Progetto “APRS”

Una ricerca teorico-pratica e sperimentale sulla rete radioamatoriale per geolocalizzazione e gestione delle “unità operative” in situazione di emergenza.

Papadopol Lucian-Ioan matricola 320648

May 24, 2025

Contents

1 Introduzione **3**
1.1 Il radioamatore, il nominativo e la motivazione del progetto . . . 3
1.2 Struttura della relazione . . . 4

2 Parte teorica **5**
2.1 Che cos'è l'APRS . . . 5
2.2 Stack ISO/OSI e stack APRS . . . 6
2.2.1 Livello fisico . . . 8
2.2.2 Livello collegamento . . . 12
2.2.3 Livello applicazione . . . 17

3 Parte pratica **19**
3.1 Realizzazione di una stazione APRS sperimentale . . . 19
3.1.1 Il software . . . 19
3.1.2 L'hardware . . . 19
3.2 Interfacciamento via socket con Direwolf . . . 23
3.2.1 Il framing KISS di Direwolf e socket TCP . . . 23
3.2.2 Ambiente di sviluppo e librerie . . . 23
3.2.3 Invio di un messaggio - aprs_sender.py . . . 24
3.2.4 Ricezione e decodifica di una posizione GPS - aprs_decoder_GPS.py . . . 25
3.3 Realizzazione di un digipeater con Raspberry Pi 3 B . . . 27
3.3.1 Hardware . . . 27
3.3.2 Software . . . 28
3.3.3 Installazione e test di funzionamento . . . 28
3.4 APRS-soup . . . 30
3.4.1 Requisiti funzionali . . . 30
3.4.2 Dettagli dell'implementazione . . . 30
3.4.3 Formato e struttura del database . . . 32
3.4.4 Interfaccia utente . . . 33
3.4.5 Workaround per il sistema di mappatura e gestione offline delle tile . . . 34

4 Conclusioni **36**

1 Introduzione

1.1 Il radioamatore, il nominativo e la motivazione del progetto

I radioamatori[1] sono operatori autorizzati che utilizzano sistemi di radiocomunicazione per fini personali, scientifici, sperimentali e di pubblica utilità, nel rispetto delle normative internazionali.
Si distinguono per la loro competenza tecnica, per l'interesse verso le comunicazioni via etere e per l'apertura alla sperimentazione.
A differenza delle comunicazioni professionali o commerciali, quelle radioamatoriali sono svolte su base volontaria e non lucrativa, ma seguono regolamenti ben definiti e richiedono il superamento di un esame teorico e tecnico per ottenere la relativa licenza.

Ogni radioamatore dispone di un nominativo di chiamata[2] univoco assegnato dall'autorità nazionale competente. Questo identificativo, chiamato "call-sign", è utilizzato per tutte le comunicazioni radio ed è formato da una sigla che identifica il paese e una parte alfanumerica personale.
Nel **mio caso**, il nominativo **principale** è IZ6NNH, utilizzato per le attività quotidiane e di sperimentazione, mentre il nominativo IB6P è associato a **operazioni in ambito sportivo**, come i contest radio internazionali.

Il mio percorso nel mondo della radio è iniziato subito dopo la fine delle scuole superiori: affascinato dalla tecnica e motivato da un profondo interesse per l'elettronica e le telecomunicazioni, ho conseguito la patente di radioamatore all'età di 18 anni, immediatamente dopo l'esame di maturità.
Anche il mio percorso professionale ha seguito la stessa direzione.
Per oltre quindici anni ho lavorato come tecnico on-field per aziende in appalto a Telecom Italia, specializzandomi nella realizzazione e manutenzione di reti di comunicazione su rame e tecnologie digitali come PSTN, ADSL, SHDSL e HDSL.
Questa esperienza ha consolidato il mio interesse per le infrastrutture di rete, sia cablate sia radio, rafforzando il legame tra la mia attività lavorativa e la mia passione da radioamatore.

Negli anni ho approfondito in particolare il settore delle comunicazioni dati via radio in condizioni estreme, come ad esempio l'utilizzo di fenomeni naturali quali lo scattering troposferico, oppure tecniche più avanzate come il moonbounce (EME), che sfrutta la Luna come riflettore passivo per rimbalzare segnali radio verso la Terra.
Un ulteriore ambito che trovo particolarmente stimolante è l'utilizzo di ripetitori radio a bordo di satelliti in orbita bassa (LEO) e persino a bordo della Stazione Spaziale Internazionale (ISS).

Questo interesse nasce da una fascinazione per le missioni spaziali, in particolare per il progetto Voyager, che mi ha colpito sin da giovane per la capacità di inviare dati scientifici da distanze siderali utilizzando unicamente segnali radio. L'idea che la radio possa fungere da ponte invisibile verso l'ignoto, anche nelle condizioni più estreme, ha continuato ad alimentare la mia curiosità e il desiderio di sperimentazione.

Proprio per questi motivi ho scelto di dedicare questo progetto allo studio dell'APRS, un sistema che unisce i miei interessi per la radiotecnica, le reti di comunicazione e l'elaborazione di dati in mobilità. APRS rappresenta per me un perfetto punto di incontro tra teoria e pratica, tra la dimensione personale e l'applicazione collettiva della comunicazione via radio.

[1] https://it.wikipedia.org/wiki/Radioamatore
[2] https://it.wikipedia.org/wiki/Indicativo_di_chiamata

1.2 Struttura della relazione

La presente relazione è organizzata in due sezioni principali: una parte teorica e una parte pratica, pensate per fornire sia il contesto concettuale di riferimento, sia l'analisi e documentazione delle attività sperimentali svolte.

La **parte teorica** è dedicata alla descrizione dei fondamenti su cui si basa il progetto.
In essa vengono illustrati il funzionamento del sistema APRS ed il protocollo AX.25 che ne costituisce la base a livello di collegamento.
Viene inoltre presentato un confronto tra i modelli teorici delle architetture di rete, in particolare il modello ISO/OSI e la struttura reale della pila protocollare utilizzata in APRS, con l'obiettivo di evidenziarne le peculiarità.
In questa sezione si chiariscono inoltre i principi di funzionamento dei meccanismi di instradamento dei pacchetti, la funzione dei digipeater, il ruolo del path, e la logica di propagazione delle informazioni all'interno della rete radioamatoriale APRS.

La **parte pratica** è invece focalizzata sull'implementazione e sulla sperimentazione diretta di un sistema APRS completo, impiegando strumenti software e hardware accessibili.
Vengono descritte le fasi di configurazione, test e verifica di funzionamento di:

- una stazione client basata su computer e software TNC Direwolf;
- un digipeater realizzato su piattaforma Raspberry Pi;
- script per la creazione e decodifica di pacchetti APRS;
- un applicazione web completa per mappare le stazioni APRS georeferenziate ricevute;

L'attività sperimentale ha l'obiettivo di analizzare la trasmissione e ricezione di pacchetti APRS dal vero, comprendere in modo pratico il meccanismo di instradamento dei messaggi, comprendere l'uso dei socket TCP in Python e testare le varie possibilità offerte dalla rete APRS.

2 Parte teorica

2.1 Che cos'è l'APRS

L'APRS[1], acronimo di Automatic Packet Reporting System, è un protocollo di comunicazione ideato nei primi anni '90 da Bob Bruninga (WB4APR) per soddisfare la necessità della comunità radioamatoriale di trasmettere dati in tempo reale.
Nato inizialmente come sistema per la localizzazione automatica delle stazioni mobili, si è progressivamente evoluto fino a diventare una piattaforma di comunicazione capace di trasmettere una varietà di informazioni geografiche e testuali.

Per comprendere il significato e la portata dell'APRS è utile inserirlo nel contesto più ampio dei sistemi packet radio, nati a partire dagli anni '80 come evoluzione della telegrafia e fonia tradizionale.
I sistemi packet consentono la trasmissione di dati digitali via radio utilizzando il protocollo AX.25[2], sviluppato come adattamento radioamatoriale del protocollo HDLC[3].
In questi sistemi, la comunicazione avviene generalmente in modalità punto-a-punto, con meccanismi di handshaking, numerazione di pacchetti e ritrasmissione in caso di errore, simili a quanto avviene in una connessione TCP.
APRS si differenzia da questa impostazione tradizionale per la sua natura uno-a-molti e non orientata alla connessione.
Non mira a stabilire una sessione affidabile tra due stazioni, bensì a diffondere informazioni a chiunque sia in ascolto.
Questo approccio lo rende più vicino a un sistema di tipo broadcast, simile a una rete di sensori distribuiti piuttosto che a un sistema di comunicazione bidirezionale convenzionale.

La distribuzione immediata delle informazioni a più destinatari contemporaneamente rende particolarmente utile l'APRS in contesti operativi dinamici come gli eventi sul campo, le situazioni di emergenza o le attività di monitoraggio ambientale.
Il sistema è concepito per essere leggero, veloce e facilmente accessibile anche con mezzi semplici. Non si basa su meccanismi di connessione affidabile come il TCP, ma punta sulla ridondanza e sulla diffusione costante dei pacchetti affinché l'informazione sia ricevuta almeno da qualcuno nel network.
L'assenza di una vera e propria logica di sessione o di trasporto viene compensata dall'efficienza del modello a diffusione.

L'APRS consente l'invio di dati in forma compatta e strutturata, comprendenti:

- la posizione geografica della stazione, rilevata via GPS;
- messaggi testuali brevi, anche confermabili;
- oggetti e simboli visualizzabili su mappe digitali come ambulanze, veicoli, eventi, ostacoli;
- dati meteorologici provenienti da stazioni automatiche;
- bollettini e annunci a diffusione generale.

Queste informazioni possono essere ricevute e visualizzate su software dedicati, che consentono una rappresentazione grafica in tempo reale del territorio e delle entità in movimento o in postazione fissa.

[1] https://en.wikipedia.org/wiki/Automatic_Packet_Reporting_System
[2] https://it.wikipedia.org/wiki/AX.25
[3] https://it.wikipedia.org/wiki/High-Level_Data_Link_Control

L'impiego dell'APRS è molto variegato. Viene usato in ambito radioamatoriale per finalità operative, logistiche e sperimentali.

Tra gli scenari applicativi più comuni si trovano:

- le attività di emergenza e protezione civile, dove è importante tracciare e coordinare squadre di intervento o mezzi mobili;
- gli eventi sportivi o pubblici, per il monitoraggio e il supporto logistico;
- il tracciamento di sonde, palloni aerostatici e veicoli sperimentali;
- la raccolta e diffusione di dati meteorologici in tempo reale da postazioni dislocate sul territorio.

Negli anni il sistema ha conosciuto una larga diffusione a livello internazionale, supportato da numerose implementazioni software per diversi sistemi operativi e integrato in apparati radio dedicati.
Il collegamento con la rete APRS-IS ha poi consentito l'estensione del sistema anche su Internet, offrendo la possibilità di raccogliere e visualizzare i dati trasmessi via radio in forma globale attraverso portali pubblici quali ad esempio aprs.fi[4].

2.2 Stack ISO/OSI e stack APRS

Il modello ISO/OSI, International Organization for Standardization / Open Systems Interconnection, è un riferimento teorico a sette livelli utilizzato per descrivere e standardizzare il funzionamento delle reti di telecomunicazione e dei protocolli di comunicazione.
Ogni livello svolge un ruolo specifico nella trasmissione dei dati da un'applicazione all'altra, attraverso una rete.
Nel contesto del protocollo APRS, la struttura a sette livelli del modello ISO/OSI non viene implementata completamente, bensì APRS si appoggia a una combinazione più semplice di protocolli, in cui solo alcuni livelli dello stack ISO/OSI sono realmente rappresentati.

