodo di Gold. Scelta la lunghezza dei registri, L in questo caso, il numero di elementi nella sequenza di Gold sarà pari a $N = 2^L - 1$, ovvero la lunghezza del ciclo della M-sequence. Inoltre, la cross-correlazione fra le M-sequences così generate presenterà 3 valori non normalizzati mostrate nelle colonne della *Tabella 2*.

L	N_c	normalized 3-value crosscorrelation	Frequency of occurence
Odd	$2^L - 1$	$-1/N_c$ $-(2^{(L+1)/2} + 1)/N_c$ $(2^{(L+1)/2} - 1)/N_c$	~ 0.50 ~ 0.25 ~ 0.25
Even (not k.4)	$2^L - 1$	$-1/N_c$ $-(2^{(L+2)/2} + 1)/N_c$ $(2^{(L+2)/2} - 1)/N_c$	~ 0.75 ~ 0.125 ~ 0.125

Tabella 3 Espressioni per i tre valori normalizzati della cross-correlazione [8]

Dalla *Tabella 3* si può osservare come il valore di L impatti sulla ampiezza dei tre picchi di cross-correlazione normalizzati. Tale cosa permette di avere a priori una stima delle interferenze che si verranno a creare durante la propagazione sul mezzo condiviso. L permette dunque di definire dei limiti controllati a tali interferenze.
Dalla *Tabella 3* si osserva anche la percentuale con cui questi picchi si presentano nella funzione di cross-correlazione.
Un altro fatto è importante evidenziare. Le M-sequences prodotte dall'algoritmo

di Gold non sono tutte bilanciate. Per cui alcune presenteranno la proprietà 2) e altre no.

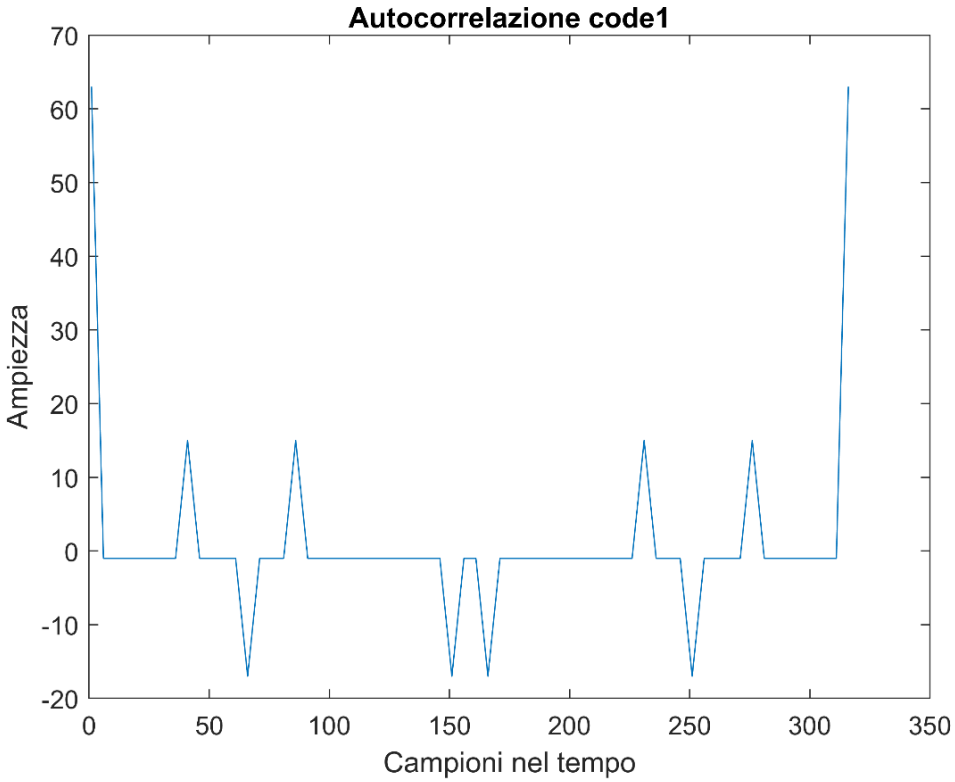

Figura 20 Funzione di autocorrelazione del primo codice generato con Gold

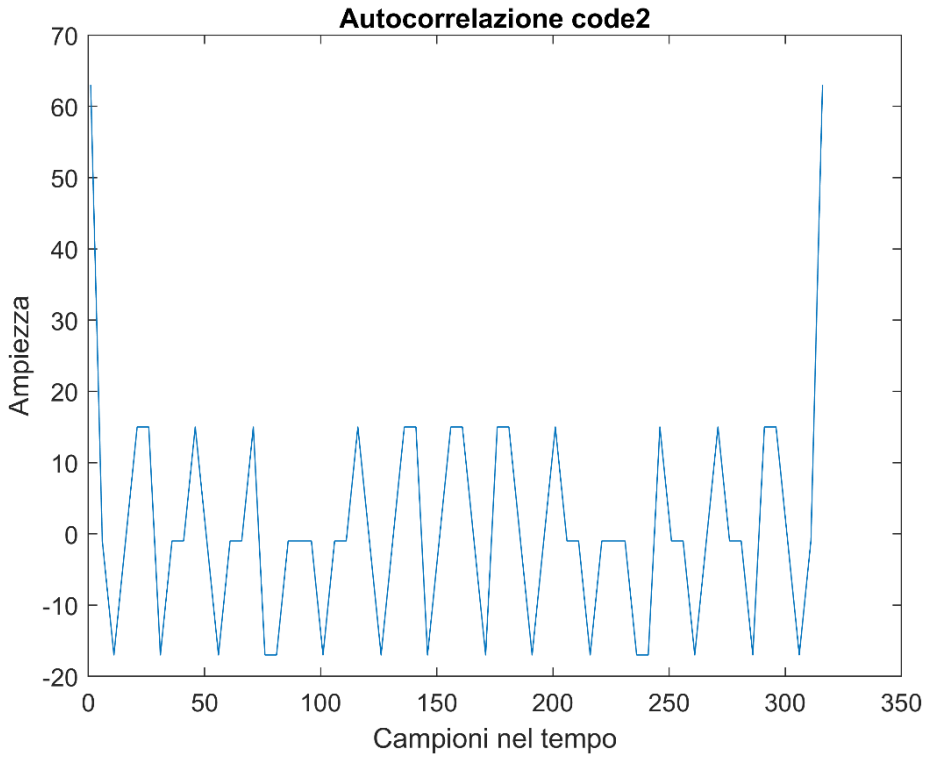

Figura 21 Funzione di autocorrelazione del secondo codice generato con Gold

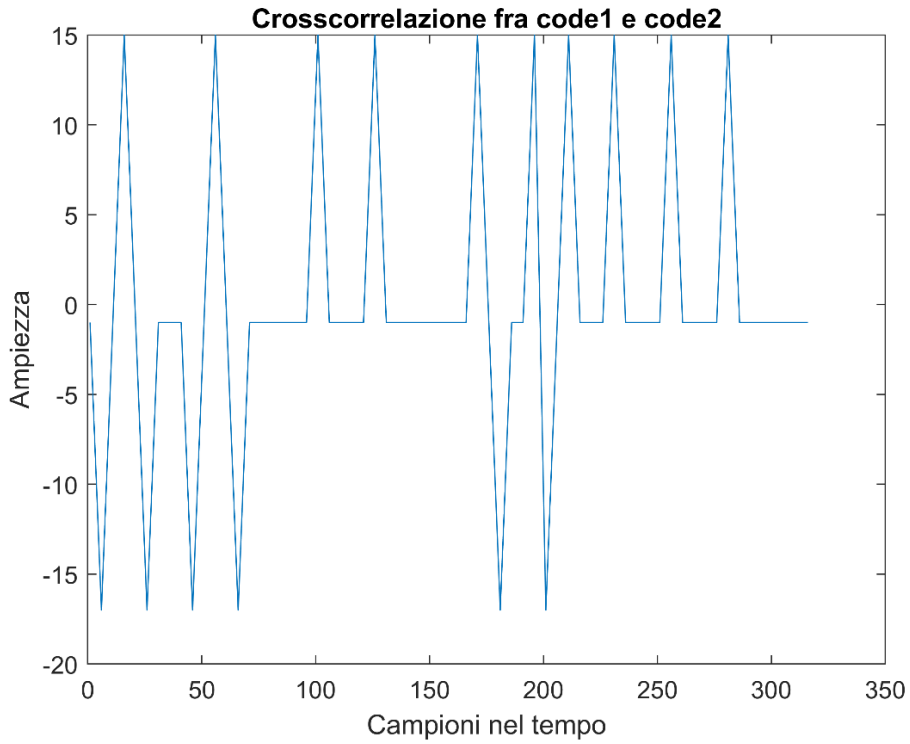

Figura 22 Funzione di cross-correlazione fra il primo e il secondo codice generati con Gold

Nella *Figura 22* si può osservare la presenza dei tre valori non normalizzati della funzione di cross-correlazione. Tali valori corrispondono a quelli presenti nella *Tabella 2* per L pari a 6.
Di seguito vengono riassunti i passaggi principali per la generazione delle M-sequences attraverso l'algoritmo di Gold:

1. scelta di due registri LFSR con gli opportuni tappi in retroazione per la generazione delle due M-sequences di partenza;
2. generazione di una nuova M-sequence ottenuta tramite l'operazione di xor fra gli elementi delle prime due M-sequences generate al passo 1.;
3. shift di un elemento della prima, o seconda, M-sequence e ripetizione dal passo 2.;

I passi 2. e 3. vengono ripetuti per generare ognuna delle restanti $2^L - 1$ M-sequences.

I vantaggi dell'algoritmo di Gold sono:

1. semplice implementazione hardware o software dell'algoritmo;
2. elevato numero di codici prodotti, $2^L + 1$;
3. cross-correlazione fra i codici controllata dal parametro L;

ALGORITMO DI KASAMI

Un altro metodo per la generazione dei codici per la codifica CDMA è l'algoritmo di Kasami. Tale procedura è molto simile a quella vista per la generazione dei codici di Gold, però Il numero di sequenze generate con Kasami è minore per un fissato L. Nonostante ciò, con questo approccio si possono generare sequenze che hanno migliori caratteristiche di autocorrelazione e cross-correlazione rispetto a Gold.
Una famiglia di codici di Kasami è derivata da una M-sequence con grado pari L, dunque ogni codice avrà lunghezza:

$$lm = 2^L - 1$$

Questa M-sequence sarà decimata con un *fattore di decimazione* pari a:

$$d = 2^{L/2} + 1$$

Tale cosa genera una sequenza chiamata *support-sequence* (s-sequence) con una lunghezza pari a:

$$ls = 2^{L/2} - 1$$

Le ulteriori M-sequence, in aggiunta alla prima, saranno generate attraverso l'operazione di xor della prima M-sequence con tutte le versioni ciclicamente traslate della s-sequence.
Il numero complessivo di M-sequences così generate è:

$$M = 2^{L/2}$$

Tale famiglia di M codici viene definita *small set*.

Un altro algoritmo, qui non descritto, permette di ottenere un numero maggiore di codici. Per farlo viene combinata la procedura di Gold con quella di Kasami per ottenere il cosiddetto *large set* di Kasami.

Come nel caso della procedura utilizzata con Gold, l'algoritmo di Kasami è facilmente riproducibile sia via software che via hardware.

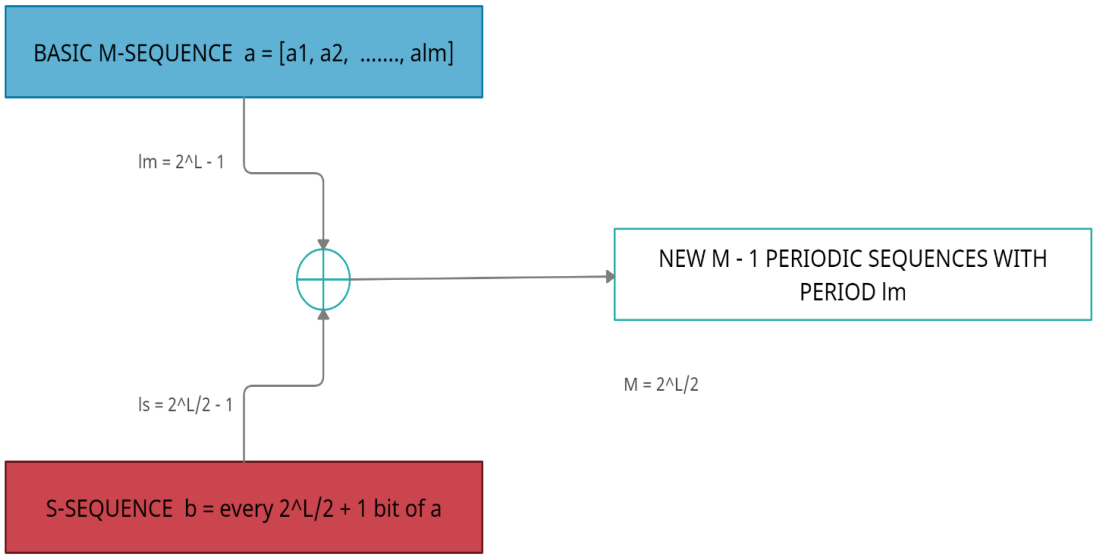

Figura 23 Schema a blocchi per la generazione dei codici di Kasami

Nel caso di Kasami la cross-correlazione è a tre valori come era per i codici generati con Gold, però i valori non normalizzati in questo caso seguono la seguente espressione:

$$\left\{-1, -\left(2^{\frac{L}{2}}+1\right), 2^{\frac{L}{2}}-1\right\}$$

Dunque, il picco massimo nella funzione di cross-correlazione per ogni coppia di sequenze risulterà essere

$$R_{c\_max} = 2^{\frac{L}{2}} + 1$$

Di conseguenza nel caso di Kasami la funzione di cross-correlazione risulta essere migliore rispetto al caso di Gold, perché il valore massimo di cross-correlazione è più basso.

$$R_{cKasami_{max}} = 2^{\frac{L}{2}} + 1 \text{ VS } R_{cGold_{max}} = 2^{\frac{L}{2}+1} + 1$$

Di seguito vengono mostrate alcune immagini relative ai grafici della funzione di autocorrelazione e di cross-correlazione generati tramite il software MATLAB.

