Alma Mater Studiorum - Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica Per Il Management

SVILUPPO DI UN APPLICATIVO MULTITHREADING PER LA DECODIFICA DI MESSAGGI ADS-B

Tesi di Laurea in Algoritmi e Strutture Dati

Relatore:
Ill.mo Dott. Moreno Marzolla

Presentata da:
Tartaglione Alberto

Correlatore:
Prof. Matteo Zanzi
Ing. Alessandro Mirri

I Sessione
Anno Accademico 2010/2011

Introduzione

Attraverso la ricerca e lo sviluppo di una piattaforma informatica su cui far convergere tutte le possibili informazioni caratterizzanti un determinato spazio aereo, è stato sviluppato il progetto **EPICAS** il cui obiettivo è l'introduzione di un sistema di monitoraggio semplice,facilmente adattabile alle diverse esigenze, unico e integrato per la sorveglianza e gestione del traffico aereo, indirizzato a piccoli aeroporti e aviazione generale, movimentazione di terra, paracadutisti ecc.
Questo progetto è sostenibile grazie al progresso della tecnologia **Global Navigation Satellite System** (GNSS), che rende possibile una georeferenziazione accurata e sicura dei dati in ingresso (uno per tutti il posizionamento dei velivoli), unitamente al progresso della tecnologia sensoristica e delle telecomunicazioni, che permette l'integrazione e la fusione dei dati geospaziali con quelli di altri sensori come ad esempio il Secondary Surveillance Radar (SSR) e in genere i radar di sorveglianza o i sistemi di multilaterazione (MLAT). Sapendo che la misura effettuata da un Global Position System (GPS) presente nella strumentazione di un aeromobile è affetta da un errore, il laboratorio Lasim ha sviluppato il progetto **Traffic Report with Enhanced Control of (AIRPORT) Surface** (T-RECS) che riceve dall'antenna ADS-B i dati di posizionamento di tutti i veicoli i quali dovranno essere corretti al fine di trovare la vera posizione di un veicolo.

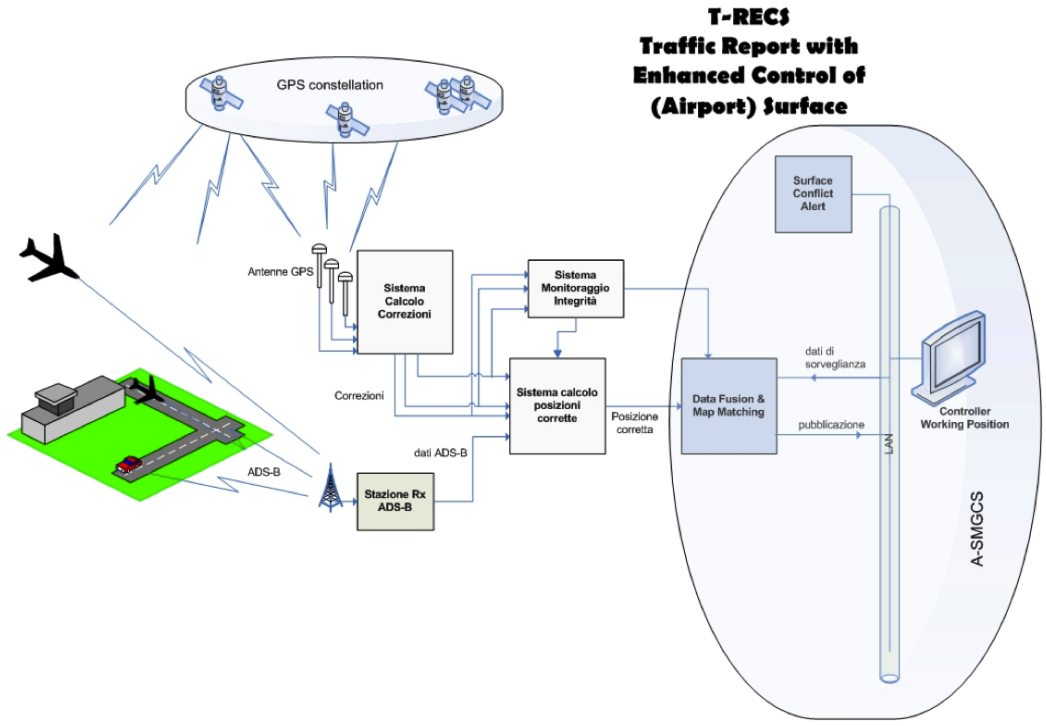


Figura 1 Illustrazione grafica progetto T-Recs [2]

Nella figura 1 è riassunto lo schema della stazione T-RECS che, grazie ad una corretta georeferenziazione dei ricevitori GNSS, è in grado di quantificare gli errori ionosferici, troposferici e quelli dovuti alle geometrie apparenti della costellazione GNSS che affliggono anche i ricevitori di bordo degli aeromobili. Con questi dati T-RECS è in grado di apportare le correzioni al dato posizionale fornito dal sensore ADS-B. Tale informazione viene trasmessa in input alla componente data fusion che elabora le cosiddette "tracce di sistema" da presentare alla postazioni controllore. Il progetto GSLogger presentato in questa dissertazione si colloca all'interno dello spazio denominato "Stazione Rx ADS-B" e si occuperà di ricevere, decodificare e salvatare in un database tutti i messaggi ADS-B provenienti da veicoli di terra e aria. Il linguaggio di

implementazione scelto è il Java, il quale permette di ottenere applicazioni portabili e indipendenti dalla piattaforma sottostante quindi sfruttando la sua portabilità sarà possibile utilizzare il GSLogger su tutti i sistemi operativi più diffusi. L'utente tramite una interfaccia grafica può controllare e gestire tutti i comandi e le opzioni del progetto come l'intervallo e la velocità di ricezione dei messaggi, impostare la porta di connessione e la quantità di messaggi da visualizzare nel grafico dei messaggi ricevuti e nella tabella degli aerei.

Indice

Capitolo 1

Le tecniche di sorveglianza

La sorveglianza è lo strumento di base per l'erogazione dei servizi di controllo del traffico aereo (ATC) al fine di mantenere le separazioni di sicurezza tra un velivolo e l'altro e gestire efficacemente lo spazio aereo. Ne discende, quindi, che le procedure di controllo in un definito spazio aereo sono funzione delle tecniche di sorveglianza attualmente disponibili. Brevemente queste possono essere suddivise in tre categorie:

1. **Sorveglianza di tipo "dipendente"**: il controllore ha necessità della collaborazione attiva del pilota al fine di stabilire la stima della sua posizione;

2. **Sorveglianza di tipo "indipendente"**: il controllore dispone di informazioni del radar primario per il rilevamento della posizione dei velivoli;

3. **Sorveglianza di tipo "cooperante"**: il controllore dispone anche delle informazioni del radar secondario che coopera con il transponder di bordo in modo automatico;

Per definire il nuovo concetto Communication Navigation Surveillance/Air Traffic Management (CNS/ATM) ed il relativo scenario tecnologico, bisogna

integrare le funzioni di Comunicazione, Navigazione e Sorveglianza con l'introduzione di nuove tecnologie già maturate e sperimentate in settori diversi da quello aereo. Nel dettaglio:

1. **Comunicazione**: introduzione di collegamenti in data-link (che utilizzano tecniche di modulazione digitale) per garantire un flusso di dati tra sistemi digitali di terra e di bordo che utilizzeranno canali radio nelle bande già assegnate al servizio aeronautico.

2. **Navigazione**: introduzione di un sistema di assistenza che fornisca adeguate informazioni alla navigazione per tutte le fasi del volo con copertura a livello intercontinentale: il GNSS (Global Navigation Satellite System) costituito dall'integrazione delle costellazioni satellitari GPS (Global Position System - USA) e GALILEO (Europa);

3. **Sorveglianza**: introduzione di applicazioni/funzionalità di sorveglianza tra le quali quella dipendente Automatic Dependent Survelliance (ADS) con trasmissione dati selettiva tra velivolo e centro di controllo (ADS-Addressed), oppure circolare (ADS-Broadcast). L'ADS-Contract riguarda la trasmissione della posizione, dell'identità e di altre informazioni dall'aereo a uno o più sistemi di terra per cui realizza un data link point-to-point secondo un flusso di comunicazione bidirezionale affidabile, per cui il mittente viene a conoscenza se il messaggio è stato ricevuto o meno.

