POLITECNICO DI TORINO

CORSO DI LAUREA

INGEGNERIA FISICA

ELABORATO FINALE LAUREA TRIENNALE

Realizzazione di un ricevitore di segnali Wireless ADS-B per il controllo e monitoraggio del traffico aereo attraverso la programmazione di un dispositivo RTL-SDR e MATLAB

CANDIDATO
Lorenzo Maria Scarrone

RELATORE
Prof. Andrea Carena

Anno accademico 2017-2018

Lorenzo Maria Scarrone matricola:213692.
Elaborato prova finale 3° anno di Ingegneria Fisica

Decodificare segnali wireless utilizzando un ricevitore RTL-SDR e Matlab

Indice degli argomenti trattati:

Lorenzo Maria Scarrone matricola:213692.
Elaborato prova finale 3° anno di Ingegneria Fisica

Introduzione:

Automatic Dependent Surveillance - Broadcast è una tecnica operativa di controllo del traffico aereo utilizzata per l'identificazione di aeromobili al fine di gestire il traffico aeroportuale e di evitare collisioni tra velivoli in volo (tecnica affiancata ai sistemi radar). Ogni velivolo dotato di un transponder ***Mode-S*** qualora sia collegato a un dispositivo *GPS,* può inviare periodicamente in *broadcast* (a tutti gli "ascoltatori") un segnale contenente informazioni sul nome del velivolo, posizione, altitudine, velocità e altri dati utili a chi si dovesse trovare a bordo di un altro velivolo in volo. Poichè la comunicazione avviene in *broadcast,* non esiste un target specifico e quindi, chiunque dotato di un ricevitore ADS-B è in grado di monitorare il traffico aereo.

I segnali che si andranno a decodificare seguono lo standard Mode-S che è uno standard di comunicazione aderente alle normative ed alle raccomandazioni dell'Organizzazione Internazionale Aeronautica Civile (ICAO - *International Civil Aviation Organization*) e segue una specifica codifica basata su una modulazione PPM.

Il metodo di segnalazione Mode-S usa messaggi di tipo *squitter* che sono definiti come messaggi non richiesti: il dispositivo emittente non aspetta una richiesta per segnalare la propria posizione ma la comunica comunque a chiunque sia in ascolto.

Mode-S ha le seguenti proprietà:

- Transmit Frequency: 1090 MHz
- Modulation: Pulse Position Modulation
- Data Rate (R_s): 1 Mbit/s
- Short Squitter Length: 56 microseconds
- Extended Squitter Length: 112 microseconds

I messaggi di tipo short squitter contengono le seguenti informazioni:

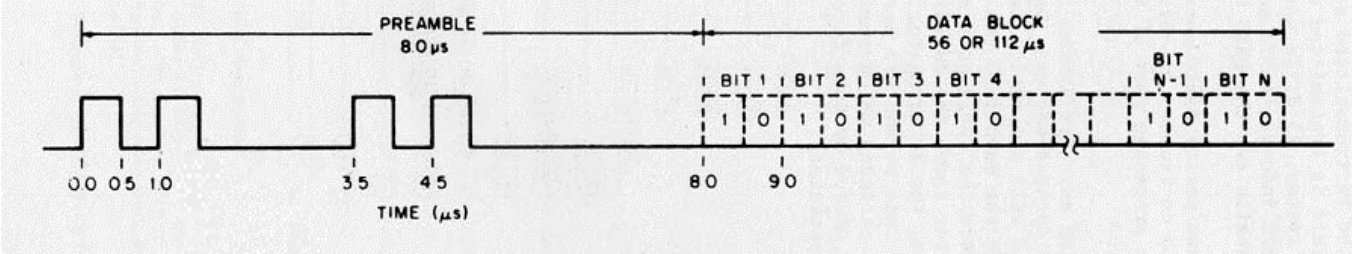
- Downlink Format (DF)
- Capability (CA)
- Aircraft ID (Unique 24-bit sequence)
- CRC Checksum

I messaggi *Extended squitter* (ADS-B) contengono tutte le informazioni contenute nei messaggi di tipo *short squitter* e a seconda del valore di un particolare parametro può contenere alcune di queste informazioni:

- Altitude
- Position
- Heading
- Horizontal and Vertical Velocity

I segnali nel formato Mode-S hanno una sequenza di allineamento di 8 microsecondi e a seconda che siano *Extended* o *short*, una lunghezza di 112 o 56 microsecondi come illustrato di seguito:

- **preambolo** (*durata* $8\ \mu s$) ed è formato da quattro impulsi P1,P2,P3,P4 di $500\ ns$ ciascuno. In particolare per gli impulsi da P1 a P2 e da P3 a P4 la distanza è di 1 μs, mentre tra P2 e P3 ci sono 2 μs.
- **campo dati** (*durata* $56\ \mu s$ o $112\ \mu s$) La durata è legata al numero di bit del payload che può essere di 56(*short)* o di 112bit (*extended)* ogni bit dura $1\ \mu s$. Si utilizza una modulazione PPM dove ogni bit viene letto come 1 logico se il segnale è presente per la prima metà del periodo ($500\ ns$) oppure 0 se presente nella seconda metà.



Acquisizione del segnale

Per realizzare il ricevitore si utilizza un dispositivo RTL-SDR reciver con la frequenza centrale impostata a 1090MHz.

