

Facoltà di Ingegneria Corso di Laurea Magistrale in Ingegneria Elettronica

Analisi, studio e prototipazione di un sensore wearable per la geolocalizzazione

Analysis, study and prototyping of a wearable sensor for geolocation

Candidato: Gianni Falleroni

Relatore:

Prof. Ennio Gambi

Correlatore:

Ing. Emiliano Anceschi

Università Politecnica delle Marche Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Elettronica Via Brecce Bianche – 60131 Ancona (AN), Italy



Ringraziamenti

In primis ringrazio il mio relatore prof. Ennio Gambi, per la disponibilità offerta e per avermi permesso di realizzare questo lavoro, aiutandomi e consigliandomi nella scelta del progetto di tirocinio. Ringrazio il mio correlatore, Emiliano Anceschi, per avermi accolto nella sua azienda con entusiasmo, permettendomi di cimentarmi in questo progetto, in un ambiente formativo e stimolante, e per poter continuare a lavorarci anche dopo la laurea. Un ringraziamento particolare all'Ing. Rocco d'Aparo, che mi ha seguito nel progetto, dandomi i suoi consigli di esperto: grazie per le chiamate delle dieci di sera (e non solo), e per essere stato il mio Virgilio nella selva, all'inizio oscura (ora grazie a te un po' più luminosa) del design di PCB. Grazie anche all'Ing. Marco Giammarini, che mi ha seguito nei primi test del firmware. Ringrazio l'Università Politecnica delle Marche, che in questo percorso di cinque anni ha reso possibile oltre alla parte formativa, anche la nascita di nuove amicizie, e la conoscienza di persone fantastiche che mi hanno accompagnato in questo lungo cammino. Quest'ultimo anno sicuramente è stato il più difficile, con l'emergenza Covid che ci ha chiuso tutti dentro casa, ed ha ahimè, abbattuto i contatti umani, ricordo con rimpianto i momenti di divertimento passati al bar della facoltà di ingegneria, tra un caffè e uno scherzo con Matteo, Renat, Il Fedi, Rupo, il Magis, e tanti altri; senza di voi sarebbe stato tutto più noioso! In questo periodo particolare, voglio ringraziare poi l'amico d'infanzia, e della porta accanto Marco, per le chiacchierate e i "drumì" post pausa pranzo, rari momenti di svago, concessi da questa situazione. Ringrazio poi la mia fidanzata Valentina, che ormai da più di 4 anni a questa parte mi sopporta e mi supporta, confortandomi nei momenti difficili. Grazie per aver sopportato ansie, preoccupazioni, frustrazioni e per essere riuscita a farmele dimenticare nei momenti trascorsi insieme, gli unici di spensieratezza e serenità; grazie amore perchè ci sei sempre stata. Infine per ultima, ma non per importanza, ringrazio la mia famiglia, mia madre e mio padre, per tutto quello che mi hanno dato, per avermi sempre supportato e guidato in ogni scelta, e aver sempre creduto in me incondizionatamente, anche quando io stesso ero pieno di dubbi. Fonte di sostegno e di coraggio, grazie per avermi trasmesso la forza e la voglia di raggiungere questo traguardo più di qualsiasi altra cosa; se sono arrivato fino a qui è soprattutto merito vostro.

Loreto, Gennaio 2021

Gianni Falleroni

Sommario

In questo lavoro di tesi si è realizzata l'analisi, lo studio e la prototipazione di un sensore wereable alimentato a batteria, che includesse comunicazione LoraWan, sensore GPS, sensore di distanza Ultra Wide Band(UWB), interfaccia Wi-Fi, interfaccia Bluetooth Low Energy(BLE), comunicazione RFID e circuito di ricarica. Tra le funzionalità principali del dispositivo vi sono quella della localizzazione di precisione, di prossimità e geografica, grazie ai sensori GPS e UWB, nonchè una possibile implementazione del monitoraggio del distanziamento sociale, sfruttando il Bluetooth. Per realizzare il dispositivo sono stati scelti ed integrati, il chip Semtech LR1110[1], che include interfaccia Wi-Fi, sensore GPS e comunicazione Lora; il modulo Decawave DWM1004C[2] che presenta al suo interno il sensore di distanza UWB, ed il chip ST25R95[3] per la comunicazione NFC. Se il modulo DWM1004C è controllato da un processore interno, per il controllo dei restanti dispositivi è stata scelta una MCU della famiglia STM32WB55[4], che fornisce già integrata l'interfaccia Bluetooth. La prima parte del lavoro si è incentrata sulla realizzazione dello schema elettrico e del PCB, mentre nella seconda è stato svolto uno studio e test dei driver dei vari componenti che permettono di implementare le funzionalità dei vari chip, utili per lo sviluppo futuro del firmware del dispositivo, con particolare attenzione al circuito integrato LR1110.

Indice

1	Intro	oduzioi	ne	1
2	Ove	rview o	del sistema	5
	2.1	Schen	na a blocchi	5
	2.2	LR11	10	6
	2.3	DWM	[1004C	10
	2.4	STM3	32WB55	19
	2.5	ST25I	R95	21
3	Scho	ema ele	ettrico	25
	3.1	Realiz	zzazione della libreria	25
	3.2	Shema	a elettrico	28
		3.2.1	Schema elettrico LR1110	28
		3.2.2	Schema elettrico MCU principale	34
		3.2.3	Schema elettrico DWM1004C	40
		3.2.4	Schema elettrico ST25R95	40
		3.2.5	Ricaricatore, LDO e Ship Mode	42
		3.2.6	Componenti aggiuntivi	45
4	Layo	out del	РСВ	49
	4.1	Scelta	dell'enclosure e forma del PCB	49
	4.2	Placer	ment dei componenti	50
	4.3	Stack-	-up e Design rules	54
	4.4	Routi	ng	56
5	Sch	eda pe	r lo sviluppo del DWM1004C	61
6	LR1	110 dr	ivers	65
	6.1	Host-	Controller Interface	65
		6.1.1	Write Commands	65
		6.1.2	Read Commands	65
	6.2	Scans	ione GNSS	66
		6.2.1	Introduzione generale al sistema GNSS	66
		6.2.2	Principi di funzionamento	67
		6.2.3	Driver GNSS	68

Indice

	6.3	Wi-Fi	Passive scanning	82
		6.3.1	Driver Wi-Fi	83
		6.3.2	WifiGetNbResults	87
		6.3.3	WifiReadResults	87
	6.4	Sub G	Hz Radio	91
		6.4.1	Driver Radio	91
		6.4.2	Modem Configuration	100
		6.4.3	LoRa Modem	101
7	Test	LR111	10	111
	7.1	Codice	e C	112
		7.1.1	gnss_scan.c/h	113
		7.1.2	Wi-Fi_scan.c/h	116
		7.1.3	application.c/h	118
		7.1.4	$\mathrm{main}()\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	119
	7.2	Script	Python	120
	7.3	Misure	e effettuate	127
		7.3.1	Indoor/outdoor detection	127
		7.3.2	Outdoor assisted vs. autonomous	130
		7.3.3	Localizzazione Wi-Fi	133
		7.3.4	Localizzazione GNSS vs. Wi-Fi	136
8	Con	clusioni	i	139

Elenco delle figure

2.1	Schema a blocchi del dispositivo
2.2	Principio di funzionamento delle scansioni Wi-Fi
2.3	Diverse fasi delle scansioni Wi-Fi
2.4	Schema GNSS modalità assistita
2.5	Schema base per sfruttare le funzionalità Wi-fi, GNSS e LoRa del
	dispositivo LR1110[1]
2.6	Schema del modulo DWM1004C, riportato dal datasheet [2] 11
2.7	Confronto tra segnale tradizionale e impulso UWB
2.8	Con un segnale costituito da un impulso molto stretto è facile al
	ricevitore distinguere i vari contributi in arrivo
2.9	Schema TDoA
2.10	Schema Two Way Ranging
2.11	Schema a blocchi del front-end a RF[4]
2.12	Schematico semplificato della parte a RF[4]
2.13	Schema ST25R95
3.1	Footprint del dispositivo LR1110 utilizzato nel progetto 25
3.2	Esempio di stencil
3.3	Esempio di serigrafia (linee bianche) e solder mask (verde) che copre
5.5	e protegge il PCB
3.4	Simbolo del dispositivo LR1110
3.5	Parte radio sub-GHz
3.6	Struttura rete di matching degli amplificatori
3.7	Struttura rete di matching RX
3.8	Struttura rete GNSS
3.9	Alimentazione antenna attiva
3.10	Stadio Wi-Fi
	Stadio Wi-Fi
	Alimentazione in modalità DC-DC
	Schema Cristalli MCU
	Schema pins GPIO
5.18	Schema elettrico DWM1004c

Elenco delle figure

3.19 Cristallo ST25R95 e alimentazione
3.20 Comunicazione ST25R95
3.21 IRQ ST25R95
3.22 Struttura necessaria al collegamento dell'antenna
3.23 Schema elettrico rete antenna
3.24 Schema elettrico circuito di ricarica
3.25 Schema LDO
3.26 Schema Ship Mode
3.27 Schema Shipping Mode
3.28 Schema barometro
3.29 Schema memoria
3.30 Schema buzzer
3.31 Connettore per il debug
3.32 Connettore per la tastiera
3.33 Test points
3.34 Connettori sulla board
4.1 Specifiche enclosure [5]
4.2 Outline del PCB
4.3 Antenne sulla scheda
4.4 Placement componenti sulla scheda
4.5 Reti di mactching
4.6 Circuiteria LR1110
4.7 Circuiteria ST25R95
4.8 Circuiteria STM32Wb55
4.9 Top layer e Bottom Layer
4.10 Layer 3 (segnali) e 2 (GND)
4.11 Layer 4 e layer 5
4.12 File gerber per lamina e solder mask
4.13 File gerber per la serigrafia
5.1 Schema elettrico board di sviluppo 6
5.2 LDO utilizzato per alimentare la scheda
5.3 Segnali DWM1004C
5.4 Board di sviluppo DWM1004C
0.4 Board di Svindppo D W M1004C
6.1 Timing delle line e $BUSY$ e SPI durante un Write Command 6
6.2 Timing delle line e $BUSY$ e SPI durante un Read Command 6
6.3 Overview del sistema GNSS
6.4 Struttura NAV message
6.5 Possibile diagramma di utilizzo dei driver
6.6 Sequenza Wi-Fi passive scanning
6.7 Schema della parte radio

6.8	Rx Duty cycle	97
6.9	Rx duty cycle con rilevamento di preambolo	97
6.10	Ordine delle istruzioni da eseguire per configurare il modem	101
6.11	Banda di un segnale LoRa	102
6.12	Formato pacchetto LoRa	102
7.1	Schema a blocchi dell'applicazione realizzata	111
7.2	Impostazione dei pin del microcontrollore con STMCubeMx. $\ \ .$	112
7.3	Log script Python della scansione autonoma per costellazione GPS e	
	BeiDou	128
7.4	Log script Python della scansione assistita per costellazione GPS e	
	BeiDou	128
7.5	Log script Python della scansione autonoma per costellazione GPS	129
7.6	Log script Python della scansione assistita per costellazione GPS	129
7.7	Scansioni autonoma e assistita in ambiente indoor con alcuni satelliti	
	rilevati	130
7.8	Risposta del server	130
7.9	Scansioni autonoma e assistita outdoor	131
7.10	Visualizzazione dei risultati scansioni GNSS	132
7.11	Scansioni autonoma e assistita outdoor con numero massimo di satelliti	
	da ricercare uguale a sette.	132
7.12	Errore GPS Time	133
7.13	Log scansioni Wi-Fi	134
7.14	Visualizzazione delle coordinate ottenute su Google Earth	135
7.15	Visualizzazione delle coordinate ottenute con 4 MAC e 2 MAC	135
7.16	Log scansioni Wi-Fi	135
7.17	Visualizzazione delle coordinate ottenute con GNSS e Wi-Fi. $$	137
7.18	Tracking di un percorso ottenuto con GNSS e Wi-Fi	138

Elenco delle tabelle

2.1	UWB IEEE802.15.4-2011, canali supportati dal DW1000 14
2.2	$\label{eq:continuous} \mbox{UWB IEEE802.15.4-2011, bit rates e PRF supportati dal DW1000.} \ . \ \ 14$
3.1	Collegamento Pins GPIO
3.2	Collegamento Select Serial Interface
6.1	SetGNSSConstellationToUse
6.2	GnssSetMode
6.3	GNSS Autonomous
6.4	GNSS Assisted
6.5	GnssAssistancePosition
6.6	Comando SetTcxoMode
6.7	WiFiScan Command
6.8	Formato risultati base per indirizzi MAC
6.9	Comando WifiGetNbResults
6.10	Comando WifiReadResults
6.11	Descrizione dei bytes del tempo della scansione
6.12	Comando WifiCumulTimings
6.13	Comando SetRfFrequency
6.14	Comando SetRx
6.15	Comando SetTx
6.16	Comando AutoTxRx
6.17	Comando SetRxTxFallbackMode
6.18	Comando SetRxDutyCycle
	Comando StopTimeoutOnPreamble
6.20	Comando SetRxBoosted
6.21	Comando SetPacketType
6.22	Comando SetModulationParam
6.23	Comando SetPacketParam
6.24	Comando SetCadParam
6.25	Comando SetPaConfig
6.26	Comando SetTxParams
7.1	GnssAssisted vs GnssAutonomous
7.2	Valori medi dei consumi e tempi della indoor detection
7.3	GnssAssisted vs Wi-Fi tempi.

Elenco delle tabelle

7.4	GnssAssisted vs Wi-Fi Consumi	137
7.5	GnssAssisted vs Wi-Fi errore	137

Capitolo 1

Introduzione

La digitalizzazione è un fenomeno di portata globale, che sta attraversando tutti i settori dell'impresa, compreso quello delle costruzioni. Per digitalizzazione dell'impresa delle costruzioni si intende l'adozione di tecnologie innovative che permettono di dematerializzare i documenti in informazioni digitali e di trasformare i processi aziendali con l'obiettivo di ottimizzare il business. Abbandonare gli strumenti tradizionali per automatizzare le procedure e rendere più snelli i flussi di lavoro, o impiegare tecnologie innovative per cambiare in meglio il lavoro delle persone, significa produrre vantaggi significativi considerevoli. L'indice DESI, indice di economia e società digitale che monitora il progresso digitale dei paesi membri dell'Unione Europea, posiziona l'Italia al quart'ultimo posto rispetto agli indici di connettività, capitale umano, uso di internet, impiego da parte delle aziende di tecnologie ad alta innovazione e servizi pubblici digitali. Si evidenzia così un gap digitale rispetto agli altri paesi, che se colmato, potrebbe portare a innumerevoli vantaggi. Mentre le grandi aziende sono alle prese con la trasformazione digitale, il tessuto imprenditoriale di dimensioni più piccole dimostra di essere digitalmente immaturo rispetto al vantaggio di assumere un modello strategico fondato su processi digitali automatizzati e sull'impiego di tecnologie innovative. L'emergenza da Covid-19 ha sicuramente spinto sull'acceleratore della digitalizzazione portando allo sviluppo di soluzioni digitali, ma di fatto il mercato chiede un maggiore impegno, poichè introduce a un ritmo sempre più veloce processi innovativi che richiedono competenze da aggiornare continuamente [6].

Cantiere digitale La trasformazione digitale e tecnologica ha investito anche il settore delle costruzioni, con la necessità di semplificare e rendere più efficienti il flusso di informazioni e la gestione dei processi. Il traguardo di questo processo di rivoluzione è il cantiere digitale: un ambiente digitale che mette al centro tutti i dati che riguardano l'opera, le informazioni tecniche di progettazione e realizzazione, la gestione della sicurezza nei cantieri, dei fattori di rischio e criticità, il monitoraggio dell'andamento dei lavori, un flusso organizzato di informazioni accessibili in ogni momento e consultabili da tutte le figure coinvolte. E ancora un luogo fisico di lavoro che diventa esperienza digitale, impiegando dispositivi tecnologici intelligenti che migliorano l'operatività dal punto di vista della gestione del personale e delle risorse. Le metodologie tradizionali di gestione della sicurezza sembrano essere inadeguate e

non più sufficienti per garantire delle condizioni di tutela dei lavoratori. Sempre di più vi è la stringente necessità di una trasformazione digitale. La gestione del cantiere si è sempre basata su un approccio statico e di carattere documentale, con risultati poco efficienti e poco produttivi e con risvolti talvolta gravi in merito alla sicurezza degli operatori. Considerate le normative vigenti e il distanziamento sociale richiesto oggi nell'attività lavorativa (norme anti-contagio da Covid 19), la complessità del cantiere non può più fare a meno di un processo di transizione, già ampiamente affermato in altri settori, per gestire in maniera efficiente la sicurezza. La gestione di quest'ultima necessita di strumenti che possano rispondere alla dinamicità propria delle attività che vi si susseguono. Il cantiere digitale viene definito anche "Cantiere 4.0", per riprendere la terminologia della quarta rivoluzione industriale, ovvero il risultato della trasformazione che si sta verificando nella gestione e nell'operatività di cantiere grazie all'impiego degli strumenti della digitalizzazione e delle tecnologie innovative. Con riferimento a queste ultime, sulla base delle sperimentazioni attuali, si possono riportare alcuni esempi di sistemi innovativi che contribuiscono oggi alla definizione di Cantiere 4.0. Si possono citare ad esempio i sistemi SAPR (Sistemi Aereomobili a Pilotaggio Remoto), tecnologie avanzate applicabili in cantiere quale supporto per il monitoraggio dello stato di avanzamento dei lavori durante le fasi di costruzione e per l'individuazione di possibili interferenze e dei rischi correlati. Questi sistemi offrono flessibilità e stabilità nelle operazioni di volo, consentendo anche sorvoli di prossimità, acquisendo immagini ad alta risoluzione e fornendo un database accurato e adeguato per le elaborazioni successive. Un altro approccio metodologico introdotto di recente è quello basato sull'utilizzo degli smartphone e delle App "Social" per condividere e scambiare informazioni sulla gesione del cantiere. In questo caso esse hanno come obiettivo primario quello di agevolare la gestione documentale, favorire la condivisione dei documenti e la cooperazione tra gli stakeholders in tempo reale e indipendemente da dove essi si trovino[7].