Il modello ISO/OSI[5]
Il modello ISO/OSI è composto dai seguenti livelli, dal più alto al più basso:

- *Applicazione* – Fornisce servizi all'utente finale (es. email, browser, messaggistica).
- *Presentazione* – Traduce e formatta i dati (es. codifica, compressione, cifratura).
- *Sessione* – Gestisce le sessioni di comunicazione tra due nodi.
- *Trasporto* – Garantisce la consegna dei dati end-to-end (es. TCP, UDP).
- *Rete* – Determina il percorso dei dati nella rete (es. IP, routing).
- *Collegamento dati* – Gestisce l'indirizzamento fisico e l'accesso al mezzo.
- *Fisico* – Trasmette i bit sul mezzo fisico (es. cavo, radio, onde).

Lo stack APRS
Nel sistema APRS, molti di questi livelli non sono presenti o vengono svolti in modo semplificato:

- *Livello Applicazione*: occupato interamente da APRS, che gestisce la codifica dei messaggi, la formattazione delle posizioni GPS, i dati meteo, oggetti sulla mappa, messaggi istantanei, beacon, ecc.

[4] https://aprs.fi/#!lat=43.5697&lng=12.4732
[5] https://it.wikipedia.org/wiki/Modello_OSI

- *Livelli Presentazione, Sessione, Trasporto e Rete*: sono assenti.
  Non esiste cifratura standard, né gestione delle sessioni, né affidabilità end-to-end o instradamento dinamico.
  Il routing è gestito in maniera statica tramite il meccanismo di digipeating.
- *Livello Collegamento Dati*: implementato con il protocollo AX.25, che fornisce indirizzamento con call-sign, controllo degli errori e instradamento attraverso digipeater.
- *Livello Fisico*: si basa tipicamente sulla modulazione AFSK su onde radio in banda VHF/UHF che è il mezzo fisico, utilizzando dispositivi come Terminal Node Controller (TNC) hardware oppure software.

Il seguente grafico mostra un confronto diretto tra i livelli del modello ISO/OSI e quelli implementati nello stack APRS; viene anche evidenziata, per ciascun livello, l'unità di dato associata.

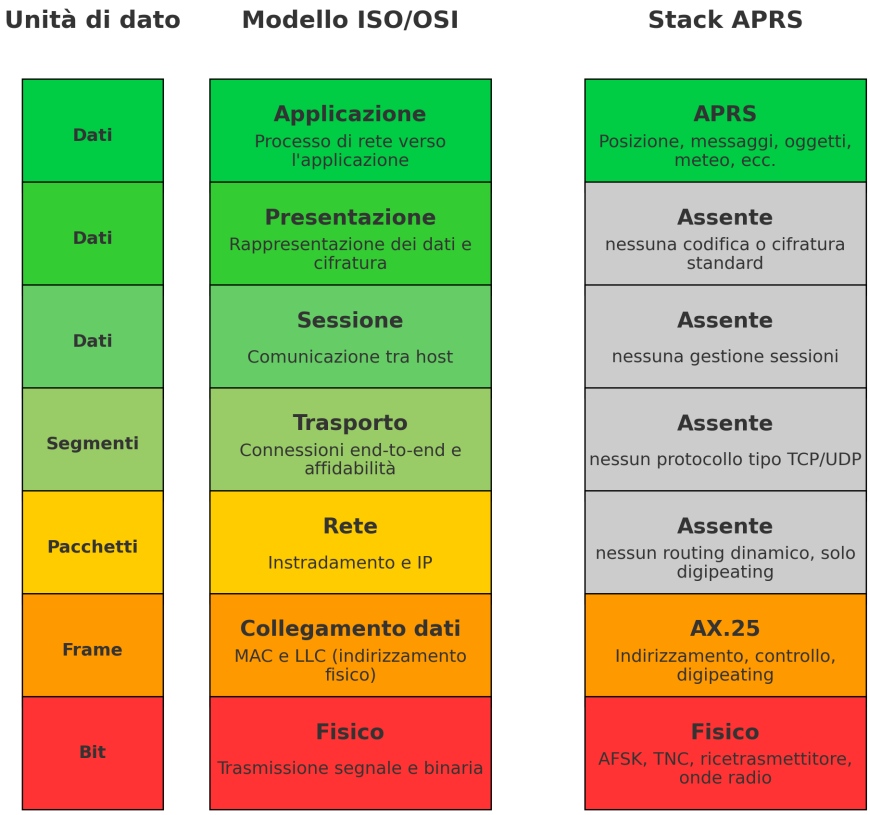

Figure 2.1: Stack ISO/OSI e stack APRS

2.2.1 Livello fisico

Il livello fisico rappresenta l'insieme del mezzo di trasmissione, delle sue proprietà intrinseche e delle caratteristiche tecniche della modulazione impiegata che regolano la trasmissione effettiva dei dati.
Qui trattiamo il mezzo trasmissivo e le sue caratteristiche, la modalità di modulazione, la larghezza di banda utilizzata e i tempi di propagazione del segnale.

Mezzo di trasmissione

L'APRS opera principalmente sulla banda VHF[6] alla lunghezza di onda di 2 metri, in particolare sulla frequenza 144.800 MHz in Europa.

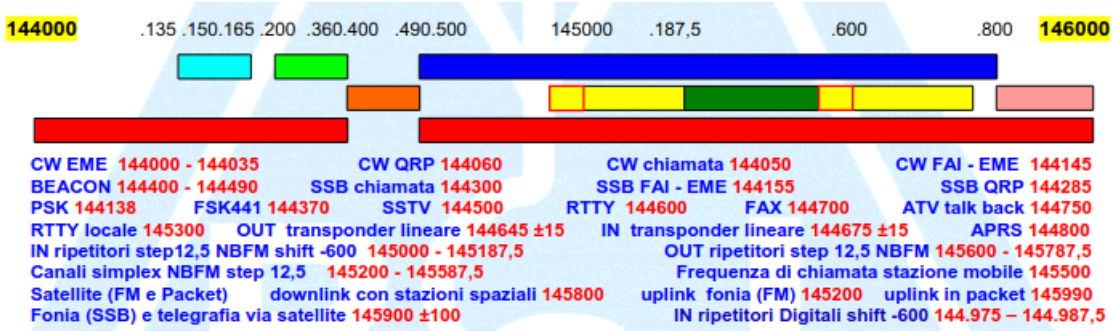

Figure 2.2: Estratto del band-plan VHF & UP. Banda dei 2m da 144Mhz a 146Mhz

Il mezzo di trasmissione è costituito dall'onda elettromagnetica irradiata utilizzando la modulazione di frequenza FM.
Il segnale digitale viene convertito in segnale audio modulato in AFSK[7] ovvero Audio Frequency-Shift Keying, seguendo lo standard Bell 202[8], sfruttando due toni distinti:

- 1200 Hz per il bit 1
- 2200 Hz per il bit 0

Il segnale audio così modulato viene iniettato nel trasmettitore che lo converte in modulazione di frequenza e lo irradia via antenna.
Il collegamento avviene tipicamente in modalità half-duplex, ovvero una stazione può trasmettere o ricevere, ma non contemporaneamente.

[6] https://it.wikipedia.org/wiki/Very_high_frequency
[7] https://en.wikipedia.org/wiki/Frequency-shift_keying
[8] https://en.wikipedia.org/wiki/Bell_202

Larghezza di banda
Utilizzando una velocità di trasmissione standard di 1200 baud[9] con la modulazione AFSK Bell 202, 1 baud corrisponde ad 1 bit/s.
Facciamo qualche calcolo, sfruttando la regola di Carson[10] $B = 2 * (\Delta f + f_m)$
dove B è la larghezza di banda necessaria, Δf è la deviazione massima della portante ed f_mè la massima frequenza audio modulante,
calcoliamo B , sostituendo alle variabili i dati $\Delta f = 3kHz$ e $f_m = 2.2kHz$ otteniamo come risultato $B = 10kHz$.

La trasmissione APRS a 1200 baud, utilizzando modulazione AFSK con toni a 1200 Hz e 2200 Hz, e una deviazione di frequenza di circa 3 kHz in FM, richiede una larghezza di banda tipica di circa 10 kHz, compatibile con i canali FM standard da 12,5 kHz.
Grazie a questa limitata occupazione di spettro, l'APRS riesce a coesistere senza problemi con altri servizi radioamatoriali operanti nella stessa banda.

Ritardo/latenza
Il ritardo o latenza di trasmissione in un collegamento APRS su 144 MHz è composto da diversi contributi:

- il tempo necessario al modem TNC per modulare e preparare il pacchetto;
- il tempo di trasmissione del pacchetto stesso sul canale a 1200 baud;
- il tempo di propagazione dell'onda radio nell'atmosfera;
- il ritardo di accesso al canale.

Considerando ininfluente la velocità di elaborazione del TNC poiché grandemente maggiore rispetto alla velocità di trasmissione di 1200bit/s possiamo calcolare il tempo necessario all'invio di 330byte di dati, che è la dimensione massima del frame AX.25, quindi andiamo a valutare il caso peggiore.
Il tempo di trasmissione di un pacchetto dati è calcolato dalla relazione: $t = n_{bit}/v_{tx}$ dove t è il tempo necessario a trasmettere n_{bit}, n_{bit} è la quantità di bit da trasmettere e v_{tx}è la velocità di trasmissione in bit/s.
Sostituendo i dati a nostra disposizione abbiamo un tempo di trasmissione di 2,2s per trasmettere 2640bit (330byte) alla velocità di trasmissione di 1200bit/s standard.

La propagazione del segnale radio è praticamente istantanea su scala locale, ritardo dell'ordine di microsecondi per distanze fino a 100 km, dunque la latenza percepita è dominata quasi interamente dalla durata della trasmissione.
Dimostro: $t = \frac{d}{c} = \frac{1*10^5 m}{3*10^8} = 333us = 3.3 * 10^{-4} s$

Va poi considerato il ritardo di accesso al canale, dato che le stazioni operano in modalità condivisa con Carrier Sense Multiple Access (CSMA) p-persistente[11].

Hardware
Per permettere la trasmissione e la ricezione di frame AX.25, comunemente chiamati dai radioamatori "pacchetti" dati su onde radio, venne introdotto un dispositivo chiamato TNC, acronimo di Terminal Node Controller.
Il TNC ha rappresentato per lungo tempo l'interfaccia fondamentale tra il computer e la radio.
Collegato al computer tramite una porta seriale RS-232 o, successivamente, via USB, il TNC si occupa di modulare i dati digitali in segnali audio; i segnali audio quindi vengono trasmessi tramite un normale ricetrasmettitore VHF.
In ricezione, il flusso audio proveniente dalla radio viene demodulato dal TNC e riconvertito in pacchetti dati leggibili dal software in esecuzione sul computer.