La durata della finestra di acquisizione è stata configurata in modo da fornire abbastanza campioni da contenere n=180 messaggi squitter extended. Impostando una frequenza di campionamento a $2.4\ \mathrm{MHz}$ un messaggio Extended ha una lunghezza di 288 campioni quindi il numero di campioni che devono essere contenuti in ogni frame (SamplesPerFrame) per garantire n deve essere di 51840.

$$SamplesPerFrame = n \cdot \frac{TimeExtendedSquitter}{Tsampling} = 180 \cdot \frac{120\ \mu s}{\frac{1}{2.4\ \mathrm{MHz}}} = 51840\ \mathrm{Samples}$$

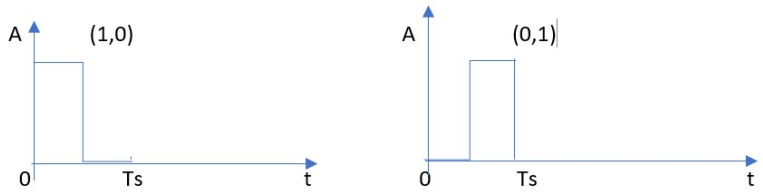
Una volta acquisito il segnale a causa della frequenza di campionamento di 2.4MHz il numero di campioni per simbolo è di 2.4 SpS; per semplificare il problema si sovra-campiona il segnale mediante un filtro di interpolazione FIR che permette di avere un segnale in uscita come se fosse stato campionato a $F_c = n \cdot f_c$ dove n nel nostro caso è pari a 5. In questo modo è come se il segnale fosse stato campionato a 12 MHz anziché a 2.4 MHz portando ad avere 12 SpS (numero intero molto più semplice da gestire):

$$SpS_{new} = \frac{n \cdot f_{sampling}}{R_s} = \frac{5 \cdot 2{,}4\ \mathrm{MHz}}{1\ \mathrm{Mbit/s}} = 12\ \mathrm{SpS}$$

Lorenzo Maria Scarrone matricola:213692.
Elaborato prova finale 3° anno di Ingegneria Fisica

Introduzione alla codifica PPM

Sapendo che il segnale ricevuto è modulato in 2-PPM ed è unipolare si hanno due segnali rispettivamente per 1 e per 0. Nella modulazione 2-PPM l'impulso ha una durata di solo mezzo periodo (Ts) del simbolo.



Modulazione PPM

Osservando i grafici riportati sopra si osserva che per la configurazione unipolare del PPM esistono due configurazioni possibili: la prima indicata con (1,0) rappresenta l'uno logico, la seconda (0,1) rappresenta lo zero logico.

Le due funzioni riportate nella notazione (1,0) e (0,1) rappresentano due funzioni porta, dove la virgola indica la metà del periodo e l'1 rispettivamente la parte alta dell'impulso e lo 0 la parte bassa o nulla. Matematicamente possono essere riportate come (dove $u(t)$ è la funzione gradino):

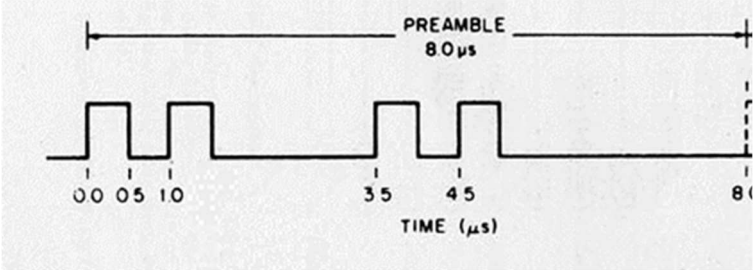
$$(1,0) = u(t) - u\left(t - \frac{T_s}{2}\right)$$
$$(0,1) = u\left(t - \frac{T_s}{2}\right) - u(t - T_s)$$

Analisi del segnale e riconoscimento del pacchetto dati ADS-B

Essendo un segnale unipolare è bene considerare solo il valore assoluto e per non avere problemi con l'ampiezza del segnale è bene normalizzarlo rispetto al massimo assoluto

```
rxAbs = abs(rxFil)/max(abs(rxFil));
```

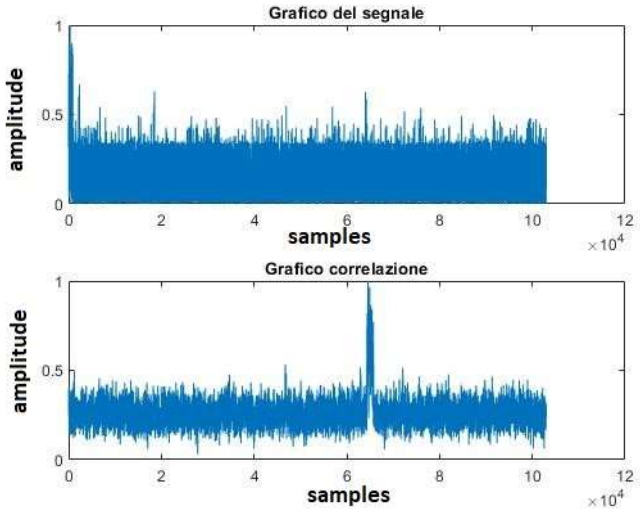
A questo punto è possibile procedere con la ricerca della sequenza di allineamento dei pacchetti: il preambolo. Quest'ultimo è definito da 4 impulsi PPM separati da un intervallo di tempo ben definito



I primi due impulsi hanno durata rispettivamente $1\ \mu s$ e dal punto di vista logico corrispondono a due 1. Tra i primi due impulsi e gli ultimi due abbiamo un silenzio rappresentato da un'assenza di segnale, la durata di questo silenzio è di $1\ \mu s$. Infine sono presenti gli ultimi due impulsi PPM interpretabili come 0 logico la cui durata complessiva è di $2\ \mu s$. Gli ultimi $3\ \mu s$ sono di nuovo legati a un'assenza di segnale.
Dopo il passaggio del segnale all'interno del filtro FIR tutti i simboli PPM sono rappresentati su 12 (SpS) campioni e i bit di silenzio su 6 (SpS/6). Questo porta a scrivere il preambolo nel seguente modo:

```
adsbParam.preamble = [ones(1,6) zeros(1,6) ones(1,6) zeros(1,6) ...
                      zeros(1,18) ones(1,6) zeros(1,6) ...
                      ones(1,6) zeros(1,36)];
```

La ricerca del preambolo all'interno del frame viene realizzata mediante un'operazione di *cross-correlation*. Mediante la quale vengono evidenziati nel frame i picchi di correlazione che evidenziano i punti in cui quella sequenza è presente.



Dal grafico si può osservare che il segnale è stato normalizzato, questo perché l'ampiezza del segnale ricevuto può variare a seconda dell'intensità del segnale ricevuto. In questo modo qualunque segnale può essere descritto tra 0 e 1 permettendo anche di definire delle soglie fisse per realizzare dei controlli in modo ripetibile.