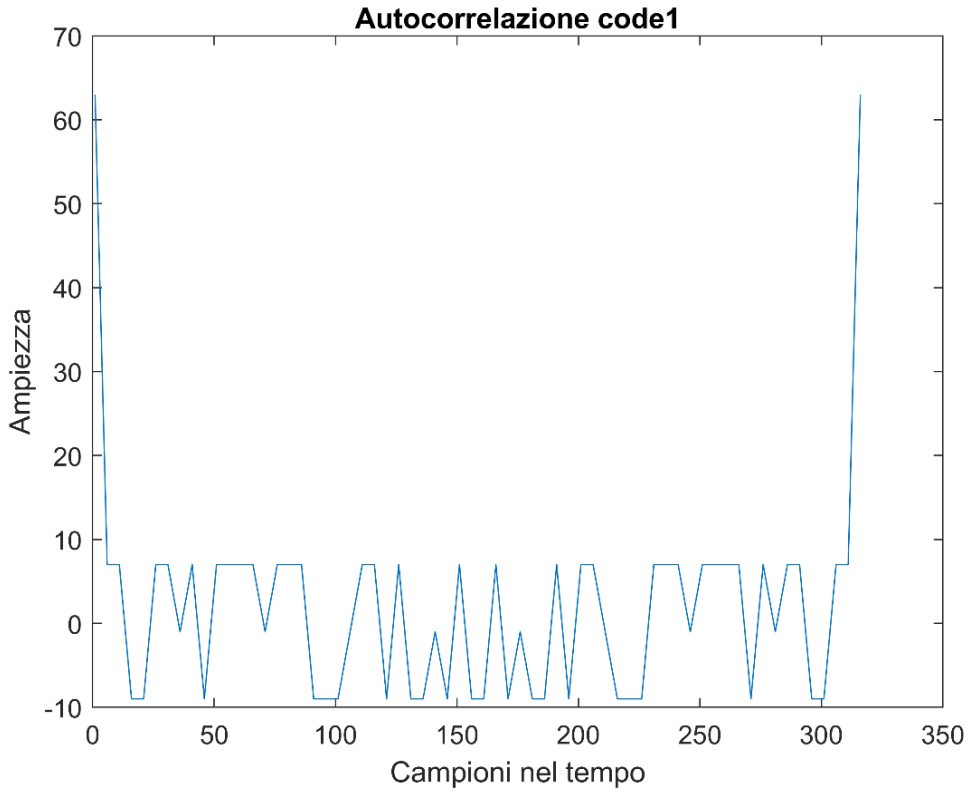

Figura 24 Funzione di autocorrelazione del primo codice generato con Kasami

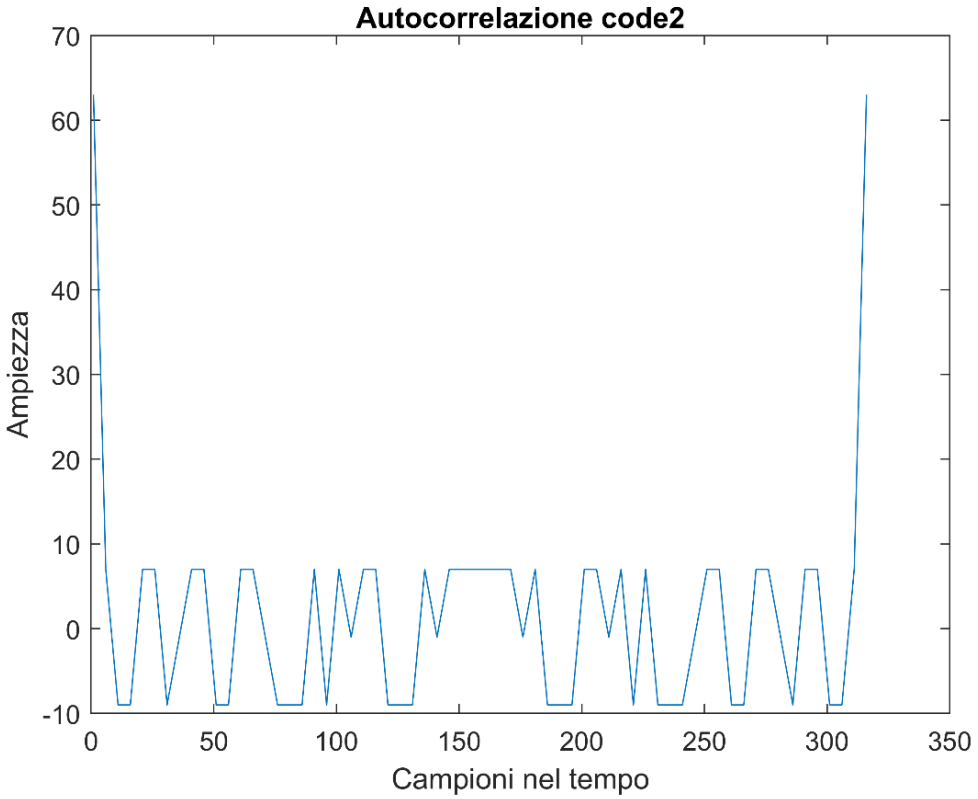

Figura 25 Funzione di autocorrelazione del secondo codice generato con Kasami

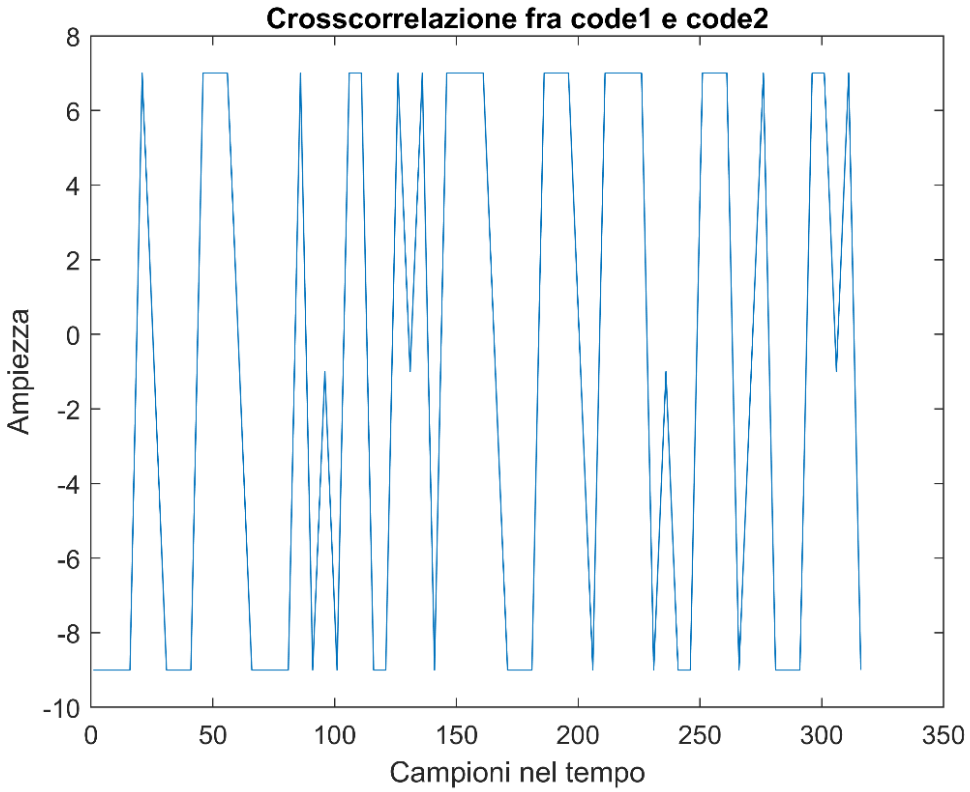

Figura 26 Funzione di cross-correlazione fra il primo codice e il secondo codice generati con Kasami

Confrontando il risultato della funzione di cross-correlazione ottenuto mediante Gold, *Figura 22*, con quello ottenuto mediante Kasami, *Figura 26*, si osserva in quest'ultimo la presenza di picchi più bassi a parità di L.

ALGORITMO DI HADAMARD-WALSH

I codici di Walsh hanno la proprietà di essere tutti ortogonali fra loro e idealmente un numero molto elevato di codici può essere prodotto in maniera semplice via software.
L'algoritmo di Walsh permette di generare $N = 2^n$ codici di lunghezza pari a N stesso. La procedura è alquanto semplice:

$$H_N = \begin{bmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{bmatrix} \text{ con } H_1 = [1].$$

Ogni riga (o colonna) della matrice H_N è un codice di Walsh.
La matrice H_N è costituita da quattro sottomatrici ognuna di dimensione N/2. Ogni sottomatrice sarà, dunque, costituita a sua volta da altre quattro sottomatrici di dimensione N/4 e così via in modo ricorsivo fino ad arrivare a H_1. Per rendere meglio l'idea si consideri il caso di N = 8:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

In ogni caso la prima riga e la prima colonna di ogni matrice è costituita da soli 1. Ogni altra riga (o colonna) contiene N/2 elementi pari a -1 e N/2 elementi pari a 1. Contando le righe a partire da 0, la riga numero N/2 è costituita da N/2 elementi pari a 1 consecutivi seguiti poi da altri N/2 pari a -1. Fra ogni coppia di righe il numero di elementi diversi è pari anch'esso a N/2, tale valore viene chiamato *distanza*.
La caratteristica di queste matrici così generate è che ogni riga (o colonna) è ortogonale a tutte le altre. Ciò assicura che non ci sia alcuna interferenza fra i segnali codificati. Nonostante ciò, lo svantaggio principale di tale algoritmo è che i codici prodotti hanno buone proprietà di ortogonalità solamente quando c'è esatta sincronizzazione nella trasmissione dei segnali dai diversi trasmettitori. Tale cosa rende meno applicabile tale algoritmo in situazioni pratiche in cui il sistema funziona in maniera asincrona, come nel caso del progetto trattato in questa tesi.

Di seguito sono mostrate le funzioni di autocorrelazione e di cross-correlazione relative a due codici generati con l'algoritmo di Walsh.

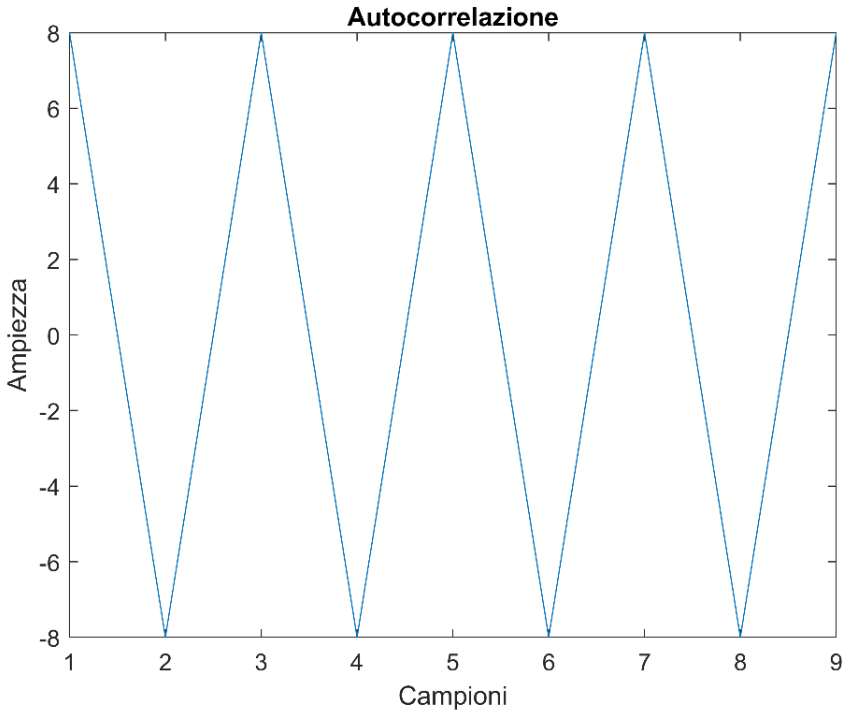

Figura 27 Funzione di autocorrelazione di un codice generato con Walsh

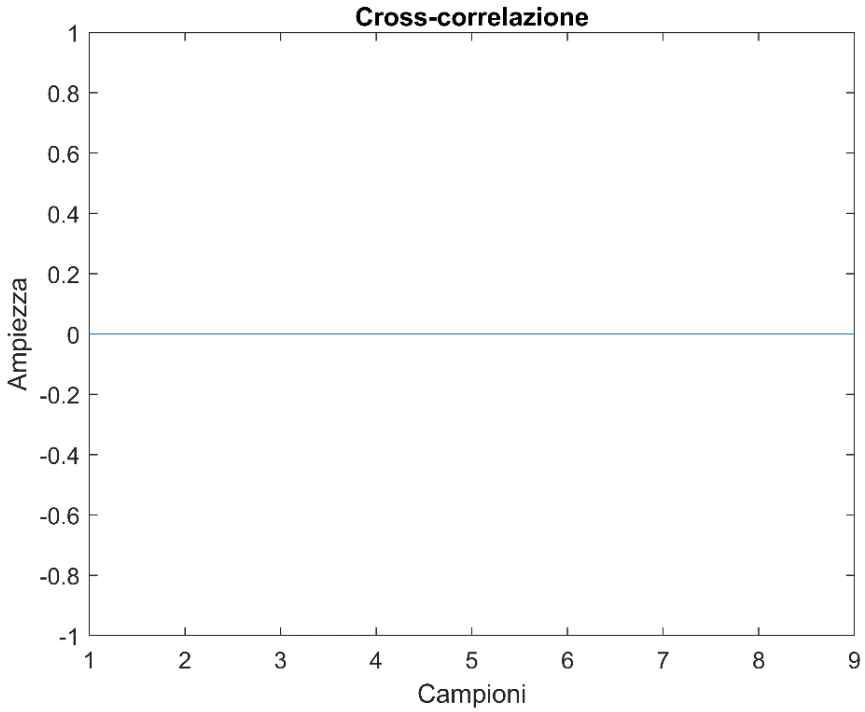

Figura 28 Funzione di cross-correlazione fra due codici generati con Walsh

Come si può osservare dalla *Figura 28*, la funzione di cross-correlazione è sempre nulla. Questo è dovuto alle proprietà di ortogonalità dei codici di Walsh. La funzione di autocorrelazione, invece, non presenta proprietà altrettanto buone, come si può osservare dalla *Figura 27*, ci sono vari valori diversi da zero che la funzione assume.
Riassumendo i principali vantaggi e svantaggi dei codici generati con l'algoritmo di Walsh sono i seguenti:

VANTAGGI

- semplicità nella generazione dei codici via software;
- ortogonalità dei codici prodotti che corrisponde, dunque, a non avere interferenze fra diversi utenti, idealmente;

SVANTAGGI

- le proprietà di ortogonalità sono soddisfatte solo per sistemi sincroni;
- la funzione di autocorrelazione presenta diversi valori non nulli;

4 VERIFICA SPERIMENTALE

4.1 INTRODUZIONE

In questo capitolo viene mostrata l'implementazione di un sistema di trasmissione e ricezione basato sugli algoritmi di Gold e di Kasami per la generazione dei codici utilizzati per effettuare la modulazione (e la demodulazione) DSSS.
Nella progettazione del sistema sono stati implementati diversi blocchi di codice fondamentali. Nello specifico si parlerà per il trasmettitore dei blocchi per la generazione del segnale da trasmettere, la costruzione delle M-sequences e, infine, la modulazione DSSS-BPSK. Mentre per il ricevitore si parlerà dei blocchi per la compensazione dello sfasamento introdotto dalla propagazione, la demodulazione DSSS-BPSK e la corretta rappresentazione dei bit tramite il decisore.
La propagazione del comportamento meccanico della struttura viene eseguita tramite software mediante un algoritmo di raytracing appositamente sviluppato. Verranno pertanto mostrati i risultati delle simulazioni considerando un tipo di materiale. Saranno considerati vari casi con la presenza di più trasmettitori e ricevitori, che saranno posizionati secondo diverse configurazioni per valutare come varia la *Bit Error Rate* (BER) del sistema. La BER è un parametro che si può ottenere una volta effettuata la decodifica del segnale ottenuto dal ricevitore, essa è il rapporto fra il numero di bit ricevuti e decodificati in modo errato e il numero di bit complessivamente trasmessi.
Infine, verrà scelto uno dei due algoritmi per la successiva implementazione sul microcontrollore.
Il riferimento al codice utilizzato per la realizzazione dei blocchi fondamentali e delle analisi si trova nella appendice A e B.

4.2 BLOCCHI FONDAMENTALI

Il sistema complessivo di comunicazione che è stato implementato può essere rappresentato con dei blocchi funzionali per la parte relativa al trasmettitore e la parte relativa al ricevitore. Di seguito viene riportato lo schema a blocchi del sistema complessivo, verranno pertanto descritte le funzioni relative a ciascun blocco funzionale.