[9] https://it.wikipedia.org/wiki/Baud
[10] https://en.wikipedia.org/wiki/Carson_bandwidth_rule
[11] https://it.wikipedia.org/wiki/CSMA

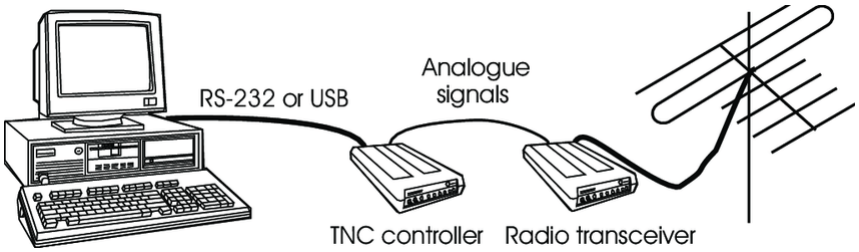

Figure 2.3: TNC hardware

La Figura 2.2 mostra chiaramente i collegamenti fisici: il computer comunica con il TNC attraverso RS-232 o USB, mentre il TNC dialoga con la radio attraverso segnali audio analogici, fino all'antenna che irradia o riceve il segnale radio.
Nel corso degli anni, con l'aumento della potenza di calcolo disponibile anche su macchine di basso costo e la diffusione di schede audio integrate o esterne è stato possibile sostituire completamente l'hardware del TNC con un'applicazione software; questo ha portato allo sviluppo di TNC software, tra cui uno dei più noti e utilizzati è Direwolf.
Direwolf è un TNC software che consente di inviare e ricevere i frame AX.25 utilizzando una normale scheda audio come interfaccia fisica verso la radio.
In pratica, la funzione di modulazione e demodulazione è svolta interamente in software.
Il programma riceve i segnali audio in ingresso dal microfono della scheda audio (o da un'interfaccia dedicata) e li elabora.
Allo stesso modo, può generare segnali audio in uscita per trasmettere dati in modalità AFSK, che la radio invierà sotto forma di onde radio.

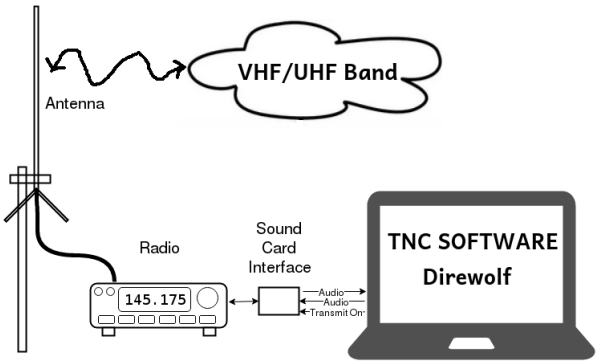

Figure 2.4: TNC software

Ricapitolando una moderna stazione radio operante in APRS è composta da:

- un ricetrasmettitore VHF, anche portatile "hand-held" o modulo custom di comunicazione radio;
- una (o più) atenna VHF omnidirezionale tipo "ground plane" oppure "collineare" o direttiva;
- interfaccia audio radio - PC;
- PC con TNC software "Direwolf";
- software applicativo di cartografia/messaggistica APRS.

Hardware per makers

Oggigiorno non è più necessario disporre di una ricetrasmittente radioamatoriale "classica" per operare in APRS e non è nemmeno più necessario utilizzare un computer "fisso" o "portatile".

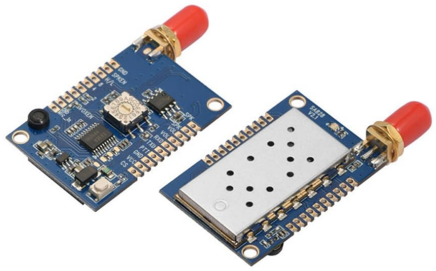

Figure 2.5: Modulo SA828 VHF

Si possono acquistare a costo irrisorio moduli radio rice-trasmittenti "SA828", operanti a soli 5V di alimentazione, di dimensioni molto ridotte e che possono sostituire in toto il complesso radio-antenna poiché dotati di un antenna flessibile da avvitare sul connettore SMA, nella figura 2.5 coperto dal cappuccio rosso.

Questi moduli sono facilmente integrabili in applicazioni custom con sforzo minimo essendo sufficiente alimentarli e programmare la frequenza di utilizzo tramite un interfaccia UART seriale.

I segnali audio di ingresso ed uscita verso la scheda audio ed il comando di trasmissione sono presenti su delle piazzole del circuito stampato rendendo agevole la connessione a single board computer quali RaspBerry Pi et simili. Ciò agevola chi vuole dedicarsi allo sviluppo di un proprio sistema custom per accedere alla rete APRS.
Rimane comunque l'obbligo di avere l'autorizzazione/licenza radioamatoriale e nominativo, per usare le frequenze radio.
Per ricevere solamente ovvero senza trasmettere, chiunque può farlo senza dover obbligatoriamente ottenere la licenza ed il nominativo da radioamatore.

2.2.2 Livello collegamento

Nel contesto dello stack di protocolli APRS, il livello dati corrisponde al secondo livello del modello ISO/OSI e rappresenta lo strato in cui i dati provenienti dal livello applicativo organizzati in pacchetti, vengono incapsulati in frame strutturati, pronti per la trasmissione sul canale fisico.
In APRS, questo livello è gestito dal protocollo AX.25, che definisce il formato e le regole di trasporto dei pacchetti dati via radio.
Tuttavia, per comprenderne il funzionamento, è utile fare riferimento al concetto più ampio di "packet radio", che ne costituisce il paradigma operativo.

Packet radio
Il packet radio è un modello di comunicazione dati che applica i principi della commutazione di pacchetto ai collegamenti radio.
Non è un protocollo specifico, ma un approccio generale che prevede la suddivisione dell'informazione in pacchetti trasmessi in maniera asincrona e indipendente, ciascuno contenente il proprio indirizzamento e i dati di controllo necessari.
Questo modello si contrappone alle trasmissioni a commutazione di circuito , tipiche della fonia su circuito PSTN ad esempio, e consente la creazione di reti dinamiche, decentralizzate e flessibili.
In APRS, l'utilizzo del packet radio consente a ogni stazione di trasmettere dati posizione, messaggi, oggetti, bollettini in forma autonoma, senza necessità di una connessione persistente.
I pacchetti possono essere ricevuti da tutte le stazioni in ascolto e rilanciati da nodi intermedi detti digipeater, che fungono da ripetitori intelligenti e permettono di estendere la portata della trasmissione.
Il packet radio ha introdotto nella pratica radioamatoriale la possibilità di:

- trasmettere dati digitali in modo affidabile su frequenze VHF/UHF o HF,
- creare reti locali e regionali a partire da nodi autonomi,
- integrare l'indirizzamento tramite call-sign radioamatoriale,
- supportare scenari operativi come il tracciamento veicolare, la messaggistica, il monitoraggio meteorologico e l'interscambio automatico di dati.

Il protocollo che implementa in modo concreto queste funzionalità nel livello dati è AX.25.

Protocollo AX.25
Il protocollo AX.25 è l'implementazione effettiva del livello dati nello stack APRS.
Derivato dal protocollo X.25, è stato adattato specificamente per l'uso radioamatoriale e si è affermato come standard per la trasmissione di pacchetti via radio in ambito radioamatoriale.
Esso definisce la struttura dei frame, le regole di indirizzamento, la gestione dei percorsi di ripetizione (path), e i meccanismi di controllo d'errore.

Un frame AX.25 ha una struttura precisa, composta dai seguenti campi:

- *Flag di inizio/fine 0x7E:* delimitano i bordi del pacchetto.
- *Destination address:* questo campo può contenere un nominativo di destinazione APRS. I dati APRS sono codificati per garantire che il campo rispetti il formato standard AX.25 per i nominativi ossia 6 caratteri alfanumerici più SSID.
  Se l'SSID è diverso da zero, specifica un percorso di digipeater APRS generico.
- *Source address:* questo campo contiene il nominativo e lo SSID della stazione trasmittente.
  Se lo SSID è diverso da zero esso può specificare un codice simbolo di visualizzazione APRS.
- *Path:* in questo campo possono essere inclusi da zero a otto nominativi di digipeater.
  Questi indirizzi di digipeater possono essere sovrascritti da un percorso di digipeater APRS generico (specificato nello SSID dell'indirizzo di destinazione).

- *Campo di controllo:* definisce la tipologia del frame, es: informativo, non numerato, ecc.
- *PID Protocol Identifier:* specifica il protocollo per layer 3; per APRS è 0xF0 ovvero nessun protocollo layer 3.
- *Payload:* contiene il payload vero e proprio, come un messaggio APRS o un pacchetto di posizione.
- *FCS Frame Check Sequence:* una sequenza CRC-16 per il rilevamento degli errori.

Nel caso specifico dell'APRS, il frame AX.25 utilizzato è un U-frame ovvero un unnumbered frame, con PID 0xF0, che indica una trasmissione non affidabile e non orientata alla connessione
La struttura complessiva del frame è la seguente:

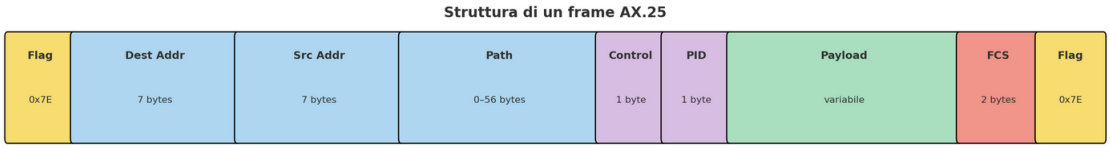

Figure 2.6: Struttura di un frame AX.25

Nota: il payload ha dimensione minima di 1 byte e massima di 256 byte.

Gestione delle collisioni

AX.25 implementa il CSMA ed è p-persistente.

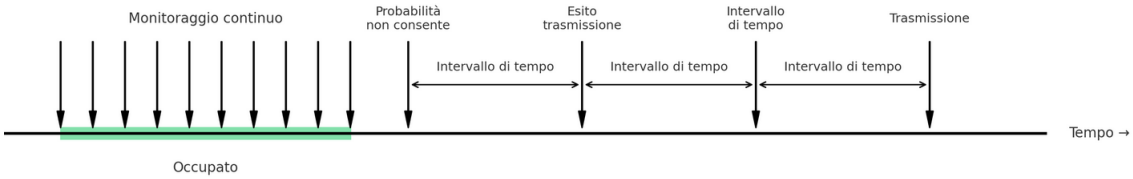

Figure 2.7: CSMA p-persistente

- Se il canale radio è libero, si trasmette subito con probabilità p, oppure rimanda la trasmissione all'intervallo successivo con probabilità q=1-p, se anche questo risultasse libero trasmette altrimenti rimanda, se invece è occupato si comporta come una collisione e attende un intervallo di tempo casuale per poi ritrasmettere;
- Se il canale radio è occupato fin dall'inizio, si attende fino all'intervallo successivo e si ripete l'algoritmo sopracitato;
- Se viene rivelata una collisione, prima di ritentare la trasmissione, si attende un intervallo di tempo casuale.

SSID in APRS

SSID sta per *Secondary Station Identifier* e serve a differenziare più dispositivi o ruoli legati allo stesso nominativo principale oppure a specificare che tipo e quale funzione svolge il dispositivo APRS. Utilizzato nel campo "destinazione" può essere usato per specificare l'instradamento del pacchetto APRS.

SSID nel campo mittente

Specifica che tipo di stazione sta inviando il packetto, consentendo a chi lo richeve di associarvi un icona.