Nell'istante in cui si verifica l'evento di correlazione (ovvero si ha una corrispondenza tra segnale e preambolo) i picchi di correlazione sono più ampi rispetto ad altri in cui la correlazione ha avuto esito negativo.

Dal punto di vista teorico l'operazione di cross-correlazione

$$R_{XY}(t) = (X * Y)(t) = \int_{-\infty}^{\infty} X^*(\tau)Y(t+\tau)d\tau$$

dovrebbe fornire un output in cui dovrebbero potersi osservare in modo chiaro gli eventi di correlazione legati al confronto tra il segnale e la sequenza cercata.

In pratica la forte presenza di rumore di fondo rende difficile l'identificazione grafica dei picchi di correlazione, quindi l'output grafico non permette di osservare fisicamente gli eventi di correlazione.

Attraverso l'utilizzo della funzione `xcorr` fornita da MatLab è possibile registrare gli eventi di correlazione all'interno della variabile `Lag`.

```
[Xcor,Lag] = xcorr(SyncSeq,Signal);
```

Poiché solo gli eventi di correlazione di maggior intensità potrebbero identificare l'inizio di un pacchetto ADS-B definisco una soglia pari a 0.8 che è abbastanza alta da escludere eventi di correlazione casuali legati a sequenze simili all'interno del segnale causate dalla sovrapposizione con il rumore, ma non troppo per evitare di perdere troppi pacchetti.

```
[~,dl] = findpeaks(Xcor/max(Xcor),Lag,'MinPeakHeight',0.8);
```

La funzione `findpeaks` permette di confrontare gli eventi di correlazione registrati nella variabile `Lag` con la soglia impostata (`'MinPeakHeight'=0.8`) e registra in `dl` la posizione del campione che verifica la condizione sull' ampiezza del picco.

Identificate le posizioni in cui si potrebbe trovare un pacchetto ADS-B si procede con l'estrazione dei dati. La posizione salvata nella variabile `dl` corrisponde all'inizio del preambolo del presunto pacchetto ADS-B.

Per poter estrarre i dati bisogna saltare i primi 96 dalla posizione di inizio della sequenza registrata nella variabile `dl`, dove 96 è il numero di campioni contenuti nel preambolo.

L'insieme di queste operazioni viene realizzato nella funzione `CorrelateAndSync` la quale restituisce i pacchetti estratti dal *frame* ed eventualmente un errore se la posizione considerata è in corrispondenza della fine della finestra di acquisizione, ovvero impedisce al sistema di cercare di leggere un dato non esistente.
La finestra di acquisizione è di 51840 che dopo essere passati per il filtro FIR è diventata di $51840 \cdot 12\ SpS$, se la posizione $dl + 96$(preambolo) $+ 120$(bit di dati) $\cdot 12\ SpS > 51840 \cdot 12\ SpS$ allora il dato viene scartato e la funzione restituisce errore in corrispondenza di quel pacchetto.

Demodulazione di segnali PPM

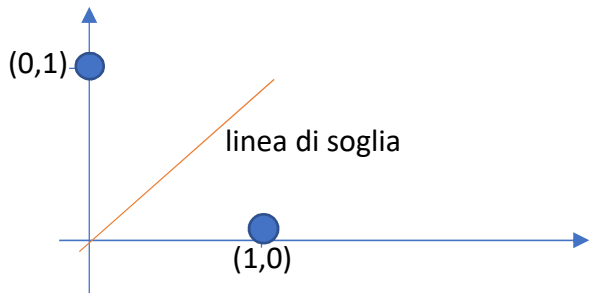
Una volta ottenuti i presunti pacchetti e verificato che non siano fisicamente danneggiati, cioè che la funzione abbia verificato che il set di bit del pacchetto sia completo, ovvero che non sia stato tagliato dalla finestra di acquisizione, si procede alla demodulazione del segnale 2-PPM.

I segnali 2-PPM a differenza di altri segnali in banda-base invece di essere unidimensionali sono bidimensionali nello spazio dei segnali poiché la base di questo spazio è composta da due differenti segnali per rappresentare i bit 1 e 0 che quindi sono rappresentati da due differenti versori.

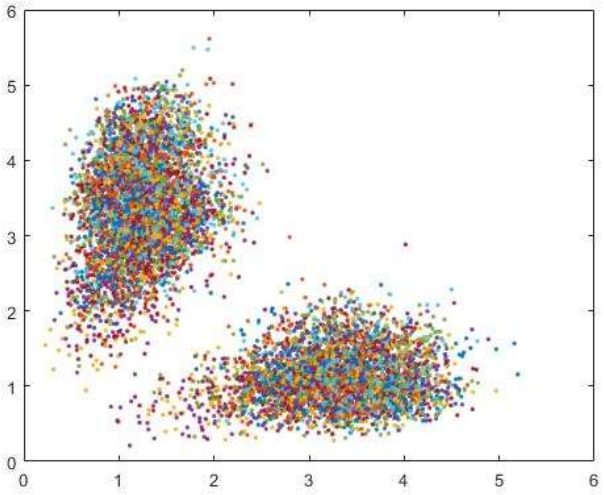
Questo comporta, nel caso della demodulazione, a dover scegliere una soglia in un piano bidimensionale.

Lorenzo Maria Scarrone matricola:213692.
Elaborato prova finale 3° anno di Ingegneria Fisica

Di seguito il grafico teorico in cui si vanno a disporre le configurazioni dei segnali (1,0) e (0,1) nello spazio bidimensionale:



Il grafico a dispersione (scattering) dei punti che identificano i due bit 1 e 0 devono essere delle "nuvole" di punti ben distinte dalle quali è possibile distinguere attraverso una retta usata come soglia i due differenti bit. Di seguito verrà mostrato il grafico a dispersione (scattering) di un segnale 2-PPM ADSB che ha passato il controllo di integrità del pacchetto (CRC).