Sistemi IoT Un altro esempio importante sono i sistemi IoT, ambito nel quale si colloca il presente lavoro di tesi, sistemi intelligenti che agiscono mediante sensori applicati alle persone o agli oggetti, che possono essere impiegati per ottenere una sicurezza attiva in cantiere. Questi sistemi permettono di rilevare e individuare persone, mezzi e cose, così da restituire ai responsabili della sicurezza informazioni preventive sulle potenziali situazioni di pericolo, o per segnalare in maniera tempestiva il verificarsi di situazioni di emergenza. Tali sistemi di monitoraggio del cantiere in tempo reale possono riferire circa lo stato di salute degli operatori, monitorando la condizione di "uomo a terra" e al tempo stesso verificare il corretto utilizzo dei dispositivi di protezione individuale da parte dei lavoratori, nonchè fornire l'informazione di distanziamento tra i vari lavoratori.

Adesso più che mai, a causa dell'emergenza Coronavirus, il tema della Smart Safety ha assunto un ruolo decisivo nel settore delle costruzioni, perchè ha costretto a riorganizzare e gestire in sicurezza tutti i luoghi di lavoro. In questo contesto si

possono individuare tre tematiche di interesse: il monitoraggio del distanziamento sociale, il controllo dei dispositivi di protezione individuale e un sistema di controllo intelligente per la riduzione dei rischi. Ad oggi per poter incrementare il livello di sicurezza ottenibile nella gestione delle risorse umane, si può ricorrere ad un tipo di sicurezza attiva, che sfrutta l'utilizzo di tecnologie intelligenti. Tra le tecnologie che possono essere messe a servizio della gestione del cantiere possiamo citarne alcune: Tecnologie dell'Informazione e della Comunicazione (ICT), Internet of Things (IoT), Smart Sensors, Body Area Network (BAN) e Body Sensor Networks (BSN), Proximity Detection (PD), Virtual Reality (VR) e Augmented Reality (AR). Le soluzioni più promettenti sono quelle che si basano sull'impiego di dispositivi elettronici wireless e tecnologie di localizzazione e tracking per il monitoraggio in tempo reale degli operatori che lavorano nel cantiere. Queste soluzioni possono presentarsi anche in forma più complessa, ovvero come piattaforme intelligenti, che vanno ad integrare differenti tecnologie e sensori per ottenere molteplici risposte sulla gestione delle risorse di cantiere. Le informazioni ottenute non riguardano soltanto la geolocalizzazione del personale di cantiere, in quanto tali piattaforme possono:

- restituire parametri di controllo sullo stato di salute (particolarmente rilevante in condizioni di lavoro in aree confinate);
- monitorare il corretto utilizzo dei dispositivi di protezione individuale;
- monitorare le interferenze tra le attività di cantiere;
- gestire gli accessi, compresa l'interdizione delle aree pericolose;
- tracciare gli spostamenti degli operatori, per scongiurare le possibili collisioni coi mezzi impiegati nello stesso tempo in cantiere.

In particolare, si potrebbe pensare ad una piattaforma digitale che controlli in modo digitalizzato che gli addetti ai lavori rispettino le procedure di protezione e prevenzione, indossando i dispositivi di protezione, e che verifichi il loro corretto utilizzo. A tal fine sui DPI degli operatori possono essere installati appositi "Smart Tag", piccoli dispositivi elettronici che comunicano in tempo reale con un dispositivo principale indossato, per riferire la propria presenza nel raggio d'azione del lavoratore. Si viene a creare così una "Wireless Body Area Network" (WBAN) tra questi dispositivi, intorno all'operatore. I dispositivi Tag essendo tecnologia attiva, oltre a segnalare la presenza dell'oggetto associato (DPI) sono in grado di monitorarne il corretto utilizzo, misurando il suo movimento e l'orientazione. Il mancato rilevamento di uno dei DPI previsti o il suo impiego non conforme, sono dati registrati dal dispositivo principale, che a sua volta provvede a inoltrarli alla piattaforma di sistema. Questa si occupa di rilevare i dati in ingresso, analizzarli ed elaborarli, al fine di restituire sull'interfaccia dell'utente le informazioni utili alla gestione dell'evento, piuttosto che le relative allarmistiche, secondo le policy definite per la gestione della sicurezza. Anche per il monitoraggio del distanziamento sociale, si può pensare a tag a basso

Capitolo 1 Introduzione

consumo, integrati in un dispositivo principale, che utlizzino il tempo di volo di un segnale UWB per misurare con precisione centimetrica la distanza tra tutti i device nelle vicinanze. Ancora si possono integrare dei sistemi di check-in automatici nel cantiere, con tag NFC, rilievo di temperatura e identificazione della mascherina sul volto, tramite sensori termici e telecamera, con il vantaggio di non dover porre un addetto per questa mansione.

Questo tipo di soluzione può offrire una serie di vantaggi che vanno dalla riduzione dei rischi (legati a infortuni dovuti ad un utilizzo improprio dei DPI), utilizzo di strumenti non invasivi e di dimensioni ridotte (dispositivi innocui per gli operatori e integrabili con i normali sistemi di fissaggio e strumenti di lavoro, non ostacolando l'operatività di campo) ed infine efficacia ed affidabilità delle informazioni grazie alle logiche implementate a bordo dei dispositivi, che possono segnalare anomalie in tempo reale[7].

Questa tesi, si colloca proprio in questo contesto, e si pone di illustrare le fasi di prototipazione e sviluppo di un nuovo dispositivo wearable, a basso consumo, che includa tutte queste funzionalità: dal monitoraggio dei dispositivi di protezione, check del distanziamento sociale, localizzazione dell'operatore, controllo accessi, e che comunichi tutte queste informazioni ad un nodo gerarchicamente più alto della rete in modo che queste possano essere sfruttate per migliorare le condizioni di lavoro.

Capitolo 2

Overview del sistema

2.1 Schema a blocchi

In fase di sviluppo di un nuovo dispositivo, prima ancora di dedicarsi alla realizzazione dello schema elettrico, si parte dalla definizione di uno schema a blocchi, che permette di visualizzare l'architettura complessiva, nonchè di individuare le componenti principali che definiscono il dispositivo, e le modalità di interfaccia tra i vari blocchi.

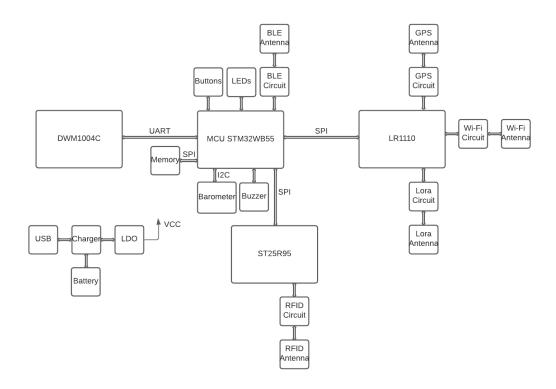


Figura 2.1: Schema a blocchi del dispositivo.

In Figura 2.1 è mostrato lo schema a blocchi del dispositivo. In un primo momento questo era stato pensato solo per avere interfaccia Bluetooth, Wi-Fi e GNSS, ed in seguito è stata aggiunta anche la funzionalità dell'RFID reader. Come si può vedere dallo schema, i componenti principali sono quattro: il modulo DWM1004C, la MCU

STM32WB, il chip LR1110 ed il chip ST25R95. La MCU STM32WB costituisce il microcontrollore principale del sistema e ad esso sono collegati tutti gli altri moduli. Tramite interfaccia seriale SPI, esso comunica con il chip LR1110, mandando il codice delle istruzioni da eseguire. Quest'ultimo implementa le funzionalità di scansione per segnali GPS, scansione Wi-Fi e comunicazione LoRa, ed è collegato alle relative antenne. Il modulo UWB comprende al suo interno il chip DWM1000, controllato da un microprocessore della famiglia STM32L0, il quale comunica con il micro principale tramite UART, per ricevere le istruzioni. Inoltre comprende anche già integrata al suo interno una antenna per le sue operazioni. Il circuito integrato ST25R95 invece è il chip che permette di implementare la tecnologia RFID, comunica tramite SPI con il micro principale ed è collegato alla relativa antenna. Alla MCU principale sono collegati sui GPIO rimanenti una tastiera, utilizzata come sorgente di interrupts, quattro LEDs che servono ad indicare lo stato del dispositivo, un buzzer per emettere eventuali segnali di allarme, un barometro per rilevare l'altezza del dispositivo ed infine una flash esterna, per non avere vincoli di memoria per il caricamento del firmware. In basso a sinistra si nota lo schema per il circuito di ricarica, che fornisce l'alimentazione a tutta la scheda e ricarica la batteria quando il dispositivo viene connesso alla rete.

2.2 LR1110

Come descritto nel datasheet del dispositivo [1], il chip LR1110 è un integrato ultra-low power, che mette a disposizione un ricetrasmettitore LoRa, uno scanner per il sistema satellitare globale di navigazione (GNSS) che supporta diverse costellazioni di satelliti, e uno scanner passivo di indirizzi MAC di access points Wi-Fi, che permettono di realizzare applicazioni di geolocalizzazione. Il chip è configurabile per supportare diverse applicazioni, utilizzando lo standard globale LoRaWAN, oppure protocolli proprietari. L'idea è quella di sfruttare le caratteristiche low-power di questo dispositivo, acquisendo informazioni come segnali satellitari o MAC Wi-Fi, e trasmetterle utilizzando una rete LPWAN al server che fornisce il servizio di geolocalizzazione. In questo modo analizzando le informazioni ricevute, la posizione del dispositivo è calcolata dal server, ottenendo un bilancio unico tra prestazioni e consumo di potenza. Nel dispositivo finale, esso dovrà comunicare tramite protocollo LoraWan con un gateway, che poi inoltrerà le informazioni al Network e Application server. Di seguito sono brevemente descritti i principi di funzionamento e le modalità del circuito integrato, come illustrate in [1] e [8].

Scansioni Wi-Fi Come mostrato in Figura 2.2 l'LR1110 è capace di rilevare access points Wi-Fi b/g/n nelle vicinanze, ed estrarre gli indirizzi MAC, al fine di geolocalizzare il dispositivo. Lo scopo è quello di ottenere almeno 2 indirizzi MAC, che è il numero minimo di indirizzi necessari per poter ottenere l'informazione sulla posizione del device, dopo averli mandati al server, che li confronterà con un database

di posizioni Wi-Fi. Per avere una buona efficienza energetica, solo una piccola parte del pacchetto Wi-Fi sarà catturata e demodulata, quella contenente l'indirizzo MAC. Una scansione Wi-Fi consta di tre parti: la ricerca di un preambolo, cattura e demodulazione, ed ottenimento dell'indirizzo MAC, se sono state rilevate delle reti. Per ottenerne più di uno, queste fasi devono essere ripetute più volte, a seconda di come si configura il dispositivo. L'indirizzo MAC è l'unica informazione necessaria per la localizzazione, ma per aumentare la precisione viene estratto anche l'RSSI che può essere anch'esso inviato al calcolatore. Le scansioni possono essere configurate anche per estrarre il country code di un access point, che è contenuto nel beacon, un frame trasmesso periodicamente da ogni access point, contenente tutte le informazioni della rete, e che serve ad annunciare la presenza della wireless LAN e a sincronizzare i membri del service set.

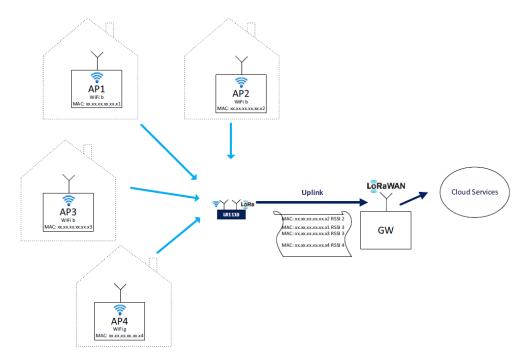


Figura 2.2: Principio di funzionamento delle scansioni Wi-Fi.

Di seguito (Figura 2.3) si possono visualizzare le tre fasi prima nominate: nella fase di ricerca del preambolo il dispositivo entra in modalità ricevitore, finchè l'inizio di un preambolo non è rilevato; in quella di cattura il dispositivo preleva la parte contenente l'indirizzo MAC; e nell'ultima fase demodula l'informazione prelevata.

La prima fase può essere di durata variabile, in base al traffico sul canale: per uno molto trafficato basterà un tempo breve per il rilevamento di un preambolo, al limite 0us se un pacchetto viene subito rilveato; invece per canali dove ad esempio si ha solo un AP (Access Point), ed è generato poco traffico, la finestra può essere lunga quanto l'intervallo beacon impostato per quello specifico AP.

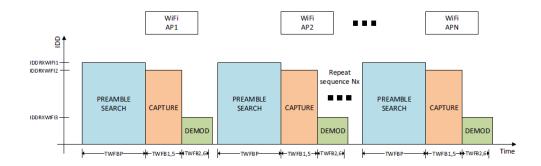


Figura 2.3: Diverse fasi delle scansioni Wi-Fi.

Scansioni GNSS L'LR1110 implementa un veloce e low power scanner GNSS (Global Navigation Satellites System). Il dispositivo cattura una piccola porzione dei segnali broadcast dei satelliti, ed estrae le informazioni richieste al calcolo della posizione - gli pseudorange (ovvero le pseudo-distanze tra satelliti rilevati e dispositivo). Queste informazioni sono assemblate in un NAV message che può essere mandato ad un sistema back-end per il calcolo della posizione. L'LR1110 supporta sia la costellazione GPSL1 e i GPS geostazionari SBAS (EGNOS e WAAS), che BeiDou B1 e BeiDou geostazionari (GEO/IGSO).

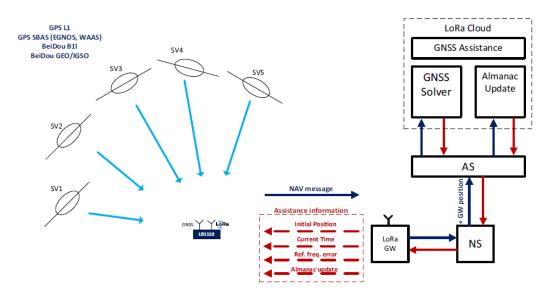


Figura 2.4: Schema GNSS modalità assistita.

Fornendo delle informazioni aggiuntive al dispositivo si possono minimizzare lo spazio di ricerca, il tempo della scansione e l'energia consumata. In particolare il dispositivo supporta una modalità assistita, che prende in ingresso le seguenti informazioni aggiuntive: una stima approssimativa della posizione, il tempo corrente, l'errore in frequenza da compensare ed una versione aggiornata delle effemeridi, richiesta per stimare la posizione dei satelliti visibili al tempo e alla posizione indicata della scansione. Tutti questi parametri hanno un contributo sull'errore totale che

diminuisce all'aumentare dell'accuratezza delle informazioni aggiuntive. Quando un segnale è catturato seguono due fasi di rilievo del segnale: una ricerca rapida dei satelliti ricevuti con un segnale forte, una ricerca più approfondita per satelliti disponibili ricevuti dal dispositivo con un segnale più debole. Nella prima fase si si ottengono gli pseudo-range dei primi satelliti e successivamente si prosegue con la seconda ricerca. Le effemeridi contengono i parametri orbitali dei satelliti, e ne descrivono il moto attorno la Terra. Grazie a queste si possono escludere satelliti irrilevanti e ridurre la finestra di scansione. Le modalità sono quindi due:

- Modalità autonoma, nella quale il dispositivo non necessita di alcuna informazione e riceve i segnali più forti dai satelliti. Questa può essere utilizzata per una classificazione di ambiente indoor o outdoor, concludendo che il dispositivo è indoor se nessun satellite è rilevato. A quel punto si può ricorrere ad altri metodi di localizzazione meno dispendiosi e più adatti, come ad esempio quella Wi-Fi.
- Modalità assistita, nella quale il dispositivo costruisce una lista di 10-12 satelliti da ricercare in base alle informazioni di tempo e spazio fornite al dispositivo. In particolare questa è implementata in due differenti tipologie: una "low power" che si limita ad una scansione all'interno della lista dei satelliti in visibilità, e continua con la ricerca più approfondita solo se almeno un satellite di questi viene rilevato; ed una "best effort" che prosegue con la ricerca anche se nessuno dei satelliti in visibilità viene rilevato. La prima ottimizza i consumi energetici, e può essere usata per una indoor/outdoor detection ottimizzata rispetto a quella della scansione autonoma, riuscendo a classificare con una ricerca su 10-12 satelliti contro 32-35 della autonoma; la seconda invece può essere usata per ambienti più difficili, dove è possibile rilevare satelliti a scapito di un maggiore dispendio energetico.