SSID	Significato
0	Stazione principale, generalmente fissa
-1	Stazione addizionale generica, digi, mobile, meteo
-2	Stazione addizionale generica, digi, mobile, meteo
-3	Stazione addizionale generica, digi, mobile, meteo
-4	Stazione addizionale generica, digi, mobile, meteo
-5	Altre reti (Dstar, Iphones, Androids, Blackberry's)
-6	Attività speciale, operazioni via satellite, camping, banda 6 metri
-7	Portatile
-8	Barca o seconda stazione mobile
-9	Stazione mobile principale
-10	Internet, Igates, echolink, winlink, AVRS, APRN
-11	Pallone, aereo
-12	Tracker
-13	Stazione meteo
-14	Autocarri, Camion
-15	Stazione addizionale generica, digi, mobile, meteo

Table 2.1: Tabella codici SSID APRS - campo mittente

Nominativo - SSID	Significato
IZ6NNH-0	Stazione base / generica
IZ6NNH-1	Stazione fissa secondaria
IZ6NNH-7	PDA / palmare / tablet
IZ6NNH-9	Mobile
IZ6NNH-11	APRS via computer
IZ6NNH-13	"WX station" stazione meteo
IZ6NNH-14	Tracker portatile

Table 2.2: Esempio di uso dell'SSID APRS con il mio nominativo IZ6NNH

SSID nel campo destinatario
Lo SSID nel campo Indirizzo di Destinazione di tutti i pacchetti è codificato per specificare il percorso dei digipeater APRS.
Se lo SSID dell'Indirizzo di Destinazione è –0, il pacchetto segue il percorso standard AX.25 dei digipeater ("VIA") contenuto nel campo Indirizzi dei Digipeater dell'frame AX.25.
Se lo SSID dell'Indirizzo di Destinazione è diverso da zero, il pacchetto segue uno dei 15 percorsi generici di digipeater APRS.
Il campo SSID nell'Indirizzo di Destinazione (ossia nel settimo byte dell'indirizzo) è codificato come segue:

SSID	Percorso
-0	Usa il percorso specificato nel campo "path"
-1	WIDE1-1
-2	WIDE2-2
-3	WIDE3-3
-4	WIDE4-4
-5	WIDE5-5
-6	WIDE6-6
-7	WIDE7-7
-8	Percorso verso Nord
-9	Percorso verso Sud
-10	Percorso verso Est
-11	Percorso verso Ovest
-12	Percorso Nord + WIDE
-13	Percorso Sud + WIDE
-14	Percorso Est + WIDE
-15	Percorso Ovest + WIDE

Table 2.3: Tabella codici SSID APRS - campo destinatario

Instradamento e digipeating
Nel sistema APRS, il termine "digipeating", per esteso "digital repeating", indica il processo attraverso cui una stazione radioamatoriale appositamente configurata per svolgere tale ruolo chiamata **digipeater**, riceve, interpreta e successivamente ritrasmette un pacchetto APRS.
Questo meccanismo permette ai pacchetti APRS di superare le limitazioni fisiche del singolo collegamento radio, estendendo la copertura geografica e rendendo possibile la diffusione delle informazioni attraverso una rete ad hoc, costituita da nodi autonomi, i digipeater.
Un digipeater APRS opera secondo il protocollo AX.25 al livello Data Link, utilizzando le informazioni contenute nell'indirizzamento del pacchetto, il campo chiamato "path", per decidere se ripetere o meno il pacchetto ricevuto.

Un pacchetto APRS trasmesso tipicamente contiene una specifica sequenza di identificatori di instradamento all'interno del campo "path", ad esempio:

IZ6NNH–9>APRS,WIDE1–1,WIDE2–2:payload

La sequenza "WIDE1-1,WIDE2-2" rappresenta un tipico path APRS.
Questo path indica che il pacchetto deve essere ripetuto inizialmente da digipeater locali o "fill-in" WIDE1-1, e poi eventualmente da digipeater di rete più estesi WIDE2-2.

Ogni identificatore nel path ha la struttura generale "WIDEn-N", dove:

- WIDE è l'identificativo convenzionale per digipeater generici

- n indica il numero massimo originale di ripetizioni previste
- N rappresenta il numero di ripetizioni ancora consentite, viene decrementato a ogni passaggio

Per esempio, partendo dal path WIDE1-1,WIDE2-2:

1. Il primo digipeater che riceve questo pacchetto e riconosce WIDE1-1 decrementa questo valore a WIDE1-0.
   A questo punto, dato che è arrivato a zero, il pacchetto non può più essere ritrasmesso utilizzando quell'identificatore, che viene sostituito dal nominativo reale del digipeater stesso (es. DIGI1*). Il pacchetto ritrasmesso diventa così:

   IZ6NNH−9>APRS, DIGI1 ∗, WIDE2−2: payload

2. Successivamente, quando il pacchetto raggiunge un secondo digipeater, questo riconosce l'identificatore WIDE2-2, lo decrementa di uno (WIDE2-1) e lo ritrasmette sostituendo il proprio nominativo:

   IZ6NNH−9>APRS, DIGI1 , DIGI2 ∗, WIDE2−1: payload

3. Un ulteriore digipeater, incontrando WIDE2-1, decrementa a zero (WIDE2-0) e sostituisce anche questo con il proprio nominativo. Il pacchetto diventa:

   IZ6NNH−9>APRS, DIGI1 , DIGI2 , DIGI3 ∗: payload

4. Ora il path è esaurito e il pacchetto non può più essere ripetuto ulteriormente.

L'instradamento APRS basato sul digipeating è sostanzialmente differente dall'instradamento dinamico utilizzato nelle reti IP, la tabella 2.4 ci fornisce un confronto immediato.

Instradamento APRS	**Instradamento IP**
È statico e determinato dal mittente tramite il campo path	È tipicamente dinamico e basato su tabelle di routing
Non prevede decisioni dinamiche basate sullo stato della rete o algoritmi di routing complessi	Utilizza indirizzamenti gerarchici, indirizzo IP, per determinare percorsi ottimali
Non implica meccanismi di recupero di errori o ritrasmissioni controllate	Include meccanismi di controllo di errore e recupero tramite protocolli come TCP a livello trasporto.
È broadcast cioè ogni pacchetto inviato è ricevibile da chiunque in ascolto sul canale radio	È prevalentemente unicast, ossia il pacchetto è destinato esplicitamente a un destinatario singolo

Table 2.4: Instradamento APRS vs IP

2.2.3 Livello applicazione

Il livello applicazione nello stack APRS si occupa dell'organizzazione e della gestione dei dati informativi scambiati attraverso la rete APRS. Esso definisce il contenuto logico e il formato delle informazioni che vengono inserite nel campo payload dei pacchetti AX.25 utilizzati dal protocollo.

Formato generale dei pacchetti APRS
I pacchetti APRS hanno la struttura generale seguente

```
CALLSIGN-SSID>DEST,PATH:payload
```

- CALLSIGN-SSID è il nominativo e il tipo della stazione trasmittente
- DEST è tipicamente un identificativo generico utilizzato dal software APRS (ad esempio APRS)
- PATH indica il percorso desiderato per l'instradamento tramite digipeater (ad esempio WIDE1-1,WIDE2-2)
- payload contiene il contenuto informativo (posizione, testo, meteo, ecc.)

Tipi di dato
I principali tipi di dati definiti dal livello applicazione APRS sono:

- Posizioni geografiche
- Messaggi testuali
- Oggetti e simboli
- Dati meteorologici
- Bollettini o annunci generali

Posizioni geografiche
Informazioni sulla localizzazione geografica della stazione, espresse generalmente in coordinate di latitudine e longitudine.
Queste informazioni provengono tipicamente da ricevitori GPS integrati o collegati al sistema APRS.
Esempio di pacchetto posizione

```
IZ6NNH-9>APRS,WIDE1-1,WIDE2-2:!4254.66N/01341.77E>000/000/A=000123
```

- 4254.66N e 01341.77E indicano latitudine e longitudine della stazione IZ6NNH-9
- 000/000 indicano direzione e velocità del movimento (0°/0 nodi)
- A=000123 indica l'altitudine in piedi

Messaggi testuali
Brevi messaggi testuali scambiati direttamente tra due stazioni APRS, con possibilità opzionale di conferma di ricezione.
Esempio di pacchetto messaggio

```
IZ6NNH-9>APRS::IR6A      :Test messaggio APRS{001
```

- IR6A indica la stazione destinataria del messaggio
- "Test messaggio APRS" è il contenuto testuale
- {001 è un identificativo sequenziale che può essere utilizzato per conferma di ricezione

Oggetti e simboli
Sono pacchetti utilizzati per rappresentare specifici elementi sulla rete APRS, come veicoli, stazioni meteo, punti d'interesse, situazioni di emergenza o altre entità identificabili tramite simboli grafici definiti dal protocollo APRS.
Esempio di pacchetto simbolo

```
IZ6NNH-0>APRS:;PUNTO01   *111111z4254.66N/01341.77E-Test oggetto
```

- ;PUNTO01 definisce il nome dell'oggetto
- * indica che l'oggetto è attivo
- 111111z è l'orario UTC di creazione o aggiornamento dell'oggetto
- Coordinate e testo forniscono posizione e descrizione

Dati meteorologici
Dati generati da stazioni meteorologiche automatiche APRS, che includono valori come temperatura, umidità, pressione atmosferica, velocità e direzione del vento, e quantità di precipitazioni.
Esempio di pacchetto meteo

```
IZ6NNH-13>APRS:@111111z4254.66N/01341.77E_180/005
g010t020r000p000P000h50b10150
```

- 180/005 indica direzione e velocità media del vento (180°, 5 nodi)
- g010 è la velocità delle raffiche di vento (10 nodi)
- t020 indica la temperatura (20 °C)
- r000, p000, P000 sono le precipitazioni
- h50 indica l'umidità relativa (50%)
- b10150 è la pressione atmosferica (1015.0 hPa)

Bollettini e annunci generali
Messaggi inviati a tutte le stazioni in ascolto, utilizzati per fornire comunicazioni di carattere generale o avvisi operativi.
Esempio

```
IZ6NNH-0>APRS::BLN1      :Evento APRS oggi ore 17:00 presso sede ARI.
```

dove:

- BLN1 indica un bollettino numerato.
- Il testo seguente contiene l'annuncio generale.

3 Parte pratica

3.1 Realizzazione di una stazione APRS sperimentale

Ho deciso di iniziare a sperimentare la rete APRS accedendovi dal mio laboratorio di casa, quindi in modalità "stazione fissa".
Ciò ha comportato l'acquisizione o la realizzazione e la successiva installazione dell'hardware necessario: antenne, cavi coassiali, ricetrasmettitore e interfaccia radio-PC.
Inoltre, avendo già trattato nella parte teorica di questo lavoro i TNC software, in particolare Direwolf, ne ho compilato il codice e installato il programma sul mio PC Linux.
Di seguito descrivo nel dettaglio le componenti hardware e software.