Dal grafico è possibile definire una soglia identificata da una retta in uno spazio 2D per identificare rispettivamente i bit 1 e 0. È possibile proiettare in uno spazio 1D quanto appena visto.
Supponendo di poter esprimere i vettori della base bidimensionale in 3 dimensioni, i vettori o versori (in quanto il loro modulo è unitario e costituiscono una base) diventano:

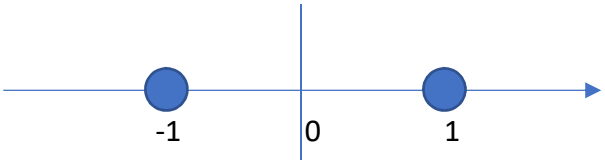
$$\overrightarrow{a_1} = (1,0,0) \text{ e } \overrightarrow{a_2} = (0,1,0)$$

Adesso procedo proiettando il vettore $\overrightarrow{a_2}$ su $\overrightarrow{a_1}$ attraverso un terzo vettore $\overrightarrow{a_3} = (0,0,1)$:

$$\overrightarrow{a_2}' = \overrightarrow{a_3} \times \overrightarrow{a_2} = det \begin{bmatrix} \vec{u} & \vec{v} & \vec{k} \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = -1 \cdot \vec{u} = (-1,0,0)$$

A questo punto i due versori che identificano i bit 1 e 0 si trovano nello stesso spazio unidimensionale, ma sono identificati rispettivamente da due diversi simboli 1 per 1 logico e -1 per lo 0 logico.

La configurazione nuova che verrà descritta nella notazione (1,-1) è rappresentabile sul piano in modo schematico come segue:



Il numero di bit contenuti nella sezione dati di ogni pacchetto ADS-B è 112. Sapendo che un pacchetto ADS-B ha una durata di $120\ \mu s$ e che il preambolo ha una durata di $8\ \mu s$ e che ogni bit ha la durata di $1\ \mu s$:

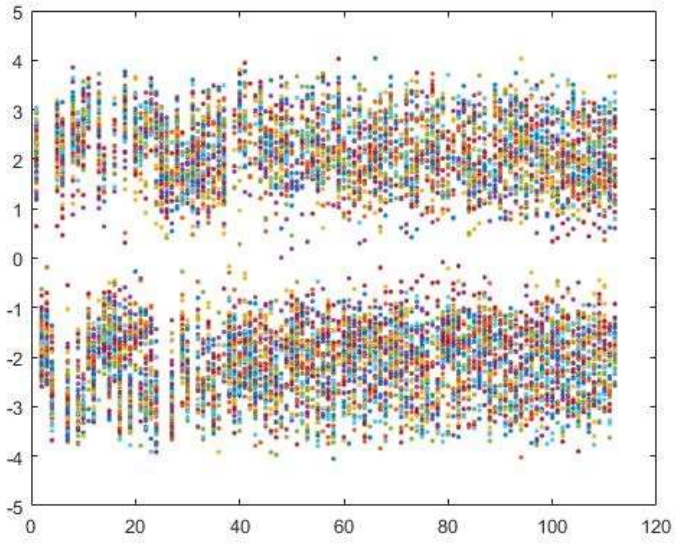
$$NumBitDati = \frac{(120-8)\mu s}{1\ \mu s} = 112\ \text{bit}$$

È noto che il payload del pacchetto, se si tratta di un pacchetto ADS-B extended, ha una lunghezza complessiva di 112 bit, ogni bit a causa del passaggio del segnale attraverso il filtro FIR ha un equivalente di SpS(12) campioni, ovvero i campioni totali del payload sono 1344.
Si dispone il vettore contenente l'informazione da demodulare (y) in una matrice con SpS colonne e 112 righe dove ogni riga può presentare la configurazione 2-PPM (1,0) o (0,1).

Moltiplicando la matrice che rappresenta i dati nello spazio a due dimensioni per il vettore (1,-1), si ottiene un vettore monodimensionale che descrive il segnale. Trattandosi di prodotto tra matrici $[112 \times 12] \cdot [12\ \times 1] = [112\ \times 1]$ il vettore risultante è monodimensionale.

$$\begin{bmatrix} y_1 & \cdots & y_{12} \\ \vdots & \ddots & \vdots \\ y_{1332} & \cdots & y_{1344} \end{bmatrix} \cdot [1 \dots 1 - 1 \dots - 1]$$

Di seguito è riportato il grafico di scattering dei dati dopo il prodotto per la nuova base, ora si è in grado di determinare la soglia che divide gli 1 dagli 0 logici.



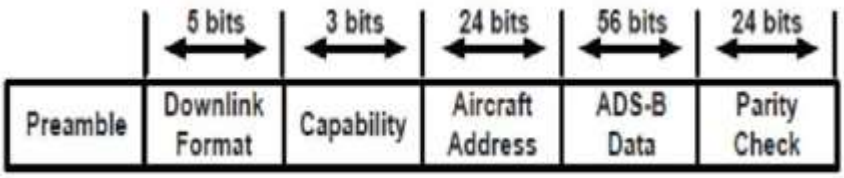
Dove la soglia risulta essere a 0. Confrontando il vettore del segnale con la soglia, ricavata dal grafico, è possibile ricostruire il vettore dei bit.

L'insieme di queste operazioni è realizzato dalla funzione `adsbDemod` la quale accetta in ingresso il segnale contenente il pacchetto dati da demodulare e restituisce in uscita il vettore di bit corrispondente.

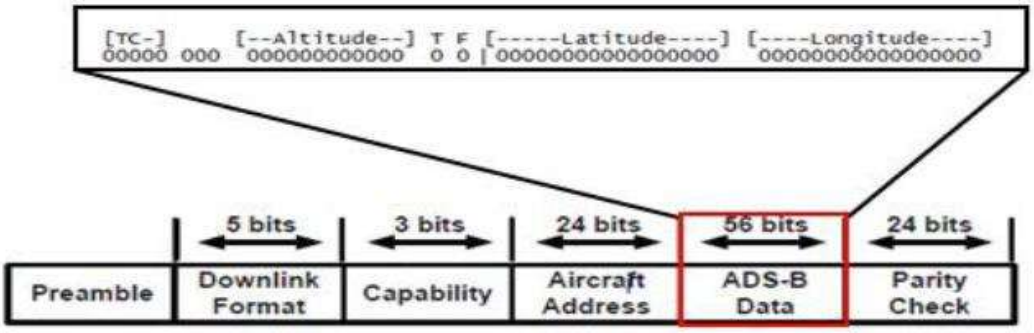
Decodifica del pacchetto dati ADS-B

Una volta ottenuto il vettore di bit è necessario, prima di procedere con la decodifica dei dati, e verificare l'integrità del pacchetto. Quest'operazione viene eseguita mediante il CRC check che consiste nell'eseguire uno XOR tra una sequenza nota (`generator`) e i bit del vettore contenente i dati. Se il controllo fornisce un risultato privo di errori, il pacchetto è valido altrimenti restituisce un errore e il pacchetto viene scartato. La funzione `CRCcheck` realizza quest'operazione.