1.1 Sorveglianza e classificazione delle tecniche di sorveglianza

La sorveglianza è un modo di acquisire la posizione dei velivoli per mantenerne le separazioni di sicurezza e gestirli in maniera efficiente (sicura,

ordinata e rapida).
Come detto le procedure di controllo in un definito spazio aereo sono funzione delle tecniche di sorveglianza disponibili classificabili in:

1. **Indipendente dai sistemi on-board di navigazione e avionici (di trasmissione, transponder)**: il controllore acquisisce la posizione dei velivoli dal radar primario (PSR e SMR);

2. **Cooperante con i sistemi on-board**: il controllore acquisisce la maggior parte delle informazioni dei velivoli, tra le quali quella di posizione dal radar secondario e alcune dai sistemi on board (SSR modo A/C ed S e multilaterazione);

3. **Dipendente:**: il controllore acquisisce la posizione dei velivoli fornita dal sistemi di navigazione on-board via radio tramite il voice reporting del pilota o in automatico (manuale o ADS-B rispettivamente).

Le principali tecniche di sorveglianza utilizzate negli aeroporti sono le seguenti:

1. La *MPR* è una tecnica di sorveglianza usata nelle regioni dove non è presente una copertura radar. L'equipaggio attraverso una comunicazione radio di tipo HF comunica i suoi dati.

2. Il *Primary Surveillance Radar (PSR)* è una tecnica di sorveglianza che attraverso l'ausilio di un radar permette di conoscere distanza, elevazione e azimut[1] del target di riferimento, ed essendo una tecnica indipendente non richiede al velivolo un equipaggiamento di bordo. Il suo funzionamento si basa sulla trasmissione della stazione di terra di un

[1]Indica un angolo compreso fra un punto e un piano di riferimento dove solitamente è la distanza angolare compresa tra il Nord e la direzione dove cade la perpendicolare di un punto calcolato in senso orario.

segnale radio e sulla ricezione dell'eco di questo segnale riflesso dal target. Calcolando il cammino di andata e ritorno tra stazione di terra, target ed il tempo impiegato è possibile ricavare l'informazione della posizione sul piano orizzontale.

3. L' *Secondary Surveillance Radar (SSR) modo A/C* è una tecnica che si basa sull'invio ad una precisa frequenza (1030 MHz) di un treno di impulsi codificati che costituiscono l'interrogazione del sistema. All'interno del velivolo è presente un transponder che riceve gli impulsi e risponde automaticamente una risposta codificata (12 bit) su un'altr frequenza standardizzata a livello internazionale (1090 MHz). Questo segnale viene decodificato e interpretato per essere associato ad un eco primario e presentato con le relative informazioni alfanumeriche. Queste interrogazioni vengono fatte attraverso il modo A/C che grazie alla spaziatura dei segnali trasmessi in modo alternato, viene richiesto al velivolo di fornire il proprio identificativo composto da una risposta di 4 cifre ottali assegnate al controllo ATC da terra al pilota (4096 codici) e caricate su un pannello di controllo del transponder (associato alla Call Sign cioè un nominativo dellaeromobile standard)

Nella figura 1.1 vediamo che il radar PSR emette un segnale che "rimbalza" indietro dall'aereo restituendo così al radar un segnale che servirà per determinare la sua posizione. Qui, l'aereo non ha dato alcun segnale di per sè, ha fatto tutto il radar.
L'SSR invece emette un segnale che l'aereo riceve, rileva i processi e il piano di risposta al segnale ricevuto inviando una nuova forma di risposta contenente le informazioni che il radar ha richiesto. Solitamente le informazioni richieste sono l'altitudine, velocità, il codice di identificazione del velivolo, e così via.

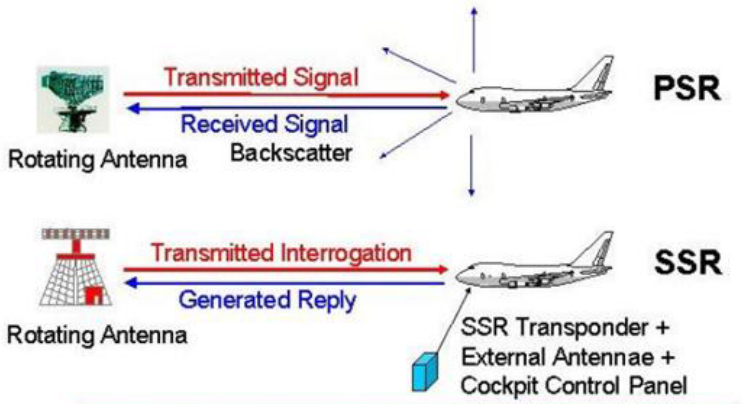


Figura 1.1 Differenza PSR e SSR [3]

1.2 Come funziona il radar SSR e le principali differenze con il radar PSR

Possiamo riassumere il funzionamento del radar SSR così:

1. La stazione radar modula alla frequenza di lavoro gli impulsi e gli invia;
2. Il transponder dell'aereo amplifica il segnale e lo demodula;
3. Il decoder estrae le informazioni e prepara il codificatore per inviare una risposta contenente l'informazione ricevuta;
4. Il codificatore codifica la risposta e la trasmissione dell'aereo e la invia alla frequenza di risposta;
5. Il ricevitor di terra amplifica il segnale ricevuto, lo decodifica e elimina le interferenze;
6. Il decodificatore processa le informazioni e le mostra sul display.

Radar PSR e SSR hanno vantaggi e svantaggi gli uni sugli altrie, vediamone le differenze:

PSR	SSR
Rileva tutti i bersagli nel cielo	Rileva solo i bersagli che rispondono al messaggio
Potenza di trasmissione molto elevata	Potenza Trasmessa Limitata
Ricevitore molto sensibile (amplificatore a microonde)	Non hai bisogno di un ricevitore molto sensibile
Target tracking complicato	È capace di un numero maggiore di tracce
Non è possibile sapere l'altezza del velivolo	Conosce l'altezza del velivolo
Non è possibile conoscere l'identità del velivolo	Conosce l'identità del velivolo

Un altro possibile problema dei radar SSR è la probabilità maggiore di generare un errore causato dalla ricezione dei messaggi di risposta alla trasmissione inizale in modo errato pur avendo una potenza di lavoro minore.

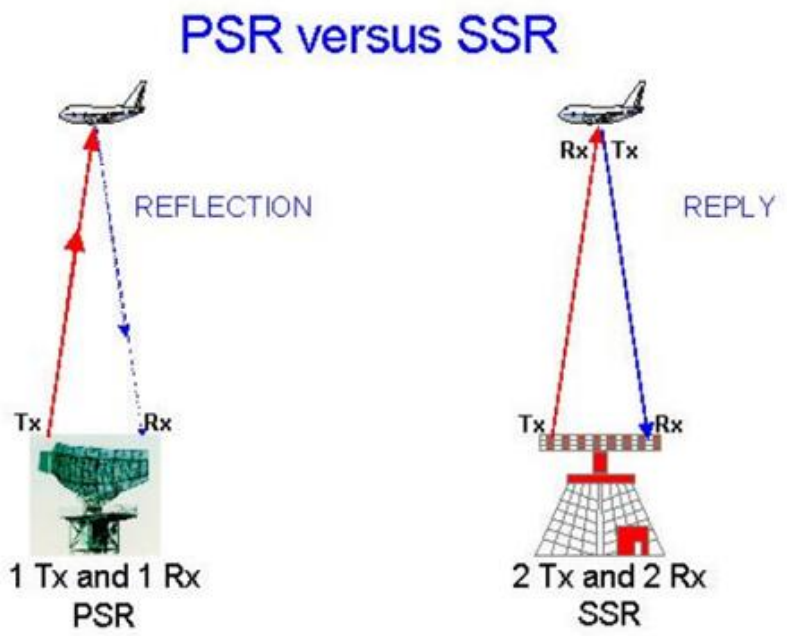


Figura 1.2 Trasmissioni segnali e risposte [3]

Il concetto viene illustrato nella figura 1.2. Tx è la trasmissione iniziale, Rx è la risposta. Si nota che nel PSR dal radar viene inviato un solo segnale

che viene riflesso dall'aereo, mentre nel SSR dal radar di terra vengono inviate due trasmissioni e l'aereo rrisponderà con un segnale di risposta corretto. Entrambi i radar sono complementari e attualmente si utilizza una combinazione di entrambi ai diversi compiti del controllo del traffico aereo. Si sta cercando però di sostituire questi sistemi di controllo con alcuni più moderni e complessi e più precisi come il GPS, il sistema GALILEO e così via

1.3 Standard delle tecniche di sorveglianza

Le competenze normative dei diversi enti standardizzatori aeronautici sono rappresentate nella figura 1.3.