3.1.1 Il software

L'ambiente software utilizzato è il sistema operativo Linux Ubuntu 24.04, installato su un computer portatile.
L'installazione di Direwolf è avvenuta senza particolari difficoltà partendo dai sorgenti ufficiali, seguendo attentamente la guida ufficiale[1] disponibile online.
La configurazione successiva è stata svolta tramite il file direwolf.conf.
Questo file è stato modificato minimamente, inserendo esclusivamente il nominativo personale IZ6NNH, indicando correttamente la sorgente audio di ingresso e uscita, e impostando il metodo di controllo della linea PTT (quella che manda in trasmissione la radio) tramite il segnale RTS della porta seriale utilizzata per la comunicazione CAT.
L'unico ostacolo incontrato è stato nella selezione della corretta sorgente audio, a causa della presenza di più microfoni integrati nel computer portatile.
Il problema è stato risolto utilizzando i comandi ALSA aplay ed arecord per identificare le designazioni corrette delle varie sorgenti disponibili, procedendo per tentativi fino all'identificazione della sorgente audio corretta.
Inoltre, è stato necessario regolare con attenzione il livello audio in ingresso tramite il mixer di ALSA, evitando saturazioni del segnale audio che avrebbero impedito la corretta decodifica dei pacchetti APRS.

3.1.2 L'hardware

Ricetrasmettitore Kenwood TS-790S
Per la sezione radio è stato utilizzato un ricetrasmettitore Kenwood TS-790S, un apparato multimodo progettato per operare nelle bande VHF/UHF dei 144Mhz, 432Mhz e 1296Mhz, caratterizzato da elevate prestazioni in ricezione ed una potenza di uscita RF di 50W a 144Mhz.
Questo apparato è dotato di interfaccia CAT acronimo di Computer Aided Transceiver, per la gestione automatizzata delle frequenze e modalità operative.

[1] https://raw.githubusercontent.com/wb2osz/direwolf/master/doc/User-Guide.pdf

Figure 3.1: Stazione radio con Kenwood TS790E in cima, TS850S per HF sotto ed interfacce Radio-PC nel pannello a destra

È stato scelto proprio per la sua precisione e stabilità in frequenza, essenziali nelle comunicazioni digitali, nonché per la facilità di integrazione con il sistema informatico tramite la porta seriale CAT, oltre al fatto fondamentale che avevo solo questa radio disponibile per la banda VHF.

Interfaccia audio/CAT/PTT

La connessione tra radio e computer è stata realizzata utilizzando una interfaccia audio/CAT/PTT progettata e realizzata personalmente[2].
Questa interfaccia dal design più che collaudato, si basa su un progetto originale nato per rispondere alle specifiche esigenze dei soci ARI Pesaro per attività radio in digitale, principalmente nelle onde corte.

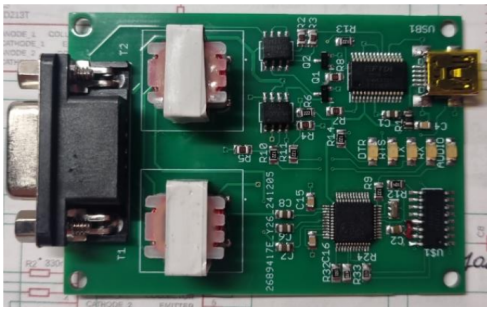

Figure 3.2: Interfaccia Radio - PC

La scheda integra un codec audio dedicato CM108AH che gestisce direttamente la parte analogica audio in ingresso e uscita.
Per la comunicazione seriale CAT è stato utilizzato un chip UART FT232 che, tramite la linea RTS, gestisce anche la commutazione PTT della radio.
Poiché sia il codec audio che il chip UART comunicano con il computer tramite porte USB separate, è stato necessario integrare un chip HUB USB sulla scheda per consentire il collegamento attraverso una singola porta USB al PC.
I segnali audio sono isolati galvanicamente tramite trasformatori di isolamento audio a 600 ohm, mentre le linee CAT e RTS sono isolate mediante optoaccoppiatori, assicurando così una protezione efficace contro disturbi e interferenze provenienti dalla radio o dal computer.

Antenna Yagi GTV2-11LT

L'antenna utilizzata è una Yagi direttiva a 11 elementi, costruita seguendo attentamente il progetto GTV2-11LT [3]disponibile sul sito del radioamatore DG7YBN.

[2]https://www.madexp.it/wp-content/uploads/2021/04/manuale.pdf
[3]http://www.dg7ybn.de/144MHz/144_GTV2_11LT.htm

Questa antenna, progettata originariamente per attività EME acronimo di Earth-Moon-Earth, presenta un guadagno dichiarato di 13 dBd con angoli di apertura del fascio di 37° sul piano orizzontale e 34° sul piano verticale.

Figure 3.3: L'antenna Yagi da 11 elementi per 6m di lunghezza

L'antenna è installata su un rotore Yaesu G-450, che consente la rotazione completa a 360°. Questa possibilità di direzionare l'antenna è essenziale per ottimizzare il collegamento con digipeater APRS distanti, sfruttando al massimo la direttività e il guadagno offerti dalla Yagi.

Test Operativo

La prova finale di verifica dell'intero sistema è avvenuta orientando l'antenna verso il digipeater APRS posizionato sul Monte Titano, nella Repubblica di San Marino.
Questa scelta è stata motivata dalla distanza non eccessiva e dalla nota affidabilità di tale digipeater, rendendolo un banco di prova ideale per ottenere un sicuro successo iniziale.
Al primo avvio di Direwolf, è stato immediatamente possibile ricevere e decodificare con successo il traffico APRS proveniente dal digipeater remoto.
La visualizzazione dei pacchetti APRS in tempo reale nella finestra del software ha rappresentato una prova immediata e convincente della corretta funzionalità sia del sistema hardware che della configurazione software adottata.

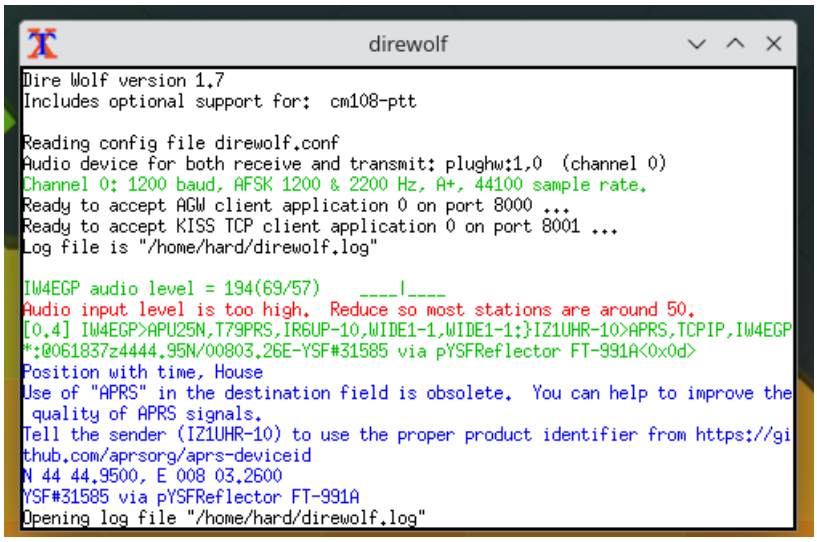

Figure 3.4: I primi pacchetti APRS ricevuti

Questo risultato iniziale conferma l'operatività del sistema, permettendomi quindi di passare oltre ovvero sperimentare con la ricezione ed invio di pacchetti APRS via software scritto da me.
Ruotando l'antenna in varie direzioni ho notato come riuscissi "ad attivare" differenti digipeater, ad esempio

in Emilia, in Abruzzo ed addirittura in Croazia.
Una semplice analisi visiva dei pacchetti APRS mostrati nella schermata di Direwolf mi ha dato indicazione in tempo reale dei nominativi dei ripetitori digipeater che mi "ripetevano" il messaggio da me mandato, poichè il nominativo radioamatoriale per com'è strutturato e standardizzato a livello internazionale ha in se un indicazione geografica di appartenenza ben precisa.
Ad esempio "T79PRS" che è il nominativo del digipeater di R.S.M ha in se l'intestazione "T79" che indica appunto la Repubblica di San Marino mentre, "IW4EGP" indica "IW" per Italia e "4" per "Zona 4" ovvero Emilia-Romagna.

3.2 Interfacciamento via socket con Direwolf

Ho implementato alcuni script Python che si interfacciano con Direwolf sulla sua porta TCP 8001 per inviare e ricevere pacchetti APRS. Gli script in questione sono quattro:

- aprs_sender.py
- aprs_decoder.py
- aprs_sender_GPS.py
- aprs_decoder_GPS.py

Mi limiterò a trattare solo aprs_sender.py e aprs_decoder_GPS.py che ritengo essere esempi significativi poichè il primo è un esempio chiaro di come inviare un pacchetto APRS mentre il secondo permette di ricavare la posizione GPS dai pacchetti APRS ricevuti che è la "base" di una tipica applicazione APRS desktop.
Non volevo appesantire il testo con codice ma questi due script sono veramente brevi e dimostrano il funzionamento base dello script che svilupperò alla fine di questa ricerca, prima di procedere però bisogna specificare che tipo di protocollo usa Direwolf per dialogare con le applicazioni tramite la porta 8001 TCP.

3.2.1 Il framing KISS di Direwolf e socket TCP

All'avvio, Direwolf crea un'interfaccia di comunicazione basata su socket e protocollo TCP, rendendo disponibile un servizio di scambio dei frame AX.25 incapsulati in frame di tipo KISS, separato dall'elaborazione audio.
KISS è l'acronimo di "Keep It Simple Stupid" ed è uno dei due protocolli di framing che Direwolf rende disponibili per la connessione.
Vediamo come Direwolf permetta la connessione da processi differenti anche non neccessariamente sullo stesso host in cui è in esecuzione.
Il funzionamento di questo meccanismo può essere descritto in tre fasi principali:

1. **Apertura della porta TCP 8001.**
   Quando viene lanciato, Direwolf esegue le chiamate di sistema necessarie per creare un socket TCP in ascolto sulla porta 8001 del sistema locale.

2. **Attesa di connessioni da altre applicazioni**.
   Dopo aver messo il socket in ascolto, Direwolf rimane in attesa di richieste di connessione da parte di client esterni.
   Qualsiasi applicazione in esecuzione sulla stessa macchina o, se configurato diversamente, anche da remoto può aprire una connessione TCP verso 127.0.0.1:8001 o l'indirizzo IP scelto.

3. **Scambio di frame KISS.**
   Le applicazioni client che si connettono sulla porta 8001, inviano e ricevono frame AX.25 utilizzando il protocollo KISS

Lo studio e la trattazione del protocollo di framing KISS esula dalla mia ricerca poichè specifico del software Direwolf e non della rete APRS.

3.2.2 Ambiente di sviluppo e librerie

Inizialmente mi sono ispirato agli esempi di programmazione socket in Python presenti nel libro "Reti di calcolatori e internet, un approccio top down - F. Kurose, W. Ross", per scrivere codice che mi permettesse di stabilire una connessione TCP a Direwolf e di recuperare i frame KISS.
Successivamente ho provato a gestire direttamente il framing KISS per estrarne il pacchetto APRS ma, ho scoperto l'esistenza di una libreria Python dedicata appunto al framing KISS che ha notevolmente semplificato sia il mio lavoro sia il codice.
Infine ho sviluppato i miei script per Python 3 avvalendomi delle librerie *"ax253"* per la gestione dei frame AX.25 e *"kiss"* per il framing KISS.