In conclusione, l'LR1110 permette di implementare un ricevitore GNSS efficiente e veloce, con il quale si può realizzare una classificazione di indoor/outdoor, ma richiede dei componenti aggiuntivi per la geolocalizzazione, demandando ad altre applicazioni l'effort computazionale che farebbe perdere al dispositivo la sua caratteristica di low power. In particolare sono necessari i seguenti tre componenti per completare il sistema di geolocalizzazione:

- Calcolatore della posizione: un'applicazione che a partire dal NAV message calcoli la posizione del dispositivo.
- GNSS Almanac Update: un componente che si occupi di gestire le richieste di aggiornamento delle effemeridi del dispositivo, che devono essere rinnovate almeno ogni trenta giorni.
- GNSS Assistance Component: per operare in modalità assistita, occorre un sistema che fornisca al dispositivo le informazioni richieste.

Di seguito è riportato uno schema base per implementare il dispositivo LR1110 e poter sfruttare le sue funzionalità Wi-Fi, GNSS e RF tranceiver. Bisogna disporre almeno di un riferimento in frequenza a 32Mhz, costituito da un TCXO (Temperature compensated Crystal Oscillator), obbligatorio per le operazioni GNSS, uno switch per la parte del tranceiver, che collegherà l'antenna LoRa al ricevitore e trasmettitore (per il quale sono selezionabili un amplificatore High Power e uno Low Power), con rispettive reti di adattamento; un'antenna GNSS con relativo amplificatore a bassa cifra di rumore e rete di adattamento; ed infine l'antenna per le operazioni Wi-Fi con relativo stage di filtraggio. Chiaramente è necessario anche un microcontrollore esterno a cui l'LR1110 sarà collegato e che invierà tramite interfaccia SPI i comandi al dispositivo.

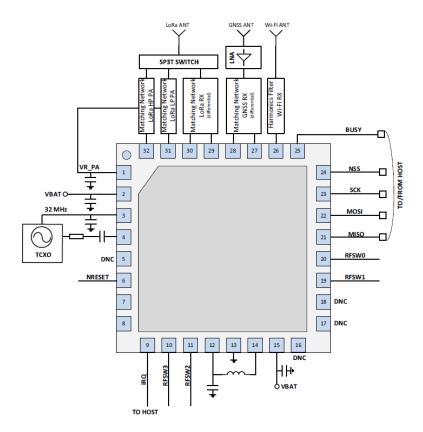


Figura 2.5: Schema base per sfruttare le funzionalità Wi-fi, GNSS e LoRa del dispositivo LR1110[1].

2.3 DWM1004C

Il DWM1004C è un modulo UWB basato sul circuito integrato DW1000, che implementa lo standard UWB IEE 802.15.4-2011. Designato specificatamente per applicazioni TDoA (Time Distance of Arrival) è anche flessibile e si presta ad altre applicazioni, versatile ad ogni scenario UWB possibile come ad esempio il Two Way

Ranging (TWR). Nel modulo è già montata l'antenna necessaria alle operazioni ed il DWM1000 è connesso tramite SPI ad un microcontroller della famiglia STM32L04, che serve per il controllo del modulo UWB. Inoltre collegato al micro, è presente un accelerometro, l'STM LIS3DH (Figura 2.6).

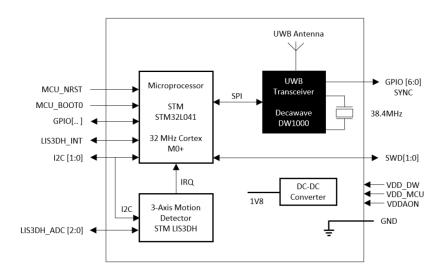


Figura 2.6: Schema del modulo DWM1004C, riportato dal datasheet [2]

Come si evince dal datasheet del dispositivo [2], il modulo non richiede nessun design a RF, in quanto include già l'antenna e tutti i componenti associati ad essa on board. Poichè al tranceiver UWB per le operazioni serve un riferimento in frequenza a 38.4MHz, il cristallo è già integrato nel modulo. Inoltre è presente una memoria Always-On (AON) che può essere utilizzata per trattenere le configurazioni del DW1000 durante le modalità low power della scheda. A seconda delle applicazioni, il modulo potrebbe aver bisogno di una calibrazione. Per fare ciò sono presenti delle funzioni interne che possono essere abilitate, come una trasmissione a onda continua e a pacchetto continuo. Il cristallo è calibrato in fase di produzione, tipicamente l'offset in frequenza dopo questa fase è meno di 3ppm. Per misurare i range in maniera accurata, è necessario un preciso calcolo dei timestamp. Per fare questo è necessario conoscere il delay dell'antenna. Il modulo permette di calibrare questo ritardo. Il valore di default calcolato in fase di produzione viene salvato nella memoria OTP (One Time Programmable). Per la calibrazione, basta misurare una distanza nota utilizzando due moduli, ed aggiustare il delay finchè la distanza calcolata coincide con quella effettiva. I valori delle calibrazioni sono tutte salvate nella memoria OTP.

Localizzazione UWB Il modulo sfrutta la tecnologia UWB, che permette di ottenere una precisione di localizzazione a livello decimetrico. A differenza delle tecnologie radio tradizionali (come Bluetooth o Wi-Fi), l'UWB opera con il cosiddetto tempo di volo (Time of Flight) del segnale, piuttosto che con l'RSSI (Received Signal Strength Indicator), rendendo capaci i dispositivi che sfruttano questo tipo di tecnologia di

Capitolo 2 Overview del sistema

calcolare distanze con una precisione centimetrica/decimetrica. Le proprietà chiavi di questa tecnologia sono:

- Una elevata precisione nella localizzazione
- Non interferisce con altri sistemi di comunicazione radio
- Resistente al multipath del canale radio
- Resistente al rumore
- Può essere implementata sfruttando dei dispositivi low power.

Lo standard che definisce le caratteristiche di questa tecnologia è il IEEE 802.15.4. Questo è uno standard del 2006, poi aggiornato a IEEE 802.15.4a nel 2007 e definisce entrambi i layer MAC (collegamento) e PHY (fisico) del modello di riferimento ISO/OSI. Lo standard originale introdusse nuovi metodi di comunicazione nelle piccole WPAN (Wireless Personal Area Networks), reti che coprono piccole aree e bassi datarate. Queste tecnologie, come ZigBee per esempio, sono ottime per la realizzazioni delle cosiddette reti WSN (Wireless Sensor Network). L'aggiornamento all'802.15.4a aggiunse due nuove metodologie al livello fisico dello standard: CSS (Chirp Spread Spectrum) e l'Ultra Wideband. Queste due aggiunte hanno introdotto nuove caratterestiche, che mancavano nello standard precedente, ovvero data-rate più alti e anche la caratteristica di ranging dell'UWB. Nel 2011 è stata introdotta la versione IEE 802.15.4-2011, servita a revisionare gli standard originali ed includerli in un unico documento. Lo standard specifica tre bande di frequenze, con un totale di 16 canali radio. Dipende poi da ciascuno stato quali di queste bande possono o non possono essere usate, a seconda dell'area geografica in cui ci si trova. Inoltre si specificano 4 data rates per l'UWB PHY: 110kbps, 850 kbs, 6.8 Mbps e 27 Mbps. La tecnologia UWB utilizza un treno di impulsi piuttosto che una sinusoide modulata per trasmettere l'informazione. Questa caratteristica unica la rende perfetta per applicazioni in cui serve determinare delle distanze. Dal momento che l'impulso occupa un'ampia banda, il suo fronte di salita è molto ripido e questo permette al ricevitore di misurare in modo molto accurato il tempo di arrivo del segnale. Inoltre gli impulsi sono molto stretti, tipicamente non più di due nanosecondi.

Grazie alla natura di questi segnali, gli impulsi UWB possono essere distinti anche in ambienti rumorosi, sono resistenti agli effetti multipath (in quanto stretto l'impulso e le sue repliche, sarà facile distinguerli e capire quale è il primo contributo arrivato e da considerare) ed hanno un alta capacità di penetrazione attraverso i muri. Tutte queste caratteristiche conferiscono alla tecnologia UWB un grande vantaggio in applicazioni in cui bisogna misurare delle distanze, rispetto ai classici sengali a banda stretta. Inoltre secondo la maschera spettrale, la potenza di trasmissione è al livello del rumore, il che significa che l'UWB non interferisce con le altre comunicazioni radio operanti alle stesse frequenze, poichè incrementa solo il livello di rumore, in modo molto simile alle tecniche spread spectrum.

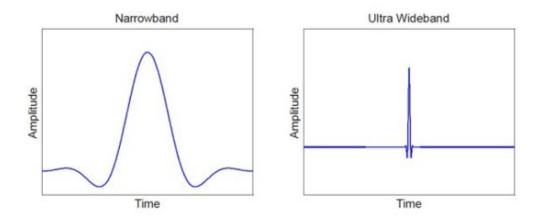


Figura 2.7: Confronto tra segnale tradizionale e impulso UWB.

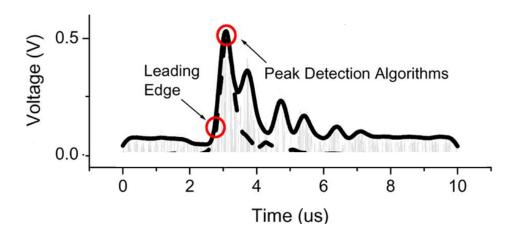


Figura 2.8: Con un segnale costituito da un impulso molto stretto è facile al ricevitore distinguere i vari contributi in arrivo.

Capitolo 2 Overview del sistema

Il circuito integrato DW1000 utilizza sei canali radio in totale, sia nella parte alta che nella parte bassa della banda. Questo consente al modulo di essere flessibile, in quanto se ci sono delle interferenze in un canale, si può facilmente passare in un altro. I canali 1,2,3 e 5 sono definiti da una larghezza di banda di 500MHz, mentre i canali 4 e 7 offrono una banda di più di 1 Ghz. Inoltre usa i primi tre data rates dell'IEEE 802.15.4-2011, ovvero 110 kbps, 850 kbps e 6.8 Mbps (Tabella 2.1 e Tabella 2.2).

Tabella 2.1: UWB IEEE802.15.4-2011, canali supportati dal DW1000.

UWB Channel Number	Centre Frequency(MHz)	Band(MHz)	Bandwidth(MHz)
1	3494.4	3244.8 - 3744	499.2
2	3993.6	3774 - 4243.2	499.2
3	4492.8	4243.2 - 4742.4	499.2
4	3993.6	3328 - 4659.2	1331.2
5	6489.6	6240 - 6739.2	499.2
7	6489.6	5980.3 - 6998.9	1081.6

Tabella 2.2: UWB IEEE802.15.4-2011, bit rates e PRF supportati dal DW1000.

UWB Channel Number	Centre Frequency(MHz)	
16	0.11	
16	0.85	
16	6.81	
64	0.11	
64	0.85	
64	6.81	

Differenti sono le tecnologie capaci di operare una localizzazione indoor. Ciascuna ha i suoi vantaggi e svantaggi riguardo accuratezza, scalabilità, copertura del segnale ecc. Mentre le tecnologie tradizionali come Wi-Fi, Bluetooth e Active RFID ottengono una accuratezza di alcuni metri, l'UWB è più adatto per applicazioni in cui la precisione è un parametro critico. Generalmente ci sono due modi per misurare la distanza tra due dispositivi. Il primo è basato sull'RSSI. Sappiamo che la potenza del segnale diminuisce all'aumentare della distanza dal trasmettitore in modo deterministico e basato su formule pratiche. Con questa assunzione si può stimare la distanza tra ricevitore e trasmettitore, anche se si presentano con questo approccio differenti svantaggi: dal momento che il canale radio cambia continuamente, lo stesso accade per il parametro RSSI, portando così inaccuratezza al sistema. Inoltre l'RSSI può essere degradato anche dalla propagazione multipath e altri fenomeni comuni del

canale radio (selettività in frequenza ecc.). I risultati possono essere migliorati da un processo di stima del canale, tuttavia a causa del cambiamento rapido dell'ambiente, questo deve essere fatto frequentemente per migliorare i risultati. Le teconologie come Wi-Fi e Bluetooth, si basano proprio su questa tecnica. Il secondo metodo invece si basa sull'uso del Time of Flight del segnale. Questo porta a una migliore accuratezza del risultato in ambienti Line of Sight, e grazie ad esso si può ottenere la precisione a livello centimetrico, a seconda della frequenza e della natura del segnale utilizzato. Questo secondo metodo è utilizzato dalla tecnologia UWB: combinando la misura del tempo di volo da differenti dispositivi, si può ottenere una posizione accurata del device che si vuole localizzare. Le performance possono essere degradate in assenza di una linea di vista tra i dispositivi, ma rimangono comunque migliori a quelle del metodo utilizzante l'RSSI. Generalmente per realizzare la localizzazione di nodi mobili, c'è bisogno di fare riferimento ad un numero di nodi con posizione nota, chiamati "Anchors". Tipicamente ne servono almeno tre per operare una localizzazione in due dimensioni, invece almeno quattro non disposti sullo stesso piano per una localizzazione in tre dimensioni. La disposizione degli "Anchors" deve essere tale per cui almeno quattro di esse siano sempre nel range di comunicazione del nodo mobile "Tag", che si vuole localizzare. L'area di copertura dipende dal data rate e dalla lunghezza del preambolo, la scelta dei quali è influenzata dalla densità dei nodi e dai consumi di potenza. Gli schemi principali per sistemi di localizzazione UWB sono due e sono lo schema Time Difference of Arrival (TDoA) e Two Way Ranging (TWR).

Time Difference of Arrival Lo schema TDoA è basato su una misura precisa della differenza di tempo di arrivo (TDoA) di un segnale che giunge a diversi dispositivi Anchors. Il tag manda periodicamente un messaggio "blink", che viene ricevuto dai nodi fissi nelle sue vicinanze. Quando questi hanno il clock sincronizzato e il tempo di arrivo del blink message può essere confrontabile, prendendo un nodo anchor come riferimento e confrontando la differenza dei tempi di arrivo a ciascun anchor del sistema, si definiscono delle superfici iperboliche sulle quali giace il tag. La localizzazione 3D si ottiene intersecando queste superfici ottenute dalla misura di almeno quattro TDoA. Ovviamente in questo schema il punto cruciale è che gli Anchors devono essere accuratamente sincronizzati: questo può essere ottenuto ad esempio con una distribuzione cablata dello stesso clock a tutti i dispositivi, oppure tramite tecniche wireless. La prima risulta vantaggiosa quando si ha un'alta densità di tags, e permette ai nodi fissi di rimanere sempre in ascolto dei blink message, senza la possibilità di perderne qualcuno. La seconda prevede che alcune Anchors prescelte, e con copertura ottima dette "Master Anchors", mandino dei segnali broadcast di sincronizzazione. Le Anchors nell'area di copertura del Master definiscono la cella in cui tutte sono sincronizzate per un certo periodo di tempo. Il rischio è che alcuni di questi messaggi potrebbero collidere con i blink message, tuttavia si ha il vantaggio di non dover cablare il sitema. Questo sistema basato sul TDoA ha diverse

caratteristiche favorevoli:

- I tag non comunicano singolarmente con un solo anchor e quindi si può prolungare la vita della batteria, in quanto è sufficiente mandare un solo messaggio per essere localizzati (comparato ad un numero di messaggi variabile richiesti in un sistema TWR, in cui la comunicazione avviene con ciascun Anchor singolarmente).
- Poichè non c'è una associazione tra tag e anchor, come nel caso TWR, il numero di anchors è totalmente scalabile.
- I tags impiegano solo una piccola porzione di tempo per mandare il messaggio blink, e quindi si possono avere un grande numero di tags nel sistema, anche qui scalabili a seconda delle necessità.

Dunque i tags trasmettono ad intervallo regolare un breve blink message che è processato da tuti gli anchors nel range di comunicazione. Grazie alla loro sincronizzazione temporale queste calcolano tutte un timestamp di arrivo del messaggio, e possono determinare la posizione del device con una tecnica di multilaterazione. Per calcolare la posizione del Tag servono almeno tre timestamps di tre Anchors sincronizzate.