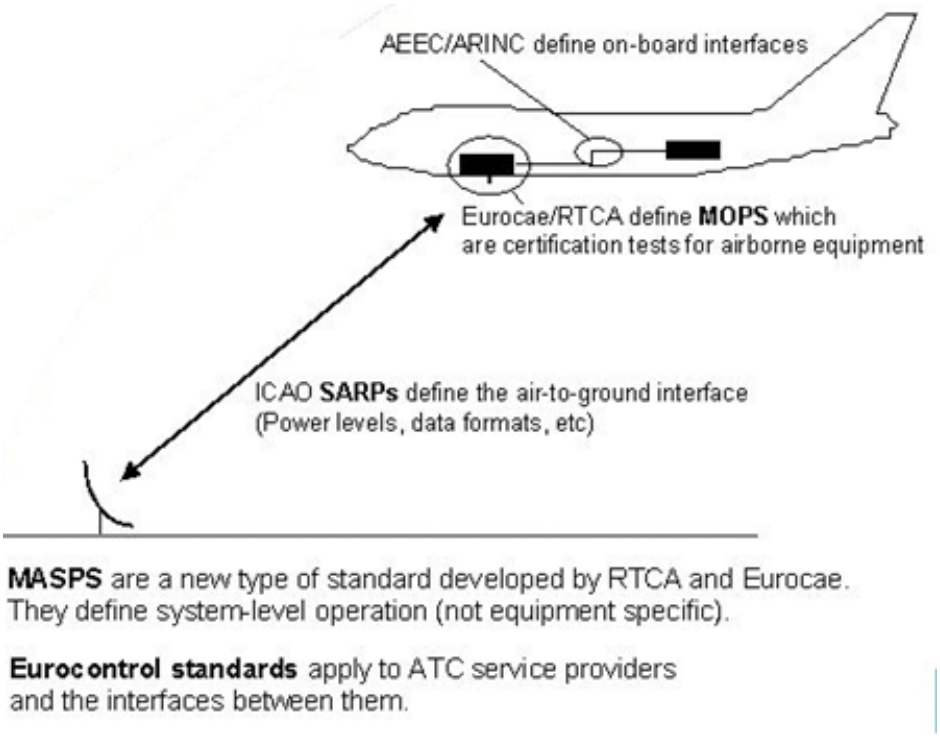


Figura 1.3 Aree di competenza degli enti standardizzati [2]

L'ente MASPS è un nuovo tipo di standard creato da RTCA e Eurocae

e definisce un insieme di operazioni a livello di sistema ed assieme a Eurocae/RTCA definisce le caratteristiche degli apparati di terra e di bordo e gli standard di Eurocontrol si applicano ai sistemi utilizzati nel controllo del traffico aereo e possiamo dividere la sorveglianza a terra e in aria. ICAO/-SARPs si occupa di definire le interfacce aria/terra e AEEC/ARINC definisce le interfacce a bordo di un velivolo.

Capitolo 2

Il Modo S

Il Mode S utilizzato dal radar SSR è standardizzato a livello internazionale ed è impiegato nella sorveglianza del traffico aerei di velovoli dotati di transponder. È definito standard di comunicazione ed è aderente alle normative imposte dall'ICAO, Organizzazione Internazionale dell'Aviazione Civile che ha il compito di sviluppare i principi e le tecniche della navigazione aerea internaionale, delle rotte degli aerei e degli aeroporto e promuovere la progettazione e sviluppo del trasporto aereo internazionale rendendolo più sicuro e ordinato [2].

2.1 Tipi di interrogazioni modo S

Potendo interrogare il velivolo in maniera selettiva attraverso un indirizzo univoco a 24 bit, differenzia questa tecnologia dalle altre chiamate Modo A e Modo C consentendo due tipi di interrogazioni:

1. *All Call*: Il prinicipio di funzionamento è simile a quello delle interrogazioni standard nel quale un segnale da terra è inviato senza indirizzo specifico del destinatario, o con un indirizzo specifico per questo tipo di interrogazioni, al fine di acquisire nuovi tutti i target modo A/C.

2. *Modo S (Roll Out)*: consente di interrogare selettivamente i bersagli modo S al fine di ridurre l'affollamento del canale. Poiché, inoltre, i due sistemi devono anche garantire l'interoperabilità, vi sono più modalità per le quali i dispositivi di bordo possono iniziare a trasmettere.

Nella figura 2.1 vengono riassunte le modalità di interrogazione alle quali rispondono i transponder.

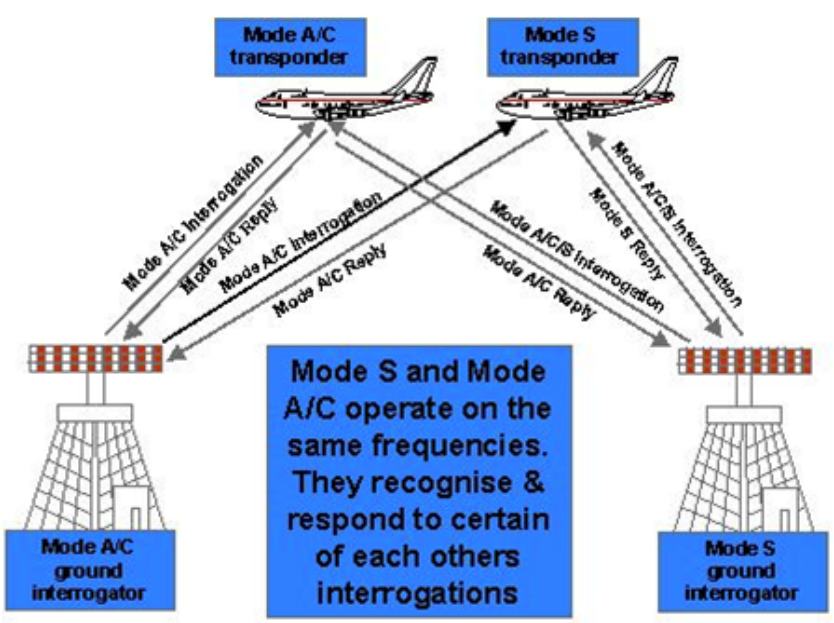


Figura 2.1 Modalità interrogazione transponder [2]

Nella figura 2.1 notiamo i due radar Mode A/C e Mode S che operano nella stessa frequenza e due velivoli con a bordo un transponder modo A/C e uno modo S. Quando il radar modo A/C invia una interrogazione di tipo A/C all'aereo con il transponder A/C e S riceverà una risposta dello tesso tipo della interrogazione iniziale. Il radar Mode S invece, quando invia una interrogazione mode A/C/S all'aereo con transponder A/C riceverà una risposta del tipo A/C, a differenza di quello con il transponder modo S che risponderà in modo S.

2.2 Caratterizzazione degli impulsi

Possiamo distinguere tre tipo di impulsi o interrogazioni inviati da terra verso il velivolo con a bordo il transponderEsistono tre distinte sequenze di imulsi, o tipi di interrogazione, da terra verso bordo:

- *All call modo A/C*;
- *All Call Intermodali*: riconoscibili sia dai transponder tipo A/C che da quelli tipo S;
- *Modo S*: Interrogazioni all call o selettive ricevibili solo dagli equipaggiamenti modo S; Possiamo generalizzare che un transponder risponderà solo alle interrogazioni definite nelle proprie SARP[1].