3.2.3 Invio di un messaggio - aprs_sender.py

In questo primo test vogliamo inviare un un generico messaggio di saluto verso la rete APRS.
Dopo l'invio, osservando la finestra di Direwolf dedicata ai pacchetti ricevuti/trasmessi, ci aspettiamo di vedere lo stesso messaggio ripetuto dai digipeater.
Mi aspetto di vedere nella finestra di Direwolf qualcosa come:

```
Trasmesso: IZ6NNH>IR6A,WIDE1-1,WIDE2-2:>Test invio APRS via KISS3 + Python
Ricevuto:  IZ6NNH>IR6A,IR6A*,WIDE2-2:>Test invio APRS via KISS3 + Python
```

Il messaggio che trasmetto è "test invio APRS via KISS3 + Python", specifico il mio nominativo IZ6NNH, specifico come destinatario IR6A che è un digipeater e specifico anche due path di digipeating ovvero WIDE1-1 e WIDE2-2.

Il listato dello script è il seguente:

```python
#!/usr/bin/env python3
"""
Invia un frame KISS via TCP per essere trasmesso e ripetuto.
"""
import os
from ax253 import Frame
import kiss

MYCALL = os.environ.get("MYCALL", "IZ6NNH")
KISS_HOST = os.environ.get("KISS_HOST", "localhost")
KISS_PORT = os.environ.get("KISS_PORT", "8001")

def main():
    ki = kiss.TCPKISS(host=KISS_HOST, port=int(KISS_PORT), strip_df_start=True)
    ki.start()
    frame = Frame.ui(
        destination="IR6A",
        source=MYCALL,
        path=["WIDE1-1,WIDE2-2"],
        info=">Test invio APRS via KISS3 + Python",
    )
    ki.write(frame)

if __name__ == "__main__":
    main()
```

Nell'immagine sottostante si possono vedere i pacchetti APRS ricevuti in verde, trasmessi in rosa e ricevuti e decodificati in formato leggibile da Direwolf in blu.

Figure 3.5: Prima prova di trasmissione

Modificando il codice è possibile inviare le proprie coordinate GPS. Sfruttando il servizio offerto dal sito

aprs.fi ho verificato quindi che presso la mia posizione geografica compaia l'icona della mia stazione APRS a distanza di qualche minuto dalla trasmissione radio, e ciò è effettivamente avvenuto.

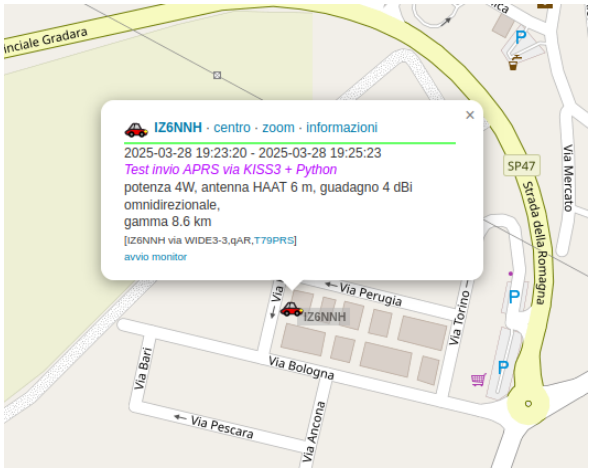

Figure 3.6: Test invio posizione GPS e feedback da aprs.fi

3.2.4 Ricezione e decodifica di una posizione GPS - aprs_decoder_GPS.py

Con questo script ho combinato varie funzionalità non trivali che utilizzerò successivamente per sviluppare una web application con backend Python + Flask:

1. Deframing KISS.
2. Deframing AX.25 ovvero estrazione del payload APRS.
3. Estrazione coordinate GPS dal pacchetto APRS e stampa di debug in formato *"{mittente} > POSIZIONE: {latitudine} {longitudine}"*.

```
#!/usr/bin/env python3
"""
Legge frame KISS da un socket TCP e stampa, per i pacchetti APRS di posizione,
la riga NOMINATIVO > POSIZIONE: lat long
"""

import os
import re
from ax253 import Frame
import kiss

KISS_HOST = os.environ.get("KISS_HOST", "localhost")
KISS_PORT = int(os.environ.get("KISS_PORT", "8001"))

# Regex per estrarre coordinate
POSITION_RE = re.compile(
    r'(?:[!=/@])?'                  # uso futuro?
    r'(?:\d{6}z)?'                  # timestamp zulu (hhmmssz) uso futuro?
    r'(?P<lat_deg>\d{2})'
    r'(?P<lat_min>\d{2}\.\d{2})'
    r'(?P<lat_dir>[NS])'
    r'[/\\]'
    r'(?P<lon_deg>\d{3})'
    r'(?P<lon_min>\d{2}\.\d{2})'
    r'(?P<lon_dir>[EW])'
)

def extract_source(aprs_msg: str) -> str | None:
    """Restituisce il nominativo (campo source) dal messaggio APRS."""
    return aprs_msg.split('>', 1)[0] if '>' in aprs_msg else None

def parse_position(aprs_msg: str) -> tuple[float | None, float | None]:
    """Estrae latitudine e longitudine (gradi decimali) dal messaggio APRS."""
    match = POSITION_RE.search(aprs_msg)
    if not match:
```

```
        return None, None

    try:
        lat_deg = int(match.group("lat_deg"))
        lat_min = float(match.group("lat_min"))
        latitude = lat_deg + lat_min / 60
        if match.group("lat_dir") == "S":
            latitude = -latitude
        lon_deg = int(match.group("lon_deg"))
        lon_min = float(match.group("lon_min"))
        longitude = lon_deg + lon_min / 60
        if match.group("lon_dir") == "W":
            longitude = -longitude
        return latitude, longitude
    except (ValueError, TypeError):
        return None, None

def handle_frame(raw_frame: bytes) -> None:
    """Callback per kiss.TCPKISS.read: deframma, estrae e stampa la posizione."""
    frame = Frame.from_bytes(raw_frame)
    aprs_msg = str(frame)

    source = extract_source(aprs_msg)
    latitude, longitude = parse_position(aprs_msg)

    if source and latitude is not None and longitude is not None:
        print(f"{source} > POSIZIONE: {latitude:.5f} {longitude:.5f}")

def main() -> None:
    """Apre la sessione KISS/TCP e resta in ascolto finchè non riceve Ctrl-C."""
    ki = kiss.TCPKISS(host=KISS_HOST, port=KISS_PORT, strip_df_start=True)
    ki.start()

    try:
        ki.read(callback=handle_frame)
    except KeyboardInterrupt:
        pass
    finally:
        ki.stop()

if __name__ == "__main__":
    main()
```

Il risultato dell'esecuzione di questo script è il seguente:

```
hard@Zenbook:~/Desktop/APRS project/APRS-Soup$ python3 aprs_decoder_gps.py
I4GOS-3 > POSIZIONE: 44.05983 12.55067
I4GOS-3 > POSIZIONE: 44.05983 12.55067
IZ3SLR > POSIZIONE: 45.43150 12.11717
IW4EGP > POSIZIONE: 38.60167 15.90283
IW4EGP > POSIZIONE: 46.02183 7.74583
IW4EGP > POSIZIONE: 38.60167 15.90283
S55YAR-10 > POSIZIONE: 46.43133 14.17517
S55YAR-10 > POSIZIONE: 46.43133 14.17517
IQ4HI-2 > POSIZIONE: 44.28533 11.74500
IQ4HI-2 > POSIZIONE: 44.28533 11.74500
IQ4HI-2 > POSIZIONE: 44.28533 11.74500
IQ4HI-2 > POSIZIONE: 44.28533 11.74500
```

Figure 3.7: Ricezione coordinate GPS

3.3 Realizzazione di un digipeater con Raspberry Pi 3 B

Per approfondire la rete APRS ho realizzato un sistema ad hoc con funzione di digipeater. Ho coinvolto la sezione di Pesaro [4] dell'A.R.I. Associazione Radioamatori Italiani[5].
L'A.R.I è un ente morale fondato da Guglielmo Marconi per unire i radioamatori italiani. La sezione di Pesaro di cui sono socio, mi ha offerto la possibilità di installare il mio digipeater presso il ripetitore VHF IR6A: all'interno dello shelter posso così sfruttare l'antenna omnidirezionale già in quota e l'alimentazione elettrica esistente.

3.3.1 Hardware

Il digipeater è stato ingenerizzato ed assemblato cercando di sfruttare hardware low-cost oppure di recupero facilmente adattabile allo scopo:

1. Ricetrasmettitore hand held tipo "walkie-talkie" Baofeng UV-5R.
2. Single board computer Raspberry Pi 3b e scheda audio USB "generica".
3. Interfaccia radio-pc e sistema di alimentazione realizzati artigianalmente su di una basetta per prototipazione elettronica.

Figure 3.8: Digipeater IR6A con il coperchio aperto per mostrare la sua componentistica

Nella foto si vedono la ventola installata per la ventilazione forzata e il portafusibile che protegge l'impianto da cortocircuiti, evitando possibili principi d'incendio.
Il connettore d'antenna è montato sulla parte superiore del coperchio apribile (non visibile in foto) e raggiunge la radio tramite un breve tratto di cavo coassiale RG-174 terminato con connettori SMA.

[4] https://www.aripesaro.it/
[5] https://it.wikipedia.org/wiki/Associazione_Radioamatori_Italiani
https://www.ari.it/

3.3.2 Software

Per la componente software ho potuto riutilizzare quasi tutto ciò che avevo già sperimentato sul mio PC portatile: sia lì che sul Raspberry Pi eseguo Debian Linux, quindi la procedura è stata pressoché identica.

TNC software – Direwolf
Ho installato nuovamente Direwolf e, nel file direwolf.conf, ho abilitato la funzione DIGIPEAT per trasformarlo in digipeater.
Un'unità systemd (/etc/systemd/system/direwolf.service) avvia l'applicazione al boot e ne assicura il riavvio automatico in caso di crash o di riavvio del sistema.

Accesso remoto senza Internet sul sito
Il sito non dispone di connettività WAN, quindi per gestione e diagnostica ho riconfigurato il Raspberry Pi come access-point Wi-Fi stand-alone:

- Il software RaspAP fornisce un AP 802.11 n e una web dashboard completa.
- Dal parcheggio, mi basta collegarmi alla rete Wi-Fi dedicata, aprire la dashboard o fare login via SSH per interventi da shell. Il servizio Telnet è disabilitato.
- Per sicurezza ho impostato credenziali distinte per l'AP, per la dashboard e per l'utente SSH; inoltre l'autenticazione SSH accetta solo chiavi ed è limitata agli indirizzi della subnet Wi-Fi.

Questa configurazione mi permette di controllare i log di Direwolf o modificarne la configurazione senza dover aprire lo shelter o coinvolgere il custode.

3.3.3 Installazione e test di funzionamento

Il digipeater è stato installato in località Montegaudio, nel comune di Pesaro, municipalità di Monteciccardo, a circa 380 m s.l.m.