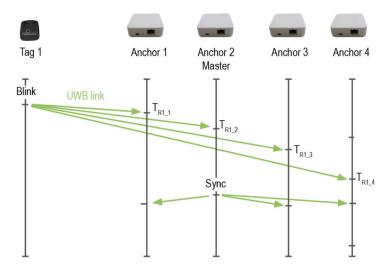


Figura 2.9: Schema TDoA.

Considerando un Tag che emette un segnale in posizione ignota (x, y, z) e quattro Anchors con posizioni note (A1, A2, A3, A4), il tempo impiegato dal segnale a raggiungere ogni ricevitore dal punto ignoto può essere ricavato dividendo lo spazio per

la velocità del segnale, approssimata alla velocità della luce c:

$$T_{1} = \frac{1}{c} \cdot \sqrt{(x - x_{1})^{2} + (y - y_{1})^{2} + (z - z_{1})^{2}}$$

$$T_{2} = \frac{1}{c} \cdot \sqrt{(x - x_{2})^{2} + (y - y_{2})^{2} + (z - z_{2})^{2}}$$

$$T_{3} = \frac{1}{c} \cdot \sqrt{(x - x_{3})^{2} + (y - y_{3})^{2} + (z - z_{3})^{2}}$$

$$T_{4} = \frac{1}{c} \cdot \sqrt{(x - x_{4})^{2} + (y - y_{4})^{2} + (z - z_{4})^{2}}$$

$$(2.1)$$

Se definiamo il quarto Anchor come origine del sistema allora

$$T_4 = \frac{1}{c} \cdot \sqrt{x^2 + y^2 + z^2} \tag{2.2}$$

Quindi le differenze dei tempi di arrivo (TDoA) rispetto al riferimento scelto saranno:

$$\tau_{1} = T_{1} - T_{4} = \frac{1}{c} \cdot \left(\sqrt{(x - x_{1})^{2} + (y - y_{1})^{2} + (z - z_{1})^{2}} - \sqrt{x^{2} + y^{2} + z^{2}} \right)
\tau_{2} = T_{2} - T_{4} = \frac{1}{c} \cdot \left(\sqrt{(x - x_{2})^{2} + (y - y_{2})^{2} + (z - z_{2})^{2}} - \sqrt{x^{2} + y^{2} + z^{2}} \right)
\tau_{3} = T_{3} - T_{4} = \frac{1}{c} \cdot \left(\sqrt{(x - x_{3})^{2} + (y - y_{3})^{2} + (z - z_{3})^{2}} - \sqrt{x^{2} + y^{2} + z^{2}} \right)$$
(2.3)

Dove (x_1, y_1, z_1) (x_2, y_2, z_2) e (x_3, y_3, z_3) , sono le posizioni delle Anchors note a priori e c è la velocità del segnale, approssimata con quella della luce. Risolvendo il sistema in (x, y, z) si individua un'unica soluzione, che rappresenta la posizionde del Tag che ha mandato il segnale.

Two Way Ranging Questa tecnica determina il tempo di volo del segnale UWB e calcola la distanza tra i nodi moltiplicando il tempo per la velocità della luce. Il metodo può essere applicato tra un Anchor e un Tag prescelto, oppure due tag, e coinvolge due dispositivi alla volta. Ovviamente per la localizzazione il procedimento dovra essere ripetuto per diversi Anchors presenti nel sistema. Ciascuna distanza definisce una superficie sferica su cui giace il Tag, centrata nell'Anchor. La localizzazione 3D è ottenuta intersecando le superfici sferiche ottenute dalla misura del TOF (Time Of Flight) di almeno quattro Anchors.

Per misurare la distanza è necessario uno scambio di tre messaggi: il Tag detto "initiator" manda un "Poll message" ad un certo anchor al tempo TSP (Time of Sending Poll). L'anchor registra il TRP (Time of Poll Reception) e risponde al tag al tempo TSR (Time of sending Response). Ricevuta la risposta al tempo TRR (Time of Receiving Response) il Tag prepara un messaggio finale con il suo ID, TSP,TRR e TSF (Time of sending Final message). Basandosi sul TRF (Time of Receiving Final message) e sulle altre informazioni contenute nel messaggio, l'Ancora determina il

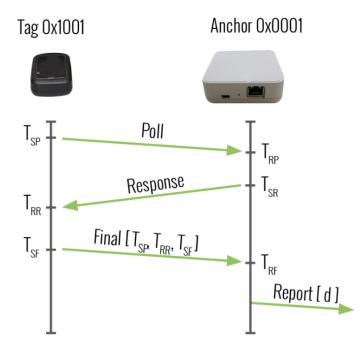


Figura 2.10: Schema Two Way Ranging.

tempo di volo del segnale UWB. Opzionalmente il risultato della distanza può essere rimandato indietro al Tag, o a qualsiasi altro dispositivo UWB nelle vicinanze. Di seguito sono riportate le relazioni per il calcolo della distanza:

$$D = ToF \cdot c \tag{2.4}$$

$$ToF = \frac{\left[(TRR - TSP) - (TSR - TRP) + (TRF - TSR) - (TSF - TRR) \right]}{4} \tag{2.5}$$

Un primo vantaggio di questa tenica è chiaramente la possibilità di localizzare un dispositivo senza il bisogno della sincronizzazione degli Anchors, che è invece il punto cruciale nella tecnica TDoA. D'altro canto il sistema è meno scalabile dell'altro, infatti ciascun Tag deve conoscere il range di indirizzi dei vari Anchor, poichè gradualmente invierà i suoi messaggi a tutti quelli presenti, e deve quindi anche conoscere quali anchors sono in prossimità. Per il calcolo della posizione sarà necessario che il Tag invii il proprio messaggio individualmente a ciascun Anchor nel sistema, risultando in un dispendio di energia maggiore, e un tempo variabile della procedura, a seconda della grandezza del sistema. Se risulta più dispendioso per la localizzazione tuttavia, questo metodo si potrebbe prestare per check sulla distanza reciproca tra due tag, per monitorare il distanziamento sociale, soprattutto se entrambi i terminali sono mobili.

In un sistema RTLS (Real Time Localization System) l'accuratezza dei timestamps ottenuti dal dispositivo DW1000, possono dare una risoluzione sotto ai 10cm. Tuttavia il posizionamento degli anchors rispetto al tag potrebbe degradare la precisione della posizione. L'ascolto e la ricezione dei messaggi è la fase più dispendiosa in termini di energia per il DW1000. Per i dispositivi che devono rimanere sempre in ascolto e che sono alimentati a batteria, occorre introdurre degli schemi per ridurre in qualche modo il tempo di ascolto. Una soluzione potrebbe essere quella di mettersi in ascolto per delle brevi finestre temporali alla ricerca di un preambolo, e se questo non viene rilevato tornare alla low power mode; oppure un'altra può essere quella di mandare un messaggio ed attivare una breve finestra di ricezione per una eventuale risposta.

2.4 STM32WB55

Dalle caratteristiche del microprocessore STM32WB55 riportate nel datasheet [4], esso offre una interfaccia potente e ultra-low-power, compatibile con il Bluetooth Low Energy e lo standard IEEE 802.15.4-2011. Il micro è designato per operare in condizioni estreme di low-power, ed è basato su un core Arm Cortex-M4 a 32-bit RISC (CPU1), che lavora ad una frequenza fino a 64MHz. Il vantaggio di questo dispositivo è che integra già al suo interno un processore multi-standard e ultra-low-power per gli standard Bluetooth. Lo stack Bluetooth Low Energy gira su un core Arm Cortex-M0+ (CPU2). Lo stack è memorizzato nella memoria Flash, che è condivisa con l'applicazione che gira sulla CPU1.

Il front-end a radiofrequenza è basato su una modulazione diretta della portante al trasmettitore, e usa una architettura a bassa frequenza intermedia al ricevitore. Grazie al trasformatore interno sui pins RF, il circuito si potrebbe interfacciare direttamente all'antenna, con una impedenza vicina a 50Ω . La natura passabasso del trasformatore interno, semplifica la circuiteria che deve essere implementata all'esterno per il filtraggio delle armoniche e delle interferenze fuori banda. In modalità trasmettitore, la potenza massima in uscita è selezionabile dall'utente tramite l'LDO programmabile dell'amplificatore. In ricezione il circuito può essere usato sia in modalità a performances elevate che in modalità a ridotto consumo energetico. sempre selezionabile dall'utente. Grazie all'utilizzo di un filtraggio complesso e una archittetura I/Q altamente accurata, il dispositivo è caratterizzato da un'alta sensibilità e linearità. La BOM necessaria per l'implementazione delle funzionalità RF è ridotta al minimo, il riferimento in frequenza è sintetizzato da un cristallo esterno a 32MHz e non necessità di capacità esterne, poichè ce ne sono due interne e programmabili. Il blocco può fungere sia da processore master che slave, ed è compatibile con le specifiche dello standard Bluetooth 5.0 (2Mbps). Utilizzando l'SMPS (Switched Mode Power Supply) il consumo di corrente alla potenza in uscita massima (+6dBm) si mantiene moderato con una $I_{tmax} = 8.1 mA$. Sono disponibili modalità sleep ultra-low-power, con transizioni tra le varie modalità molto brevi, che si traducono in bassi consumi medi, risultando in una durata maggiore della batteria.

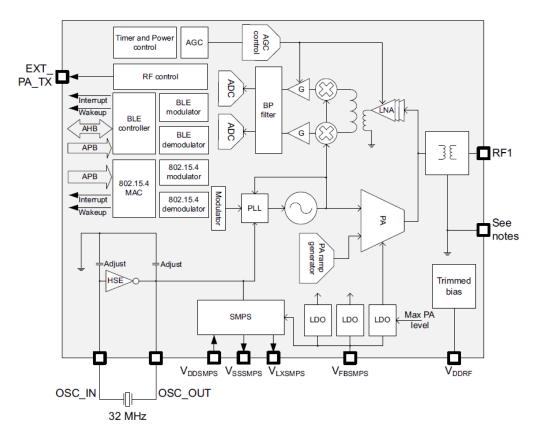


Figura 2.11: Schema a blocchi del front-end a RF[4].

Di seguito è riportato uno schema base con gli elementi necessari per lo sviluppo della parte RF: come già evidenziato, grazie alle caratteristiche del dispositivo la lista di componenti esterni è veramente ridotta.

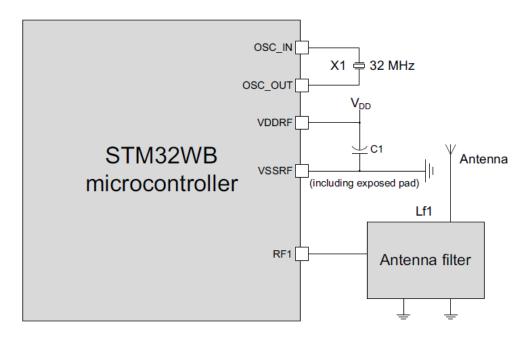


Figura 2.12: Schematico semplificato della parte a RF[4].

2.5 ST25R95

Il circuito integrato ST25R95 è un ricetrasmettitore integrato, grazie al quale si possono sviluppare applicazioni contactless. Con un package a 32 pins e un interfaccia SPI fino a 2MBps per la comunicazione con un controllore esterno, grazie ad esso è possibile implementare le principali tecniche RFID e NFC a 13.56MHz. Il chip supporta la comunicazione ISO/IEC 14443 Type A in modalità reader e card emulation, e la comunicazione ISO/IEC 14443 Type B, ISO/IEC15693, e FeliCa in modalità reader. Inoltre supporta anche il rilievo, lettura e scrittura di tags NFC Forum Type 1,2,3,4,5 ed integra al suo interno un front end analogico a 13.56MHz.

Di seguito sono riassunte le principali funzionalità del dispositivo facendo riferimento al datasheet[3]:

- Parte della famiglia ST25, include funzionalità NFC/RFID tag e reader
- Modalità operative supportate: Reader/Writer e Card emulation (ISO/IEC 14443-3 Type A)
- Frame controller interno dedicato
- Front end a radiofrequenza per la comunicazione integrato

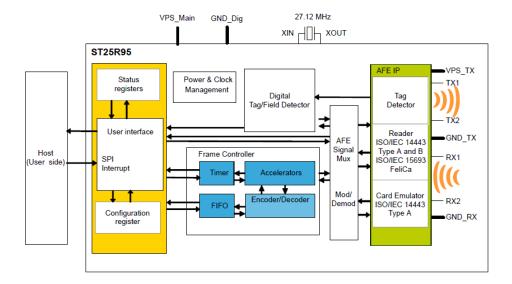


Figura 2.13: Schema ST25R95.

- Gestione della potenza ottimizzata
- Modalità tag detection e field detection
- NFC-A / ISO14443A, NFC-B / ISO14443B, NFC-F / FeliCa e NFC-V / ISO15693 reader mode
- $\bullet\,$ NFC-A / ISO14443A card emulation mode
- MIFARE Classic compatible

Tecnologia RFID Radio-frequency identification (RFID), è una tecnologia per l'identificazione e/o memorizzazione automatica di informazioni, basata sulla capacità di particolari circuiti elettronici, chiamati tag, e sulla capacità di questi a rispondere all'interrogazione a distanza da parte di appositi apparati fissi o portatili chiamati reader. Questa identificazione avviene mediante radiofrequenza, grazie alla quale un reader è in grado di comunicare e/o aggiornare le informazioni contenute nei tag che sta interrogando; infatti nonostante il suo nome, un reader non è in grado solamente di leggere ma anche di scrivere informazioni. Quindi si possono realizzare dei sistemi di lettura e/o scrittura senza fili con svariate applicazioni. Ad estensione degli standard si sta affermando negli ultimi anni anche la tecnologia NFC (fino a 10cm ma con velocità di trasmissione dati 424kbit/s). Dunque un sistema RFID è costituito da tre elementi fondamentali:

- Uno o più tag;
- Un reader,
- Un sistema informativo per la gestione dei dati da e verso i readers.

L'elemento principale è il tag, che è costituito da un microchip che contiene i dati in memoria, un'antenna e un supporto fisico che tiene insieme i due componenti precedenti, che può essere realizzato in diversi materiali. Se il tag contiene solo un microchip privo di alimentazione elettrica e un'antenna, allora si ha un tag passivo. Al passaggio di un lettore che emette un segnale a RF, questo attiva il microchip e gli fornisce l'energia necessaria per rispondere, ritrasmettendo un segnale contenente le informazioni memorizzate, oppure aggiorna i dati presenti sulla memoria. Se il tag dispone di una o più antenne e una batteria di alimentazione allora è detto tag attivo. In questo caso si hanno delle distanze operative maggiori, che possono arrivare fino a 200m. Infine si ha la versione semi-passiva, in cui l'alimentazione a batteria alimenta solo il microchip o apparati ausiliari (sensori), ma non il trasmettitore che si comporta come nel caso passivo; e la versione semi-attiva con batteria che alimenta sia chip che trasmettitore, ma che viene attivata solo quando interrogato. Per questioni progettuali, quali i form factor dei vari dispositivi che implementano la tecnologia RFID in commercio, e le antenne che poi si monteranno sul PCB, si è scelto di integrare sulla scheda il chip ST25R95 che opera alla frequenza di 13.56MHz, tuttavia esistono numerose bande di frequenza operative ben definite e normate da standard ISO, come ad esempio la 433MHz, la 860-960MHz o la 2.4GHz. L'RFID è una valida alternativa sia alle tecnologie di personal identification tradizionali, sia alle tecnologie di strong authentication basate sul riconoscimento degli attributi biometrici di un individuo. A differenza di tali tecnologie non richiede contatto visivo per l'identificazione e permette il riconoscimento anche "a distanza". L'identificazione tramite RFID oltre a rendere più agile l'impiego di varchi automatizzati, permette di distinguere gli ingressi dalle uscite e verificare automaticamente l'elenco delle presenze all'interno di una determinata zona. Il dispositivo montato sulla board è stato pensato proprio per questo tipo di funzionalità, attivando la lettura/scrittura del tag alla pressione di un pulsante sulla scheda.

Capitolo 3

Schema elettrico

3.1 Realizzazione della libreria

Ancora prima di passare alla realizzazione dello schema elettrico, è necessario operare un passaggio preliminare: la realizzazione della libreria contenente tutti i componenti che saranno poi utilizzati sulla scheda. Ciascuno di essi è composto da tre parti: il footprint, il simbolo e il dispositivo vero e proprio. Il footprint consiste nel disegno dei pads di rame, che andranno posizionati sul PCB e sui quali poi sarà saldato il singolo dispositivo.

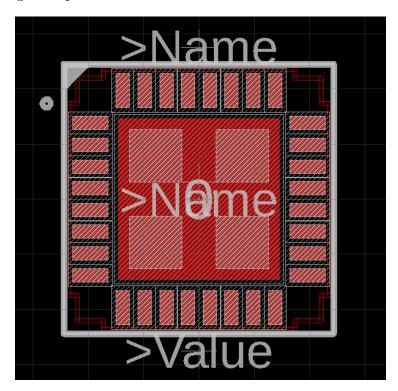


Figura 3.1: Footprint del dispositivo LR1110 utilizzato nel progetto.