Per riconoscere le interrogazioni modo S e All Call il dispositivo controlla:

- La spaziatura esistente tra il primo impulso ed i successivi;
- La lunghezza degli impulsi successivi al primo;

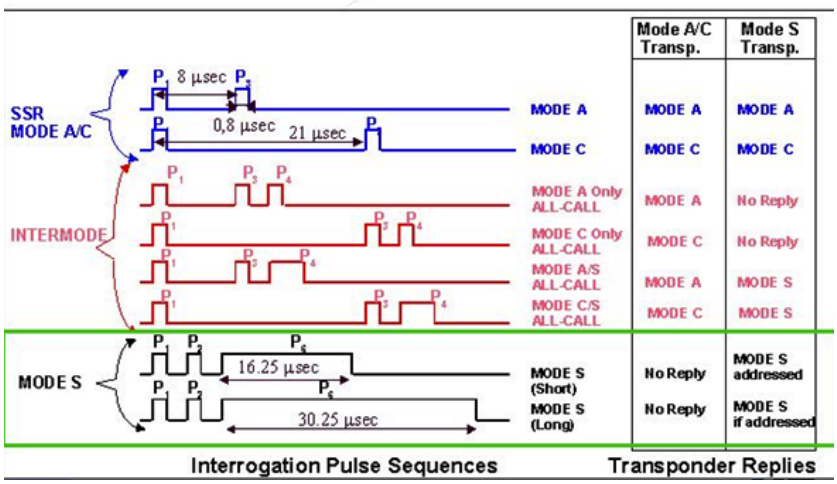


Figura 2.2 Interrogazioni per i messaggi modo A/C/S[2]

[1]Synthetic Aperture Radar Processor

Nella figura 2.2 sono rappresentate le sequenze di impulsi che corrispondono alle varie interrogazioni. Le prime due sequenze rappresentano le classiche interrogazioni modo A/C alle quali rispondono indistintamente i velivoli equipaggiati con i transponder modo A/C o modo S. Come si può notare un impulso P1 seguito da un impulso P3 distanziato di 8 μs è una richiesta modo A mentre se è separato di 21 μs è una richiesta modo C. Le successive interrogazioni sono di tipo intermodale e possono essere generate solamente da interrogatori di tipo S. Questo tipo di interrogazione può essere trasmessa unicamente durante i periodi All-Call. Come detto i transponder modo S e modo A/C possono funzionare efficacemente in un ambiente di tipo "misto" durante i periodi All Call intermodali: i dispositivi A/C non percepiscono differenze con le interrogazioni classiche, mentre quelli S aspetteranno di controllare se l'impulso P4 è lungo o corto per decidere se devono emettere o meno il loro segnale.

Capitolo 3

Il sistema ADS-B

3.1 Definizione e caratteristiche

ADS-B [2] *(Automatic Dependent Surveillance-Broadcast)* è una funzione di sorveglianza per utenti A/V[1]. Secondo la definizione data dall'ICAO[2] *è una applicazione che trasmette parametri di sorveglianza come posizione, pista e velocità al suolo attraverso un collegamento dati in modalità broadcast ad invervalli prestabiliti per esere usati da veiocoli in aria o a terra.* Quindi l'ADS-B si basa sulla trasmissione regolare di report di posizione attraverso una radiodiffusione automatica dove questi report possono essere ricevuti da tutti i velivoli a terra, velivoli trasmittenti o stazioni di terra. Se sono ricevuti da altri transponder possono essere utilizzati per integrare le informazioni di sorveglianza disponibili e visualizzarle ai piloti o controllori di terra. È possibile utilizare inoltre la trasmissione dei dati in maniera aria/aria, in modo da non dover utilizzare stazioni di terra per eseguire compiti di sorveglianza. Ricostruendo i report ricevuti si può ricreare il traffico presente nelle vici-

[1]Acronimo di Aircraft/Vehicle, intesi come aerei in volo o a terra e veicoli operativi nell'area di manovra aeroportuale

[2]International Civil Aviation Organization - Organizzazione Internazionale Aviazione Civile

nanze del velivolo e visualizzarlo all'interno degli strumenti a disposizione. Esistono due tecnologie che supportano l'ADS-B:

1. *State Vector [SV]*: contiene informazioni sullo stato cinematico dell'A/V.

2. *Mode Status Report [MS]*: contiene le informazioni degli utenti che stanno trasmettendo.

3. *Target State Report [TSR]*: contiene le informazioni sui targer rilevati dal velivolo

4. *Air Referenced Velocity Report [ARV]*: contiene le informazioni su velocità, prua [3] e intenzioni del velivolo.

3.2 Il sottosistema trasmettitore

Queto sistema è costituito da una funzione di generazione e una di scambio del messaggio. La prima comprende a sua volta le sottofunzioni di interfaccia agli ingressi e quella di assemblaggio/codifica del messaggio. La funzione di scambio comprende le sottofunzioni realizzate dalla apparecchiatura radio (modulatore/trasmettitore) e dall'antenna trasmittente a 1090MHz. Il sistema ADS-B può essere:

1. Integrato in un transponder modo S che mantiene le sue funzioni e vi aggiunge la funzione ES.

2. Stand alone senza funzioni modo S transponder.

In un *sottosistema transponder-based* tutte le sottofunzioni sono integrate nel transponder eccetto l'antenna che è esterna. Nel *non-transponder-based* può essere integrata.

[3] Letteralmente è inteso muso dell'aereo

3.3 Il sottosistema ricevitore

Questo sottosistema costituito da una funzione di scambio del messaggio e una di assemblaggio del report. La funzione di scambio è simile a quella del trasmettitore formata cioè dall'antenna ricevente e dall'apparecchiatura ricevitore/demodulatore.
La funzione di assemblaggio del report è formata da una sottofunzione di decodifica del messaggio e allemblaggio del report e interfaccia di uscita.

3.4 Formato dei messaggi

Ogni messaggio modo S Extended Squitter (ES) è formato da due parti:
Preambolo: per una risposta modo S è formato da una sequenza di quattro impulsi all'interno di un periodo di 8μs. Identificando questa sequenza di impulsi del preambolo il ricevitore ADS-B si sincronizza con la trasmissione e riceve i 56 bit che formano il blocco dati.

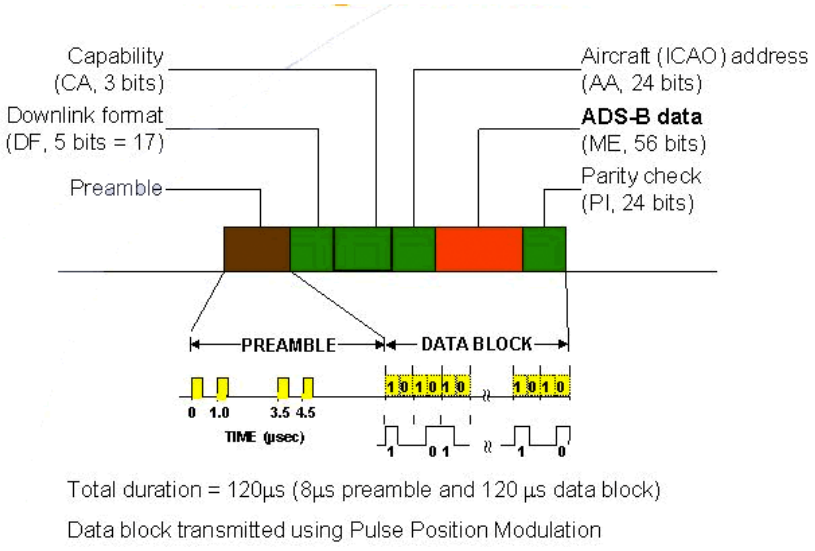


Figura 3.1 Formato messaggi ADS-B [2]

Nella figura 3.1 viene illustrato il formato dei messaggi ADS-B e del blocco dati cheè trasmesso con una **modulazione PPM** [4] i cui bits sono a 1 o 0 a seconda dell'impulso alto che si trova nella prima o seconda metà corrispondente al blocco di durata 1μs detto chip. Di seguito viene riportato il formato del blocco dati del messaggio ADS-B:

1. **Downlink format** [**DF**]: indica il tipo di messaggio trasmesso, impostato a 17 per i messaggi ES.

2. **Capability** [**CA**]: relativa alla capacità del transponder di trasmettere dati o meno.

3. **Aircraft Address** [**AA**]: indirizzo del velivolo con codice ICAO.

4. **ADS-B Data** [**ME**]: campo da 56 bit che contiene i dati ADS-B a seconda del tipo di messaggio ES che viene trasmesso.

5. **Parity check** [**PI**]: contiene un codice di rilevazione degli errori.

Quindi ogni messaggio ES contiene 56 bit di informazioni ADS modo S e 56 bit di informazioni proprie del sistema raccolte nel campo ME e inserite tra il codice ICAO dell'aereo e le informazioni di parità. Il campo AA è unico in tutto il mondo ed è possibile inserirlo anche al momento del decollo.

3.5 Registri BDS

I messaggi ES*(Extended Squitter)* sono definiti come registry BDS *(Binary Data Store)* e si riferiscono ad uno dei 255 registri che sono parte del servizio GICB *(Ground Initiated Comm B)*[5] Ogni registro BDS lungo 56 bit

[4]Pulse Position Modulation.