Figure 3.9: Lo shelter

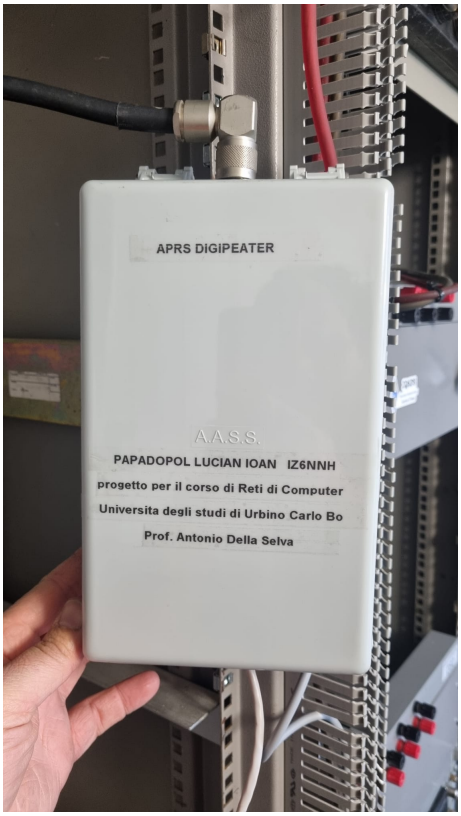

Figure 3.10: Il digipeater

Per valutarne la copertura ho utilizzato la web-app aprs.fi: il servizio raccoglie i pacchetti APRS inoltrati dai digipeater e connessi a Internet, li geolocalizza e li visualizza su mappa insieme a diverse statistiche, in particolare, la funzione "coverage map" disegna una griglia colorata che rappresenta l'intensità del segnale ricevuto nei punti in cui è presente almeno un Internet APRS Gateway. I colori impiegati sono:

- rosso chiaro – segnale ottimo, pacchetti ricevuti con margine;
- gradualmente verso il viola – intensità via via più bassa;
- assenza di colore – nessun dato perché non vi sono ricevitori Internet in quella cella.

Analizzando questa mappa nei giorni successivi all'attivazione ho potuto confermare che il digipeater IR6A-Montegaudio copre l'area urbana di Pesaro, la fascia costiera fino a Fano e parte dell'entroterra marchigiano, con segnali solidi (rosso) nel raggio di 20–25 km e decrescenti (viola) fino a oltre 50 km in linea di vista.

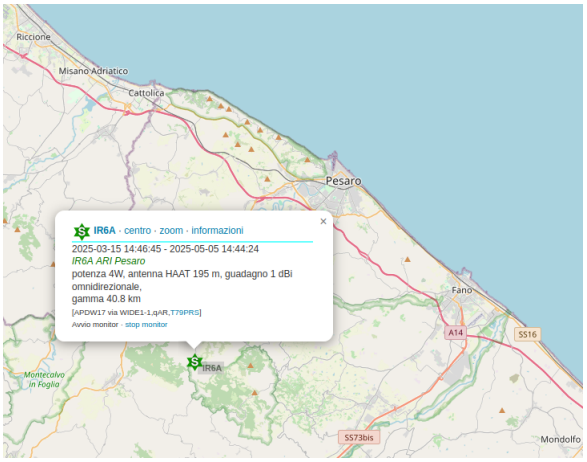

Figure 3.11: Punto geografico dove è situato IR6A

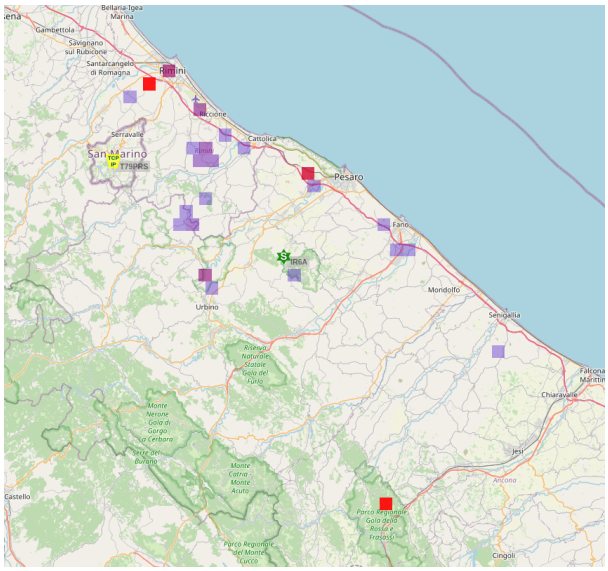

Figure 3.12: Mappa di copertura radio

3.4 APRS-soup

Acquisita la capacità di inviare e ricevere pacchetti APRS e di estrarne le coordinate geografiche, mi sono reso conto di non disporre di un'applicazione in grado di mostrare su una mappa le stazioni APRS ricevute.
Ho quindi pensato di realizzare un piccolo progetto software sperimentale che facesse questo e altro.

3.4.1 Requisiti funzionali

I requisiti di base, oltre alla semplice visualizzazione delle stazioni sulla mappa, sono:

- Portabilità sulle piattaforme Linux e Windows.
- Possibilità di ricevere e inviare messaggi di testo da e verso le altre stazioni.
- Registrazione (logging) dei dati ricevuti per analisi future.

3.4.2 Dettagli dell'implementazione

Architettura software in dettaglio
L'applicazione APRS-soup è stata progettata seguendo un'architettura client-server.
Questa scelta consente una chiara separazione tra l'interfaccia utente (front-end) e la logica di elaborazione dati (back-end).

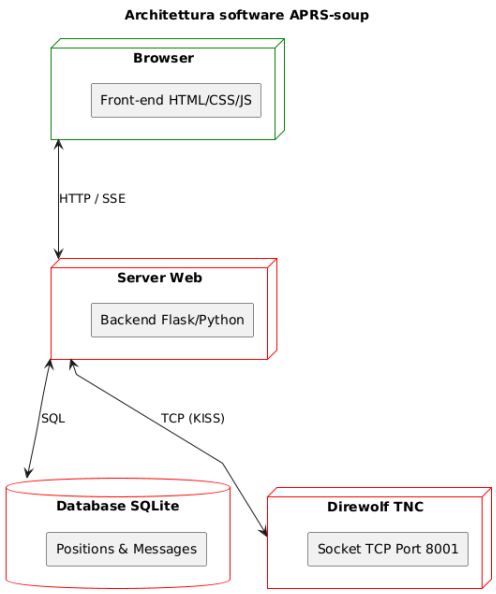

Figure 3.13: Architettura software. In verde il lato Client, in rosso il lato server

Il sistema è composto da tre componenti principali:

- *Direwolf:* utilizzato come modem software gestisce la modulazione e demodulazione dei pacchetti APRS e li espone tramite un'interfaccia TCP usando il protocollo KISS.
- *Back-end Python:* utilizza il framework Flask per offrire un server web che interagisce con Direwolf.
  Il back-end si occupa di decodificare e interpretare i pacchetti APRS ricevuti, registrare le informazioni rilevanti in un database SQLite e aggiornare in tempo reale il front-end attraverso Server-Sent Events (SSE).
- *Front-end Web:* realizzato in HTML, CSS e JavaScript, visualizza su una mappa interattiva le stazioni APRS ricevute e gestisce l'invio e la ricezione di messaggi testuali.

La comunicazione tra il back-end e Direwolf avviene attraverso un collegamento TCP sulla porta 8001.
I dati aggiornati in tempo reale vengono inviati ai client connessi mediante la tecnica SSE, permettendo una visualizzazione immediata e continua delle informazioni.

Funzionamento interno del back-end Python
All'avvio il back-end stabilisce una connessione TCP con Direwolf sulla porta 8001 utilizzando la libreria Python kiss.
I frame AX.25 ricevuti vengono immediatamente processati attraverso la libreria aprslib, che consente di estrarre facilmente i dati APRS contenuti.
Nel dettaglio, quando il back-end riceve un pacchetto AX.25, lo analizza per identificare due principali tipologie di informazioni:

- *Posizioni:* quando nel pacchetto sono presenti le coordinate geografiche (latitude e longitude), queste informazioni sono salvate in una tabella SQLite dedicata (positions) assieme al nominativo della stazione (callsign), SSID e timestamp.
  Il backend notifica immediatamente questa nuova posizione a tutti i client connessi tramite SSE.

- *Messaggi testuali:* quando il pacchetto contiene un messaggio ovvero presenta i campi from, to, e info ma non coordinate, queste informazioni sono salvate nella tabella messages.
  Anche in questo caso, il messaggio viene notificato ai client per aggiornare istantaneamente l'interfaccia web.

Il back-end inoltre espone alcune API JSON per consentire al front-end di recuperare facilmente le ultime posizioni e messaggi ricevuti.

Funzionamento del front-end web
L'interfaccia utente è realizzata tramite un documento HTML (map_template.html) e uno script JavaScript (APRS-soup.js).
Il front-end visualizza una mappa interattiva generata con la libreria JavaScript Leaflet.
Le stazioni ricevute vengono visualizzate dinamicamente sulla mappa, con la possibilità per l'utente di filtrare la visualizzazione in base al tempo ovvero ultimi 15 minuti, 30 minuti, 1 ora, ecc. tramite un apposito controllo presente nell'interfaccia utente.
La comunicazione in tempo reale è gestita sfruttando la tecnologia Server-Sent Events: il front-end apre una connessione persistente verso il back-end Python per ricevere aggiornamenti immediati ogni qualvolta viene rilevato un nuovo pacchetto APRS.
Lo script JavaScript si occupa inoltre della gestione della messaggistica.
L'utente può inviare messaggi APRS inserendo il destinatario e il testo del messaggio in un pannello dedicato.
Questi dati sono inviati al backend Python tramite una chiamata POST.
Una volta inviato, il messaggio è inoltrato a Direwolf, che si occupa della trasmissione vera e propria.

3.4.3 Formato e struttura del database

Per la memorizzazione persistente dei pacchetti APRS ricevuti e per supportare funzionalità di consultazione storica e visualizzazione su mappa, il progetto APRS-soup utilizza un database locale su file basato su SQLite 3.
La scelta è ricaduta su questo motore per via della sua leggerezza, portabilità, semplicità di integrazione con applicazioni Python e assenza di necessità di configurazione lato server.
Il database è costituito da due tabelle principali:

Tabella positions
Questa tabella memorizza le posizioni geografiche delle stazioni APRS ricevute, incluse le informazioni basilari di identificazione.
Ogni record rappresenta una stazione in un dato istante.

Campo	Tipo	Descrizione
id	INTEGER	Identificatore univoco (chiave primaria)
callsign	TEXT	Nominativo della stazione ricevuta
latitude	REAL	Latitudine in gradi decimali
longitude	REAL	Longitudine in gradi decimali
time	TEXT	Timestamp del pacchetto ricevuto

Table 3.1: Tabella positions

Questa tabella è utilizzata dal front-end per mostrare le stazioni sulla mappa.
I dati vengono aggiornati in tempo reale non appena Direwolf, tramite il back-end Python, riceve un nuovo pacchetto contenente informazioni geografiche valide.

Tabella messages
Contiene i messaggi testuali APRS scambiati tra stazioni. Ogni record rappresenta un singolo messaggio inviato o ricevuto.

Campo	Tipo	Descrizione
id	INTEGER	Identificatore univoco
from_call	TEXT	Nominativo della stazione mittente
to_call	TEXT	Nominativo della stazione destinataria
message	TEXT	Contenuto testuale del messaggio
time	TEXT	Timestamp del pacchetto ricevuto

Table 3.2: Tabella messages

Il back-end salva automaticamente ogni nuovo messaggio ricevuto.
La tabella è inoltre interrogabile dal front-end per mostrare lo storico dei messaggi, o per notificare l'utente alla ricezione di nuovi testi.

Gestione del database
L'accesso al database avviene nel back-end Python tramite la libreria sqlite3 di Python.
Al primo avvio del sistema viene automaticamente creato il file database "positions.db" se non già presente.
Le operazioni di scrittura sono effettuate in tempo reale alla ricezione di ciascun pacchetto valido.