In Figura 3.1 si può vedere un esempio di footprint per il dispositivo LR1110: nel datasheet di ciascun componente è presente il landpad consigliato, ovvero la posizione e la distanza tra i vari pads, nonchè la loro dimensione consigliata. Questa è una operazione fondamentale in quanto se si realizza un footprint errato, non

Capitolo 3 Schema elettrico

sarà possibile in fase di montaggio saldare il componente sulla scheda. Come si può notare sempre dalla Figura 3.1 oltre ai pads in colore rosso, sono presenti anche altri elementi in altri colori. Il CAD Eagle utilizzato per il progetto, permette di ragionare per layer [9], che corrispondono ai layer fisici che realizzano il PCB. I pads sono in colore rosso e vanno diseganti sul layer Top, che nella scheda fisica corrisponderà allo strato di rame superficiale sulla faccia della scheda; analogamente ci sarà un layer Bottom, che corrisponderà al rame sulla faccia inferiore della scheda. In rosso puntinato si può invece notare il layer tKeepOut, che non corrisponde ad un layer fisico, ma è utile in fase di progetto perchè permette al software di segnalare all'utente eventuali sovrapposizioni tra diversi componenti. Le aree ricoperte da linee oblique che vanno a sovrapporsi ai pads, corrispondono a due layers diversi, e sono aggiungti in automatico dal CAD e modificabili in modo custom in fase di progetto: il tCream e il tStop. Il primo, per i pad piccoli si sovrappone perfettamente ad essi, mentre per quelli più grandi li ricopre solo parzialmente e corrisponde alle zone dove andrà applicata la pasta saldante su cui verrà posizionato il dispositivo per essere saldato. In particolare grazie alla presenza di questo layer, in fase di produzione si realizzerà un una piastra metallica con dei buchi in corrispondenza dei pads, che faciliterà la stesura della pasta saldante solo nei punti desiderati (Figura 3.2).

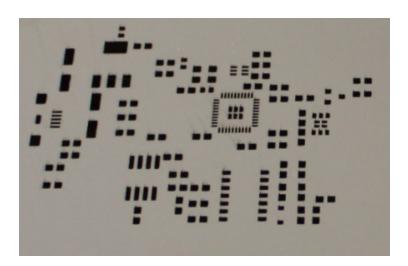


Figura 3.2: Esempio di stencil.

Sopra al PCB andrà poi applicata la solder mask, uno strato colorato (solitamente verde), che oltre a dare colore al PCB, serve in fase di montaggio per evitare che si creino dei corti tra i conduttori, e in fase di utilizzo per proteggere la scheda dall'ambiente e dagli agenti atmosferici, che altrimenti aggredirebbero la vetronite del substrato e le piste. Chiaramente essa non andrà applicata sui pads dove saranno saldati i componenti, e proprio a questo serve il layer tStop, che tiene conto delle parti che non dovranno essere ricoperte da questa protezione. Si hanno poi il layer tName e tValue, i quali si può scegliere se mostrare o meno sulla scheda, e che serviranno per dare un nome ed un valore (ad esempio nel caso di resistori, capacità ecc.), e

che serviranno per la generazione della BOM (Bill Of Materials). In linea bianca più sottile è presente il tDocu, che serve per riportare sul footprint delle informazioni come le dimensioni meccaniche del dispositivo e del suo package, che non saranno riportate sulla scheda ma sono utili in fase di progetto e reviewing del PCB. Infine per ultimo, ma non meno importante, si ha il tPlace che è il layer che verrà poi stampato sul PCB, nel quale si possono inserire l'outline dei singoli componenti, dove questi andranno posizionati, e informazioni riguardandi ad esempio la numerazione dei pin (di solito si evidenzia il pin numero uno degli integrati, oppure il catodo dei diodi ecc.) per evitare ambiguità in fase di montaggio.

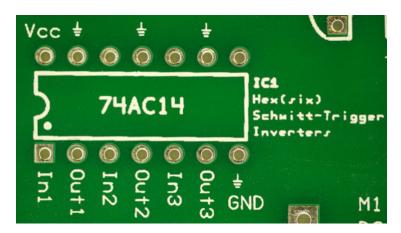


Figura 3.3: Esempio di serigrafia (linee bianche) e solder mask (verde) che copre e protegge il PCB.

Questi sono i layer principali e fondamentali per realizzare il footprint di un dispositivo, poi ce ne sono molti altri che portano informazioni aggiuntive come ad esempio il tGlue che indica delle zone dove va depositato uno strato di colla per fissare i componenti che possono essere più soggetti a stress durante l'uso (come switches, connettori ecc.), oppure dei layer che non sono utilizzati per il footprint ma che poi saranno usati nel progetto della scheda come ad esempio il tRestrict, che indica delle zone da dove il rame deve essere rimosso (ad esempio per ragioni di carattere termico), oppure il layer Drills contenente tutte le informazioni sui fori conduttivi presenti sulla scheda, o ancora il layer Unrouted il quale mostra in fase di routing le connessioni ancora non effettuate tra i vari pads.

Una volta realizzati tutti i footprint dei dispositivi si passa alla creazione del simbolo del componente, che sarà visualizzato in fase di progetto dello schema elettrico della scheda.

Anche qui come per il footprint, Eagle permette di ragionare per layers, si ha il layer *Pins* dove vengono inseriti i pin che presenta il dispositivo, i layer *Names* e *Values* per l'inserimento di nomi e valori, che sarano poi inseriti in automatico per tutti i dispositivi uguali, e così via. Importante è inserire nel simbolo tutti i pin necessari per le connessioni del dispositivo e nominarli coerentemente per facilitare il

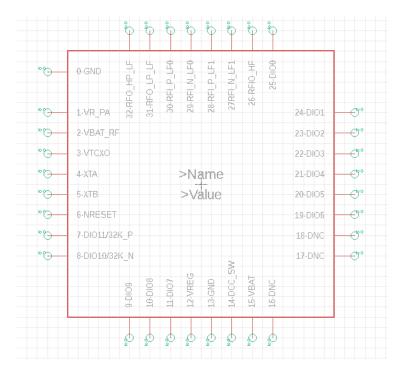


Figura 3.4: Simbolo del dispositivo LR1110.

progetto. Una volta creati footprint e simbolo, l'ultimo step necessario è quello di creare il vero e proprio dispositivo: si crea una corrispondenza tra uno o più footprint ed un simbolo, collegando ciascun pad fisico del footprint ad uno dei pin del simbolo che si posizionerà nello schema elettrico. Questa è un'operazione fondamentale per il corretto collegamento dei vari dispositivi fisici sulla scheda. Uno stesso dispositivo poi, può avere differenti package tra i quali è possibile scegliere in fase di progetto (ad esempio resistore 0402 o 0805 ecc.) ai quali corrispondono lo stesso simbolo, ma differenti footprint. Inoltre è possibile inserire per ciascun dispositivo degli attributi che lo caratterizzano, di norma si deve almeno inserire una breve descrizione, un ID univoco del prodotto fornito dal produttore, e un ID del prodotto del distributore. Questi attributi sono mandatori per poter produrre alla fine del progetto una BOM che sarà consegnata a chi dovrà assemblare il PCB.

3.2 Shema elettrico

3.2.1 Schema elettrico LR1110

Per la composizione dello schema elettrico relativo al dispositivo LR1110, si sono seguite le linee guida presenti nel datasheet del dispositivo [1], nonchè si è preso spunto dallo schematico della evaluation board venduta da Semtech [10], con i quali sono stati effettuati i test preliminari sulla parte GNSS, Wi-Fi e LoRa.

Questa parte è stata progettata in una configurazione multi-band. Si usano sia l'amplificatore high power interno (+22dBm) sul pin RFO_HP_LF che quello low

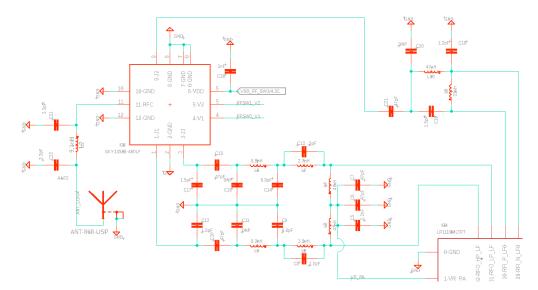


Figura 3.5: Parte radio sub-GHz.

power (+15dBm) sul pin sul pin RFP_LP_LF, in questo modo possono essere implementate due configurazioni con diverse potenze in uscita e frequenze diverse sullo stesso PCB. Entrambi gli amplificatori, sono accoppiati al ricevitore grazie allo switch a radiofrequenza SKY13588SP3TRF che permette performance ottimizzate sia in trasmissione che in ricezione. Entrambi gli amplificatori hanno la stessa struttura di matching in comune:

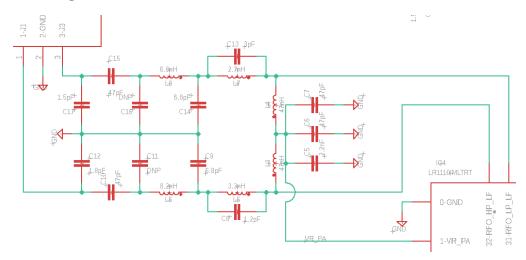


Figura 3.6: Struttura rete di matching degli amplificatori.

Ciascun amplificatore è alimentato dal pin VR_PA tramite un indudttore di 47nH (L3). Il pin VR_PA deve essere disaccopiato, e questo è fatto grazie alle capacità C5, C6 e C7, usando condensatori NP0. Particolare attenzione deve essere posta nel progetto di questa parte, per minimizzare le radiazioni indesiderate, rendendo le piste che collegano a VR_PA più corte possibile. La rete di matching ha lo scopo di ottenere l'impedenza di carico ottima all'uscita del pin dell'amplificatore,

al fine di generare la massima potenza in uscita alla frequenza di lavoro, con la migliore efficienza (e quindi il minimo consumo di potenza possibile) minimizzando le armoniche dell'amplificatore. La rete di adattamento è composta da due sezioni:

- lo stadio di adattamento dell'impedenza, per avere un'impedenza di carico ottima al pin dell'amplificatore;
- uno stadio di filtraggio passa basso, per ridurre il livello delle armoniche dell'amplificatore.

Inoltre può essere aggiunto un ulteriore stadio di filtraggio dopo lo switch (C31, L17, C32); la capacità serie C10 invece serve per rimuovere la componente DC dell'amplificatore e proteggere lo switch. Ciascuna delle due reti di adattamento è implementata seguendo i valori presenti nella BOM dell'evaluation kit fornito da Semtech per le frequenze Europee 850-928MHz (è presente anche un'altra BOM per la banda a 490MHz). L'amplificatore opera sempre nel modo più efficiente, grazie alla possibilità di avere un trasmettitore adattativo: i parametri dell'amplificatore (TXPWR, PaDutyCycle, PaHPSel) si adattano alla frequenza di lavoro, mantenendo la potenza in uscita, le componenti armoniche e l'efficienza ottimizzate ad ogni frequenza. Lo stadio di ricevitore RF è implementato in una struttura differenziale a quattro elementi sui pin RFI N LFO e RFI P LFO con l'uso di un balun per l'adattamento e per il passaggio dalle linee differenziali a singola linea. Il path del ricevitore è poi collegato all'RF switch insieme al path del trasmettitore. In questo modo si può ottenere la più bassa figura di rumore al ricevitore, e la migliore sensibilità. La capacità serie C21 è inserita per prevenire danni all'LR1110 da componenti in continua.

L'LR1110 permette di realizzare un input path per le scansioni GNSS, per permettere la localizzazione outdoor. Similmente al path RX precedentemente illustrato, lo stadio RX GNSS è implementato con una struttura differenziale a 4 elementi, sui pin RFI_N_LF1 e RFI_P_LF1 con l'uso di un balun per il passaggio a linea singola ed adattamento (L11/L12/C22/C23/C24). In questo modo si ottengono un rumore più basso al ricevitore e ottime performance nella ricezione di segnali GPS. Un filtro passa basso (L13/C26) del primo ordine provvede ad un ulteriore filtraggio RF.

Per permettere di operare con un'antenna passiva, è inserito un LNA (Low Noise Amplifier) esterno. La board può essere realizzata anche per operare con un'antenna attiva, che integra già al suo interno un amplificatore, in questo caso questo componente deve essere bypassato. L'LNA utilizzato è un chip Infineon (BGA524N6), alimentato da VDD_RF_SWITCH, un'alimentazione condivisa tra gli switch e diversa da quella principale dell'LR1110. Esso è controllato dal pin LNA_CTRL_ON, connesso al micro principale. L'LR1110 fornisce una tensione di 3.3V sulla linea RF del path GNSS. Quindi se si vuole alimentare una antenna attiva o un altro LNA direttamente sul connettore SMA, non è richiesta un'altra alimentazione. L'alimentazione dell'antenna attiva è attivata sempre dal pin LNA_CTRL_ON.

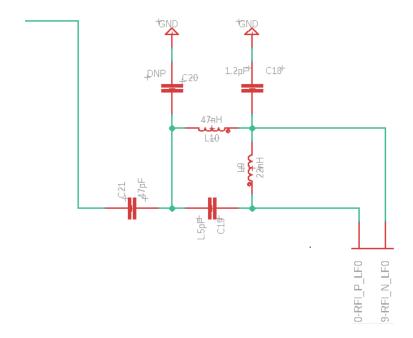


Figura 3.7: Struttura rete di matching RX.

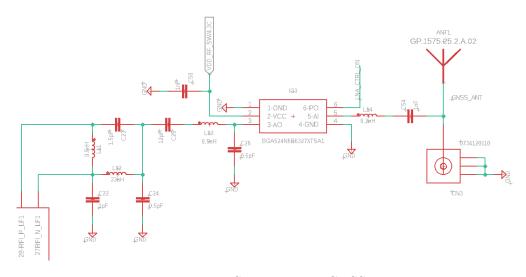


Figura 3.8: Struttura rete GNSS.

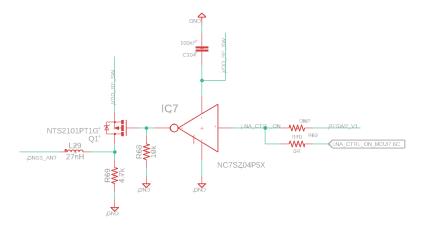


Figura 3.9: Alimentazione antenna attiva.

Sull'LR1110 è possibile implementare anche un path per scansioni Wi-Fi che permettono di eseguire una localizzazione indoor e outdoor. Lo stadio Wi-Fi è implementato su una singola linea sul pin RFIO_HF. I componenti SMD in questo stadio servono per operare un filtraggio del segnale a radiofrequenza, e la C29 per bloccare le componenti in DC che potrebbero danneggiare lo stadio Wi-Fi sull'LR1110.

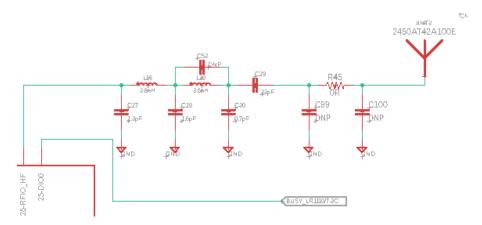


Figura 3.10: Stadio Wi-Fi.

Sulla board sono implementati un oscillatore a 32MHz che fornisce il clock principale al sistema e a 32.768 kHz, usato come RTC (Real Time Clock) che può sostituire l'oscillatore RC usato di default. Per quello a 32MHz il dispositivo supporta sia un cristallo che un TCXO (Temperature Compensated Crystal Oscillator). In ambienti con molte variazioni di temperatura, per ottenere una migliore precisione di frequenza è consigliato utilizzare un TCXO, e questo è obbligatorio per sfruttare le funzionalità GNSS del chip per minimizzare il consumo di potenza richiesto per una geolocalizzazione outdoor. Dunque nella board è stato implementato un TCXO connesso al pin XTA tramite una 100R, e viene alimentato dal regolatore interno all'LR1110 sul pin VTCXO, sul quale la tensione è programmabile, mentre il pin XTB deve essere lasciato aperto. Nel caso in cui non si volesse sfruttare la modalità

GNSS è sufficiente un cristallo connesso tra i pin XTA e XTB, che viene comunque implementato nella board ma non connesso. Inoltre è possibile ottenere un segnale di clock sul pin DIO8, il che rende possibile implementare anche un solo cristallo, riducendo i costi della BOM.

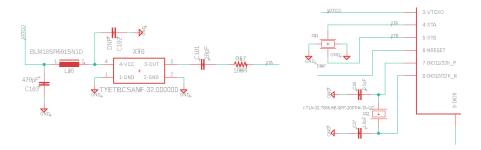


Figura 3.11: Stadio Wi-Fi.