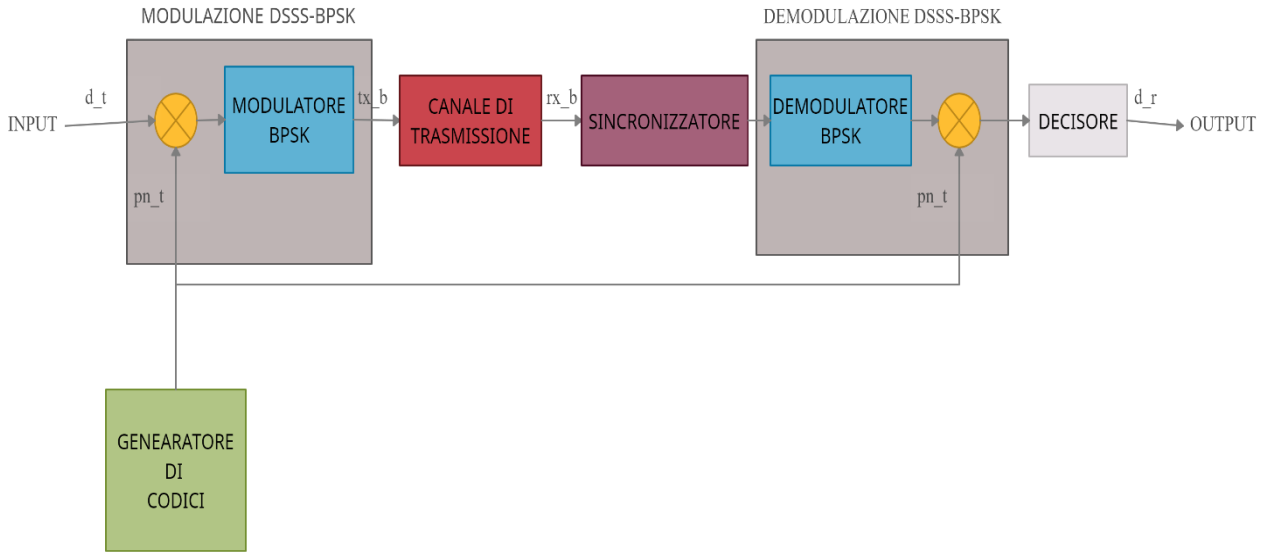

Figura 29 Schema a blocchi del sistema di modulazione e demodulazione

GENERAZIONE DEL SEGNALE DA TRASMETTERE

Per prima cosa, il sistema prevede la trasmissione di un segnale che deve soddisfare certe caratteristiche. Il segnale di partenza viene generato in modo casuale e con una lunghezza fissata. Dunque, una volta definita la lunghezza della M-sequence e il numero di campioni per ogni chip di essa, il segnale generato dovrà avere un numero di campioni per ogni simbolo pari a quello della M-sequence con cui si effettuerà la modulazione. Inoltre, è necessaria la presenza di un preambolo iniziale, che servirà poi per la corretta demodulazione nel lato del ricevitore. Si aggiungerà, infine, del ritardo variabile, in modo da simulare la trasmissione asincrona. Il ritardo nei segnali è dovuto al fatto che il sistema è asincrono, pertanto alcuni segnali potrebbero essere trasmessi prima di altri. Tale considerazione va fatta perché nella scelta degli algoritmi per la generazione dei codici è risultato che alcuni di essi generavano sequenze con buone caratteristiche solo per sistemi sincroni (si veda l'algoritmo di Walsh). Pertanto, nella valutazione delle prestazioni della modulazione DSSS è stato necessario considerare eventuali ritardi nei segnali per simulare la non sincronicità del sistema.
Successivamente i simboli saranno convertiti in -1 e 1 in quanto più adatti per la modulazione DSSS-BPSK.

Per la realizzazione del segnale è stata utilizzata la funzione *Signal_Gen*. La funzione prende come parametri di ingresso Sb (numero di campioni per ogni bit), Nbit (numero complessivo di bit da trasmettere), P_Len (lunghezza del preambolo) e delay (flag per l'inserimento del ritardo). Il codice genera una sequenza casuale di bit, ognuno con Sb campioni, che vengono poi convertiti nei simboli -1 e 1, a seconda che il campione sia uno 0 o un 1.

Di seguito viene mostrato il segnale così generato.

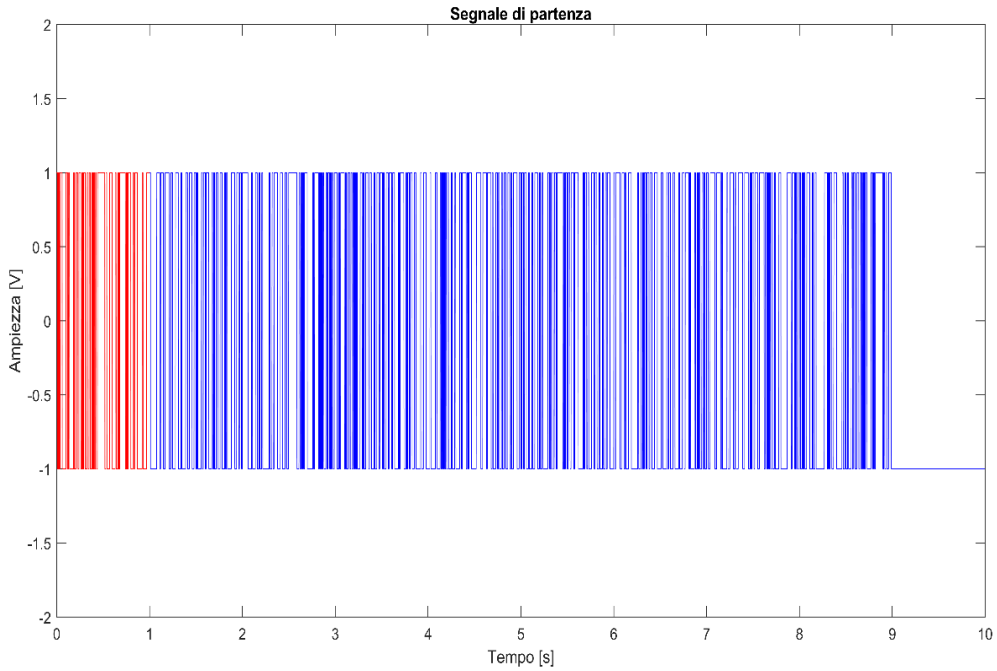

Figura 30 Segnale di partenza generato con la funzione Signal_Gen

Dalla *Figura 30* si può osservare uno dei possibili segnali generati con la funzione *Signal_Gen*.
In questo caso sono stati generati 1000 bit ognuno con $Sb = 315$ campioni. La porzione rossa rappresenta il preambolo formato dai primi 100 bit del segnale.
Il preambolo rappresenta una sequenza di bit generata in maniera casuale, o comunque definita in una fase preliminare, ma nota sia al trasmettitore che al ricevitore. Tale sequenza sarà essenziale per ridurre al minimo lo sfasamento introdotto dalla propagazione sul mezzo simulato, si parla di *riallineamento del segnale*. Durante la fase di demodulazione viene effettuata una funzione di cross-correlazione fra il preambolo e il segnale ricevuto. Questa operazione permette di identificare un picco di massima correlazione in corrispondenza del quale si ha il ritardo, *lag*, introdotto dalla propagazione del segnale. Pertanto, grazie a questo metodo è possibile eliminare il ritardo del segnale e proseguire con la decodifica.

GENERATORE DI CODICI

Questo blocco prevede l'utilizzo di un metodo per la produzione dei codici usati nella modulazione DSSS. Un modo consiste nell'utilizzo di una sequenza generata in modo randomico presa da una distribuzione Gaussiana univariata. Tale sequenza pseudo-casuale è la sequenza pn. Siccome essa presenta caratteristiche simili a quelle del rumore bianco, la correlazione con versioni traslate di sé stessa risulta essere molto bassa a parte nel caso di traslazione nulla dove

l'autocorrelazione è massima.

Un altro metodo consiste nel generare le M-sequences. Un primo algoritmo per la generazione delle M-sequences è quello di Gold.
Facendo riferimento al codice in appendice, la funzione *M_Seq_Gen* si occupa della creazione della matrice contenete i codici per la modulazione.
Come nel caso della generazione del segnale di partenza, S_b rappresenta il numero di campioni per bit. L, invece, è il numero di flip-flop nei registri LFSR che vengono impiegati in questa funzione per la generazione delle prime due M-sequences. In uscita viene restituita la Gold_mat, la matrice contenente le M-sequences disposte per riga.
Un altro algoritmo che può essere utilizzato nel blocco per la generazione dei codici è quello di Kasami. In questo caso viene sempre usata la funzione *M_Seq_Gen*, però adattata al nuovo metodo. I parametri sono gli stessi del caso di Gold, in uscita, però, viene fornita la Kasami_mat. Quest'ultima è una matrice contenente i codici di Kasami disposti nelle righe.

Siccome la scelta dell'algoritmo in questa parte del sistema incide sul numero di codici prodotti e siccome tali codici sono inseriti come righe per le matrici Gold_mat e Kasami_mat, è chiaro, dalla trattazione avvenuta nel capitolo precedente, che scegliendo l'implementazione del blocco mediante l'algoritmo di Kasami la matrice risultante avrà un numero di righe inferiore rispetto al caso dell'algoritmo di Gold. (Per esempio, scegliendo L = 6 si avrà per Kasami un numero di righe inferiore di circa otto volte quello che si avrebbe nel caso di Gold).

MODULAZIONE DSSS-BPSK

A prescindere dall'algoritmo utilizzato, il blocco successivo consiste nell'effettuare la modulazione DSSS-BPSK. A questo punto, ogni simbolo del segnale di cui si vuole effettuare la trasmissione viene moltiplicato per una sequenza scelta fra le possibili presenti nella matrice dei codici (Gold_mat o Kasami_mat a seconda dell'algoritmo implementato nel blocco precedente). Tale moltiplicazione agisce sullo spettro del segnale modulato, allargandone la banda e abbassandone l'ampiezza in modo dipendente dal numero di chip utilizzati per la sequenza, N_c. Successivamente viene applicata la modulazione BPSK (Binary Phase Shift Keying), in modo tale da effettuare una traslazione dello spettro a frequenze maggiori. Questo viene fatto per adattare meglio le caratteristiche spettrali del segnale a quelle del canale di trasmissione. Tale modulazione consiste nel moltiplicare il segnale per una sinusoide, portante, a frequenza f_{cr}.

Di seguito viene mostrata la trasformata di Fourier del segnale durante le varie fasi della modulazione DSSS-BPSK. In questo caso è stato scelto l'utilizzo dell'algoritmo di Kasami per la generazione delle M-sequences. Il numero di flip-flop nel registro LFSR, L, è pari a 6, dunque il numero di chip risultante è $N_c = 2^6 - 1 = 63$.

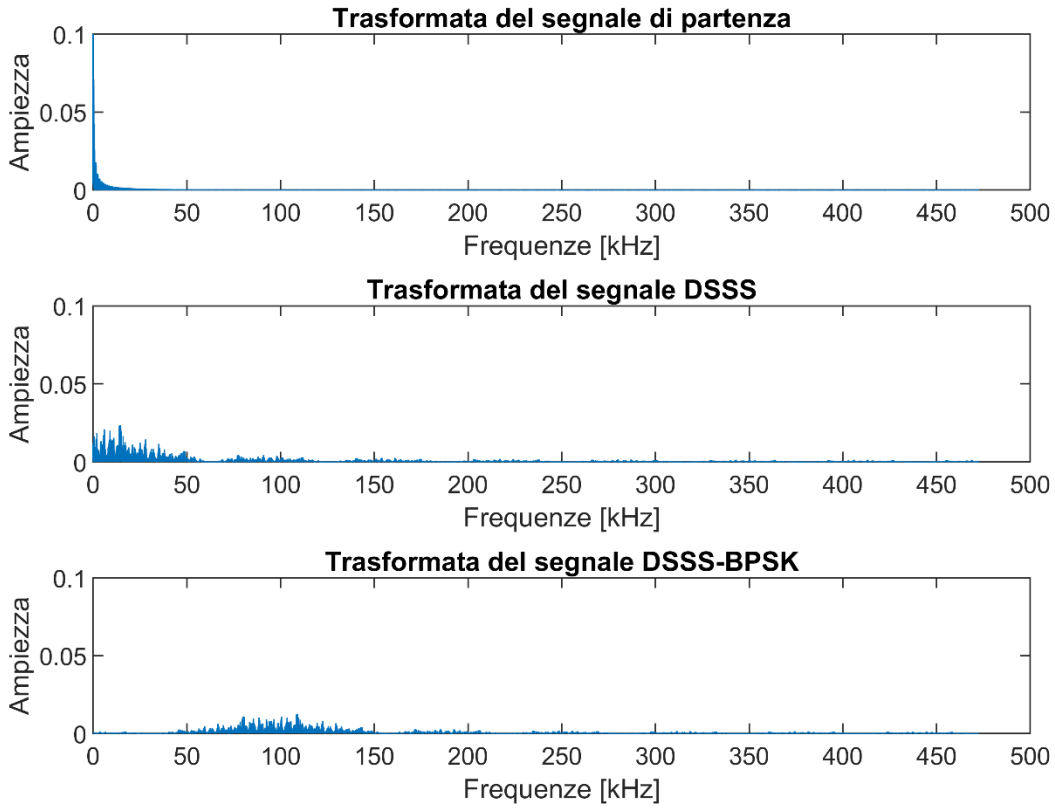

Figura 31 Trasformate di Fourier del segnale durante le varie fasi della modulazione DSSS-BPSK

CANALE DI TRASMISSIONE

Il canale di trasmissione è stato simulato tramite la funzione *Dispersion_Simulation*. L'algoritmo per la simulazione prevede la conoscenza delle curve di dispersione associate al mezzo. Oltre alla riproduzione della dispersione del segnale, vengono riprodotti anche gli effetti di interferenza. Per rendere la simulazione più realistica è stato riprodotto anche il rumore elettronico dovuto ai dispositivi, per farlo è stata aggiunta una componente di rumore additivo gaussiano ai segnali generati in modo tale che il rapporto segnale-rumore (SNR) fosse circa pari a 20 dB.

Di seguito viene mostrata una porzione del segnale trasmesso sul canale.

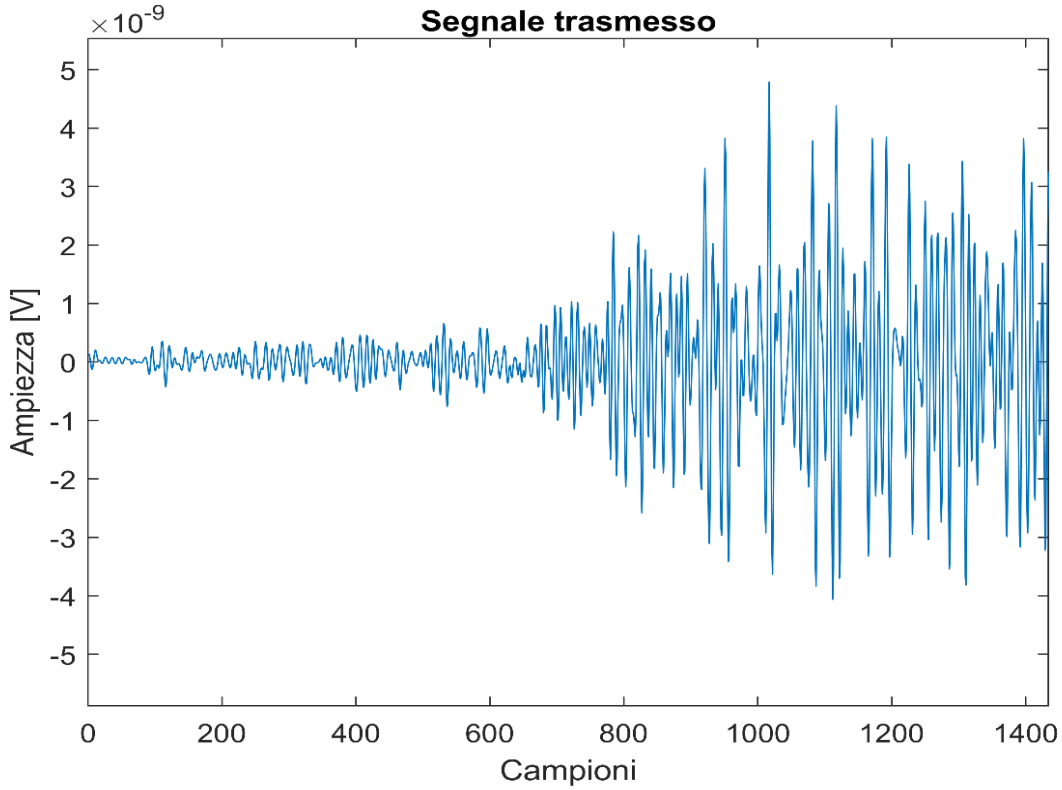

Figura 32 Porzione del segnale trasmesso nel canale

SINCRONIZZATORE

Durante la fase di demodulazione, il primo passo consiste nel compensare lo sfasamento introdotto dalla propagazione del segnale lungo il canale trasmissivo. Per farlo si è scelto di effettuare la funzione di cross-correlazione fra il segnale trasmesso e il rispettivo preambolo. In corrispondenza del massimo picco della funzione di cross-correlazione si ha lo sfasamento introdotto dalla propagazione. Di conseguenza si considera quello come istante iniziale del segnale ricevuto.