3.4.4 Interfaccia utente

L'interfaccia utente di APRS-soup è stata progettata per permettere il monitoraggio in tempo reale delle stazioni APRS ricevute.
Inizialmente, come visibile nella Figura 3.14 che è uno screenshot della prima versione, l'interfaccia mostrava semplicemente alcuni marcatori puntuali posizionati sulla mappa, con i relativi nominativi, utilizzando mappe fornite da OpenStreetMap e renderizzate tramite la libreria JavaScript Leaflet.

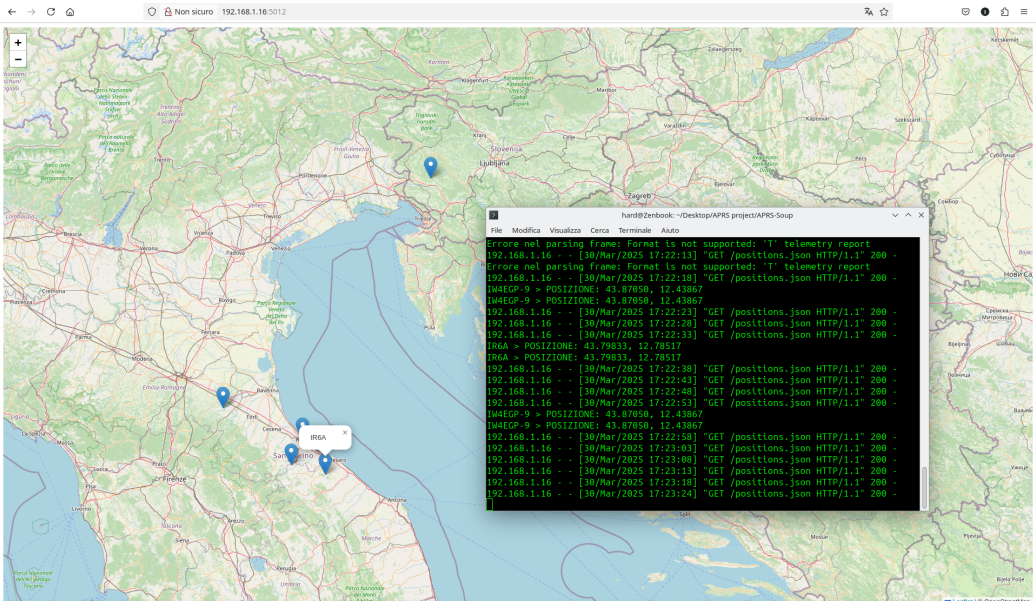

Figure 3.14: Prima versione di APRS-soup

Questa prima versione, sebbene già operativa, aveva funzionalità limitate: i marcatori venivano aggiornati in tempo reale tramite Server-Sent Events ma mancavano interazioni dirette con le stazioni e strumenti di filtro. Successivamente, il progetto è stato ampliato e migliorato, come mostrato nella Figura 3.Y che è uno screenshot dell'interfaccia aggiornata.

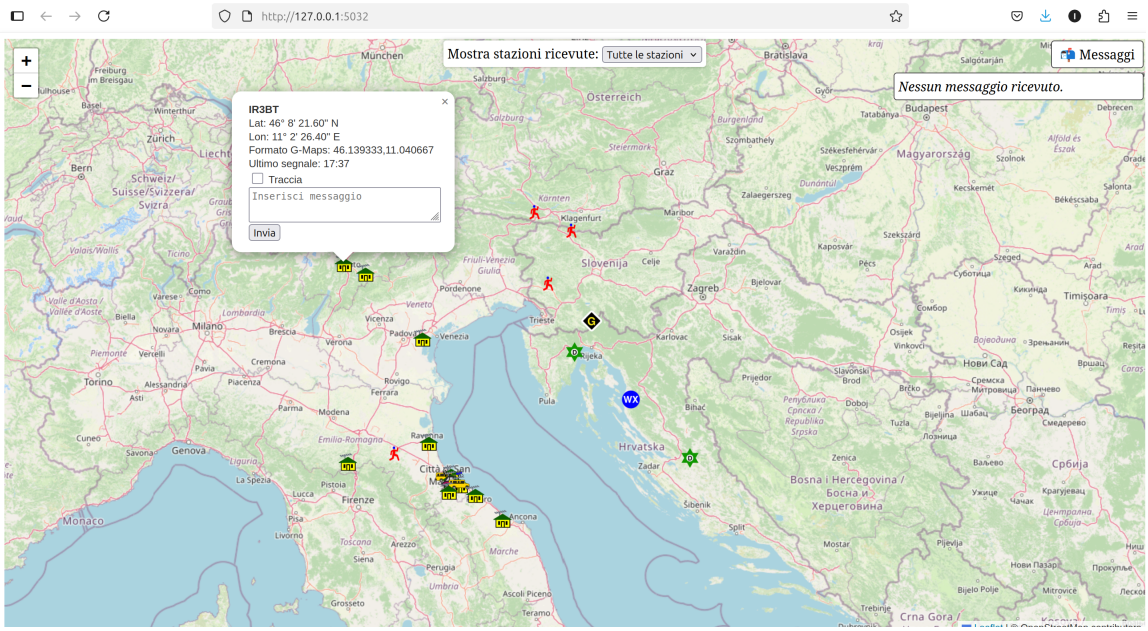

Figure 3.15: Ultima versione di APRS-soup

Le nuove funzionalità introdotte includono un popup interattivo per ciascuna stazione. Cliccando su un marcatore viene aperta una finestra contenente:

- Il nominativo APRS della stazione.
- Le coordinate in formato decimale e in formato gradi, minuti e secondi.
- Il timestamp dell'ultimo pacchetto ricevuto.
- Le coordinate in formato compatibile Google Maps.
- Una casella per l'invio diretto di un messaggio APRS a quella stazione.
- Un'opzione per attivare la tracciatura della stazione nel tempo.

Ho inoltre aggiunto:

- Menù di selezione stazioni: in alto a destra, è stato aggiunto un selettore a tendina che permette di filtrare la visualizzazione tra tutte le stazioni ricevute oppure solo quelle ricevute in un dato intervallo di tempo.
- Pannello messaggi: consente di consultare i messaggi APRS ricevuti.
  Quando un messaggio è disponibile, esso viene mostrato nel pannello stesso, fornendo una modalità semplice per la gestione della messaggistica.
- Icone differenziate: i simboli APRS sono ora rappresentati con immagini distinte ad esempio WX per stazioni meteo, automobili, pedoni, antenne per digipeater etc, permettendo di riconoscere a colpo d'occhio la tipologia di ogni stazione..

L'interazione avviene completamente dal browser, senza necessità di ricaricare la pagina o installare plug-in aggiuntivi.
Tutti gli aggiornamenti sono gestiti in tempo reale tramite eventi push da parte del server, garantendo reattività.

3.4.5 Workaround per il sistema di mappatura e gestione offline delle tile

Le mappe visualizzate nell'interfaccia utente sono basate su un sistema di tile raster pre-renderizzati, organizzati in una gerarchia multistrato in base al livello di zoom.

Ogni mappa è composta da una griglia di immagini .png, ciascuna corrispondente a un determinato riquadro geografico (tile) ad un preciso livello di dettaglio.
Questo approccio è simile a quello adottato dai principali servizi cartografici online.

Tutti i tile necessari sono stati precaricati localmente all'interno di una cartella denominata geo, integrata nel progetto.
Il pacchetto completo occupa oltre 3 GB di spazio su disco, e copre un'area estesa dell'Europa, con un livello di zoom sufficiente per applicazioni APRS su scala regionale e nazionale.

A differenza di altri progetti che utilizzano i tile direttamente da server remoti ad esempio quelli di OpenStreetMap, si è scelto di evitare dipendenze da connessione Internet.

L'obiettivo principale era infatti garantire operatività anche in ambienti privi di accesso alla rete, ad esempio in installazioni remote o scenari di emergenza.
Poiché OpenStreetMap non fornisce direttamente tile scaricabili in formato immagine PNG, è stato necessario elaborare un workaround.

In particolare, sono state seguite le seguenti fasi:

- Download del pacchetto dati vettoriali dell'Europa da MapTiler[6], contenente dati cartografici aggiornati basati su OpenStreetMap in formato vettoriale.
- Importazione nel software GIS open-source QGIS, che permette la manipolazione e visualizzazione dei dati geografici, oltre a offrire strumenti avanzati per il rendering cartografico.
- Utilizzo di un plugin specifico di QGIS per la generazione automatica dei tile raster.
  Il plugin ha provveduto a creare la gerarchia di cartelle in formato compatibile con Leaflet (livello di zoom / X / Y.png) e a esportare i tile in locale.
- Integrazione nel server APRS-soup: il server web è stato configurato per servire i tile locali al browser, evitando quindi richieste esterne e aumentando la reattività dell'applicazione.

[6] https://data.maptiler.com/downloads/europe/

4 Conclusioni

L'obiettivo iniziale di questo progetto era duplice: da un lato comprendere a fondo il funzionamento del sistema APRS e dall'altro sviluppare una stazione operativa in grado di ricevere, decodificare, visualizzare e interagire con i dati APRS tramite strumenti software e hardware progettati, realizzati e configurati in autonomia.

Dal punto di vista tecnico, tutti gli obiettivi prefissati possono ritenersi pienamente raggiunti. La realizzazione della stazione APRS, basata su un sistema Linux con Direwolf come TNC software, ha permesso la ricezione affidabile di pacchetti APRS via radio, la loro decodifica e l'analisi in tempo reale.
L'integrazione con un'antenna direttiva ad alto guadagno ha reso possibile la ricezione di stazioni remote, permettendo verifiche di portata radio e sperimentazione delle modalità di instradamento dei frame.

Il progetto APRS-soup, sviluppato come estensione funzionale, ha rappresentato un'evoluzione significativa rispetto all'idea iniziale.
Inizialmente pensato come semplice visualizzatore cartografico, il sistema si è evoluto in una vera e propria piattaforma APRS locale, dotata di interfaccia web, backend asincrono, memorizzazione su database, gestione dei messaggi e sistema cartografico offline basato su mappe locali.
Tutti questi elementi hanno richiesto competenze in ambiti quali reti di calcolatori, gestione di database, programmazione web e GIS, dimostrando un buon livello di interdisciplinarità e un'applicazione concreta delle conoscenze acquisite nei diversi corsi di questa Laurea in Informatica Applicata.

La portabilità del sistema, la possibilità di utilizzo offline e l'interazione con stazioni reali su vasta scala rendono APRS-soup uno strumento funzionale e potenzialmente estendibile, sicuramente ancora perfettibile.

Dal punto di vista personale, lo sviluppo del progetto ha offerto l'occasione per consolidare e applicare in modo pratico molte delle competenze teoriche acquisite, ma anche per affrontare problematiche reali legate all'integrazione di sistemi eterogenei e alla progettazione di software funzionale, stabile e riutilizzabile.
È stata inoltre un'opportunità per valorizzare competenze pregresse nell'ambito della radiotecnica, combinandole con strumenti software moderni.

In conclusione, il progetto ha rispettato gli obiettivi iniziali, superandoli in termini di complessità e qualità dei risultati. Rimane ampio spazio per futuri miglioramenti ed estensioni, ma la base realizzata rappresenta un sistema funzionale, solido e versatile.