Passato il controllo d'integrità si procedere con la decodifica del pacchetto. Un pacchetto ADS-B è così composto:



Dove il campo DF (Dowlink format) indica il tipo di pacchetto ricevuto. Se DF = 11 allora si tratta di un pacchetto di tipo short squitter e non contiene il campo dati, mentre se DF=17 siamo in presenza di un pacchetto di tipo extended. Dal campo CA (Capability) si può ricavare il modello del trasmettitore e quindi anche risalire alla tipologia di aereo a cui è associato il dispositivo (Civile, Militare, Carico,ecc …)
L'Aircraft Address è un codice univoco per ogni aereo assegnato dall'ICAO.
Il campo dati è ulteriormente suddiviso in altre sezioni



TC o anche Type Code indica il contenuto del pacchetto, se è in un valore compreso tra 9 e 18 indica che il contenuto del pacchetto è quello mostrato nella figura sovrastante e nella tabella seguente:

Bits	N-bit	Abbr	Content
1 - 5	5	DF	Downlink format
33 - 37	5	TC	Type code
38 - 39	2	SS	Surveillance status
40	1	NICsb	NIC supplement-B
41 - 52	12	ALT	Altitude
53	1	T	Time
54	1	F	CPR odd/even frame flag
55 - 71	17	LAT-CPR	Latitude in CPR format
72 - 88	17	LON-CPR	Longitude in CPR format

se invece TC=19 il pacchetto contiene informazioni relative alla velocità, alla direzione, altitudine barometrica. I dati del contenuto sono riportati di seguito in una tabella con riferimento anche alla posizione nel vettore di bit contenente i dati:

MSG Bits	Data Bits	Len	Abbr	Content
33 - 37	1 - 5	5	TC	Type code
38 - 40	6 - 8	3	ST	Subtype
41	9	1	IC	Intent change flag
42	10	1	RESV_A	Reserved-A
43 - 45	11 - 13	3	NAC	Velocity uncertainty (NAC)
46	14	1	S_ew	East-West velocity sign
47 - 56	15 - 24	10	V_ew	East-West velocity
57	25	1	S_ns	North-South velocity sign
58 - 67	26 - 35	10	V_ns	North-South velocity
68	36	1	VrSrc	Vertical rate source
69	37	1	S_vr	Vertical rate sign
70 - 78	38 - 46	9	Vr	Vertical rate
79 - 80	47 - 48	2	RESV_B	Reserved-B
81	49	1	S_Dif	Diff from baro alt, sign
82 - 88	50 - 66	7	Dif	Diff from baro alt

Nel caso esaminato il programma analizza solo la prima tipologia di pacchetti ADS-B quelli che portano l'informazione della posizione e altitudine.
L'analisi del pacchetto non è complicata in quanto basta suddividere il pacchetto in sezioni seguendo le informazioni fornite dalla tabella. È importate sottolineare che i bit, almeno nel primo caso, devono essere tutti letti tenendo conto che l'MSB si trova sempre sulla sinistra della sequenza considerata.

Di seguito l'esempio di un pacchetto ADS-B codificato in esadecimale.

ADS-B packet: 8D A05ADF 991988A2E88C38 EA155D

Di seguito viene illustrato il processo mediante il quale avviene la decodifica delle informazioni contenute nella sezione DATA del pacchetto. Riconsideriamo la struttura base del pacchetto:

[TC] [000] [ALTITUDE] [T] [F] [LONGITUDE] [LATITUDE]

Per ridurre l'incertezza della misura di posizione è necessario utilizzare due pacchetti ADS-B, attraverso una media. Distinguere due pacchetti ADS-B è possibile grazie al bit F detto anche flag

bit, il quale permette di definire due tipi di pacchetti uno pari (even) con F a 0 e uno dispari (odd) con F a 1. In questo modo, se i due pacchetti sono vicini temporalmente è possibile mediare la posizione e ridurre la sua incertezza.
La posizione viene riportata all'interno del pacchetto in un formato compatto detto CPR (Compact position reporting) che permette di comprimere l'informazione occupando meno bit.
Di seguito sono riportate le equazioni e le funzioni utilizzate per estrarre la latitudine e la longitudine da due pacchetti ADS-B (even) pari e (odd) dispari:

$$Lat_{cprEven} = (bit_{latitudeEven})_{10}/131072$$
$$Lat_{cprOdd} = (bit_{latitudeOdd})_{10}/131072$$

NZ è una costante che rappresenta il numero di zone geografiche tra l'equatore e un polo. Per la codifica CPR Mode-S NZ=15.

La funzione NL(lat) determina il "numero di zone di longitudine" data una certa latitudine e restituisce un intero compreso tra 1 e 59

$$NL(lat) = \text{floor}\left(\frac{2\pi}{\cos^{-1}\left(1 - \left(\frac{1-\cos\left(\frac{\pi}{2\cdot NZ}\right)}{\cos^2\left(\frac{\pi}{180}\cdot lat\right)}\right)\right)}\right)$$

L'indice di latitudine:

$$j = floor\left(59 \cdot Lat_{cprEven} - 60 \cdot Lat_{cprOdd} + \frac{1}{2}\right)$$

Poi si considerano rispettivamente le due costanti

$$dLat_{Even} = \frac{360}{4 \cdot NZ}$$
$$dLat_{Odd} = \frac{360}{4 \cdot NZ - 1}$$

A questo punto si procede con il calcolo della latitudine

$$Lat_{Even} = dLat_{Even} \cdot [mod(j, 60) + Lat_{cprEven}]$$
$$Lat_{Odd} = dLat_{Odd} \cdot [mod(j, 59) + Lat_{cprOdd}]$$

A questo punto si deve assicurare che il valore sia compreso nel range [-90, +90]

$$Lat_{Odd} = Lat_{Odd} - 360 \rightarrow se\ Lat_{Odd} \geq 270$$
$$Lat_{Even} = Lat_{Even} - 360 \rightarrow se\ Lat_{Even} \geq 270$$

Dopo di che si dovrebbe verificare quando sono arrivati i pacchetti e quale sia il più recente, per come è stato fatto il programma il pacchetto con il flag Odd è quello più recente quindi si utilizza la latitudine calcolata da quel pacchetto.