[5]Il protocollo GICB è un protocllo che consente all'utente di terra di estrarre oltre le informazioni estratte da una interrogazione modo S, anche quele contenute nei campi ME del messaggio ES come velocià e prua.

deve essere aggiornato in tempo reale perchè il suo contenuto sia ritenuto valido. Nella figura 3.2 vengono riassunti i registri BDS che contengono le informazioni ES.

Numero registro BDS (in esadecimale)	Nome	Panoramica dei contenuti
05_{16}	Ext. sq. posizione aerea	Posizione del velivolo trasmessa in volo
06_{16}	Ext. sq. posizione a terra	Posizione e velocità del velivolo trasmessa mentre è a terra
07_{16}	Ext. sq. stato	Capacità e stato dell'ES
08_{16}	Ext. sq. identificativo e categoria	Identificativo del velivolo (callsign) e categoria
09_{16}	Ext. sq. Velocità aerea	Velocità del velivolo trasmessa in aria
$0A_{16}$	Ext. sq. informazione event-driven	Dipende dall'evento
61_{16}	Ext. sq. emergenza /stato di priorita	Emergenza o stato di priorita
62_{16}	Punto di cambiamento dell'attuale traiettoria	Velivolo intento alla TCP corrente
63_{16}	Punto di cambio della prossima traiettoria	Velivolo intento alla TCP seguente
64_{16}	Coordinamento dei messaggi di operazione del velivolo	Stato corrente dei vari parametri del velivolo
65_{16}	Stato opzionale del velivolo	Classe di capacità e attuale modalità operativa delle applicazioni ATC sul velivolo
$66\text{-}6F_{16}$	Riservato per extended squitter	TBD

Figura 3.2 Registri BDS riservati ai messaggi ES

I messaggi di posizione e velocità vengono inviati due volte al secondo perchè contengono informazioni chiave per la sorveglianza mentre l'ICAO, che sarà sempre uguale durante il volo, verrà trasmesso ogni 5 secondi. Di

seguito viene riportato il formato del messaggio relativo alla posizione del velivolo sulla superficie.

Surface position squitter

Number of bits	Contents
5	Format type code
7	Movement
1	Status
7	Ground track
1	Time
1	CPR format
17	CPR encoded latitude
17	CPR encoded longitude
56 bits total	

Figura 3.3 Formato messaggio Surface Position [2]

Spiegazione campi in figura 3.3:

1. **Format type:** indica il tipo di messaggio trasmesso ad es. la precisione della misura dei dati di posizione o se è trasmessa l'altezza barometrica o geometrica (tramite GNSS);

2. **Movement:** indica la velocità dell'aereo sulla superficie usando una scala non lineare;

3. **Ground track status bit:** indica se il ground track field è valido;

4. **Ground track value field:** indica la direzione del moto sulla superficie;

5. **Latitude/Longitude fields:** indica la latitudine e longitudine dell'aereo nei due campi da 17 bits con una precisione di circa 1.25m;

6. **CPR Format bit:** Questo bit fa parte dell'algoritmo CPR ed è utilizzato per indicare se I dati si riferiscono alla griglia "pari" o "dispari".

7. **Time synchronisation bit:** Bit che indica se il tempo di validità del messaggio è sincronizzato UTC.

I campi dell'Airbone Velocity sono riassunti nella figura 3.4:

Airborne velocity squitter (subtype 1 – velocity over ground)

Number of bits	Contents
5	Format type code
3	Subtype
1	Intent change flag
1	IFR capability flag
3	Velocity uncertainty
10 + 1	East-West velocity + sign
10 + 1	North-South velocity + sign
9 + 1 + 1	Vertical rate + sign + source
2	Turn indicator
7 + 1	Geometric height diff. from barometric + sign
56 bits total	

Figura 3.4 Formato messaggi Airbone Velocity [2]

1. **Subtype:** indica il tipo di velocità trasmessa. Velocità rispetto al suolo in condizioni normali o supersoniche (1 o 2), velocità relativa e rotta in condizioni normali o supersoniche (3 o 4) quando l'aereo è in volo;

2. **Intent change flag:**settata a "1" per 18 sec quando l'intenzione dell'aereo (definita nei registri BDS 4016 e 4216) cambia;

3. **IFR capability flag:** Indica che il velivolo è dotato di transponder di classe A1.

4. **East-west velocity and sign e North-south velocity and sign** indicano rispettivamente le componenti est-ovest e nord-sud della velocità dell'aereo a terra e la direzione (tramite un bit di segno);

5. **Vertical rate and sign:** indica il rateo di variazione della quota dell'aereo con segno;

6. **Turn indicator:** indica che l'aereo sta virando;

7. **Geometric height difference from barometric and sign:** indica la differenza con segno tra l'altezza barometrica e quella geometrica.

Per il messaggio di identità possiamo riassumere i campi nella figura 3.5:

1. **Aircraft category:** indica la categoria dell'aereo: piccola (¡15500 lbs), media (da 15500 a 75000 lbs), aliante o velivoli di servizio sulla superficie.

2. **Aircract identification field:** indica gli 8 caratteri identificativi dell'ICAO dell'aereo e solitamente è la callsign.

Aircraft identification squitter

Number of bits	Contents
5	Format type code
3	Aircraft category
6	Callsign Character 1
6	Callsign Character 2
6	Callsign Character 3
6	Callsign Character 4
6	Callsign Character 5
6	Callsign Character 6
6	Callsign Character 7
6	Callsign Character 8
56 bits total	

Figura 3.5 Formato messaggi Airbone Velocity [2]

3.6 Il sistema di decodifica CPR (Compact Position Reporting)

La posizione di un aereo in un ES (Extended Squitter) è codificata tramite un sistema di decodifica chiamato Compact Position Reporting (CPR) che è impiegato per minimizzare il numero di bits richiesti per fornire la posizione del velivolo. Latitudine e longitudine tramite l'algoritmo CPR vengono convertite in valori a 19 bit,così facendo generano posizioni con una precisione di circa 0,75m.

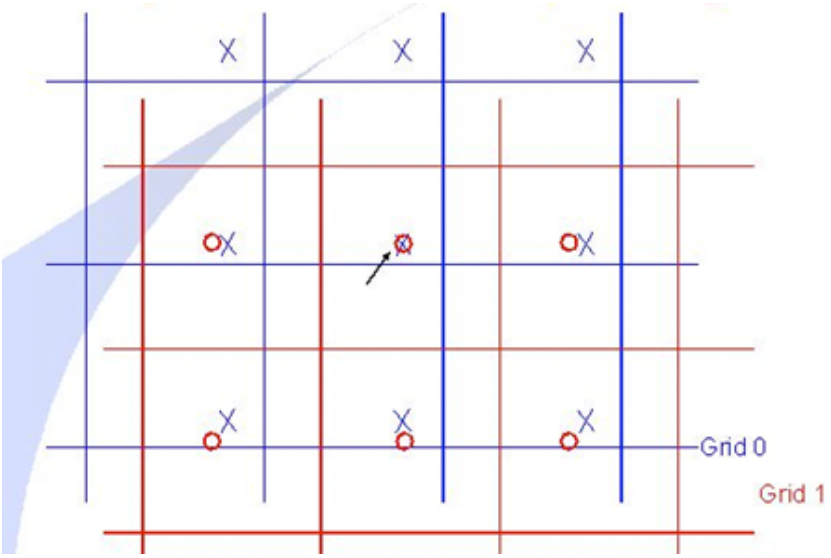


Figura 3.6 Compact Position Reporting [2]

La figura 3.6 spiega l'esecuzione dell'algoritmo CPR. Per determinare posizioni non ambigue è necessario ricevere due report, caratterizzati dal bit "CPR format/time marker" diverso e distanziati da non più di dieci secondi. Ci sono due griglie, Grid0 e Grid1 e le possibili posizioni dei target. Utilizzando l'algoritmo CPR con un messaggio solo non si riesce a determinare la posizione del target in maniera univoca ma, comparando due trasmissioni successive si nota come una sola posizione sia comune alle due griglie, permettendo in tal modo di risovere l'ambiguità sulla posizione[2].

Capitolo 4

Il Progetto GSLogger

Il GSLogger è stato sviluppato all'interno del progetto EPICAS con lo scopo di acquisire i dati dall'antenna ADS-B, decodificarli e salvarli in un database.La misura del GPS produce un errore che deve essere eliminato al fine di ricostruire la posizione "vera" dell'aereo utilizzando i dati salvati nel database. Il progetto è stato implementato in Java che, essendo un linguaggio portabile e indipendente dalla piattaforma sul quale esegue l'applicazione, permette di poter eseguire il GSlogger sui sistemi operativi più diffusi.