L'LR1110 supporta due tipi di alimentazione: sia con una configurazione DC-DC, che con LDO (Linear Dropout Regulator). L'induttore da 15uH indica come si è sfruttata la modalità DC-DC. Di questo bisognerà tenere conto nel routing, dove bisognerà prestare attenzione nel rendere il loop DCC_SW-VREG più piccolo possibile, e con resistenza serie bassa in modo da garantire un'alta efficienza per il DC-DC e un basso consumo di potenza durante le sue operazioni.

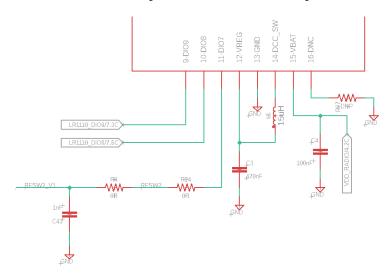


Figura 3.12: Alimentazione in modalità DC-DC.

Infine si ha l'interfaccia seriale dell'LR1110 che comunica tramite SPI (Serial Peripheral Interface) con il micro principale, per ricevere le istruzioni da eseguire. Ci sono quattro linee di comunicazione da collegare: SPI1_MISO e SPI1_MOSI, le linee dati, SPI1_CLK sulla quale il dispositivo preleva il clock per la comunicazione e la linea LR1110_SPI_CS con la quale il master della comunicazione, in questo caso il microprocessore, seleziona come slave l'LR1110. Inoltre va effettuata un ulteriore

connessione, la linea BUSY, sulla quale si può verificare lo stato del dispositivo: se alta significa che il dispositivo è impegnato in qualche operazione e non può ricevere nuovi comandi, mentre se bassa è pronto a ricevere nuove istruzioni da eseguire. I pin DIO8-9 sono collegati al MCU e possono essere utilizzati come sorgenti di interrupts per le operazioni dell'LR1110.

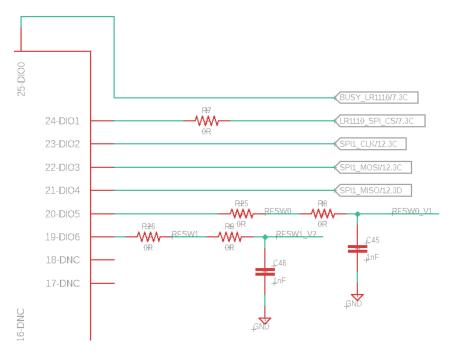


Figura 3.13: Interfaccia seriale.

3.2.2 Schema elettrico MCU principale

Avendo questo dispositivo 68 pins, piuttosto che realizzare un unico simbolo si è deciso di suddividerlo in 5 simboli ciascuno relativo ad una periferica dell'integrato: due simboli sono stati dedicati alla parte dello schema elettrico relativa all'alimentazione del micro, uno per la parte a radiofrequenza, uno per la parte dei riferimenti in frequenza ed infine un ultimo simbolo per la parte dei GPIO offerti dal dispositivo.

Per quanto riguarda l'alimentazione facendo riferimento al datasheet [11], è integrato all'interno del circuito un SMPS step-down converter per migliorare le performance low power quando la tensione di alimentazione V_{DD} è sufficientemente alta. Questo converter è in grado di essere automaticamente bypassato quando la V_{DD} scende al di sotto di una certa soglia scelta dall'utente. In particolare per un corretto funzionamento dell'SMPS è necessario introdurre la capacità C55 (capacità d'uscita dell'SMPS) del valore di 4.4uF e l'induttanza L16 da 10uH, con un'induttanza aggiuntiva L17 da 10nH per migliorare ulteriormente le prestazioni del dispositivo.

Il dispositivo prevede diverse alimentazioni e può operare nei seguenti range:

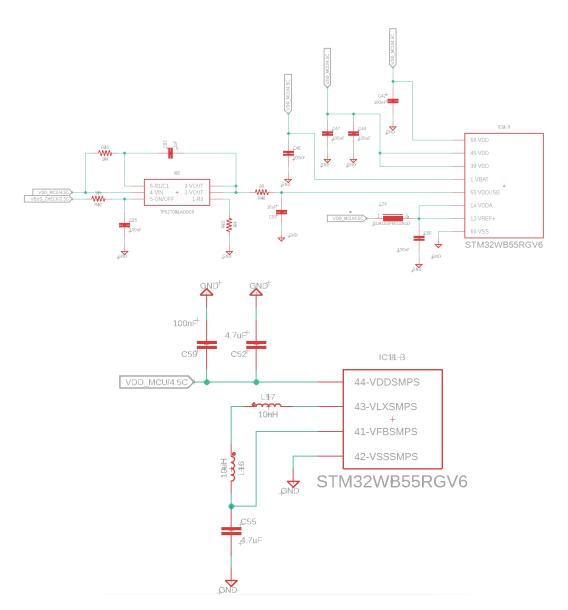


Figura 3.14: Schema alimentazione MCU.

- V_{DD} da 1.71 a 3.6V, da cui si alimentano il regolatore interno e funzioni di sistema come RF, SMPS, reset, e clock interni. Questa è l'alimentazione principale fornita alla batteria, e a questa devono essere collegati anche la VDDRF e VDDSMPS. L'alimentazione è collegata a ciascun pin tramite dei condensatori di bypass (C44, C47, C48, C42) da 100nF.
- V_{DDA} da 1,62 a 3.6V, è l'alimentazione degli ADC, in questo caso si è collegata anche questa all'alimentazione principale VDD.
- V_{DDUSB} da 3.0 a 3.6V, permette di alimentare il dispositivo tramite una connessione USB. Nello schema questo pin è collegato ad un power switch, attivato dalla linea VBUS_CHECK: se il dispositivo è connesso all'alimentazione tramite USB, la linea VBUS_CHECK va alta ed il dispositivo è alimentato tramite USB. In caso contrario la VBUS_CHECK è bassa, lo switch viene bypassato e il pin V_{DDUSB} è collegato alla alimentazione principale e non viene utilizzato.

Per quanto riguarda gli oscillatori esterni, si possono connettere due cristalli esterni al microcontrollore. L'HSE (High Speed External) con una frequenza di 32MHz è connesso ai pin 31 e 32 (OSC_{IN} , OSC_{OUT}). Non necessita di capacità aggiuntive in quanto queste sono già presenti internamente al micro e configurabili tramite i registri interni. Sarà importante poi in fase di routing posizionare il cristallo quanto più vicino possibile ai due pins. L'LSE (Low Speed External) a 32768kHz è utilizzato come RTC del sitema ed è connesso ai pin PC14 e PC15, in questo caso sono necessarie due capacità (C63 e C64) per adattarsi alla capacità di carico C0 del cristallo scelto. Con il modello di cristallo selezionato per la board è raccomandata una C0=9pF, e il valore ottimo per le due capacità C63 e C64 è di 15pF. Con una C0 bassa si ottiene una velocità in accensione maggiore e un minore consumo di potenza, mentre con una C0 più elevata si ha una migliore stabilità in frequenza. Chiaramente la capacità di carico C0 sarà anche influenzata da elementi parassiti come la capacità dei pads e le capacità parassite delle piste. Per cercare di minimizzare questi effetti è bene posizionare questi componenti più vicini possibile ai pads e connettere C1 e C2 tramite dei vias.

Per ottenere le performance migliori a radiofrequenza (in particolare massima potenza trasmessa, ottima sensibilità del ricevitore e reiezione alle interferenze) è bene implementare una rete di adattamento tra il pin RF1 e il filtro passa basso. Questa è una rete a Pigreco, subito seguita dal filtro. I componenti C65 e L18 hanno il compito di adattare l'impedenza del pin a 50Ω , l'impedenza che deve essere vista dall'antenna. La capacità C66 e il filtro hanno il compito di eliminare le armoniche indesiderate. Nel datasheet [11] si raccomanda inoltre di posizionare questa rete di adattamento il più vicino possibile al pin, e di evitare delle linee lunghe tra i componenti che compongono la rete.

Infine occorre collegare opportunamente le varie periferiche ai GPIO del micro. Particolare attenzione deve essere prestata nel connettere le linee che poi dovranno

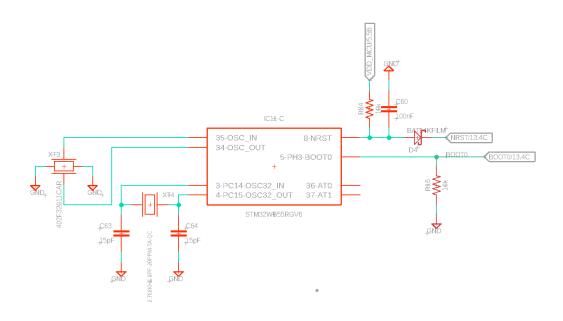


Figura 3.15: Schema cristalli MCU.

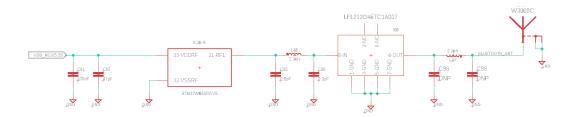


Figura 3.16: Schema RF MCU.

Capitolo 3 Schema elettrico

generare degli interrupts: il micro offre 16 linee di interrupts e bisogna fare in modo che due interrupts provenienti da periferiche differenti, non si sovrappongano sulla stessa linea. Per fare questo ci si è anche serviti del tool STM32CubeMx, che permette di visualizzare graficamente i vari pins del microcontrollore e segnala in automatico eventuali conflitti tra le periferiche. Di seguito è riportata la mappatura dei pin-linee. Queste non sono state decise a priori una volta per tutte, ma sono state modificate durante il progetto: in particolare nella fase di routing, dopo aver completato il placement di tutti i componenti, se necessario è possibile cambiare un collegamento pin-linea, se questo risulta sconveniente, se la pista taglia altri segnali ad esempio, per evitare di utilizzare dei vias dove possibile.

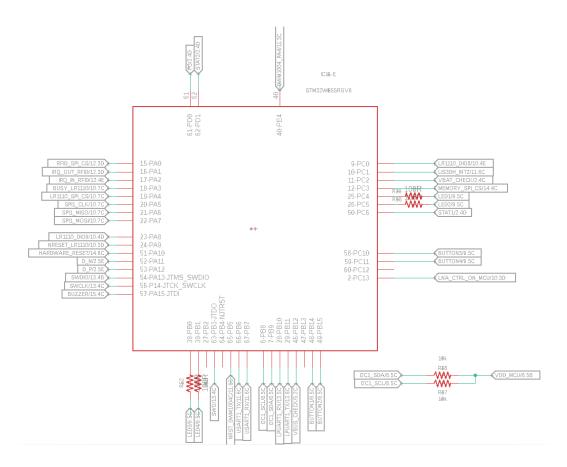


Figura 3.17: Schema pins GPIO.

La periferica SPI1 è condivisa da diversi dispositivi, ciascuno con la sua relativa linea slave select: LR1110, ST25R95, e memoria flash. L'I2C è utilizzato per la comunicazione con il barometro (sono state inserite per il funzionamento dell'I2C delle resistenze di pull-up da 10k), mentre la UART per mandare i comandi al micro interno del modulo DWM1004C. Per il debug della scheda è stata scelta la modalità SWD piuttosto che la JTAG, potendo così utilizzare 3 pins invece che 5. Le linee utilizzate per gli interrupt sono la PA1,PA8,PB12,PB14,PB15,PC10,PC11.

Tabella 3.1: Collegamento Pins GPIO

Pin Segnale Funzione PA0 RFID_SPI_CS Chip Select SPI RFID PA1 IRQ_OUT_RFID Interrupt as T25R95 PA2 IRQ_IN_RFID Interrupt verso S725R95 PA3 BUSY_LR1110 Linea Busy per controllo stato LR1110 PA4 LR1110_SPI_CS Chip Select SPI LR1110 PA5 SPII_MISO SPI linea MISO PA6 SPII_MOSI SPI linea MISO PA7 SPII_MOSI SPI linea MISO PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset flash memory PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWOIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 <th></th> <th>1000110 0111</th> <th>Conegamento i ma di 10</th>		1000110 0111	Conegamento i ma di 10
PA1 IRQ_OUT_RFID Interrupt das ST25R95 PA2 IRQ_IN_RFID Interrupt verso ST25R95 PA3 BUSY_LRI110 Linea Busy per controllo stato LR1110 PA4 LRI110_SPI_CS Chip Select SPI LR1110 PA5 SPII_CLK Linea clock SPI PA6 SPII_MOSI SPI linea MISO PA7 SPII_MOSI SPI linea MOSI PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset fash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWOIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C	Pin	Segnale	Funzione
PA2 IRQ_IN_RFID Interrupt verso ST25R95 PA3 BUSY_LR1110 Linea Busy per controllo stato LR1110 PA4 LR1110_SPI_CS Chip Select SPI LR1110 PA5 SPII_CLK Linea clock SPI PA6 SPII_MISO SPI linea MISO PA7 SPI_MOSI SPI linea MOSI PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USARTI_RX <	PA0	RFID_SPI_CS	Chip Select SPI RFID
PA3	PA1	IRQ_OUT_RFID	Interrupt da ST25R95
PA4 LR1110_SPI_CS Chip Select SPI LR1110 PA5 SPII_CLK Linea clock SPI PA6 SPII_MISO SPI linea MISO PA7 SPII_MOSI SPI linea MISO PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USARTI_TX Linea UART TX PB7 USARTI_RX Linea UART RX PB8 12C1_SDA Linea UART RX <t< td=""><td>PA2</td><td>IRQ_IN_RFID</td><td>Interrupt verso ST25R95</td></t<>	PA2	IRQ_IN_RFID	Interrupt verso ST25R95
PA5 SPII_CLK Linea clock SPI PA6 SPI_MISO SPI linea MISO PA7 SPII_MOSI SPI linea MOSI PA8 LRI110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USARTI_TX Linea UART TX PB7 USARTI_TX Linea UART TX PB8 12C1_SDA Linea JART RX PB8 12C1_SDA Linea JART RX	PA3	BUSY_LR1110	Linea Busy per controllo stato LR1110
PA6 SP11_MISO SP1 linea MISO PA7 SP11_MOSI SP1 linea MOSI PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset flash memory PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SDA Linea dati 12C PB9 12C1_SDA Love power UART RX PB10 LPUART1_TX Love power UART RX	PA4	LR1110_SPI_CS	Chip Select SPI LR1110
PA7 SP11_MOSI SP1 linea MOSI PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SDA Linea dati 12C PB9 12C1_SDA Linea dati 12C PB10 LPUART1_TX Low power UART RX PB11 LPUART1_TX Low power UART TX	PA5	SPI1_CLK	Linea clock SPI
PA8 LR1110_DIO9 Pin per interrupt LR1110 PA9 NRESET_LR1110 Reset LR1110 PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_TX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB	PA6	SPI1_MISO	SPI linea MISO
PA9 NRESET_LR1110 Reset Interval (Linear dati USB) PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USARTI_TX Linea UART TX PB7 USARTI_RX Linea UART RX PB8 12C1_SCL Clock I2C PB9 12C1_SDA Linea dati 12C PB10 LPUARTI_TX Low power UART RX PB11 LPUARTI_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 BUTTON1 Bottone tasti	PA7	SPI1_MOSI	SPI linea MOSI
PA10 HARDWARE_RESET Reset flash memory PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SCL Clock 12C PB9 PC1_SDA Linea dati 12C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PC0	PA8	LR1110_DIO9	Pin per interrupt LR1110
PA11 D_N Linea dati USB PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PC0	PA9	NRESET_LR1110	Reset LR1110
PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_TX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110	PA10	HARDWARE RESET	Reset flash memory
PA12 D_P Linea dati USB PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_TX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110	PA11	D N	Linea dati USB
PA13 SWDIO SWDIO per debug PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SCL Clock 12C PB9 I2C1_SDA Linea dati 12C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PP15 BUTTON2 Bottone tastiera PC0 LR1110_DIOS Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno			
PA14 SWCLK SWCLK per debug PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SCL Clock 12C PB9 12C1_SDA Linea dati 12C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIOS Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK			
PA15 BUZZER Attiva il buzzer PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SCL Clock 12C PB9 12C1_SDA Linea dati 12C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS </td <td>-</td> <td>1</td> <td></td>	-	1	
PB0 LED3 Led tastiera PB1 LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SP1_CS SPI Chip Select memoria PC4 LE		l i	
PBI LED4 Led tastiera PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LE		l	
PB2 - - PB3 SWO SWO per debug PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART TX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 ST			
PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera			Let tasticia
PB4 - - PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera	-	SWO	SWO per debug
PB5 NRST_DWM1004C Reset DWM1004C PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 12C1_SCL Clock I2C PB9 12C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera <		1 5₩0	SWO per debug
PB6 USART1_TX Linea UART TX PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - <t< td=""><td></td><td>NDCT DWM1004C</td><td>- Dt DWM1004C</td></t<>		NDCT DWM1004C	- Dt DWM1004C
PB7 USART1_RX Linea UART RX PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica			
PB8 I2C1_SCL Clock I2C PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo comessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica			
PB9 I2C1_SDA Linea dati I2C PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica <td></td> <td>_</td> <td></td>		_	
PB10 LPUART1_RX Low power UART RX PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica		_	
PB11 LPUART1_TX Low power UART TX PB12 VBUS_CHECK Linea per controllo comnessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PB12 VBUS_CHECK Linea per controllo connessione USB PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PB13 - - PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PB14 BUTTON1 Bottone tastiera PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica		VBUS_CHECK	Linea per controllo connessione USB
PB15 BUTTON2 Bottone tastiera PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica	-	-	-
PC0 LR1110_DIO8 Pin per interrupt da LR1110 PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PC1 LIS3DH_INT2 Interrupt da accelerometro interno modulo DWM1004C PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PC2 VBAT_CHECK Linea check batteria PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			* *
PC3 MEMORY_SPI_CS SPI Chip Select memoria PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica		I	*
PC4 LED1 Led tastiera PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			
PC5 LED2 Led tastiera PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica			*
PC6 STAT1 Controllo carica PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica		LED1	Led tastiera
PC10 BUTTON3 Bottone tastiera PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica	PC5	LED2	Led tastiera
PC11 BUTTON4 Bottone tastiera PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica	PC6	STAT1	Controllo carica
PC12 - - PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica	PC10	BUTTON3	Bottone tastiera
PC13 LNA_CTRL_ON_MCU Controllo antenna GPS PD0 PG Controllo carica PD1 STAT2 Controllo carica	PC11	BUTTON4	Bottone tastiera
PD0 PG Controllo carica PD1 STAT2 Controllo carica	PC12	-	-
PD1 STAT2 Controllo carica	PC13	LNA_CTRL_ON_MCU	Controllo antenna GPS
	PD0	PG	Controllo carica
PE4 DWM1004_PA4 GPIO micro interno al DWM1004C	PD1	STAT2	Controllo carica
	PE4	DWM1004_PA4	GPIO micro interno al DWM1004C