DEMODULAZIONE DSSS-BPSK

Il passo successivo consiste nell'effettuare la demodulazione vera e propria del segnale ricevuto. Pertanto, viene effettuata prima la demodulazione BPSK per mezzo di un integratore (di fatto si integra il segnale per estrarne la componente informativa contenuta nella fase). Di seguito si procede con la demodulazione DSSS, essa viene eseguita mediante la moltiplicazione del segnale con la stessa M-sequence utilizzata per la precedente modulazione DSSS.
Tale procedura riporta lo spettro del segnale alla sua forma di partenza.

DECISORE

Il decisore rappresenta il blocco finale nella catena di modulazione e demodulazione. Il suo scopo consiste nello scegliere se il valore del simbolo i-esimo debba essere un -1 o un 1. Per farlo viene effettuata una media sugli S_b campioni per ogni simbolo ricevuto: se questa risulta essere < 0 allora si tratta di un -1, viceversa si tratta di un 1.

Di seguito viene riportato il grafico relativo a una demodulazione effettuata utilizzando i passaggi descritti in precedenza.

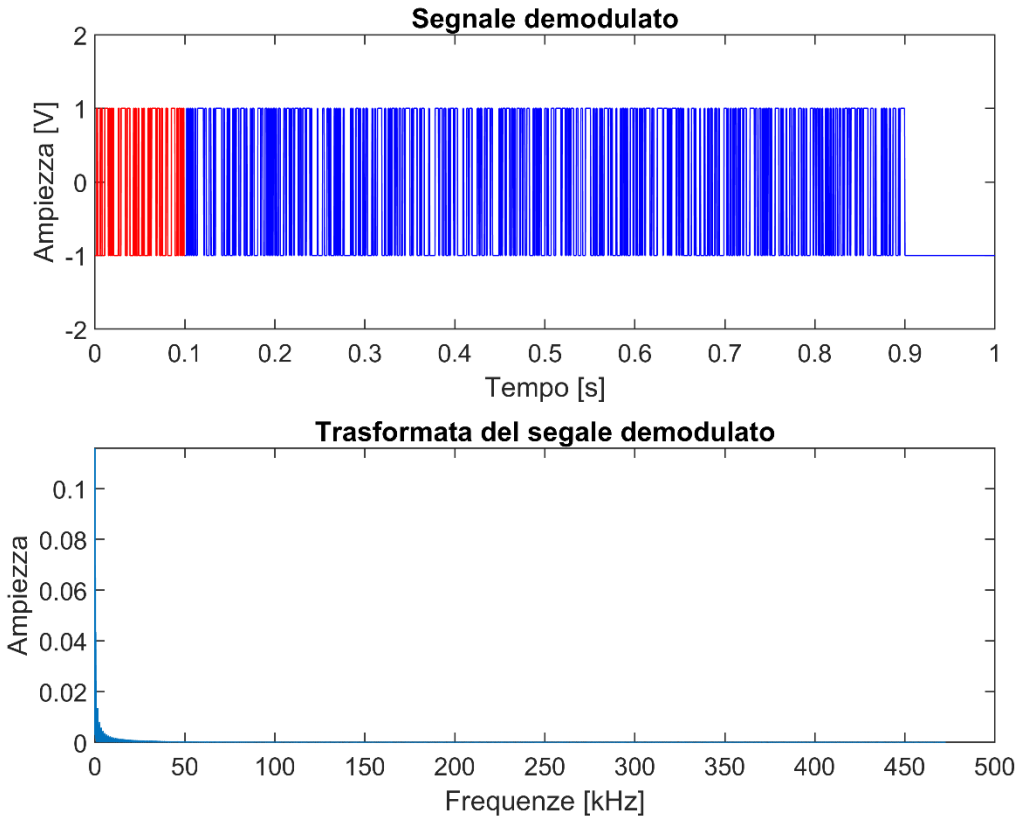

Figura 33 Segnale demodulato nel tempo e relativa trasformata di Fourier nelle frequenze

Il segnale così demodulato può presentare degli errori causati dalle non idealità del sistema implementato. Tali errori verranno considerati nella Bit Error Rate.

4.3 RISULTATI DELLE SIMULAZIONI

Di seguito verranno mostrati alcuni risultati ottenuti mediante le simulazioni supponendo di avere a disposizione una lastra quadrata in alluminio con spessore di 3mm. I risultati mostrati saranno relativi al caso dell'implementazione mediante i codici di Gold e il caso mediante i codici di Kasami.

TEST CON ALLUMINIO DI SPESSORE 3mm

I test sono eseguiti considerando come materiale una lastra di alluminio di spessore 3mm.
Di seguito viene mostrato il grafico rappresentativo delle curve di dispersione del materiale utilizzato.

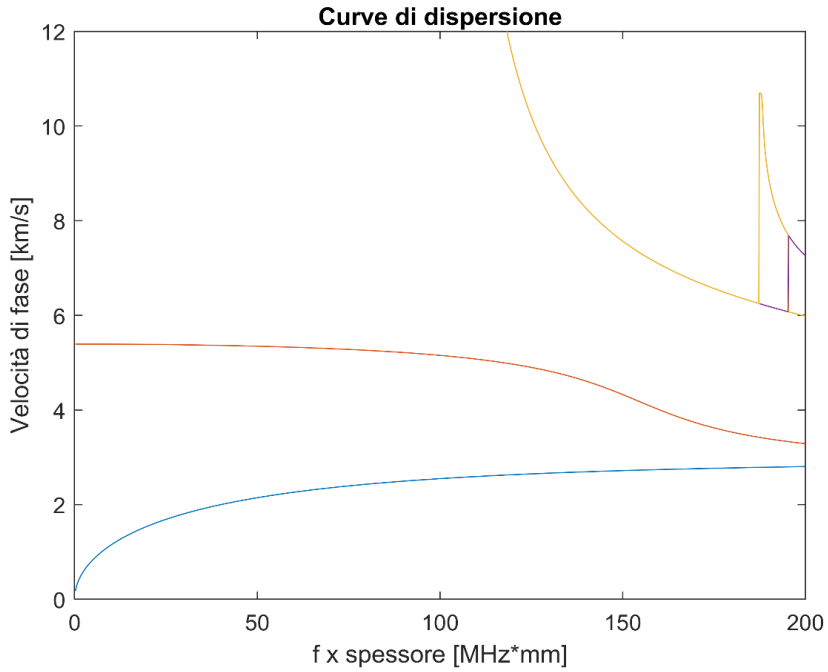

Figura 34 Curve di dispersione per l'alluminio con spessore 3mm

La *Figura 34* mostra 4 modi, i primi due sono il modo simmetrico e antisimmetrico di ordine zero, i successivi sono i primi due modi di ordine superiore. È interessante osservare che, come ci si aspettava dalla teoria, i modi A0 e S0 sono presenti a tutte le frequenze. Inoltre, il modo S0 (curva rossa) possiede una velocità di fase maggiore rispetto a quella del modo A0 (curva blu).

Inizialmente vengono posizionati tre trasmettitori e un unico ricevitore. I segnali trasmessi vengono fatti propagare sul mezzo sfruttando le caratteristiche del materiale. Per simulare meglio i fenomeni di dispersione, riflessione e interferenza, sono state scelte tre geometrie distinte. Ogni geometria è stata scelta per mettere alla prova situazioni tipiche che possono avvenire in ambienti reali.
Di seguito sono riportate le tre geometrie scelte.

1. Nella prima geometria è stata scelta una distanza elevata fra i trasmettitori e il ricevitore. Tutti i dispositivi sono alla stessa altezza, y, ma con diverse angolazioni rispetto al ricevitore.

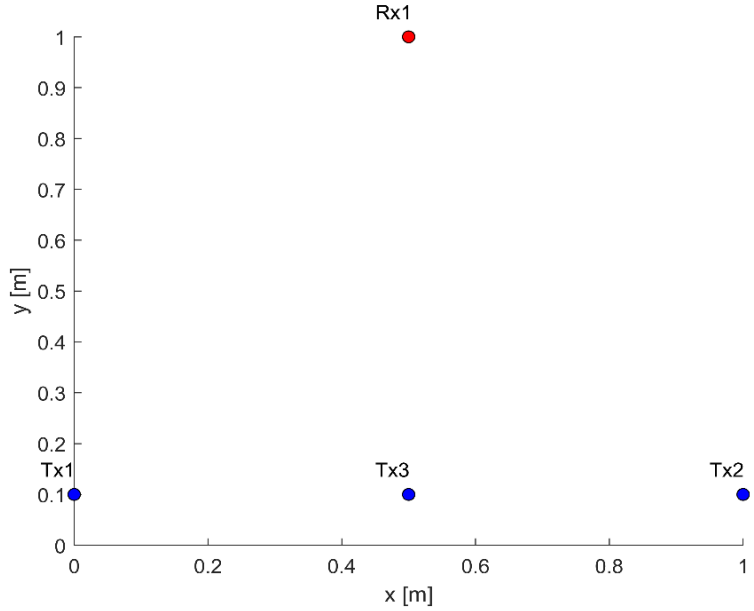

Figura 35 Locazione 1 dei dispositivi di trasmissione e ricezione

2. Il ricevitore è posizionato centralmente rispetto agli altri dispositivi. La geometria in questione è la più simmetrica e non prevede differenze fra i trasmettitori in termini di posizioni più o meno favorevoli alla ricezione del proprio segnale.

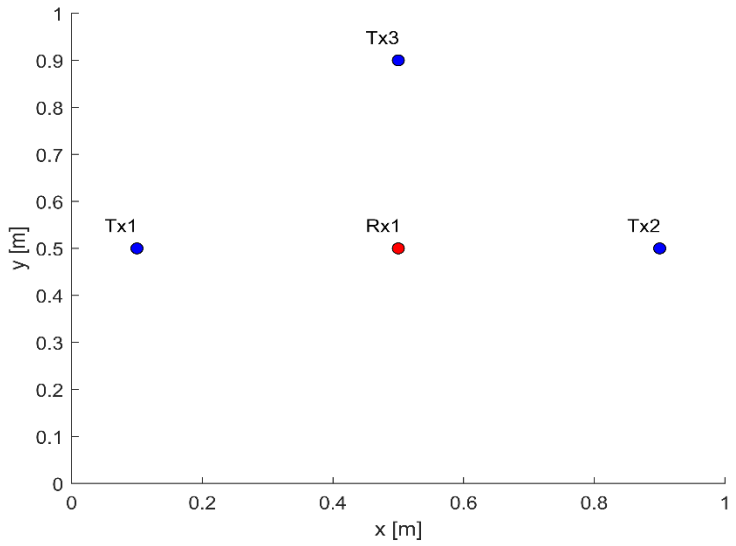

Figura 36 Locazione 2 dei dispositivi di trasmissione e ricezione

3. La terza geometria prevede due trasmettitori posizionati in modo più favorevole alla ricezione del proprio segnale e un terzo trasmettitore più distante. Inoltre, la simmetria viene a mancare in questa configurazione, il trasmettitore Tx3 è leggermente più vicino al ricevitore rispetto a tutti gli altri dispositivi presenti.

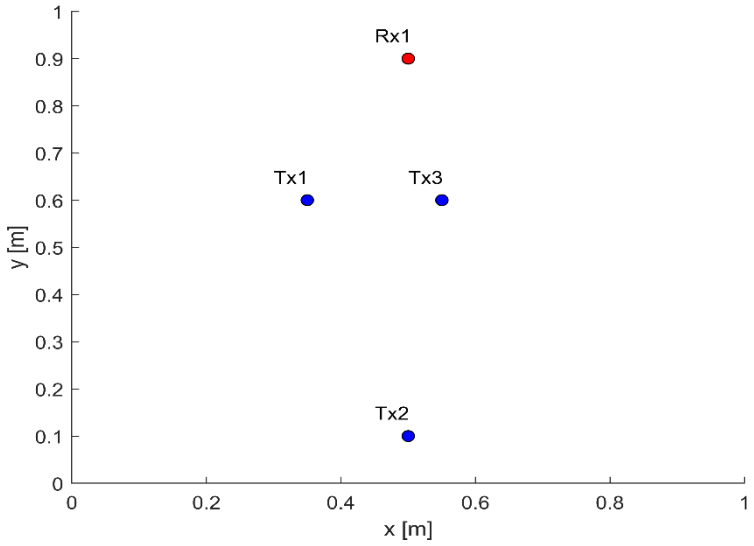

Figura 37 Locazione 3 dei dispositivi di trasmissione e ricezione

Per le prove si è scelto di utilizzare i seguenti parametri:

Nbit **(Numero di bit da trasmettere)**	1000
L **(Numero di flip-flop nei registri LFSR)**	6
N_c **(Numero di chip nella M-sequence)**	63
S_c **(Numero di campioni per chip)**	5
P_Len **(Numero di bit usati per il preambolo)**	100

Tabella 4 Parametri per la simulazione

Di seguito vengono messi a confronto i risultati ottenuti mediante Gold e Kasami. Nello specifico sono state considerate diverse frequenze di trasmissione e un rumore gaussiano additivo in modo tale da avere un SNR circa pari a 20 dB. Sono state, inoltre, messe a confronto le tre diverse geometrie.
I risultati ottenuti fanno riferimento alla decodifica relativa al segnale generato dal trasmettitore 2, Tx2.