Per il calcolo della longitudine si considera il pacchetto scelto per determinare la latitudine nel caso in esame quindi Odd:

$$ni = \max(NL(Lat_{Odd}), 1)$$

Lorenzo Maria Scarrone matricola:213692.
Elaborato prova finale 3° anno di Ingegneria Fisica

$$dLon = \frac{360}{ni}$$

$$m = floor\left\{ Lon_{cprOdd} \cdot [NL(Lat_{Odd}) - 1] - Lon_{cprOdd} \cdot NL(Lat_{Odd}) + \frac{1}{2} \right\}$$

$$Lon = dLon \cdot \left(mod(m, ni) + Lon_{cprOdd}\right)$$

Se il risultato è maggiore di 180 gradi

$$Lon = Lon - 360$$

Infine resta da calcolare l'altitudine. Quest'ultima è codificata nel modo seguente:

```
1100001 1 1000
        ^
      Q-bit
```

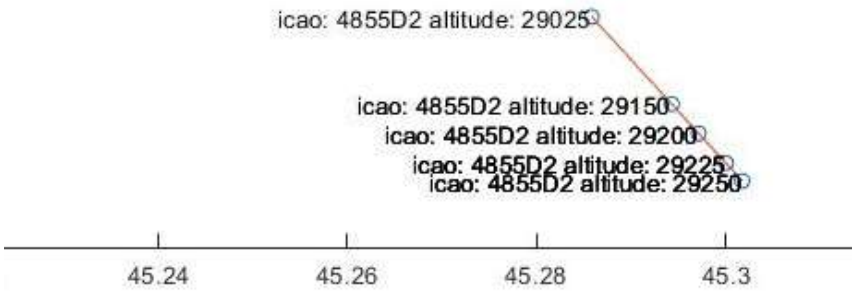
Dove il Q-bit indica che l'altitudine è codificata su multipli di 25 o 100 piedi rispettivamente per Q-bit = 1 o Q-bit =0. Dunque si considera il vettore senza il Q-bit e si procede come segue

$$mul = \begin{cases} 100 \,, Q_{bit} = 0 \\ 25 \quad , Q_{bit} = 1 \end{cases}$$

$$altitude = (bits_{altitude})_{10} \cdot mul(Q_{bit}) - 1000$$

Di seguito alcuni risultati ottenuti dopo una cattura:

	ICAO	Latitude	Longitude	Altitude
1	'4D00F3'	45.1689	5.6742	38000
2	'4D00F3'	45.1749	5.6696	38000
3	'4D00F3'	45.1795	5.6661	38000
4	'4855D2'	45.2858	5.6290	29025



È possibile verificare l'ICAO dell'aereo presso il sito: https://flightradar24.com e monitorare i dati in tempo reale.