Il progetto denominato GSLogger è composto da diverse classi, nel dettaglio:

- *GSLoggerView.java*: è l'interfaccia grafica del programma.

- *KernelThread.java*: è il thread principale del programma che ha la funzione di acquisire dalla porta seriale il flusso di bytecode inviato dal trasnsponder degli aerei, ha il compito di selezionare i messaggi con DF17 o DF18 validi e aggiornare l'ArrayList2d della classe QueueSync contenente i messaggi grezzi che dovranno poi essere decodificati dalla VehicleManager.

- *VehicleManager.java*: è il thread che ha il compito di decodificare i messaggi contenuti nell'ArrayList2d e generare il database per il salvataggio di uno storico dei messaggi non decodificati e decodificati.

- *Vehicle.java*: classe che contiene le variabili che identificano un veicolo.

- *MessageEvent.java*: gestisce la ricezione di un evento relativo a un nuovo messaggio DF17/DF18[1] valido.

- *QueueSync.java*: classe che gestisce la sincronizzazione fra la classe VehicleManager e KernelThread per la scrittura (PUSH)/ lettura (POP) dell'ArrayList2d contenente i messaggi di 16 byte non decodificati.

- *UpdateKLMFile.java*: classe che si occupa di generare il file KLM per GoogleEarth dove potranno essere visibili le rotte degli aerei.

4.0.1 GSLoggerView

GSLoggerView è la classe che genera la GUI del programma come mostra la figura 4.1.
Attraverso questa interfaccia l'utente può gestire le le impostazioni disponibili per il controllo del programma,:

- Nella *"Planes List"* vengono visualizzate tutte le informazioni decodificate relative ad un aereo. La GSLoggerView riceve un evento dalla classe UpdateKLMFile quando ha un aggiornamento degli stessi dati.

- Nel pannello *"Track Management"* l'utente può impostare il numero di posizioni da visualizzare e la cancellazione delle informazioni dopo un certo periodo di tempo in secondi. Quando l'utente inserisce queste informazioni la GSLoggerView lancia un evento che riceverà la

[1] Definizione al par. 2.2

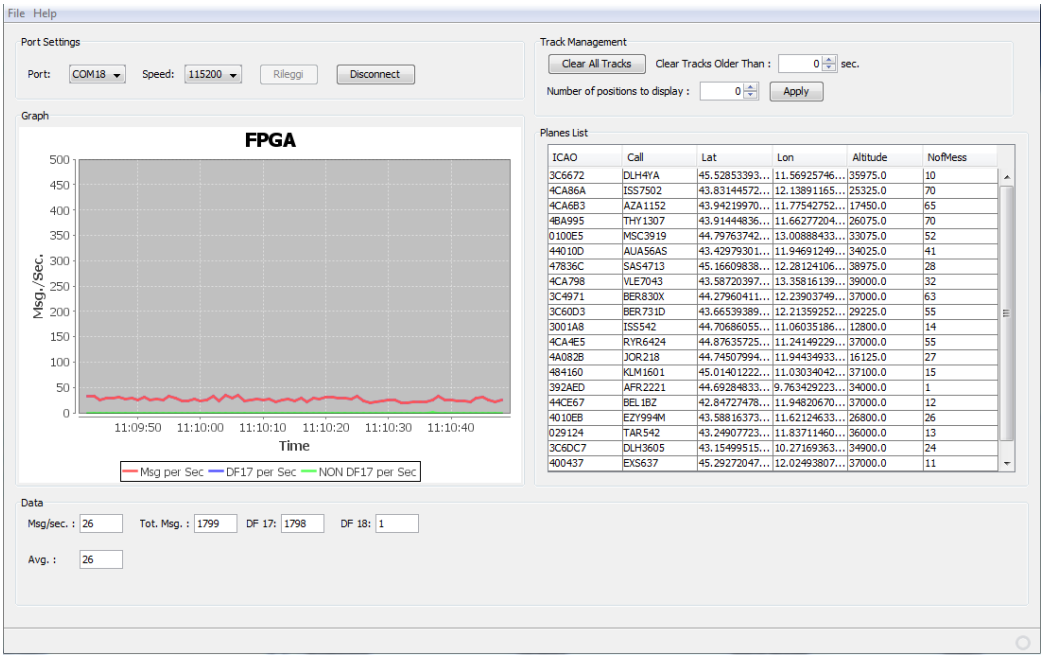


Figura 4.1 Schermata principale programma GSLogger

UpdateKLMFile ed aggiornerà il file KLM secondo i dati impostati nell'interfaccia.

- Nel pannello *"Port Settings"* sono presenti i campi per scegliere quale porta seriale collegare, la velocità di lettura dei dati e i bottoni per la connessione e rilettura del segnale della porta. Per poter connettere un programma scritto in Java con una porta seriale, è stata utilizzata la libreria RxTx che supporta la Comm API della SUN, ed è stato possibile acquisire dati dalla porta I/O
  Attraverso la classe *gnu.io.CommPortIdentifier* si ottengono tutte le porte del sistema alle quali è possibile connettersi.

- Nel pannello *Graph* attraverso l'uso della libreria jFreeChart[2] verranno plottati i dati riguardanti i messaggi per secondo, i messaggi DF17 e

[2]JFreeChart è una libreria Open Source che viene utilizzata per generare i grafici relativi ai dati dei messaggi DF17 e NONDF17 e per i messaggi al secondo ricevuti.

NONDF17 per secondo.

4.0.2 KernelThread

Attravero l'utilizzo della libreria RXTX[3] la classe acquisisce dalla porta seriale il flusso di bytecode che dovrà essere decodificato al fine di estrarne i mesaggi ADS-B che ogni veicolo invia. Se nel buffer in analisi trova un valore esadecimale corrispondente a 0xFF o 0xAA, valori fissi salvati nelle variabili private:

```
1 private Integer START_BYTE =Integer.valueOf(255);
2 private Integer START_BYTE2 = Integer.valueOf(170);
```

allora un messaggio è ritenuto valido, viene controllato che la stringa contiene 16 byte e successivamente viene salvata nell'ArrayList *inDataArray*. Il metodo *parseInDataBuffer()* cerca un allineamento di 16 byte e presuppone che ci sia un messaggio valido, quindi richiama il metodo removeDataToIndex() che dall'inDataArray estrae i 16 byte e richiama il metodo *treatMessage*. Pezzo di codice del metodo relativo all'identificazione di un DF17 o DF18 valido:

```
1 //creo una maschera di 248 bit per lo shifting destro e se
    trovo un df=17 incremento i contatori e lancio l'evento
    tramite la MessageEvent che ho trovato un nuovo messaggio
    DF17 o DF18
2
3     int DF_MASK = 248;
4     int df = ((Integer)msg.get(1)).intValue();
5     System.out.println("DF" + ((df & DF_MASK) >> 3) + "
          trovato");
6     df = (df & DF_MASK) >> 3;
7     if (df == 17) {
8       this.nOfDF17Msg += 1;
9       this.totalNofDF17 += 1;
```

[3]RXTX è una libreria Java che fornisce la comunicazione seriale e parallela per la JDK.

```
10      }
11      else {
12        this.nOfNONDF17Msg += 1;
13        this.totalNofNONDF17 += 1;
14      }
15  //se il df = 17 lancio l'evento "Nuovo messaggio DF17"
16      if (df == 17) {
17        fireEvent(new MessageEvent(this, msg, "New DF17
              Message"));
18      }
19  //se il df = 18 lancio l'evento "Nuovo messaggio DF18"
20      else if (df == 18)
21        fireEvent(new MessageEvent(this, msg, "New DF18
              Message"));
```

Dopo che un messaggio è ritenuto valido e trovato il DF17 o DF18,tramite una PUSH() nell'ArrayList2d creato nella classe QueueSync, la KernelThread aggunge il mesaggio che dovrà poi essere decodificato dalla classe VehicleManager.

4.0.3 VehicleManager

È un thread sempre attivo che controlla a intervalli periodici, se nella coda di messaggi grezzi contenuta nell'ArrayList2d della classe QueueSync ci sono nuovi messaggi da decodificare. Se trova nuovi messaggi nella coda, tramite il metodo POP() svuota la lista e comincia a decodificare i messaggi aggiornando l'ArrayList *vehicleList* che contiene i veicoli identificati dove:

- Se il veicolo è già stato riconosciuto allora aggiorna i dati.
- Se non l'ha ancora identificato lo aggiunge alla lista assieme al messaggio.