RISULTATI BER% CON GOLD					
	fs = 10kHz	fs = 20kHz	fs = 30kHz	fs = 63kHz	fs = 315kHz
GEOMETRIA 1	0.70	1.90	0	0.60	21.60
GEOMETRIA 2	0	0.20	0	0	4.60
GEOMETRIA 3	0	0	0	0.10	2.0

Tabella 5 Risultati della BER% ottenuti con Gold. fs è la frequenza dopo il chipping del segnale

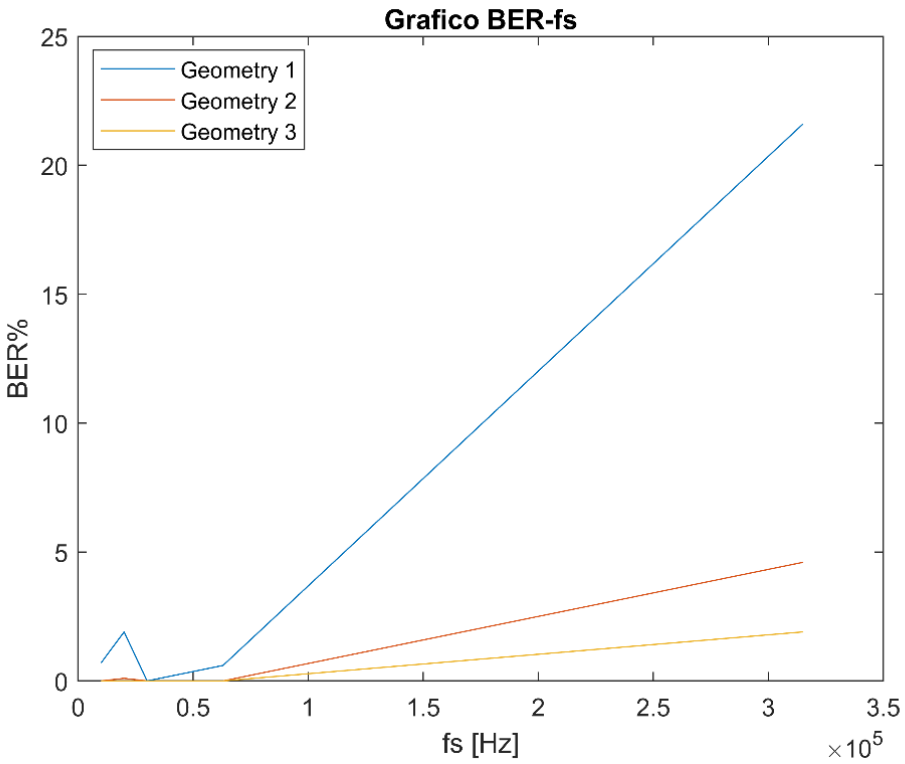

Figura 38 Grafico della BER% nel caso di Gold al variare della geometria

RISULTATI BER% CON KASAMI					
	fs = 10kHz	fs = 20kHz	fs = 30kHz	fs = 63kHz	fs = 315kHz
GEOMETRIA 1	0	0	0	0.90	13.70
GEOMETRIA 2	0	4.60	0.30	0	6.20
GEOMETRIA 3	0	0	0	0	2.90

Tabella 6 Risultati della BER% ottenuti con Kasami. fs è la frequenza dopo il chipping del segnale

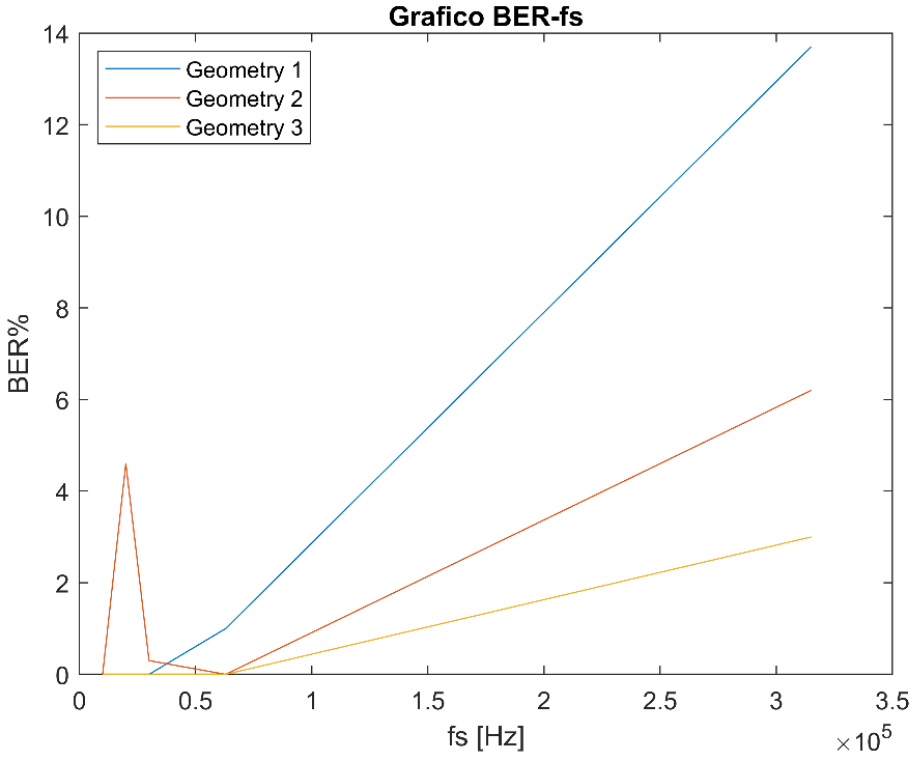

Figura 39 Grafico della BER% nel caso di Kasami al variare della geometria

Osservando le tabelle e i grafici relativi alla BER% in funzione della frequenza di trasmissione e della geometria utilizzata, si possono notare i seguenti fatti:

1. Nel caso dell'algoritmo di Gold la geometria 1 risulta la più problematica. Probabilmente ciò è dovuto alla distanza fra i trasmettitori e il ricevitore, inoltre, i tre trasmettitori sono posti alla stessa altezza, y, ciò potrebbe causare maggiori problemi di interferenza MAI fra gli utenti. Nel caso di Kasami si ha, invece, una maggiore resistenza alle problematiche descritte, ciò risulta in una BER nettamente migliore per la geometria 1.
2. La frequenza $f_s = 315\ kHz$ risulta essere la più problematica per tutte le configurazioni utilizzate. Tale cosa è dovuta al fatto che all'aumentare della frequenza aumenta anche l'interferenza inter-simbolo tra due chip successivi, quindi le "code" dei simboli precedenti contaminano il contenuto dei simboli successivi e causa questo non si riesce più a discriminare in maniera accurata il contenuto di un simbolo rispetto quello di un altro.
3. In generale le BERs% ottenute mediante l'utilizzo dell'algoritmo di Kasami risultano essere più uniformi, sulle tre geometrie considerate, rispetto ai risultati ottenuti con Gold. Il valore alla frequenza $f_s = 20\ kHz$ nel caso della geometria 2, è leggermente più alto di quello atteso per Kasami. Nonostante ciò, con i successivi test sono stati ottenuti risultati migliori anche per quella casistica.

In conclusione, l'algoritmo di Kasami porta all'ottenimento di Bit Error Rate migliori rispetto al caso dell'algoritmo di Gold. Tale fatto è dovuto alla migliore funzione di cross-correlazione ottenuta mediante Kasami, la quale presenta picchi di minore ampiezza rispetto al caso di Gold.
L'unico svantaggio del metodo di Kasami riguarda il minor numero di codici generati. Tale cosa può essere problematica nel caso in cui il numero di utenti della rete dovesse aumentare. Pertanto, per situazioni che necessitano di un elevato numero di trasmettitori e ricevitori, l'utilizzo dell'algoritmo di Gold per la generazione dei codici è consigliato, mentre in tutte le altre situazioni l'utilizzo del metodo di Kasami risulta essere migliore.

I successivi test proseguono con l'utilizzo dell'algoritmo di Kasami e la geometria scelta prevede l'aggiunta di un ricevitore nella rete. Inoltre, la locazione dei dispositivi è stata scelta in modo tale da considerare le diverse problematiche dovute all'interferenza MAI e ai fenomeni di riflessione. Si vuole pertanto valutare come varia la Bit Error Rate a seconda del ricevitore indirizzato.

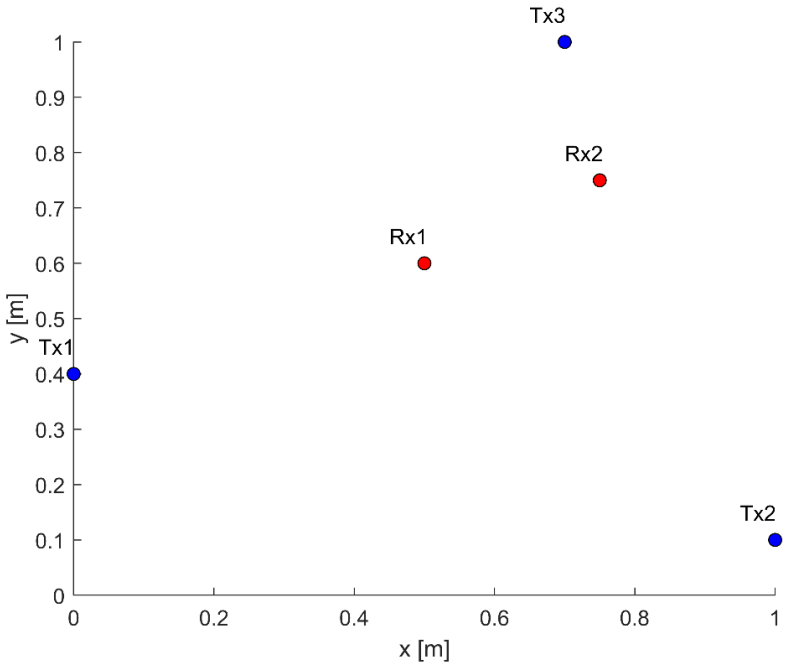

Figura 40 Locazione dei dispositivi di trasmissione e ricezione

RISULTATI BER% CON KASAMI (DECODIFICA DI Tx2 e L = 6)					
	fs = 10kHz	fs = 20kHz	fs = 30kHz	fs = 63kHz	fs = 315kHz
Ricevitore 1	0	0	0	0	5.40
Ricevitore 2	0	0	0	0.50	6.20

Tabella 7 Risultati della BER% ottenuti con Kasami, caso Tx2. fs è la frequenza dopo il chipping del segnale

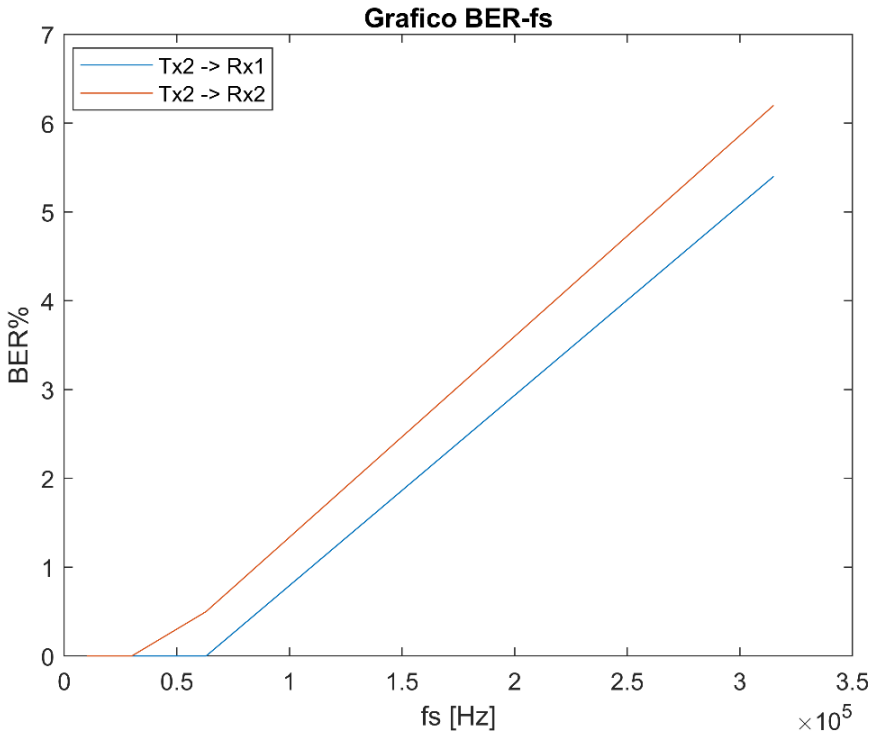

Figura 41 Grafico della BER% nel caso di Kasami e di due ricevitori. Trasmissione da Tx2

I risultati ottenuti mostrano quanto sia incisiva la locazione dei ricevitori rispetto a quella dei trasmettitori nell'ottenimento di una buona BER.
In questo caso specifico il ricevitore Rx2 ha maggiori problemi nel decodificare correttamente il segnale spedito dal trasmettitore Tx2 per la frequenza $f_s = 315\ kHz$. Questa cosa è dovuta al fatto che, mentre Rx1 si trova in una posizione più centrale rispetto a tutti i trasmettitori, Rx2 è stato posizionato in modo tale da essere nettamente più vicino a Tx3. Pertanto, il segnale trasmesso da Tx2 verso Rx2, in caso di trasmissione contemporanea dei tre trasmettitori, percorrerà un tragitto più lungo rispetto al segnale proveniente da Tx3, dunque sarà soggetto a maggiore interferenza.

Considerando adesso il caso del segnale trasmesso da Tx3 verso i due ricevitori, otteniamo i seguenti risultati.

RISULTATI BER% CON KASAMI (DECODIFICA DI Tx3 e L = 6)					
	fs = 10kHz	fs = 20kHz	fs = 30kHz	fs = 63kHz	fs = 315kHz
Ricevitore 1	0	0	0	1.90	5.10
Ricevitore 2	0	0	0	2.60	4.80

Tabella 8 Risultati della BER% ottenuti con Kasami, caso di Tx3. fs è la frequenza dopo il chipping del segnale

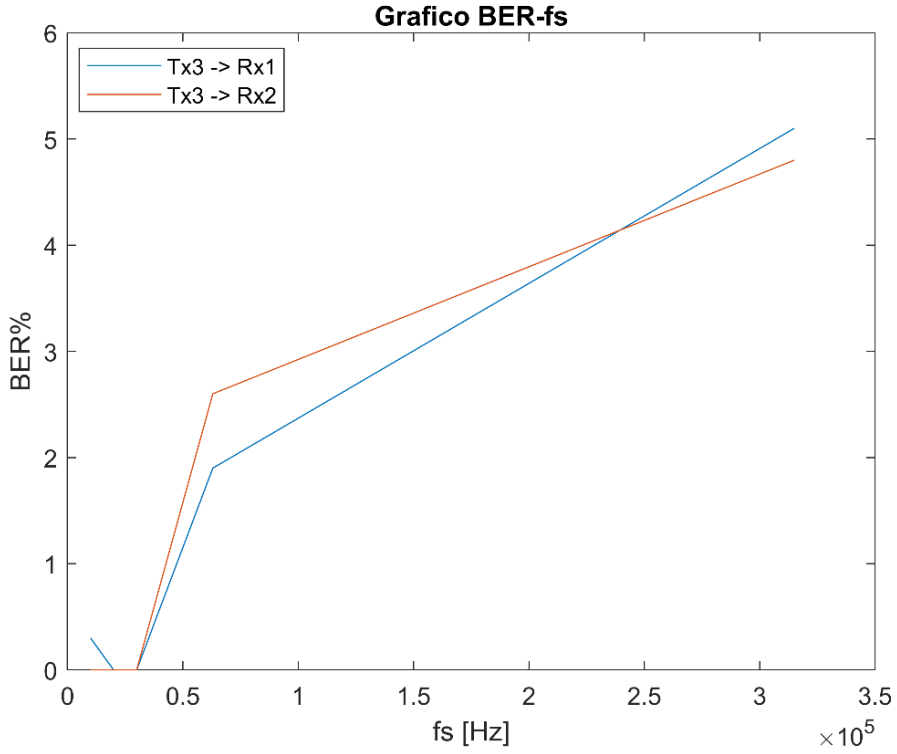

Figura 42 Grafico della BER% nel caso di Kasami e di due ricevitori. Trasmissione da Tx3

In quest'ultimo caso si osserva un leggero miglioramento per la frequenza $f_s = 315\ kHz$ e un altrettanto leggero peggioramento per la frequenza $f_s = 63\ kHz$.

Un'altra prova interessante prevede la diminuzione del parametro L. Nello specifico si vuole osservare come varia la Bit Error Rate scegliendo $L = 4$ e lasciando inalterati gli altri parametri.

RISULTATI BER% CON KASAMI (DECODIFICA DI Tx2, L = 4)					
	fs = 10kHz	fs = 20kHz	fs = 30kHz	fs = 63kHz	fs = 315kHz
Ricevitore 1	9.30	4.70	4.30	11.90	30.60
Ricevitore 2	8.60	4.10	3.80	11.50	30.70

Tabella 9 Risultati della BER% ottenuti con Kasami,nel caso di Tx2 con L = 4. fs è la frequenza dopo il chipping del segnale

I risultati ottenuti mostrano un notevole incremento della BER% dovuto alla diminuzione del parametro L. Tali risultati sono ragionevoli in quanto diminuendo L diminuisce anche il numero di chip delle M-sequences, pertanto i picchi normalizzati della funzione di cross-correlazione aumentano creando una maggiore interferenza fra i segnali.

Di seguito viene riportato anche per questa casistica il grafico relativo alla Bit Error Rate percentuale.