Di seguito è possibile consultare il codice MATLAB:

```
%Tesi - Laurea Triennale ingegneria Fisica
%
%Titolo: Decoding Live Wireless Signals with MATLAB and RTL-SDR
%
%Scritto da Lorenzo Maria Scarrone
%
%Corso di riferimento a scelta
%         INTERNET E COMUNICAZIONI
%
%referente: prof. Andrea Carena


%CARICO LA CONFIGURAZIONE PER IL RICEVITORE ADSB
adsbParam = ConfigAdsbReciverAndStandardParam();
dataCursor = 1;

%SCANSIONE PACCHETTI ADSB (ETERE)
%ADSBrx = RTLreciver(adsbParam);

%CARICO PACCHETTO DATI PREACQUISITO (offline)
%load('acquisizionisalvate/rxADSB_3aerei.mat');

for cur=1:adsbParam.NBlocks
    %sovracampionamento del segnale ricevuto: ottengo SpS intero
    % 2.4MHz x 5 = 12MHz --> 12MHz/(1Mb/s) = 12 SpS
    rxFil = adsbInterpolator(ADSBrx(cur,:)',adsbParam);
    %normalizzo il segnale interpolato
    rxAbs = abs(rxFil)/max(abs(rxFil));

    %ottengo i pacchetti dati sincronizzando il flusso dati con il preamble
    [adsbPacket,errors] = CorrelateAndSync( rxAbs ,adsbParam);

    %ANALISI DEI PACCHETTI ESTRATTI
    for ii=1:size(adsbPacket,1)
        %VERIFICO L'ESISTENZA DEL PACCHETO
        if errors(ii,1) ~= 1
            %SI PROCEDE CON LA DEMODULAZIONE DEL PACCHETTO
            adsbBits= adsbDemod(adsbPacket(ii,:)',adsbParam);
            %VERIFICA DELLA VALIDITA' DEL PACCHETTO
            invalidPacket = CRCcheck(adsbBits);
            if invalidPacket == false
                %SUDDIVISIONE DEL PACCHETTO PER LA DECODIFICA
                ADSBdata(dataCursor) = decodeData(adsbBits);
                dataCursor = dataCursor+1;
            end
        end
    end
end
%DECODIFICA DEI PACCHETTI ESTRATTI
if exist('ADSBdata')
    DecodedData = decodeHexData(ADSBdata);
    struct2table(DecodedData)
    plotData(DecodedData);
else
    warning('Nessun pacchetto ADS-B rilevato')
```

```
end

%----------------------------. . . . . . . . . ----------------------------
%------------------------- . . .  FUNZIONI . . . -------------------------
%----------------------------. . . . . . . . . ----------------------------

function plotData(decData)
    cellData = struct2cell(decData);
    ICAO_list = unique(cellData(1,1,:));

    figure();
    for i=1:length(ICAO_list)
        n=1;
        for j=1:length(cellData)
            if isequal(ICAO_list(i),cellData(1,1,j))
                x(n) = cellData(2,1,j);
                y(n) = cellData(3,1,j);
                h(n) = cellData(4,1,j);
                n=n+1;
            end
        end
        x1=cell2mat(x);y1=cell2mat(y);
        plot(x1,y1,'-o');
        hold on
        plot(x1,y1);
        for k=1:length(x1)
            txt = "icao: "+cell2mat(ICAO_list(1))+" altitude: "+num2str(cell2mat(h↙
(k)));
            text(x1(k),y1(k),txt,'HorizontalAlignment','right');
        end
        hold on
        clear x y
    end
end

%FUNZIONE PER ESTRARRE I PACCHETTI DA FRAME ACQUISITO
function [sync,error] = CorrelateAndSync(Signal, param)
    SyncSeq = param.preamble; %CARICO LA SYNC SEQUENCE
    SpS = param.SpS;
    %CALCOLO LA CORRELAZIONE PER OTTENERE I PICCHI DA CUI ESTRARRE I DATI
    [Xcor,Lag] = xcorr(SyncSeq,Signal);
    %RICERCA DEI PICCHI DI CORRELAZIONE CON IL TEST DI SOGLIA
    [~,dl] = findpeaks(Xcor/max(Xcor),Lag,'MinPeakHeight',0.8);
    %SALVO LA POSIZIONE DEI PICCHI NEL FRAME
    Pos = abs(dl);
    %RICERCA DEI PACCHETTI
    for ii=1:length(Pos)
        start_pos = Pos(ii)+length(SyncSeq)+1;
        error(ii,1)=0;
        %VERIFICO CHE LA DIMENSIONE DEL PACCHETTO NON ESCA DAL FRAME
        if length(Signal) >= start_pos+ param.LongPacketNumBits*SpS
            %RESTITUISCO IL PACCHETTO NUMERATO
            sync(ii,:) = abs(Signal(start_pos:start_pos+param.LongPacketNumBits*SpS-↙
1));
        else
```

```
            error(ii,1)=1;
            sync(ii,:) = (1:1344)*0;
        end
    end

end

%FUNZIONE INTERPOLATRICE PER AUMENTARE IL NUMERO DI CAMPIONI
function z = adsbInterpolator(y, param)
%rendo la funzione InterpFil persistente = non deve essere rigenerata tutte
%le volte
    persistent interpFil
    if isempty(interpFil)
        %USO FILTRO FIR INTERPOLATOR PER AUMENTARE IL NUMERO DI CAMPIONI
        %ref; https://it.wikipedia.org/wiki/Finite_impulse_response
        interpFil = dsp.FIRInterpolator(param.InterpolationFactor,...
            param.InterpolationFilterCoefficients);
    end
    if param.InterpolationFactor > 1
        z = interpFil(y);
    else
        z=y;
    end
end

%FUNZIONE PER LA DECODIFICA DEL PACCHETTO
function [structData,error] = decodeData(BitVector)
    bits = logical(BitVector)';
    %Pacchetto ADSB vettore esadecimale completo
    structData.hexVec = binaryVectorToHex(bits);

    %Composizione del pacchetto ADSB Extended Squitter
    %[DF 1:5][Capability 6:8][ICAO 9:32][DATA 33:88][CRC 89:112]
    %Composizione del pacchetto ADSB Short Squitter
    %[DF 1:5][Capability 6:8][ICAO 9:32][CRC 33:56]

    structData.DF = bi2de(bits(1:5),'left-msb');
    structData.CA = bi2de(bits(6:8),'left-msb');
    structData.ICAO = binaryVectorToHex(bits(9:32));

    if structData.DF == 17 %Extended squitter
        structData.DATA = binaryVectorToHex(bits(33:88));
        structData.TC = bi2de(bits(33:37),'left-msb');
        structData.CPRf = bits(54); %cpr flag even odd
        structData.PI = binaryVectorToHex(bits(89:112));
    elseif structData.DF == 11 %Short squitter
        structData.PI = binaryVectorToHex(bits(33:56));
    else
        error = 1;
    end
end

%FUNZIONI PER LA DECODIFICA DEI DATI DEL PACCHETTO
function structData = decodeHexData(ADSBdataStruct)
%CARICO TUTTI I FLAG CPRf e TC DEI PACCHETTI OTTENUTI DA decodeData
```

```
    CPRf = [ADSBdataStruct.CPRf];
    TC   = [ADSBdataStruct.TC];
    cursor = 1;
    for ii=1:length(TC)-1
        for jj=ii+1:length(TC)
%PER LA DECODIFICA DELLA POSIZIONE E' ESSENZIALE CHE I PACCHETTI DATI
%APPARTENGANO ALLO STESSO AEREO QUINDI RICHIAMO IN MEMORIA ICAO E ICAO1
%RIFERITI AL PRIMO E SECONDO PACCHETTO ANALIZZATO E IL VETTORE ADSB PER
%ESTRARRE I DATI