3.2.3 Schema elettrico DWM1004C

Come descritto in precedenza, il modulo DWM1004C, include già al suo interno tutti i componenti necessari per il funzionamento del DWM1000 e l'implementazione delle sue funzionalità UWB, in più è fornito dell'antenna necessaria alle operazioni. In questo modo è possibile realizzare le applicazioni desiderate con una BOM veramente minimale. Per il controllo del ciruito integrato è inserito nel modulo un microcontrollore della famiglia STM32L0, che tramite UART è collegato al micro principale e riceverà da esso i comandi da eseguire. Sono stati inseriti anche i collegamenti tramite delle 0R per poter eventualmente implementare in un secondo momento la comunicazione I2C con l'accelerometro interno al modulo, ed è stato collegato il GPIO0 del micro interno a quello principale, sempre tramite una 0R. Anche qui per la programmazione e debug sarà sfruttata la modalità SWD che è anche l'unica messa a disposizione dal modulo. Infine sono stati collegati i pins all'alimentazione con dei condensatori di bypass.

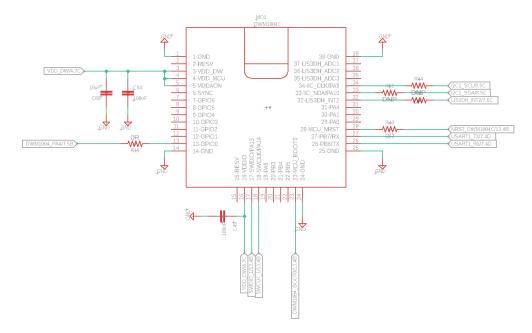


Figura 3.18: Schema elettrico DWM1004c.

3.2.4 Schema elettrico ST25R95

Questo è il modulo tramite il quale si implementerà la tecnologia RFID sulla scheda. Per il suo funzionamento l'ST25R95 ha un oscillatore RC interno a 32kHz, che genera un clock per le modalità low power del dispositivo; per il clock di sistema e le funzioni a RF invece è necessario collegare esternamente sui pin XIN e XOUT un cristallo a 27.12MHz, accompagnato da due capacità di 10pF per adattare la capacità di carico del cristallo di 6pF. Per minimizzare le capacità parassite sarà

opportuno in fase di routing posizionare il cristallo quanto più vicino possibile ai pin del dispositivo. Le alimentazioni sono due: VPS_TX e VPS. La prima si può vedere in Figura 3.19 ed alimenta la parte a radio frequenza. Essa è collegata all'alimentazione principale della scheda tramite capacità di disaccoppiamento (C82, C84, C85) e un filtro soppressore EMI.

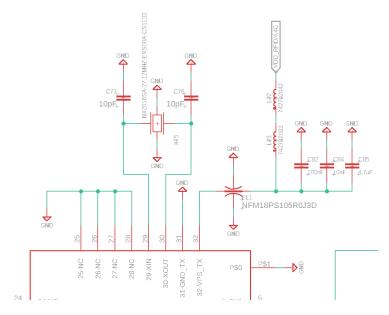


Figura 3.19: Cristallo ST25R95 e alimentazione.

Per quanto riguarda la comunicazione, il chip riceve i comandi tramite SPI, le cui linee sono collegate al micro principale. Inoltre come riportato nel datasheet [3], i pins SSI (Select Serial Interface), non possono essere lasciati floattanti, servono per selezionare la periferica SPI e devono essere configurati nel seguente modo:

Tabella 3.2: Collegamento Select Serial Interface

Pin	SPI Interface
SSI_0	1
SSI_1	0

Configurazione ottenuta collegando SSI_1 tramite una resistenza da 3.3k a GND e SSI_0 a VDD. Il pin ST_R1 è riservato e va connesso all'alimentazione tramite una resistenza da 3.3k come indicato nel datasheet.

In Figura 3.21 si possono notare i pin $IRQ_OUT\ IRQ_IN$ che servono rispettivamente il primo a mandare interrupts esterni verso il micro principale (ad esempio per segnalare che il dispositivo è pronto a mandare dei risultati), il secondo invece per ricevere degli interrupts (ad esempio per il risveglio dalle modalità low power). Entrambe le linee devono essere collegate tramite resistori di pull-up a VPS. Inoltre si vede la seconda alimentazione del chip VPS, che serve per l'alimentazione delle

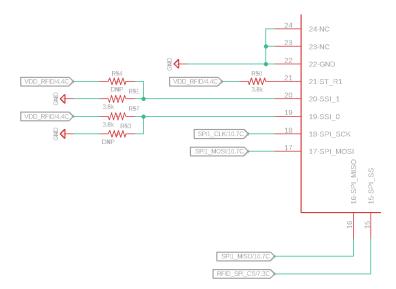


Figura 3.20: Comunicazione ST25R95.

parti digitali ed analogiche del dispositivo. Il pin ST_R0 è riservato e va connesso a GND tramite una capacità da 1nF come indicato nel datasheet.

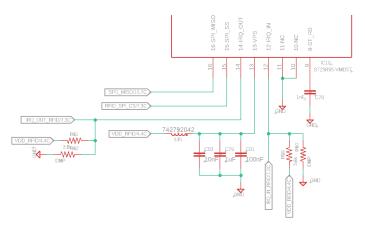


Figura 3.21: IRQ ST25R95.

Infine in Figura 3.22 si può osservare la struttura della rete necessaria per il il collegamento dell'antenna al dispositivo, di cui è mostrato lo schema elettrico in Figura 3.23. Il primo stadio composto dalle L24, L25, C69, C70, C72 e C73, implementa un filtro passabasso, un filtro EMI, per eliminare le emissioni spurie alle alte frequenze. Lo stadio successivo invece serve per adattare l'antenna all'impedenza del dispositivo e filtro. Queste capacità sono state lasciate non connesse per inserirle poi in un secondo momento dopo una fase di misure del dispositivo.

3.2.5 Ricaricatore, LDO e Ship Mode

Per il circuito di ricarica è stato scelto il dispositivo MCP73871 della Microchip. Questo è in grado di ricaricare la batteria connessa, e di fornire la tensione di

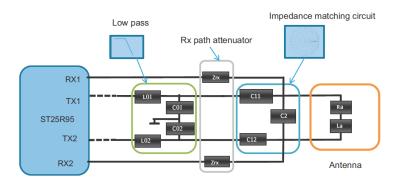


Figura 3.22: Struttura necessaria al collegamento dell'antenna.

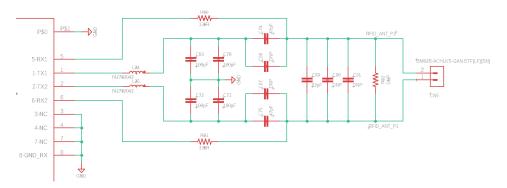


Figura 3.23: Schema elettrico rete antenna.

alimentazione alla scheda dall'uscita OUT (nel datasheet è consigliato di aggiungere una capacità di bypass dal valore minimo di 4.7 uF, Figura 3.24).

La ricarica viene effettuata tramite un connettore USB di tipo C, che viene collegato all'ingresso del caricatore. Il range di tensioni in input è compreso tra $0.3\mathrm{V}$ e $6\mathrm{V}$. Grazie alle uscite sui pin STAT1 e STAT2 è possibile monitorare lo stato di carica della batteria, mentre grazie al PG è possibile controllare quando la scheda è connessa ad una alimentazione esterna. Il terminale positivo della batteria deve essere connesso ai pin V_BAT , V_BAT2 e V_SENSE .

Per abbassare la tensione in uscita dal caricatore al valore di 3.3V è stato inserito un Low Dropout Regulator (Figura 3.25), ideale per le situazioni come questa in cui la tensione di alimentazione è molto vicina a quella fornita in output dal charger. A questo scopo è stato scelto il dispositivo TLV70233DBVR della Texas Instruments, che con un range di tensione in ingresso da 2V a 5.5V fornisce in uscita una tensione stabile di 3.3V. Per assicurare stabilità alla tensione è consigliato connettere all'ingresso e all'uscita del dispositivo una capacità di almeno 1uF.

Nei dispositivi vecchi, la batteria era estraibile, e quindi si poteva ovviare al problema della scarica della batteria del dispositivo semplicemente estraendola. Ora che la batteria è integrata, sono necessari dei componenti elettronici che mettano la batteria in modalità a bassissimo consumo, evitando che si scarichi troppo e si deteriori in fase di non utilizzo del dispositivo (Figura 3.26). Questa modalità è

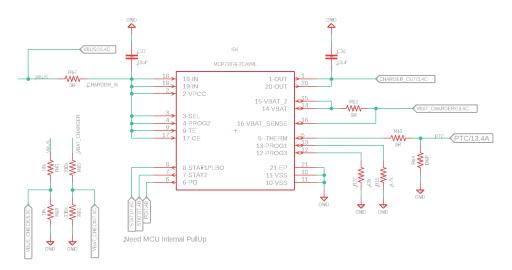


Figura 3.24: Schema elettrico circuito di ricarica.

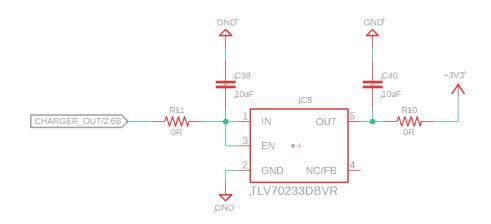


Figura 3.25: Schema LDO.



Figura 3.26: Schema Ship Mode.

implementata con uno switch, ed un reed, un interruttore che si chiude in presenza di un campo magnetico (es.calamita). Avvicinando un magnete al reed, questo si chiude la capacità C106 si scarica e lo switch IC13 si apre, scollegando la batteria dal resto del circuito. Per uscire dalla ship mode è sufficiente collegare la USB: a questo punto la capacità C106 si caricherà mandando alto l'enable dello switch, che si chiude e collega la batteria al sistema (Figura 3.27).

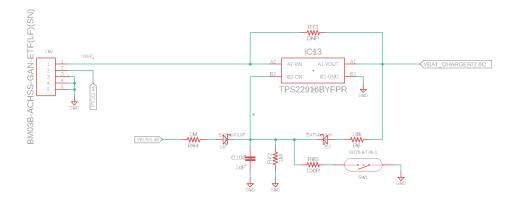


Figura 3.27: Schema Shipping Mode.

3.2.6 Componenti aggiuntivi

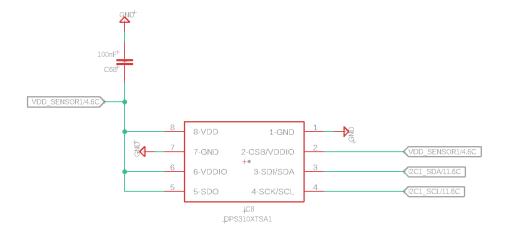


Figura 3.28: Schema barometro.

Barometro Per monitorare l'altezza del dispositivo, per verificare la condizione di uomo a terra, è importante disporre di uno strumento che sia più affidabile della localizzazione gps, che fornisce l'altitudine corrispondente ad una latitudine e una longitudine. Per questo motivo è stato inserito nel progetto un barometro. Il dispositivo Infineon sfrutta l'I2C per comunicare i risultati ottenuti al microcontrollore principale (Figura 3.28).

Capitolo 3 Schema elettrico

Memoria Per non avere dei limiti di memoria, per il caricamento del software applicativo sulla board, è stata inserita una memoria flash, collegata tramite SPI con il micro principale. In questo modo si è incrementata la capacità di storage della scheda di 16Mb (Figura 3.29).



Figura 3.29: Schema memoria.

Buzzer Per avere la possibilità di mandare dei segnali sonori in situazioni di rischio è stato aggiunto un buzzer, collegato tramite un mosfet ad un pin del micro principale (Figura 3.30).

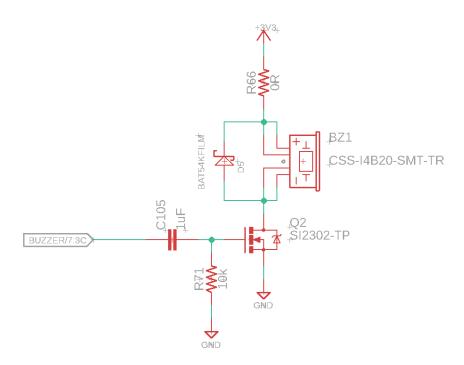


Figura 3.30: Schema buzzer.

Connettori e Test Points Infine sono stati inseriti i vari connettori: per il debug è stato scelto un cavo di tipo TC2050-IDC-NL con attacco direttamente sulla scheda, al quale sono collegati i pin dell'SWD, Reset e Boot del micro principale. In più a questo è stata collegata anche la periferica LPUART, per permettere la comunicazione

tramite UART. Per la tastiera è stato utilizzato un connettore FPC a 14 pin con pitch 0.5mm, a cui sarà poi collegata la tastiera con 4 leds e 4 bottoni. Le resistenze per limitare la corrente sui leds sono state montate direttamente sulla board con un valore di 100Ω . Tra i test points inseriti vi sono quelli per la programmazione del micro interno al modulo DWM1004C, per la misure di GND e tensione di alimentazione, e misure sulle linee dell'SPI (Figura 3.34).

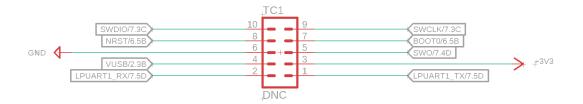


Figura 3.31: Connettore per il debug.

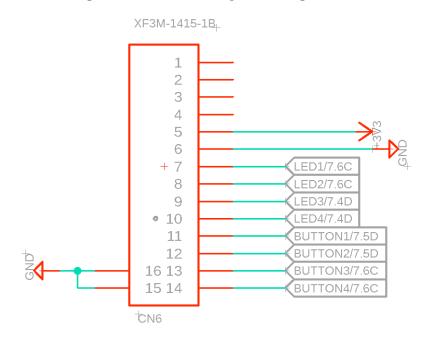


Figura 3.32: Connettore per la tastiera

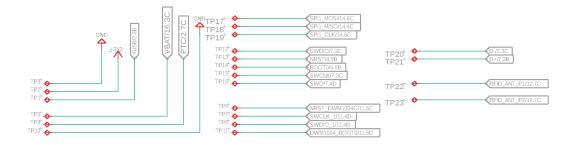


Figura 3.33: Test points.

Figura 3.34: Connettori sulla board.

Capitolo 4

Layout del PCB

4.1 Scelta dell'enclosure e forma del PCB

Una volta completato lo schema elettrico, la prima fase per la realizzazione fisica della board, consiste nella scelta dell'enclosure che conterrà il PCB e quindi della forma e dimensioni della scheda, che dovranno chiaramente essere adattate al box. Tenendo a mente a grandi linee i componenti utilizzati per il progetto (in particolare bisogna prestare particolare attenzione alle antenne GPS, RFID, e alla batteria che in questo caso sono i componenti più ingombranti che dovranno essere montati sulla scheda), si vogliono ottenere per il dispositivo delle dimensioni simili a quelle di un badge. Tra le varie possibilità di seguito è riportato il disegno tecnico dell'enclosure scelto e delle sue specifiche.