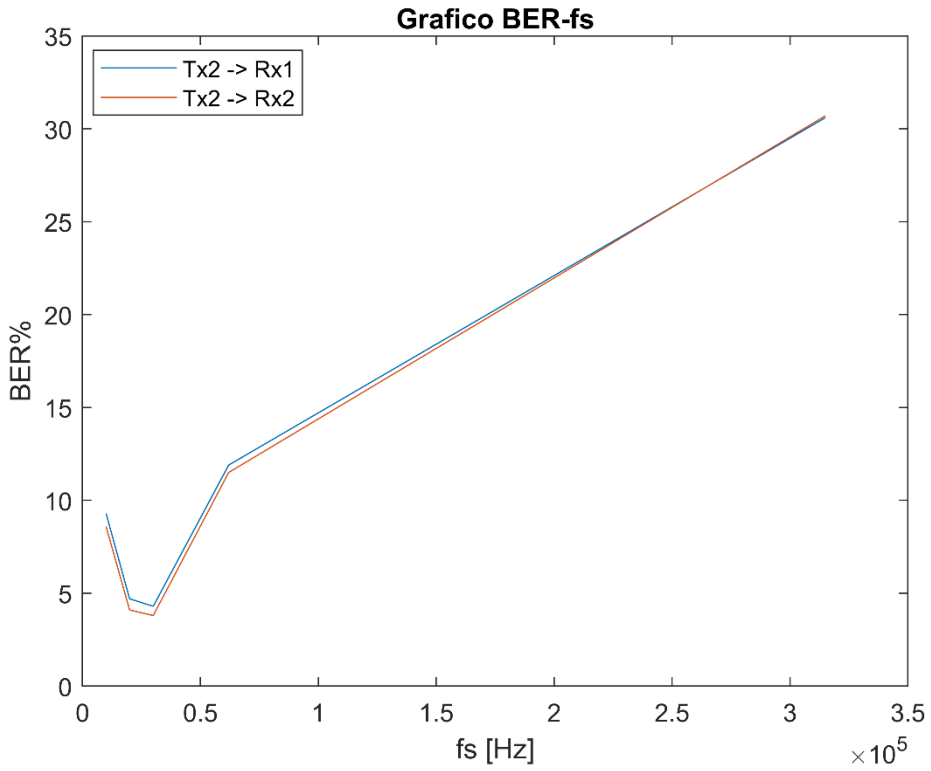

Figura 43 Grafico della BER% nel caso di Kasami e di due ricevitori. Trasmissione da Tx2 con L = 4

In conclusione, dati i buoni risultati ottenuti con la modulazione per mezzo dei codici di Kasami con la scelta di $L = 6$, quest'ultima configurazione è stata scelta per l'implementazione finale sul microcontrollore.

5 IMPLEMENTAZIONE SU MICROCONTROLLORE

5.1 SCHEDA NUCLEO-64 STM32F446RE

L'ultimo step nel presente progetto di tesi consiste nella migrazione del codice MATLAB, utilizzato per la validazione dei risultati, in codice C. Questo passo è necessario per potere implementare il sistema di modulazione e demodulazione DSSS su un microcontrollore. Il processo di trasposizione è molto delicato in quanto il nuovo codice dovrà essere eseguito su un sistema embedded che, a differenza di un comune personal computer, avrà risorse molto limitate e prestazioni inferiori.

Il microcontrollore utilizzato è un **STM32F446RE** ed è montato su una scheda **Nucleo64 di ST Microelectronics**. Le schede denominate Nucleo di ST, sono una nuova serie di schede elettroniche a basso costo adatte alla creazione di applicazioni con i microcontrollori **STM32**.

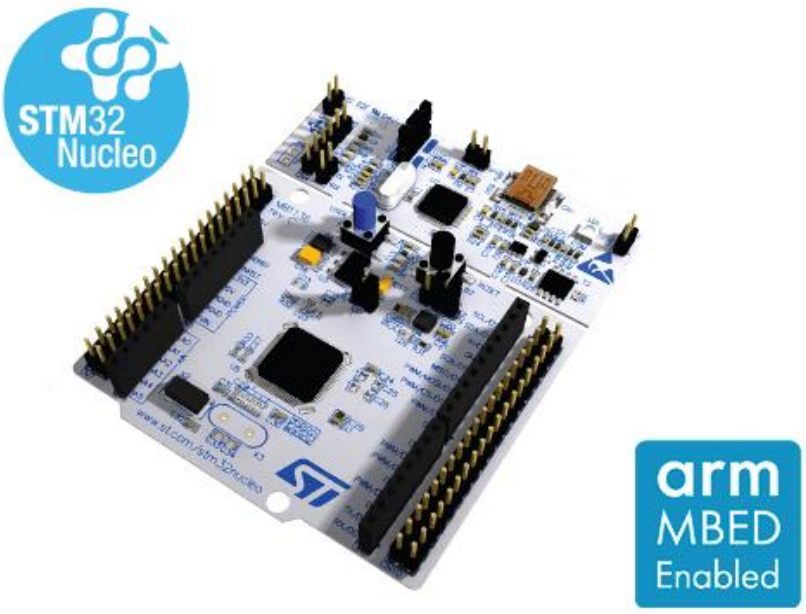

Figura 44 Scheda Nucleo64 STM32F446RE [17]

Le caratteristiche principali della serie sono:

- microcontrollore: STM32;
- header compatibile con Arduino Uno rev. 3;
- header Morpho, estensione ST per l'accesso completo agli I/O STM32;
- programmatore/Debugger ST-LINK/V2-1 on-board che può anche essere usato separatamente alla scheda tramite il connettore SWD;
- alimentazione tramite connettore USB oppure tramite alimentazione esterna;
- presenza di un LED utilizzabile dall'utente, oltre ad un LED per l'alimentazione e un LED tricolore per monitorare la comunicazione USB;
- presenza di un pulsate utilizzabile dall'utente e di uno per il reset;

La porta USB ha tre principali funzioni:

- programmazione e debugging;
- COM port virtuale;
- mass storage device per la programmazione **drag'n'drop;**

L'alimentazione può essere fornita via USB (3.3V, 5V) oppure tramite alimentatore esterno (7 – 12V).

La famiglia di circuiti integrati (IC) STM32 si basa sul microcontrollore RISC a 32bit ARM-Cortex. La seguente tabella riassume i membri della famiglia STM32:

Serie STM32	**Core ARM**
F7	Cortex-M7
F4, F3	Cortex-M4F
F2, F1, L1	Cortex-M3
L0	Cortex-M0+
F0	Cortex-M0

Tabella 10 Serie STM32

Nello specifico la scheda utilizzata per il progetto fa parte della serie F4, di cui fanno parte le schede ad alte prestazioni.

Il numero 64, in Nucleo64, indica il tipo di package, si tratta di un LQFP a 64 pin. La scheda impiegata presenta le seguenti caratteristiche principali:

-128 kB di RAM; 512 kB di FLASH; 180 MHz di frequenza di clock per la CPU;

Le principali periferiche della scheda sono le seguenti:

- 2x UART, 4x USART, 4x SPI, 4x I²C;
- timer a 16 e 32bit;
- 3x ADC e 2x DAC a 12bit;

Inoltre, possiede 2 DMA (Direct Memory Access) ad 8 canali, per il trasferimento dei dati dalle periferiche alla memoria, dalla memoria alle periferiche o dalla memoria alla memoria.

Figura 45 STM32F446RE diagramma a blocchi [18]

5.2 REALIZZAZIONE DEL CODICE IN C

L'implementazione del sistema per la modulazione e la demodulazione DSSS sul microcontrollore prevede la scrittura del codice nel linguaggio C. Per farlo è stato utilizzato l'IDE Visual Studio per la verifica della correttezza del programma. Successivamente è stato utilizzato l'IDE Atollic TrueSTUDIO e il tool STM32CubeMX per l'implementazione finale del codice sul microcontrollore.

MODULAZIONE

Il codice è strutturato nel seguente modo; all'interno del main vengono definiti i parametri per la creazione e la modulazione DSSS del segnale.

```
// DATI
short int L = 6;                    // Numero di flip-flop
short int Nbit = 15;                // Numero di bit da trasmettere
short int Nc = pow(2, L) - 1;       // Numero di chip della sequence
float Tb = 0.01;                    // Tempo di bit
float time = Nbit*Tb;               // Durata in secondi del segnale
short int Sc = 5;                   // Numero di campioni per chip
int Sb = Nc*Sc;                     // Campioni per bit
float Tc = Tb/Nc;                   // Durata di un chip nella sequence
float Ts = Tc/Sc;                   // Tempo di campionamento (dopo il chipping)
float fs = 1/Ts;                    // Frequenza di campionamento (dopo il chipping)
short int P_Len = 3;                // lunghezza preambolo
short int ID = 4;                   // Identificativo per la M-sequence da usare per la codifica DSSS
float fp = fs/10;                   // Frequenza portante per la modulazione BPSK
float BER;                          // Bit Error Rate
```

Per via delle risorse limitate, è stato scelto di terminare la trasmissione a soli 15 bit, dove i primi 3 rappresentano il preambolo.

Successivamente si procede con la generazione del segnale di partenza. Per farlo è stata richiamata la funzione *SignalGen*.

```
void SignalGen(short int *sig, short int *sigNoMod, short int *preambolo, short int *sigTX, int n, int Sb, short int
P_Len)
{

  short int rit = 0;// Ritardo nullo

  int i;
  for(i = 0 ; i < n ; i++)
  {
        if(i > n - rit - 1)
        {
              sigNoMod[i] = 0;
              sig[i] = sigNoMod[i]*2 - 1; // 0 -> -1 e 1 -> +1
        } else if(i < P_Len){
              sigNoMod[i] = rand()%2;
              sig[i] = sigNoMod[i]*2 - 1; // 0 -> -1 e 1 -> +1
              preambolo[i] = sig[i];
        } else {
              sigNoMod[i] = rand()%2;
              sig[i] = sigNoMod[i]*2 - 1; // 0 -> -1 e 1 -> +1
    }
```

```
    int j;
    for(j = 0 ; j < Sb ; j++)
    {
      sigTX[j + Sb*i] = sig[i];
    }
  }
}
```

La funzione *SignalGen* si occupa della generazione del segnale da modulare. Essa fa in modo che il segnale sia adatto alla modulazione DSSS, pertanto fa sì che ogni bit sia costituito da $n = Sb * Sc$ campioni. Oltre a ciò, inserisce un preambolo all'inizio della sequenza e infine converte i campioni 0 e 1 nei simboli -1 e 1.

Per la generazione dei codici è stato scelto l'utilizzo dell'algoritmo di Kasami. La funzione *M_Seq_Gen* realizza una matrice contenete i codici di Kasami per riga.

```
void M_Seq_Gen(short int  **codeMat, int Sb, short int L)
{
        .
        .
        .
// Creazione della prima M-sequence e della s-sequence
        int c1;
        int c2 = 0;
        for(c1 = 0 ; c1 < Nc ; c1++){
                insertElement(PN1, temp1[5], c1, Sc);
                if(c1 == j){
                        insertElement(PN2, temp1[5], c2, Sc);
                        j = j + Dec;
                        c2++;
                } // Ottenimento della s-sequence come decimazione della M-sequence
                res1 = temp1[4] ^ temp1[5];  // xor primi due tappi dello LFSR
                res2 = res1 ^ temp1[1];      // xor del risultato con il tappo successivo
                res3 = res2 ^ temp1[0];      // xor del risultato con il tappo successivo
                shifter(temp1, res3, L);     // shift e inserimento del risultato dello xor
        }

// Inserimento di PN1 come primo elemento di kasami_mat
        int c3;
        for(c3 = 0 ; c3 < Nc*Sc; c3++){
                codeMat[0][c3] = PN1[c3];
        }
```

Questa prima parte della funzione realizza la M-sequence e la s-sequence di partenza. Si può osservare l'implementazione di uno LFSR realizzato con l'xor degli elementi [0, 1, 4, 5] e la traslazione mediante la funzione *shifter*. L'output del registro viene poi inserito in PN1, la M-sequence, tramite la funzione *insertElement*. PN2, la s-sequence, è ottenuta tramite la decimazione di PN1 di un fattore $Dec = 2^{L/2} + 1$. La prima M-sequence viene poi inserita come prima riga della matrice codeMat.

Successivamente vengono realizzati gli altri $2^{L/2} - 1$ codici di Kasami che verranno inseriti come righe successive di codeMat.

```
// Aggiunta dei rimanenti 2^(L/2) - 1 codici per un tot di 2^(L/2)
        int index;
        int index2;
        int index3;
        for(index = 0 ; index < (pow(2, L/2) - 1) ; index++){

// xor fra i 2^(L/2) - 1 elementi della s-sequence con i
// 2^L - 1 della M-sequence. Dunque, 2^(L/2) + 1 passaggi
                for(index2 = 0 ; index2 < (pow(2,L/2) + 1) ; index2++){
                        for(index3 = 0 ; index3 < (pow(2, L/2) - 1); index3++){
                                s = (index3*Sc + (pow(2, L/2) - 1)*index2*Sc);
                                x = PN1[s] ^ PN2[index3*Sc];
                                insertElement(k_sequence, x, index3 + index2*(pow(2, L/2) - 1), Sc);
                        }
                }

// Inserimento della nuova M-sequence appena generata in codeMat
                int c4;
                for(c4 = 0 ; c4 < Nc*Sc ; c4++){
                        codeMat[index + 1][c4] = k_sequence[c4];
                }

// shift a sx di un elemento di Sc campioni della s-sequence
                copyArr(temp, PN2, 0, 0, Sc);

                int c6 = Sc*(pow(2, L/2) - 1);
                int c7;
                copyArr(temp_Sc2, PN2, 0, c6 - Sc, Sc); // Salva l'ultimo elemento di PN2 di Sc campioni
                for( ; c6 >= 2*Sc ; c6 -= Sc){
                        copyArr(temp_Sc1, PN2, 0, c6 - 2*Sc, Sc); // Salva l'elemento di Sc campioni prima  di
sovrascriverlo
                        for(c7 = 1 ; c7 < Sc + 1 ; c7++){
                                PN2[c6 - Sc - c7] = temp_Sc2[Sc - c7]; // Copia l'elemento di Sc campioni in una
posizione precedente nell'array
                        }
                        copyArr(temp_Sc2, temp_Sc1, 0, 0, Sc);
                }
                copyArr(PN2, temp, Sc*(pow(2, L/2) - 1) - Sc, 0, Sc); // Scrive l'ultimo elemento di PN2
        }
}
```

Per ottenere le altre M-sequences è stata eseguita l'operazione di xor degli elementi della prima M-sequence con tutte le versioni ciclicamente traslate della s-sequence. Ad ogni passaggio viene generata una nuova M-sequence che viene inserita come riga in codeMat, dunque la s-sequence viene traslata di un elemento e l'operazione viene ripetuta finché tutti i codici non sono generati. La funzione *copyArr* permette di copiare un array, o una sua porzione, in un altro array.

Una volta generati i codici di Kasami è possibile effettuare la modulazione DSSS-BPSK del segnale. Per prima cosa si estrae dalla matrice contenete i codici la riga da utilizzare per la modulazione.

```
for(int i = 0 ; i < Nbit ; i++)
 {
   for(int j = 0 ; j < Sb ; j++)
   {
     kasamiCode[j + i*Sb] = codeMat[ID][j]*2 -1;
   }
 } // Estrazione del codice dalla codeMAt
```

ID fornisce l'indice della riga di codeMat relativo al codice da utilizzare.

```
#define PI 3.14159265

void DSSS_BPSK_Coder(short int *sigTX, float *sigModTX, short int *kasamiCode, int len, float *t, float fp)
{
  // CODIFICA DSSS
  ////////////////////////////////////////////
  for(int i = 0 ; i < len ; i++)
  {
    sigTX[i] = sigTX[i]*kasamiCode[i];
  }
  ////////////////////////////////////////////

  // CODIFICA BPSK
  ////////////////////////////////////////////
  for(int i = 0 ; i < len ; i++)
  {
    sigModTX[i] = sigTX[i]*cos(2*PI*fp*t[i]);
   }
  ////////////////////////////////////////////
}
```

L'operazione consiste nel moltiplicare il segnale da trasmettere per il codice di Kasami. Dunque, il segnale risultante viene moltiplicato per una sinusoide portante per effettuare la modulazione BPSK.