            ICAO = ADSBdataStruct(ii).ICAO;
            hBITS = ADSBdataStruct(ii).hexVec;
            ICAO1 = ADSBdataStruct(jj).ICAO;
            hBITS1 = ADSBdataStruct(jj).hexVec;
            %VERIFICA ID AEREO
            if (ICAO == ICAO1)
                %VERIFICA DEL CPR FLAG E DEL TC PER OTTENERE LA POSIZIONE
                if (CPRf(ii)==0)&&(CPRf(jj) == 1)&&(TC(ii)==TC(jj))&&(TC(ii)>=9 && TC↙
(ii)<=18)

                    BITS = hexToBinaryVector(hBITS);
                    BITS1 = hexToBinaryVector(hBITS1);

                    [Latitude,Longitude,Altitude] = getADSBvalue(BITS,BITS1);
                    %salvo i dati ottenuti
                    structData(cursor).ICAO = ICAO;
                    structData(cursor).Latitude = Latitude;
                    structData(cursor).Longitude = Longitude;
                    structData(cursor).Altitude = Altitude;

                    cursor = cursor+1;
                end
            end
        end
    end
end

function [latitude,longitude,altitude] = getADSBvalue(BITS,BITS1)
                NZ = 15;
                dLat_even = 360/(4*NZ);
                dLat_odd = 360/(4*NZ-1);
                %acquisizione dei dati da valutare
                cpr_lat_even = bi2de(BITS(55:71),'left-msb')/131072;
                cpr_lon_even = bi2de(BITS(72:88),'left-msb')/131072;
                cpr_lat_odd = bi2de(BITS1(55:71),'left-msb')/131072;
                cpr_lon_odd = bi2de(BITS1(72:88),'left-msb')/131072;

                %calcolo la latitudine
                j = floor(59*cpr_lat_even-60*cpr_lat_odd+0.5);

                Lat_even = dLat_even*(mod(j,60)+cpr_lat_even);
                Lat_odd = dLat_odd*(mod(j,59)+cpr_lat_odd);

                if Lat_even >= 270
                    Lat_even = Lat_even - 360;
                end
```

```
                if Lat_odd >= 270
                    Lat_odd = Lat_odd - 360;
                end
                latitude = Lat_odd;

                %calcolo la longitudine
                ni = max(NL(cpr_lat_odd)-1,1);
                dLon = 360/ni;

                m = floor(cpr_lon_even*(NL(Lat_odd)-1)-cpr_lon_odd*NL(Lat_odd)+0.5);
                Lon_odd = dLon*(mod(m,ni)+cpr_lon_odd);

                if (Lon_odd>=180)
                    Lon_odd = Lon-360;
                end
                longitude = Lon_odd;

                %calcolo l'altitudine
                altitude = bi2de([BITS1(41:47) BITS1(49:52)],'left-msb');
                if BITS1(48)==1
                    q_bit_mult=25;
                else
                    q_bit_mult=100;
                end;
                altitude = altitude*q_bit_mult-1000;
end

function nl = NL(latitude)
    NZ = 15;
    nl = floor(2*pi/acos(1-(1-cos(pi/(2*NZ)/cos(pi/180*latitude)^2))));
end

%FUNZIONE RICEVITORE E ACQUISIZIONE DATI
function rxSig = RTLreciver(RTLParam)

    rxADSb = comm.SDRRTLReceiver('0',...
        'CenterFrequency',RTLParam.CenterFrequency,...
        'EnableTunerAGC',false,...
        'TunerGain',RTLParam.TunerGain,...
        'SampleRate',RTLParam.SampleRate,...
        'OutputDataType','single',...
        'SamplesPerFrame',RTLParam.NsBlock,...
        'FrequencyCorrection',0);

    if ~isempty(sdrinfo(rxADSb.RadioAddress))
          rxSig = zeros(RTLParam.NBlocks,RTLParam.NsBlock);
        for counter=1:RTLParam.NBlocks
            rxSig(counter,:)=step(rxADSb)';
        end
    else
        warning('No disp sdr rilevato')
    end

    release(rxADSb);
end
```

```
%DEMODULAZIONE DEL SEGNALE - PASSO IN BINARIO
function z = adsbDemod(y,param)
%LA SPIEGAZIONE DEL FUNZIONAMENTO DI QUESTA FUNZIONE E CONTENUTA NEL REPORT
%IN MODO CHIARO
%differenza tra le due modulanti (1,0)-(0,1) = (1,-1)
    diffS_12 = [ones(param.SpC,1); -ones(param.SpC,1)];
%ottengo il la matrice [numBits x SpS] ovvero [112 x 12]
    numBits = size(y,1) / param.SpS;
    yTemp = reshape(y, param.SpS, numBits)';
%prodotto tra [112 x 12 ] [12 x 1] = [112 x 1] vettore di bit dove 1 logico
%è maggiore di 0 mentre lo 0 logico è minore di 0
    ySoft = yTemp*diffS_12;
    z = uint8(ySoft > 0);
end

%CONTROLLO CRC PACCHETTI PER LA LORO VALIDITA'
function error = CRCcheck(data)

    generator = logical([1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1]);
    msg = logical(data)';

    for cursor=1:88 %112 bit -24bit per la parità
        if logical(msg(cursor)) == 1
            msg(cursor:cursor+24) = xor(msg(cursor:cursor+24),generator);
        end
    end
    CRC = msg(89:end);

    if CRC == false
        error = false;
    else
        error = true;
    end
end

%ADS-B STANDARD PARAM
function adsbParam = ConfigAdsbReciverAndStandardParam()
% definizione dei parametri standard per un ricevitore adsb fatto con
% rtl-sdr
    %PARAMETRI RADIO
    adsbParam.CenterFrequency = 1090e6;
    adsbParam.SampleRate = 2.4e6;
    adsbParam.NsBlock = 51840;
    adsbParam.TunerGain = 60;
    adsbParam.NBlocks = 1000;
    %PARAMETRI SIMBOLI
    adsbParam.NbitPerSymbol = 2;
    adsbParam.SpS = 12;
    adsbParam.SpC = 6;
    %PARAMETRI PACCHETTI
    adsbParam.LongPacketNumBits = 112;
    adsbParam.ShortPacketNumBits = 56;
    %SEQUENZA DI SYNC
    adsbParam.preamble = [ones(1,6) zeros(1,6) ones(1,6) zeros(1,6) ...
```

```
                        zeros(1,18) ones(1,6) zeros(1,6) ...
                        ones(1,6) zeros(1,36)];

    % PRESA DA ADSBExample/helperAdsbConfig.m
    adsbParam.InterpolationFilterCoefficients = ...
                    single(rcosdesign(0.5, 3, double(adsbParam.SpC)));
    % IL FATTORE DI INTERPOLAZIONE E' 5 PER PASSARE DA 2.4 A 12 SpS
    adsbParam.InterpolationFactor=5;
end
```

Link reference:

- http://mode-s.org/decode/adsb/identification.html
- https://it.mathworks.com/help/supportpkg/rtlsdrradio/examples/airplane-tracking-using-ads-b-signals.html
- http://www.lll.lu/~edward/edward/adsb/DecodingADSBposition.html
- http://article.sapub.org/10.5923.j.ajsp.20150502.01.html
- http://amslaurea.unibo.it/4850/1/Pandolfi_Carlo_Tesi.pdf