Inoltre aggiorna il database dei messaggi grezzi e decodificati. Quando un messaggio è stato decodificato lancia un evento che viene ricevuto solo dalla

GSloggerView relativo ad un aggiornamento di dati, quindi la GSLoggerView prenderà la lista dei veicoli salvata nell'ArrayList *vechicleList* e aggiornerà i suoi contatori. La classe contiene, oltre ai metodi per decodificare i messaggi, anche quello relativo alla creazione del database contenente le tabelle per l'ICAO, i messaggi non decodificati e i messaggi decodificati come mostrato nella figura 4.2.

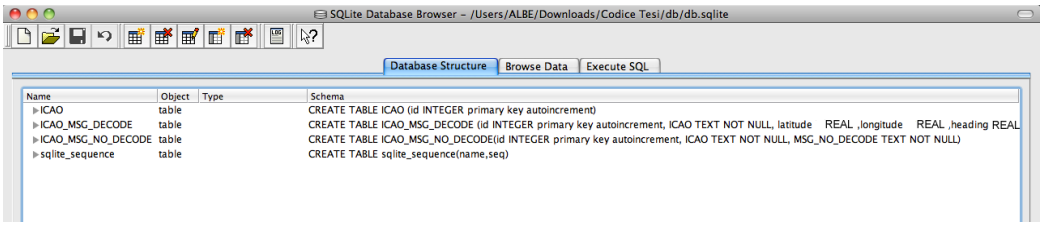


Figura 4.2 Database SQLite creato dalla classe.

Per la creazione del database nel quale salvare i messaggi decodificati, non decodificati e l'ICAO di ogni aereo è stato utilizzato SQLite [4]. Nella figura 4.2 vengono mostrare le tabelle del database scritte nella classe VehicleManager.:

- La tabella ICAO conterrà gli identificatori degli aerei che sarà chiave primaria in tutte le tabelle per unirle.
- La tabella ICAO_MSG_NO_DECODE è la tabella che contiene l'ICAO e il messaggio non decodificato.
- La tabella ICAO_MSG_DECODE contiene l'ICAO dell'aereo, e il messaggio decodificato che comprende latitude. longitude, heading, altitude e timestamp[5].

[4]SQLite è una libreria software scritta da Richard Hipp in C ma utilizzabile tramite dei binding in altri linguaggi (vedi Java) che implementa un DBMS SQL

[5]Servirà per controllare il momento (inteso come ora e data) di decodifica del messaggio.

Il codice scritto nella classe per la creazione del database è il seguente:

```
1  //connetto la classe al database sqlite
2  Class.forName(''org.sqlite.JDBC'');
3  Connection conn =
       DriverManager.getConnection(''jdbc:sqlite:db.sqlite'');
4       //preparo lo statement per la creazione della tabella
5       Statement stmt;
6       //creo la tabella
7       stmt = conn.createStatement();
8            stmt.executeUpdate(''CREATE TABLE ICAO '' +
9                ''(id INTEGER primary key autoincrement)'');
10           stmt.executeUpdate(''CREATE TABLE ICAO_MSG_NO_DECODE''
                 +
11               ''(id INTEGER primary key autoincrement, '' +
12                 ''ICAO TEXT NOT NULL, '' +
13                 ''MSG_NO_DECODE REAL NOT NULL)''+
14                 ''tstamp timestamp)'');
15            stmt.executeUpdate(''CREATE TABLE ICAO_MSG_DECODE '' +
16               ''(id INTEGER primary key autoincrement, '' +
17                 ''ICAO TEXT NOT NULL, '' +
18                 ''latitude REAL,''+
19                 ''longitude REAL,'' +
20                 ''heading REAL,'' +
21                 ''altitude REAL,''+
22                 ''tstamp timestamp)'');
23      //chiudo lo statement
24       stmp.close();
```

4.0.4 Vehicle

Vehicle è la classe che contiene le variabili di istanza che definiscono quando un messaggio è stato decodificato e quali attributi avrà un veicolo all'in-

terno dell'ArrayList *vehicle*.
Le variabili dichiarate nella classe sono:

- **icao_Address**: Identifica il transponder di uno specifico aereo.
- **latitude**: contiene il valore di latitudine.
- **longitude**: contiene il valore di longitudine.
- **altitude**: contiene il valore dell'altitudine.
- **heading**: contiene il grado di virata.
- **nOfMessages**: indica il numero di messaggi inviati.
- **identification**: si riferisce al'identificativo di un volo.
- **cprValid,cprData, cprTime**: valori che tramite l'algoritmo di decodifica CPR [6] consentono di decodificare la posizione di un velivolo.
- **lastDataUpdate**: viene salvato l'istante di ultima ricezione del messaggio di un velivolo.

4.0.5 MessageEvent

La classe MessaggeEvent riceve un evento dalla KernelThread riguardo alla ricezione di un nuovo messaggio che verrà gestito dalla VehicleManager per la decodifica. Gli attributi della classe sono l'eventType che indica il tipo di evento e le variabili statiche NEW_DF17_MESSAGE e NEW_DF18_MESSAGE.

4.0.6 QueueSync

La classe gestisce la coda sincronizzata fra VehicleManager e KernelThread. I due thread, essendo sincronizzati e quindi bloccanti, saranno gestiti

[6] Definizione al par. 3.6

in modo che quando il KernelThread accede all'ArrayList2d contenente la lista dei messaggi grezzi [7] per aggiungere un nuovo messaggio ricevuto attraverso il metodo PUSH() , completerà tutta la sua operazione mentre la classe VehicleManager attenderà la fine, prima di poter accedere alla lista tramite il metodo POP() della classe per prelevare i messaggi e svuotare la coda.