A questo punto la modulazione del segnale è stata completata e quest'ultimo può essere trasmesso verso il ricevitore per la demodulazione.

DEMODULAZIONE

Siccome il segnale ricevuto potrebbe essere affetto da ritardo causato dalla propagazione sul mezzo, la prima operazione da fare è il "riallineamento" del segnale. Pertanto, nella funzione *DSSS_BPSK_Decoder* viene inizialmente eseguita una operazione di cross-correlazione fra il segnale ricevuto e il preambolo, quest'ultimo noto a priori sia al trasmettitore che al ricevitore.

```
void DSSS_BPSK_Decoder(short int *sigDemodRX, short int *demodBit, float *sigModRX, float *t, int Sb, short int
P_Len, short int Nbit, float fs, short int *kasamiCode, float *preamboloTX, float fp, short int Nc)
{
        .
        .
        .
        xcorr(Rc, &delay, preamboloTempTX, sigModTempRX, P_Len*Sb + 100);

        for(int i = 0 ; i < Nbit*Sb ; i++)
        {
                sigModRX[i] = sigModRX[i + delay];
        }
```

La funzione *xcorr* realizza la cross-correlazione fra gli array in ingresso e restituisce un valore relativo al ritardo, delay. Questo valore corrisponde al picco

della funzione di cross-correlazione fra il segnale e il preambolo.
Di seguito si scartano gli elementi del segnale precedenti al parametro di ritardo.

Si procede, dunque, con la decodifica del segnale.
Per prima cosa viene effettuata la demodulazione BPSK.

```
// DECODIFICA BPSK
//////////////////////////////////////////
        for(int i = 0 ; i < Nbit*Sb ; i++)
        {
                sigModRX[i] = sigModRX[i]*cos(2*PI*fp*t[i]);
        }
        .
        .
        .

// Integrazione
        for(int i = 0 ; i < Nbit*Nc ; i++)
        {
                result[i] = sum(sigModRX, i*Sb/Nc, i*Sb/Nc + Sb/Nc);
                if(result[i] < 0)
                {
                result[i] = -1;
                } else {
                        result[i] = 1;
                } // Riporta i valori a -1 e 1
        } // Effettua l'integrazione negli intervalli degli impulsi di chip, Sb/Nc = Sc

//Riporta il segnale alla dimensione di partenza come numero di campioni
        for(int i = 0 ; i < Nbit*Nc ; i++)
        {
                for(int j = 0 ; j < Sb/Nc ; j++)
                {
                        sigModRX[j + i*(Sb/Nc)] = result[i];
                }
        }
```

La demodulazione BPSK consiste nell'estrazione dell'informazione contenuta nella fase del segnale ricevuto, dunque si moltiplica il segnale per la portante e si effettua una operazione di integrazione. A questo punto il segnale ottenuto è quello modulato DSSS.

Al segnale viene poi effettuata la demodulazione DSSS. Pertanto, si moltiplicano i suoi campioni per la sequenza di Kasami utilizzata in precedenza per la fase di modulazione.

```
// DECODIFICA DSSS
//////////////////////////////////////////
        for(int i = 0 ; i < Nbit*Sb ; i++)
        {
                sigModRX[i] = sigModRX[i]*kasamiCode[i]; // Demodulazione DSSS
        }
//////////////////////////////////////////
```

Infine, l'ultimo blocco di codice effettua la decisione sui campioni per minimizzare l'errore commesso.

```
// DECISORE
//////////////////////////////////////////
//Il decisore per ogni Sb campioni ne fa la somma poi li media per decidere il
//valore corretto e lo porta a 0 e 1
        printf("%c: ", 'P');
        for(int i = 0 ; i < Nbit ; i++)
        {
                demodBit[i] = sum(sigModRX, i*Sb , i*Sb + Sb)/Sb > 0 ;
                printf("%hd ", demodBit[i]);
                if(i == P_Len - 1)
                {
                printf("\n%c: ", 'S');
                }
        }
```

Il decisore somma Sb campioni alla volta del segnale e li divide per Sb, dunque se il valore risultante è maggiore di 0 il bit sarà considerato un 1, viceversa si avrà uno 0.
Il segnale risultante sarà dunque quello di partenza a meno di errori commessi durante la fase di demodulazione.

Di seguito viene mostrato un risultato ottenuto simulando con Visual Studio.

START OF PROCESS

Starting signal:

P: 0 1 1

S: 0 0 0 0 1 0 0 0 1 1 0 1

Delay = 0

Starting demodulated signal:

P: 0 1 1

S: 0 0 0 0 1 0 0 0 1 1 0 1

BER(%) = 0.0000%

END OF PROCESS

Con la trasmissione di 15 bit (3 di preambolo e 12 di segnale) e l'assenza di propagazione (Delay = 0) il segnale viene ricostruito esattamente, dunque si ottiene una Bit Error Rate nulla.

L'implementazione sul microcontrollore occupa il seguente spazio in memoria:

TRASMETTITORE (byte)				
text	data	bss	dec	Hex
29746	124	1840	31728	7BF0

RICEVITORE (byte)				
text	data	bss	dec	hex
35356	124	1840	37320	91c8

Tabella 11 Sopra lo spazio occupato dal codice per il trasmettitore, sotto quello occupato dal codice del ricevitore

Dove

- **text** rappresenta la quantità di spazio occupato nella memoria FLASH. Qui vengono memorizzate le funzioni e le variabili **const**;
- **data** è riferito alle variabili inizializzate, le quali occupano spazio nella memoria RAM;
- **bss** (Block Started by Symbol) è riferito alle variabili non inizializzate, le quali occupano spazio nella memoria RAM;
- **dec** è la somma di text, data e bss;
- **hex** è la versione esadecimale di dec;

Per concludere il codice è stato infine adattato a quello presente nei dispositivi in laboratorio.

Per la parte relativa al trasmettitore è stata fatta una maschera per la selezione dell'algoritmo relativo alla modulazione DSSS-BPSK. La maschera in questione è la 0xE0. Una volta selezionata la maschera, inizia la trasmissione dei singoli campioni che vengono prima modulati tramite la funzione *DSSS_BPSK_Coder*. Il codice viene chiamato ad interrupt, perciò ogni volta che è possibile la trasmissione di un nuovo dato il codice viene rieseguito e viene decrementato un contatore, rep, in modo da identificare il termine della trasmissione, ovvero quando rep raggiunge lo zero.

```
else if(nodeConfig.txMode & 0xE0) // Maschera per la codifica DSSS-BPSK
{
        if (chd->rep == 0) {  // Se rep è 0 si conclude la trasmissione

                chd->out = LOW;

                DSSS_BPSK_Coder(1); // Reset dei parametri della trasmissione

                } else {

                        chd->out = DSSS_BPSK_Coder(0); // Codifica DSSS_BPSK
                        chd->rep--; //Decremento il numero di campioni che mi mancano da trasmettere

                }

}
```

Per la parte relativa al ricevitore si considera la funzione *DAQ_process*. Il segnale trasmesso viene ricevuto su tre possibili canali; CH1, CH2 e CH3. Pertanto, è possibile estrarre tutti i dati ricevuti ed effettuarne la demodulazione DSSS-BPSK tramite la funzione *DSSS_BPSK_Decoder*.

```
void DAQ_process(void)
{
        .
        .
        .
        int16_t *sigRx = DaqBuff.pDataCh1; // Dati ricevuti sul canale 1
        .
        .
        .
        DSSS_BPSK_Decoder(sigDemodRX, demodBit, sigRx, t, Sb, P_Len, Nbit, fs, kasamiCode, preamboloTX, fp,
        Nc);
        .
        .
        .
}
```

In conclusione, il codice adattato è stato realizzato considerando le risorse limitate del microcontrollore utilizzato. Pertanto, oltre a limitare il numero di bit trasmessi complessivamente, sono state fatte ottimizzazioni sui tipi di dati utilizzati e sono state fatte modifiche al codice relativo al trasmettitore e al ricevitore. Nello specifico, per quanto riguarda la parte di modulazione nel trasmettitore è stato scelto di salvare in memoria FLASH sia il segnale da trasmettere sia la sequenza di Kasami. Ciò ha permesso di ridurre il tempo computazionale necessario per la generazione dei due array. Inoltre, la trasmissione avviene un bit alla volta senza avere necessariamente un array contenente tutto il segnale modulato, cosa che andrebbe a occupare molta memoria. Per quanto riguarda il ricevitore, siccome la funzione *xcorr* risulta molto onerosa in termini di tempo necessario per il completamento della esecuzione, è stato scelto di non utilizzare tutti i campioni acquisiti come ingresso, ma solo i primi $P\_Len + 100$ campioni, dal momento che il ritardo introdotto durante le simulazioni non superava mai le 100 unità di delay.

Dal punto di vista della memoria occupata, gran parte del risparmio è stato osservato per il trasmettitore. I risultati portano a un dec pari a 8300 byte, rispetto ai 31728 byte occupati precedentemente.

6 CONCLUSIONI

In questo progetto sono stati analizzati diversi algoritmi di encoding per l'implementazione di tecniche di modulazione Spread Spectrum, il cui scopo consiste nel realizzare una codifica CDMA per l'attenuazione delle interferenze che si vengono a creare su canali ad accesso multiplo. Lo sviluppo di queste tecniche trova largo impiego in ambito SHM, nello specifico nella comunicazione ad ultrasuoni basata sulle onde di Lamb, dove si sfruttano le capacità di reti di sensori per eseguire la comunicazione in modo non cablato sfruttando la guida d'onda meccanica.
Dunque, sono stati considerati i codici generati tramite una distribuzione gaussiana uniforme, le sequenze pn, poi le M-sequences generate con gli algoritmi di Gold e di Kasami e infine i codici ortogonali generati con il metodo Walsh. Analizzando le funzioni di autocorrelazione e di cross-correlazione ottenute mediante i codici generati con i vari metodi, si è giunti alla conclusione che le caratteristiche migliori e più efficaci per un sistema di tipo asincrono sono ottenute mediante gli algoritmi di Gold e di Kasami. Pertanto, sono state fatte verifiche sperimentali mediante un software di simulazione che sfrutta un algoritmo di raytracing per simulare la propagazione del segnale emesso e ricevuto da uno o più dispositivi posizionati su una lastra di alluminio. Nelle simulazioni sono stati messi a confronto i risultati, in termini di Bit Error Rate percentuale, ottenuti mediante i codici generati dagli algoritmi di Gold e di Kasami. È emerso che quest'ultimo garantisce prestazioni migliori, ma fornisce anche un minor numero di codici rispetto al caso del metodo Gold. Un'altra osservazione è relativa al parametro L, il quale rappresenta il numero di flip-flop nei registri LFSR utilizzati per la generazione delle M-sequences, e quindi anche il numero di chip che le costituiscono. Si è visto come una diminuzione di tale parametro possa influenzare notevolmente la BER%, infatti passando da $L = 6$ a $L = 4$ la BER% è aumentata di diversi punti percentuali a tutte le frequenze considerate.
Infine, nell'ultima fase di progetto si è passati all'implementazione del meccanismo di modulazione e demodulazione su microcontrollore. A tale scopo è stata utilizzata la scheda STM32F446RE. Dunque, sono state descritte tutte le parti principali del codice utilizzato in modo da mostrare i vari passaggi effettuati per la creazione del sistema di modulazione e demodulazione DSSS-BPSK.
I risultati raggiunti aprono le porte ad ulteriori sviluppi futuri. Nello specifico, sarebbe interessante testare il sistema di modulazione e demodulazione in una vera rete di nodi sensori al fine di vederne le reali prestazioni.

APPENDICE A

In questa sezione viene riportato il codice MATLAB utilizzato per la validazione dei risultati. I blocchi fondamentali descritti nel capitolo 4 fanno riferimento alle funzioni utilizzate nel seguente codice, nello specifico viene considerata l'implementazione del solo metodo di Kasami. Fare riferimento alla APPENDICE B per la parte di Gold e di Walsh.

TX_DSSS_BPSK.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%% TX_DSSS_BPSK

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione svolge tre elaborazioni diverse; genera il segnale binario da trasmettere, ne effettua la
% modulazione DSSS, fa la codifica BPSK del segnale.

% ARGOMENTI DELLA FUNZIONE
% kasami_mat: è una matrice in cui ogni riga presenta una m_sequence generata con il metodo kasami.
% Nbit: è il numero complessivo di bit che si vogliono generare.
% time: è la durata complessiva della trasmissione (comprende ritardi e preambolo).
% Sb: è il numero di campioni per bit.
% fs: è la frequenza di campionamento.
% ID: identifica il codice di kasami nella kasami_mat, la riga, da usare.
% P_Len: è la lunghezza del preambolo da generare.
% delay: è un flag per l'aggiunta dello sfasamento nel segnale.

% DATI IN USCITA
% tx: è il segnale modulato DSSS-BPSK.
% code: è il codice usato per la modulazione DSSS.
% Pn: è la ripetizione di code per la lunghezza del segnale in termini di Nbit.
% C: è la portante usata per la modulazione BPSK.
% sig_no_mod: è il segnale non modulato.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
% ----------------------------------------------------------------------

function [tx, code, Pn, C, sig_no_mod] = TX_DSSS_BPSK(kasami_mat, Nbit, time, Sb, fs, ID, P_Len, delay)

  % Creazione dell'asse dei tempi
  t = 0 : 1/fs : time - 1/fs;

%% Generazione del segnale
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  [b, sig_no_mod] = Signal_Gen(Sb, Nbit, P_Len, delay);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Modulazione DSSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Estrazione del codice dalla kasami_mat relativo all'ID scelto
   Kasami = kasami_mat(ID, :);

% Conversione degli elementi in questo modo 0 --> -1 e 1 --> 1.
   code = Kasami * 2 - 1;

% Generazione del segnale con spreading
   m = [];
   Pn = [];

   for i = 1 : Nbit
      Pn = [Pn code];
   end

   m = b.*Pn;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% CODIFICA BPSK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Creazione della portante
   fcr =  fs/10;   % frequenza della portante
   C = cos(2*pi*fcr*t);

%%Generazione del segnale modulato DSSS-BPSK da trasmettere
   tx = m.*C;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

% ---------------------------------------------------------------------
```

Signal_Gen.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Signal_Gen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione ha lo scopo di generare un segnale binario composto da un preambolo e da un
% ritardo.

% ARGOMENTI DELLA FUNZIONE
% Sb: numero di campioni per bit.
% Nbit: numero di bit da generare.
% P_Len: lunghezza del preambolo.
% delay: flag per il ritardo.

% DATI IN USCITA
% Signal: è il segnale generato e convertito in -1 e 1.
% sig_no_mod: è la versione del segnale generato con singoli campioni.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
```

```
% ------------------------------------------------------------------------

function [Signal, sig_no_mod] = Signal_Gen(Sb, Nbit, P_Len, delay)

   % Ritardo di 100 bit
   rit = 100;

   % Generazione del preambolo in modo randomico
   preambolo = randi([0 1], 1, P_Len);

   % Costruzione del segnale in base al flag delay per il posizionamento
   % del ritardo
   switch delay
      case 0
         b = [zeros(1, rit)  preambolo randi([0 1], 1, Nbit - length(preambolo) - rit)];
      case 1
         b = [preambolo randi([0 1], 1, Nbit - length(preambolo) - rit) zeros(1, rit)];
      case 2
         rit = 50;
         b = [zeros(1, rit) preambolo randi([0 1], 1, Nbit - length(preambolo) - rit)];
      otherwise
         b = randi([0 1], 1, Nbit);
   end

   % Segnale di partenza con singoli campioni
   sig_no_mod = b;

   % Si aumenta il numero di campioni di Sb volte
   Signal = [];
   for i = 1 : length(b)
      Signal = [Signal b(i)*ones(1, Sb)];
   end
   Signal = 2*Signal - 1;

end

% ------------------------------------------------------------------------
```

M_Seq_Gen.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% M_Seq_Gen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione genera i codici per la modulazione DSSS con la tecnica di Kasami e mette i risultati
% ottenuti in una matrice dove ogni riga appresenta una sequenza.

% ARGOMENTI DELLA FUNZIONE
% Sb: numero di campioni per bit.
% L: Numero flip-flop.

% DATI IN USCITA
% kasami_mat: Matrice contenente i codici generati con kasami.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
```

```
% ------------------------------------------------------------------------

function [kasami_mat] = M_Seq_Gen(Sb, L)

   % Dati
   Nc = 2^L - 1;      % Numero di chip nella M-sequence
   sr1 = zeros(1, L);  % Elementi nei registri sr1 inizializzati a zero
   sr1(L) = 1;        % Inizializzo a 1 l'ultimo elemento di sr1

   PN1 = [];          % Conterrà la M-sequence di partenza
   PN2 = [];          % conterrà la versione decimata di PN1. s-sequence
   Sc = Sb / Nc;      % Numero di campioni per ogni bit della sequenza
   kasami_mat = ones(2^(L/2), Nc*Sc);  % Inizializzo la kasami_mat
   temp1 = sr1;       % Variabile temporanea per sr1
   k_sequence = [];   % Conerrà la sequenza di kasami i-esima
   Dec = 2^(L/2) + 1;  % Fattore di decimazione per kasami
   j = 1;             % Contatore inizializzato a 1

   % Generazione dei codici di kasami
   % Generazione della M-sequence e della s-sequence con registro LFSR
   for i = 1 : Nc
      PN1 = [PN1 temp1(6).*ones(1, Sc)]; % Inserisco gli elementi per PN1 ognuno con Sc campioni

      % Ottengo PN2 da una decimazione di PN1
      if i == j
         PN2 = [PN2 temp1(6).*ones(1, Sc)];
         j = j + Dec;
      end

      % Effettuo gli xor fra i registri per la creazione della M-sequence
      res1 = xor(temp1(5), temp1(6));
      res2 = xor(res1, temp1(2));
      res3 = xor(res2, temp1(1));

      % Realizzo lo shift dei registri sr1 e inserisco come primo valore di sr1 il risultato dello xor
      % precedente
      temp1 = [res3 temp1(1 : L-1)];
   end

   % Inserisco come primo elemento di kasami_mat la M-sequence di PN1
   kasami_mat(1, :) = PN1;

   % Aggiungo i rimanenti 2^(L/2) - 1 codici per un tot di 2^(L/2)
   for index = 1 : 2^(L/2) - 1

      % Faccio xor fra i 2^(L/2) - 1 elementi della s-sequence con i
      % 2^L - 1 della M-sequence. Dunque, effettuo 2^(L/2) + 1 passaggi
      for i = 0 : 2^(L/2)
         kasami = xor(PN1(1 + (2^(L/2) - 1)*i*Sc : (2^(L/2) - 1)*(i + 1)*Sc), PN2);
         k_sequence = [k_sequence kasami];
      end

      kasami_mat(index + 1, :) = k_sequence; % Inserisco in kasami_mat
      kasami = []; % Libero kasami per la nuova iterazione
      k_sequence = []; % Libero k_sequence per la nuova iterazione

      % shift di un elemento della s-sequence
      temp = PN2(1:Sc);
      PN2 = [PN2(Sc + 1 : end) temp];
   end

end
```

```
% ------------------------------------------------------------------------
```

RX_DSSS_BPSK.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% RX_DSSS_BPSK

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione prende in ingresso il segnale codificato DSSS-BPSK e ne effettua la demodulazione per
% ottenere, a meno di errori, il segnale di partenza prima della modulazione.
% Per effettuare la demodulazione vengono usati quattro blocchi fondamentali; il rilevatore di inizio
% sequenza, il demodulatore BPSK (con integratore), il demodulatore DSSS e il decisore.

% ARGOMENTI DELLA FUNZIONE
% sig_sum: è il segnale trasmesso (viene chiamato sig_sum perchè in genere può essere composto dalla
% somma di più segnali anch'essi trasmessi nello stesso mezzo.
% time: è la durata complessiva della trasmissione (comprende ritardi e preambolo).
% Sb: è il numero di campioni per bit.
% fs: è la frequenza di campionamento.
% Pn1: è la sequenza di chipping.
% preambolo_tx: è il preambolo del segnale da decodificare.
% C: è la portante usata per la modulazione BPSK.
% Nc: è il numero di chip nella M-sequence.
% sig_no_mod: è il segnale non modulato.

% DATI IN USCITA
% ds: è il segnale demodulato.
% Ber: bit error rate.
% demod_bit: è il segnale demodulato ma con i bit a singoli campioni.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
% ------------------------------------------------------------------------

function [ds, Ber, demod_bit] = RX_DSSS_BPSK(sig_sum, time, Sb, fs, Pn1, preambolo_tx, C, Nc,
sig_no_mod)

   % Creo l'asse dei tempi
   t = 0 : 1/fs : time - 1/fs;

   rx = sig_sum; % Segnale ricevuto

%% Rilevatore di inizio sequenza
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calcolo tramite la cross-correlazione lo sfasamento dovuto alla
% propagazione del segnale. Rilevo quando inizia il preambolo, max picco
% della cross-correlazione.
   [Rc, lag] = xcorr(rx, preambolo_tx);
   [max_val, lag_index] = max(Rc);
   lag = abs(lag(lag_index))

% Estraggo da rx solo la porzione di segnale
   rx = rx(lag + 1  : length(t) + lag);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DEMODULAZIONE BPSK
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Moltiplicazione per la portante
   rx1 = rx.*C;

%%Integrazione
   int_op = [];
   for i = 0 : Sb/Nc : length(rx1) - Sb/Nc
       % Effettuo l'integrazione negli intervalli degli impulsi di chip e
       % normalizzo per Nc/Sc
      int_o = (Nc/Sb)*trapz(rx1(i + 1 : i + Sb/Nc));
      int_op = [int_op int_o];
   end

   demod = sign(int_op); % riporto i valori a essere -1/1

% Riporto il segnale alla dimensione di partenza come numero di campioni
   demod1 = [];
   for i = 1 : length(demod)
      demod1 = [demod1 ones(1, Sb/Nc)*demod(i)];
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DEMODULATORE DSSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Effettuo il despreading del segnale
   ds = demod1.*Pn1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DECISORE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Il decisore per ogni Sb campioni ne fa la somma poi li media per decidere
%il valore corretto e lo porta a essere -1 o 1
   sig = [];
   for i = Sb : Sb : length(ds)
      sig = [sig (2*(sum(ds(i - (Sb - 1) : i))/Sb > 0) - 1) ];
   end

%Porto il segnale a avere le dimensioni di Sb campioni per ogni bit
   sig1 = [];
   for i = 1 : length(sig)
      sig1 = [sig1 sig(i)*ones(1, Sb)];
   end

   ds = sig1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%CALCOLO BIT ERROR RATE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   demod_bit = (sig+1)/2;
   diff = demod_bit - sig_no_mod;
   err = sum(abs(diff));
   Ber = (err/length(demod_bit))*100;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

```
% ------------------------------------------------------------------------
```

Auto_Cross_Correlation.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Auto_Cross_Correlation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione viene usata per il calcolo della autocorrelazione e della cross-correlazione per l'ana-
lisi % delle proprietà dei codici di kasami usati.

% ARGOMENTI DELLA FUNZIONE
% code1: M-sequence1.
% code2: M-sequence2.
% norm_fact: fattore di normalizzazione per la autocorrelazione e la cross-correlazione.

% DATI IN USCITA
% Nessun dato in uscita.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
% ------------------------------------------------------------------------

function [] = Auto_Cross_Correlation(code1, code2, norm_fact)

%% CALCOLO DELLA AUTOCORRELAZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  for j = 0 : length(code1)
    Ra1(j+1) = 1/norm_fact * sum(code1.*circshift(code1, j));
  end

  for j = 0 : length(code1)
    Ra2(j+1) = 1/norm_fact * sum(code2.*circshift(code2, j));
  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% CALCOLO DELLA CROSSCORRELAZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  for j = 0 : length(code2)
    Rc(j+1) = 1/norm_fact * sum(code2.*circshift(code1, j));
  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

% ------------------------------------------------------------------------
```

APPENDICE B

In questa sezione viene riportato la porzione di codice utilizzato per l'implementazione dell'algoritmo di Gold e successivamente quello per l'implementazione dell'algoritmo di Walsh. Nello specifico, per Gold verrà considerata la funzione *M_Seq_Gen*, mentre per Walsh sarà riportata la funzione *walsh_code*.

M_Seq_Gen.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% M_Seq_Gen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione genera i codici per la modulazione DSSS con la tecnica di Gold e mette i risultati
% ottenuti in una matrice dove ogni riga appresenta una sequenza.

% ARGOMENTI DELLA FUNZIONE
% Sb: numero di campioni per bit.
% L: Numero flip-flop.

% DATI IN USCITA
% Gold_mat: Matrice contenente i codici generati con Gold.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
% -----------------------------------------------------------------------

function [Gold_mat] = M_Seq_Gen(Sb, L)
% Dati
Nc = 2^L - 1; % lunghezza m-sequence
sr1 = zeros(1, L);
sr1(L) = 1;
sr2 = zeros(1, L);
sr2(L) = 1;

txb = [];
PN1 = [];
PN2 = [];
Sc = Sb / Nc; % Numero di campioni per ogni bit della sequenza
%fs = fs*Nt;
Gold = [];
sig = [];
Gold_mat = ones(Nc, Nc*Sc);
temp1 = sr1;
temp2 = sr2;

% Generazione dei codici gold sequence.
  for i = 1 : Nc
    PN1 = [PN1 temp1(6).*ones(1, Sc)];
    res1 = xor(temp1(5), temp1(6));
    res2 = xor(res1, temp1(2));
    res3 = xor(res2, temp1(1));
    temp1 = [res3 temp1(1 : L-1)]; % Realizzo lo shift e inserisco il rsultato dello xor
```

```
    PN2 = [PN2 temp2(6).*ones(1, Sc)];
    res4 = xor(temp2(1), temp2(6));
    temp2 = [res4 temp2(1 : L-1)]; % Realizzo lo shift e inserisco il risultato dello xor
  end

  Gold_mat(1, :) = PN1;
  Gold_mat(2, :) = PN2;

  for i = 0 : Nc - 1
    Gold = xor(PN1, PN2);
    temp = PN2(end - Sc + 1: end);
    PN2 = [temp PN2(1 : end - Sc)]; %shift di della m-sequence2
    Gold_mat(i + 3, :) = Gold; %inserimento del codice gold nella matrice
    Gold = [];
  end

end

% ------------------------------------------------------------------------
```

walsh_code.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% walsh_code

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% DESCRIZIONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Questa funzione genera i codici per la modulazione DSSS con la tecnica di Walsh

% ARGOMENTI DELLA FUNZIONE
% code_length: Numero di elementi del codice da realizzare (potenza di 2).
% ncode: Identificativo per la selezione del codice.

% DATI IN USCITA
% wcode; Codice selezionato con ncode
% wmat: Matrice contenente i codici generati con Walsh.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% INIZIO DEL CODICE
% ------------------------------------------------------------------------

function [wcode, wmat] = walsh_code(code_length, ncode)

if code_length == 0
  wmat = 1;
else
  wmat = [1 1 ; 1 -1]; %Inizializzazione della matrice di Walsh
  number_codes = log2(code_length)-1;

  for i = 1 : number_codes
    wmat = [wmat wmat ; wmat -wmat];
  end
end

wcode = wmat(ncode, :);

end
```

```
% ----------------------------------------------------------------
```

RIFERIMENTI

[1] ULTRASOUND COMMUNICATION SYSTEM FOR METAL STRUCTURE AND RELATED METHODS (Tomlinson, Jr. et al. - 2010) available at https://patents.google.com/patent/US7654148B2/en

[2] Power and energy transduction analysis of piezoelectric wafer-active sensors for structural health monitoring (Bin Lin, Victor Giurgiutiu - 2012) available at https://www.researchgate.net/publication/266672645_Power_and_energy_transduction_analysis_of_piezoelectric_wafer-active_sensors_for_structural_health_monitoring

[3] Identification of Damage Using Lamb Waves From Fundamentals to Application (Zongqing Su, Lin Ye - 2009)

[4] Lamb-Wave Based Structural Health Monitoring in Polymer Composites (Rolf Lammering Ulrich Gabbert Michael Sinapius Thomas Schuster - 2017)

[5] Rayleigh and Lamb Waves: Physical Theory and Applications (Viktorov, I.A. - 1967)

[6] Lamb Waves available at https://en.wikipedia.org/wiki/Lamb_waves

[7] Complete course on wireless Available at https://www.slideshare.net/blackdevilvikas/complete-course-on-wireless

[8] Spread Spectrum introduction (Jan De Nayerlaan - 1999)

[9] Evolution of 802.11 (physical layer) (Denis Bakin - 2007)

[10] Multipath effect on radar detection of nonfluctuating targets (Y. Jang, Hyeongyong Lim, D. Yoon 2015) available at https://www.semanticscholar.org/paper/Multipath-effect-on-radar-detection-of-targets-Jang-Lim/a14ed63ff855935c2dd95b56cd784e5d1a33078e

[11] A Highly-Effective Parallelization of Statistical Time-Consuming Tests of Pseudorandom Number Generators Using CUDA (Muhammad Osama, Aziza Hussein - 2015) available at https://www.researchgate.net/publication/292130031_A_Highly-Effective_Parallelization_of_Statistical_Time-Consuming_Tests_of_Pseudorandom_Number_Generators_Using_CUDA

[12] Gold code generator using LFSRs (Mathuranathan - 2015) available at https://www.gaussianwaves.com/2015/06/gold-code-generator/

[13] Wireless Measurement System for Structural Health Monitoring With High Time-Synchronization Accuracy (Alvaro Araujo, Jaime García-Palacios, Javier Blesa, Francisco Tirado, Elena Romero, Avelino Samartín, and Octavio Nieto-Taladriz - 2012) available at https://ieeexplore.ieee.org/document/6061956

[14] Application and Challenges of Signal Processing Techniques for Lamb Waves Structural Integrity Evaluation: Part A-Lamb Waves Signals Emitting and Optimization Techniques (Zenghua Liu and Honglei Chen – 2018)

[15] Difference between FDMA, TDMA and CDMA (2020) available at https://www.geeksforgeeks.org/difference-between-fdma-tdma-and-cdma/

[16] SPREAD SPECTRUM PROCESS USING DIRECT SEQUENCE SPREAD SPECTRUM (DSSS) AND FREQUENCY HOPPING SPREAD SPECTRUM (FHSS) (Handrizal Tanjung, Noraziah Binti Ahmad, Ahmed N Abdalla, Alla – 2009) available at https://www.researchgate.net/publication/268009687_SPREAD_SPECTRUM_PROCESS_USING_DIRECT_SEQUENCE_SPREAD_SPECTRUM_DSSS_AND_FREQUENCY_HOPPING_SPREAD_SPECTRUM_FHSS

[17] Nucleo64 STM32f446re available at https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html#products

[18] STM32F446-ARM Nucleo Board User's Manual D.K. Blandford August, 2017 available at https://docplayer.net/63398506-Stm32f446-arm-nucleo-board-user-s-manual-d-k-blandford-august-2017.html

RINGRAZIAMENTI

Voglio ringraziare in primo luogo il professore e relatore Luca De Marchi e la correlatrice Federica Zonzini per avermi dato la possibilità di lavorare con loro a questo progetto di tesi e per avermi supportato durante l'intero percorso dimostrandosi sempre molto disponibili e comprensivi nei miei confronti. Il fatto di avere affrontato una tematica così attuale e importante mi rende orgoglioso del lavoro svolto e per questo gli sono grato.

Ringrazio la mia famiglia per essermi stata vicino durante il mio cammino verso la laurea magistrale in ingegneria elettronica e per avere sempre creduto in me anche nei momenti più difficili. Grazie di cuore per avermi supportato e sopportato per tutti questi anni.

Ringrazio, infine, tutti i miei compagni di corso dell'università di Bologna che hanno reso tutte le mie giornate all'università molto più belle grazie alla loro compagnia.