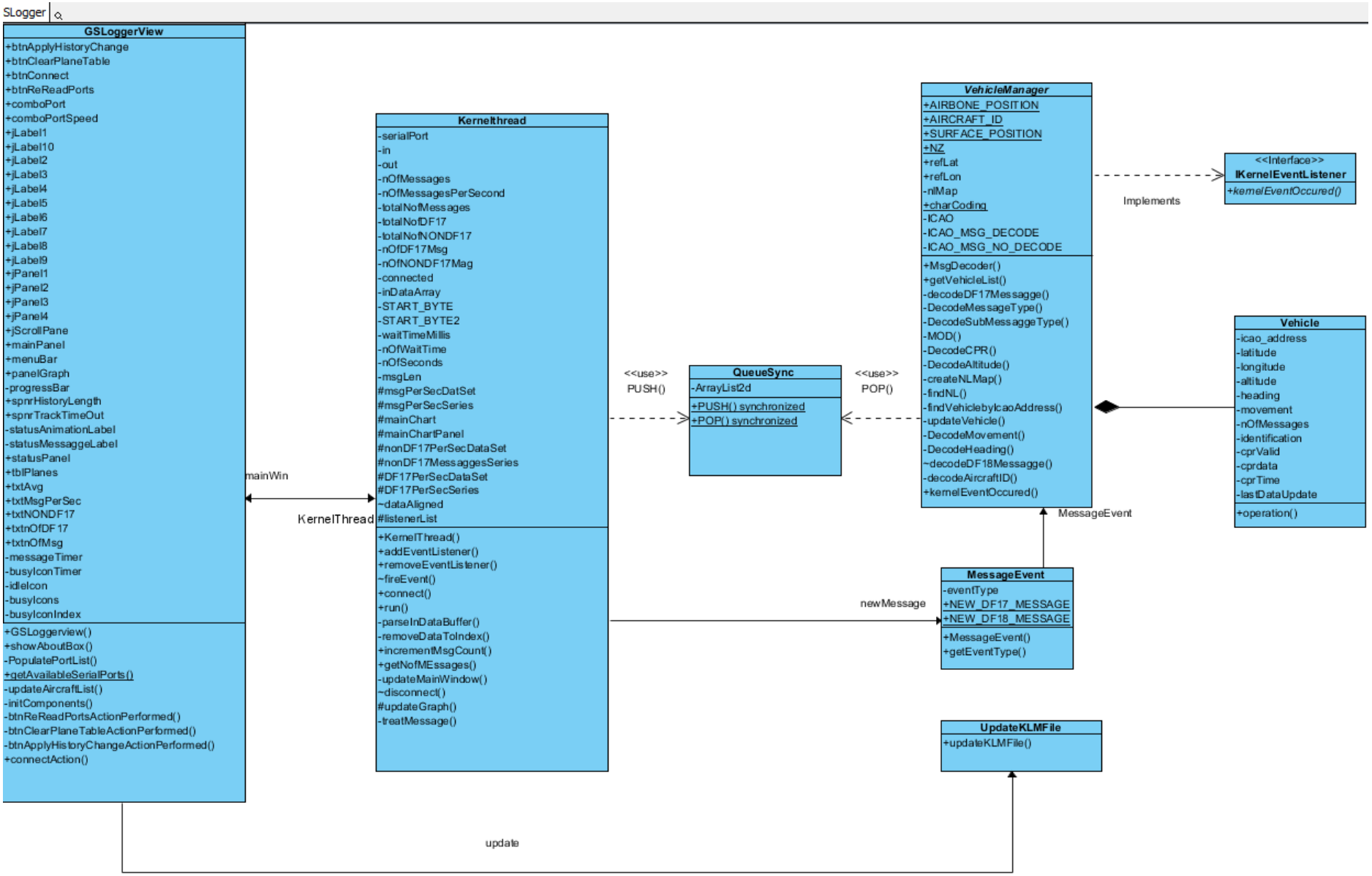
4.0.7 UpdateKLMFile

Questa classe ha il compito di creare il file KLM che, fatto leggere tramite Google Earth, visualizza le rotte degli aerei identificati Per generare il file, la classe prende dalla tabella ICAO del database l'identificativo di ogni aereo, e dalla tabella ICAO_MSG_DECODE dei messaggi decodificati tutti gli attributi che serviranno per generare la rotta nel dettaglio: latitudine, longitudine, altitudine, heading.
Quando la classe ha aggiornato il file KLM lancia un evento che riceve la GSLoggerView che dovrà aggiornare la lista degli aerei. La classe inoltre, quando dalla GUI vengono impostati dei parametri relativi alla visualizzazione degli ultimi messaggi ricevuti, estrae gli icao e tutte le posizioni più recenti dell'aereo e dovrà quindi aggiornare il file KLM secondo tali impostazioni. Una volta che il file KLM è stato creato e fatto leggere da Google Earth il risultato sarà quello mostrato nella figura 4.3.

[7] Sarà un ArrayList di ArrayList dato che la dimensione della coda non è nota

Figura 4.3 Rotte degli aerei dopo la decodifica messaggi.

4.1 Diagramma UML delle classi


SLogger
GSLoggerView
+btnApplyHistoryChange
+btnClearPlaneTable
+btnConnect
+btnReReadPorts
+comboPort
+comboPortSpeed
+jLabel1
+jLabel10
+jLabel2
+jLabel3
+jLabel4
+jLabel5
+jLabel6
+jLabel7
+jLabel8
+jLabel9
+jPanel1
+jPanel2
+jPanel3
+jPanel4
+jScrollPane
+mainPanel
+menuBar
+panelGraph
-progressBar
+spnrHistoryLength
+spnrTrackTimeOut
-statusAnimationLabel
-statusMessaggeLabel
+statusPanel
+tblPlanes
+txtAvg
+txtMsgPerSec
+txtNONDF17
+txtnOfDF17
+txtnOfMsg
-messageTimer
-busyIconTimer
-idleIcon
-busyIcons
-busyIconIndex
+GSLoggerview()
+showAboutBox()
-PopulatePortList()
+getAvailableSerialPorts()
-updateAircraftList()
-initComponents()
-btnReReadPortsActionPerformed()
-btnClearPlaneTableActionPerformed()
-btnApplyHistoryChangeActionPerformed()
+connectAction()
mainWin
KernelThread
Kernelthread
-serialPort
-in
-out
-nOfMessages
-nOfMessagesPerSecond
-totalNofMessages
-totalNofDF17
-totalNofNONDF17
-nOfDF17Msg
-nOfNONDF17Mag
-connected
-inDataArray
-START_BYTE
-START_BYTE2
-waitTimeMillis
-nOfWaitTime
-nOfSeconds
-msgLen
#msgPerSecDatSet
#msgPerSecSeries
#mainChart
#mainChartPanel
#nonDF17PerSecDataSet
#nonDF17MessaggesSeries
#DF17PerSecDataSet
#DF17PerSecSeries
~dataAligned
#listenerList
+KernelThread()
+addEventListener()
+removeEventListener()
~fireEvent()
+connect()
+run()
-parseInDataBuffer()
-removeDataToIndex()
+incrementMsgCount()
+getNofMEssages()
-updateMainWindow()
~disconnect()
#updateGraph()
-treatMessage()
<<use>>
PUSH()
QueueSync
-ArrayList2d
+PUSH() synchronized
+POP() synchronized
<<use>>
POP()
VehicleManager
+AIRBONE_POSITION
+AIRCRAFT_ID
+SURFACE_POSITION
+NZ
+refLat
+refLon
-nlMap
+charCoding
-ICAO
-ICAO_MSG_DECODE
-ICAO_MSG_NO_DECODE
+MsgDecoder()
+getVehicleList()
-decodeDF17Messagge()
-DecodeMessageType()
-DecodeSubMessaggeType()
-MOD()
-DecodeCPR()
-DecodeAltitude()
-createNLMap()
-findNL()
-findVehiclebyIcaoAddress()
-updateVehicle()
-DecodeMovement()
-DecodeHeading()
~decodeDF18Messagge()
-decodeAircraftID()
+kernelEventOccured()
Implements
<<Interface>>
IKernelEventListener
+kernelEventOccured()
Vehicle
-icao_address
-latitude
-longitude
-altitude
-heading
-movement
-nOfMessages
-identification
-cprValid
-cprdata
-cprTime
-lastDataUpdate
+operation()
MessageEvent
newMessage
MessageEvent
-eventType
+NEW_DF17_MESSAGE
+NEW_DF18_MESSAGE
+MessageEvent()
+getEventType()
UpdateKLMFile
+updateKLMFile()
update

Acronimo	Descrizione
A/V	Aircraft/Vehicle
AA	campo Announced Address
ADS-B	Automatic Dependent Surveillance Broadcast
ATC	Air Traffic Control
ATM	Air Traffic Management
CF	campo Code Format
DF	campo Downlink Format
ES	Extended squitter
HW	hardware
ME	campo MEssage
MS	Mode Status
MSB	Most Significative Bit
PI	campo Parità/Interrogator ID
PPM	Pulse Position Modulation
RF	Radio Frequency
SiGeVAMA	Sistema di Gestione Veicolare per Applicazioni Multi-Servizio Aeroportuali
SLS	Side Lobe Suppression
SPI	Special Position Indicator
SV	State Vector
TBD	To be defined
TBW	To be written
TIS-B	Traffic Information Service Broadcast
WP	Work Package

Figura 4.5 Tabella Acronimi e Abbreviazioni

Conclusioni e Sviluppi Futuri

Lo scopo dello sviluppo del programma GSLogger all'interno del progetto EPICAS è stato di acquisire i dati dall'antenna ADS-B, decodificarli e salvarli in un database. Sapendo che la misura di posizione effettuata dal GPS di un aeromobile è affetta da un errore, il dato di posizione dovrà essere ripulito per determinare la posizione "vera" dell'aereo utilizzando i dati salvati nel database. Il progetto quindi, verrà interfacciato con un ulteriore sistema che, andando a prendere le informazioni dal database, consentirà di fare la stima della posizione corretta dei mezzi nonostante il GPS sia affetto da errore. Quindi attraverso l'implementazione di questo sistema da parte degli aeroporti ove è possibile sapere più precisamente la posizione di un aereo, aumentare la sicurezza e abbassare il rischio di collisione, si può aumentare il numero di velivoli presenti nello spazio aereo e quindi incrementare i ricavi dell'aeroporto.

Sviluppi futuri

Questo progetto stabilisce la base di acquisizione di dati per sviluppare un server multithreading. In futuro si potranno fare dei test delle prestazioni in Java riguardo al tempo che intercorre tra ricezione e decodifica dei messaggi ADS-B ed una integrazione in realtime degli algoritmi di correzione al fine di poter stabilire una posizione degli aerei più precisa attraverso il binding di librerie C.

Parte I

Bibliografia

Bibliografia

[1] International Civil Aviation Organization. *Annex10 - Aeronautical Telecommunications*. 2006. Radio Navigation Aids.
URL: http://www.scribd.com/doc/5509183/Annex10-ICAO

[2] Mirri A. *Contesto Operativo ADS-B*. SIGEVAMA, 2005

[3] "Surcando Los Cielos"
URL: http://surcandoloscielos.es/blog/frequently-asked-questions-ix-el-radar-3parte/ , 2009.