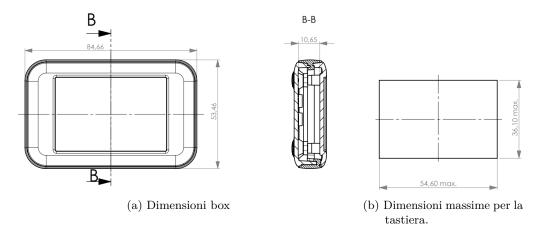


Figura 4.1: Specifiche enclosure [5].

La scatola è fornita di un ring che permette di rialzare e quindi aumentare lo spessore complessivo di 2mm, fino ad un totale di 10.65mm, ed in questo caso sarà utilizzato per permettere l'inserimento del PCB e batteria al suo interno. Per dimensionare e dare una forma alla board, si sono seguite anche qui le specifiche tecniche dell'enclosure riportate di seguito, che prevedono 4 fori per le viti di fissaggio del PCB. In Eagle una volta completato lo schema elettrico, è possibile generare in automatico una board dallo schematico, in cui verranno anche trasferiti tutti i componenti utilizzati con i loro footprints. Per dare una forma al PCB occorre

utilizzare il layer *Dimension*, sul quale si traccia il contorno della scheda con le dimensioni desiderate.

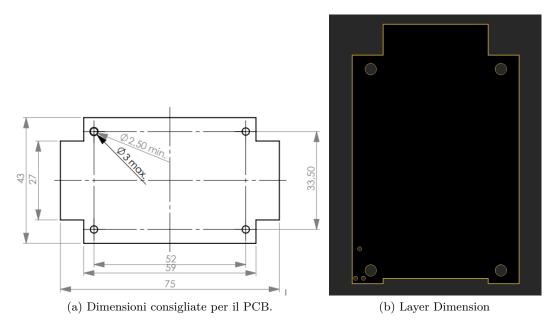


Figura 4.2: Outline del PCB.

A questo punto bisognerà anche decidere come disporre i vari componenti sulla scheda, e come sfruttare le due facce del PCB: per l'antenna GPS e il modulo UWB è stato scelto il lato bottom della scheda, mentre per il resto dei componenti è stato scelto il lato top. Per interferire il meno possibile con l'antenna GPS, la batteria sarà montata invece sul lato top, e su di essa sarà attaccata l'antenna RFID, per la quale è stato scelto un modello schermato con ferrite, che si presta in modo ottimale per essere posizionata come in questo caso sopra ad una batteria. La tastiera sarà dal lato opposto in modo che alla pressione di un pulsante, si possa attivare la funzionalità RFID del dispositivo. Ovviamente in un secondo momento bisognerà fresare opportunamente la scatola per l'inserimento della tastiera a membrana, per la quale sarà fatto realizzare un modello custom appositamente per la scatola scelta, nonchè un foro per il connettore USB di tipo C, sul lato della scatola.

4.2 Placement dei componenti

Una volta decisa la forma e le dimensioni del PCB è necessario posizionare tutti i componenti necessari sulla scheda. La fase della disposizione dei componenti nel processo di creazione del layout del PCB è, al tempo stesso, sia un'arte che una scienza e richiede una particolare attenzione allo spazio fisico presente sulla scheda. Anche se questo processo può essere difficile, il modo in cui si posizionano i componenti elettronici, determina quanto poi sarà facile o difficile produrre la scheda e quanto sarà in grado di rispettare le specifiche di progetto. Per fare questo bisogna innanzitutto

scegliere la posizione delle antenne, infatti il resto dei componenti si svilupperà attorno ad esse, ed in base ai modelli scelti, bisogna cercare di posizionarle quanto più similmente possibile a quello che è indicato nel datasheet, per non ritrovarsi con un antenna completamente disadattata. Inoltre per ciascuna bisogna tenere conto se necessita di piano di massa oppure no.

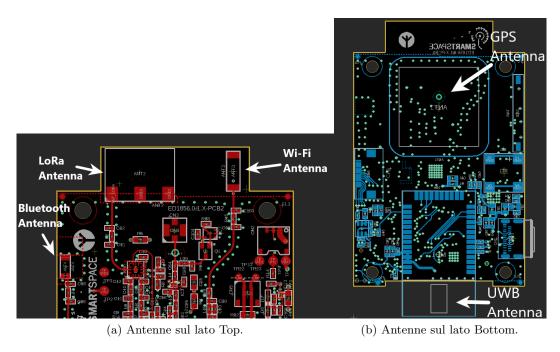
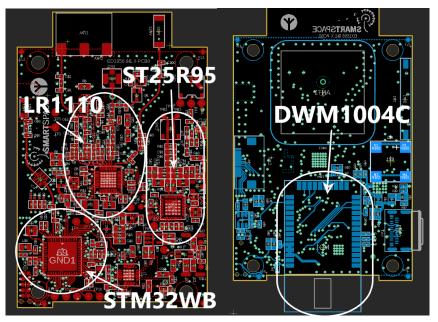


Figura 4.3: Antenne sulla scheda.

Per quanto riguarda le antenne LoRa e Wi-Fi, sono posizionate sul lato top della scheda nella parte superiore, che è la soluzione migliore per seguire il montaggio raccomandato presente nel datashhet. in quanto entrambe non necessitano di un piano di massa, esso non è stato inserito in quella zona. Inoltre è consigliato che al di sotto di esse non siano presenti delle piste, negli strati sottostanti, per evitare l'interferenza con esse. Per quanto riguarda l'antenna bluetooth, anche qui per seguire i consigli dei costruttori, essa è stata posizionata sul lato sinistro della scheda, sul lato top. Anch'essa non necessita di un ground plane, e da datasheet per un corretto funzionamento ha bisogno di una clearance area di $4.00 \times 4.25mm$. Per quanto riguarda l'antenna GPS, essendo questa di dimensioni considerevoli, non si ha molta scelta, ed è stata montata sul lato bottom e su piano di massa, come indicato nel datasheet. Anche in questo caso è bene fare in modo che al di sotto di essa si crei un piano di massa e si evitino piste che potrebbero andare ad interferire con il suo funzionamento. Infine l'antenna UWB è gia incorporata nel modulo DWM1004C, ed il suo posizionamento dipende dalla posizione di quest'ultimo. Per ottenere le migliori performances nel datasheet è specificato che l'antenna deve essere posizionata su una "keep out area" con un altezza di 10mm e larghezza 10mm minimo, ad entrambi i lati dell'antenna, senza parti metalliche all'interno, oppure

essere direttamente posta in modo da sporgere dal PCB. In questo caso particolare è stata scelta proprio questa seconda soluzione, modificando la forma della board, rispetto a quella raccomandata nel datasheet della scatola e creando una piccola rientranza, in modo da assicurare i 10mm di clearance all'antenna, ma anche di rientrare nella lunghezza totale della scatola. Per posizionare il resto degli elementi si sono seguite le varie guide presenti sui siti dei costruttori dei componenti principali, in particolare le Application Notes [11], [10] e [12], per le parti relative al micro principale, all'LR1110, e al ST25R95. In generale la regola principale da seguire per il placement dei componenti è quella di cercare per quanto possibile di creare dei percorsi convenienti per le piste che li dovranno connettere, e ove possibile cercare di posizionarli prevenendo eventuali incroci tra le linee. Inoltre è bene posizionare i componenti relativi ad un particolare circuito integrato vicino ad esso, come le resistenze e le capacità che vanno raggruppate nei pressi del device corrispondente. Di seguito sono riportate delle immagini relative alla disposizione sulla scheda dei blocchi principali, con dettagli ravvicinati sui blocchi del micro, LR1110 e ST25R95.



(a) Placement componenti sul top. (b) Placement componenti sul bottom

Figura 4.4: Placement componenti sulla scheda.

Il posizionamento dei blocchi principali è condizionato dalla posizione delle antenne: nella parte centrale della scheda è stato posizionato l'LR1110 con le varie reti di adattamento delle diverse antenne che alimenta. In particolare a sinistra c'è la rete relativa all'antenna LoRa con lo switch, centralmente quella per l'antenna GPS, ed infine a destra quella per l'antenna Wi-Fi (Figura 4.5).

I componenti dell'LR1110 sono stati raggruppati vicino al circuito integrato principale, si possono notare il cristallo a 32kHz (XT2) ed il TCXO (XT6) (Figura 4.6).

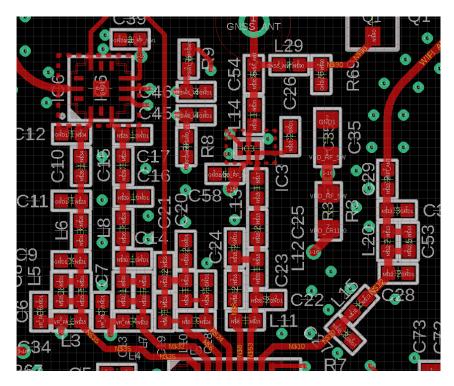


Figura 4.5: Reti di mactching.

Per il raffreddamento dell'IC sono stati inseriti dei vias al di sotto del pad di ground. Seguendo l'Application Note [10] vicino ai cristalli è stato creato un poligono di cutout, ovvero un poligono dal quale il rame è stato eliminato sia nel top layer che nei sottostanti, per isolare termicamente i cristalli.

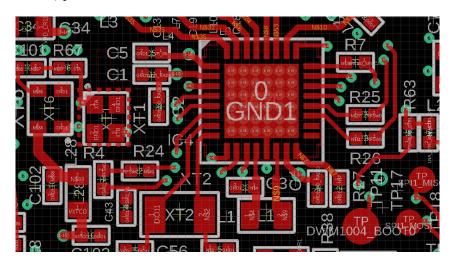


Figura 4.6: Circuiteria LR1110.

In Figura 4.7, si può osservare la parte relativa al ST25R95, con la rete di adattamento per le linee differenziali che arrivano al connettore dell'antenna, ed il cristallo XT5 a 27.12MHz per le sue operazioni. Si possono notare le piste più larghe da 0.6mm per le linee differenziali, per l'adattamento dell'impedenza dell'antenna.

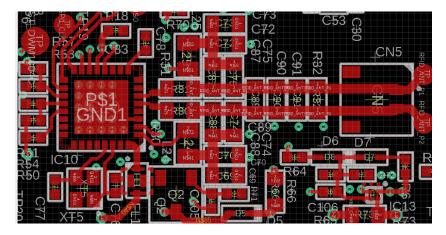


Figura 4.7: Circuiteria ST25R95.

Infine in Figura 4.8 è evidenziata la parte del microcontrollore principale, con i due cristalli XT3 ed XT4, e le varie capacità di disaccoppiamento per la sua alimentazione.



Figura 4.8: Circuiteria STM32Wb55.

4.3 Stack-up e Design rules

Stack-up In un primo momento per il PCB era stato scelto uno stack-up composto da 4 layers (rame(1)-core-rame(2)-prepreg-rame(4)-core-rame(16)) con vias 1/16, 1/2 e 3/16: il layer 1 (Top), riservato allo sbroglio dei segnali, layer interno 2 riservato prevalentemente al piano di massa, layer interno 3 riservato ai piani di alimentazione e layer 16 (Bottom), riservato allo sbroglio dei segnali della parte inferiore della scheda. Tuttavia in fase di progetto data l'elevata densità di componenti presenti sulla scheda, si è dimostrato complesso procedere con questo tipo di build-up, poichè sarebbe stato necessario portare sul layer interno 2 diverse linee, che sarebbero andate a degradare in modo considerevole il piano di massa. Inoltre il costo realizzativo della board

sarebbe stato elevato data la presenza dei blind vias (1/2 e 3/16). Per semplificare lo sbroglio si è quindi deciso di passare ad uno stack-up formato da 6 layers: (rame(1)-core-rame(2)-core-rame(3)-prepreg-rame(4)-core-rame(5)-core-rame(16)), con layer 1 (Top) riservato ai segnali, layer interno 2 riservato a GND, layer interno 3 riservato ai segnali, layer interno 4 riservato ai piani di alimentazione, layer interno 5 riservato ai segnali ed infine layer 6 (Bottom) per lo sbroglio di segnali sulla parte inferiore della scheda. In questo modo è stato possibile semplificare il routing, avendo più layers a disposizione, nonchè avere un costo del dispositivo più favorevole non essendo presenti blind vias ma solo vias passanti 1/16. Il materiale scelto per il dielettrico è FR4 stndard. Le dimensioni dei vari layers sono quelle di un PCB standard, e sono di seguito riportate:

- Copper 18 um-plating to 35um
- Dielectric 0.18 mm dielectric constant 4.29
- Copper 35 um
- Dielectric 0.2 mm dielectric constant 4.29
- Copper 35 um
- Dielectric 0.18 mm dielectric constant 4.29
- Copper 35 um
- Dielectric 0.2 mm dielectric constant 4.29
- Copper 35 um
- Dielectric 0.18 mm dielectric constant 4.29
- Copper 18 um-plating to 35um

Con queste dimensioni lo spessore totale del PCB è di 1mm.

Design Rules Eagle mette a disposizione il tool "Design rules check (DRC)", che permette di controllare in modo automatico se in tutto il circuito vengono rispettate delle regole prestabilite (come ad esempio la spaziatura minima tra le piste o la larghehzza minima delle stesse), e queste devono essere impostate manualmente, tenendo conto anche delle specifiche del progetto. Inoltre è presente anche il tool "Electric rules check (ERC)" per verificare che i collegamenti effettuati sulla board corrispondano effettivamente a quelli dello schema elettrico realizzato. Una prima rule da settare è la clearance, ovvero la distanza minima da mantenere tra gli oggetti presenti nel layer dei segnali: in particolare è stata scelta una distanza minima di 0.15mm tra piste, pads smd e vias. Si ha poi la Distance, ovvero la distanza minima tra qualsiasi oggetto nel layer dei segnali (via, pista, pad smd ecc.) e i bordi fisici

della scheda che è stato impostato a 0.25mm. Successivamente si devono definire la larghezza minima di vias e piste: per queste ultime il valore minimo è di 0.2mm, e sarà utilizzato per tutte le piste legate a segnali, mentre per le piste di alimentazione è opportuno utilizzare ove possibile delle piste più larghe, poichè devono trasportare delle correnti più elevate, e sono state progettate con una larghezza di 0.5mm. Diverso il discorso per le piste che devono trasportare i segnali a radiofrequenza, che saranno in genere più larghe di quelle dei segnali normali. Esse infatti devono essere progettate in modo da matchare l'impedenza dei componenti che collegano in genere 50Ω . Per queste utilizzando il tool "PCB Toolkit" è possibile inserendo i parametri del proprio PCB, calcolare la larghezza ottimale da utilizzare per avere un'impedenza desiderata. In questo caso si è utilizzato un valore di 0.35mm per piste singole, mentre 0.6mm per piste differenziali (come nel caso dell'antenna RFID), anche queste calcolate con il tool nell'apposita applicazione per piste differenziali. Infine occorre definire anche le dimensioni per i vias, dei quali due sono i parametri principali: il diametro del foro e l'annular ring, ovvero l'anello di rame posizionato attorno al foro del via, che viene definito in percentuale rispetto al diametro del foro. Sul PCB sono presenti dei vias di diverse dimensioni, a seconda della funzione particolare del foro: per il passaggio di segnali da un laver all'altro sono utilizzati dei vias da 0.2mm così come quelli per il collegamento del piano di massa. Per le piste di alimentazione invece sono stati utilizzati dei vias dal diametro maggiore, di 0.5mm.

4.4 Routing

Stabilite le rules si effettua il routing del PCB, collegando con delle piste i vari componenti della scheda, seguendo lo schema elettrico. Il layer "Unrouted" ci assiste in questo, mostrando con delle linee gialle i collegamenti ancora da fare. Eagle dispone anche di un "auto-router", ma è sempre bene per i collegamenti più complessi ricorrere a quello manuale. Ove possibile le piste sono state create sul layer top e bottom. Ovviamente per collegare i dispositivi più lontani tra loro, essendo la scheda molto densa di componenti, questo non è stato possibile e si è dovuto ricorrere ai layer sottostanti: in particolare nel layer interno 3 sono state trasferite le linee dell'SPI che collegano i tre dispositivi principali, I2C, e le piste che collegano i GPIO del microcontrollore principale. Il layer interno 2 è riservato al piano di ground ed è stato utilizzato solo per la pista della VUSB, che trovava degli impedimenti in tutti gli altri layer. Per alcune delle piste che non potevano essere portate sul layer 3, è stato utilizzato il layer 5 nella parte inferiore, mentre nella parte superiore è stato lasciato vuoto per realizzare il piano di massa per l'antenna GPS. Le piste a radiofrequenza sono tutte sul top layer, e si è cercato di non creare altre piste subito al di sotto di esse per evitare eventuali interferenze. Inoltre per realizzare delle curve è stata utilizzato un contorno più smussato e sono stati evitati gli spigoli come per le altre. Attorno ad esse sono stati posizionati dei vias a GND per cercare di ricreare una struttura guidata per il segnale (Figura 4.9 e Figura 4.10).

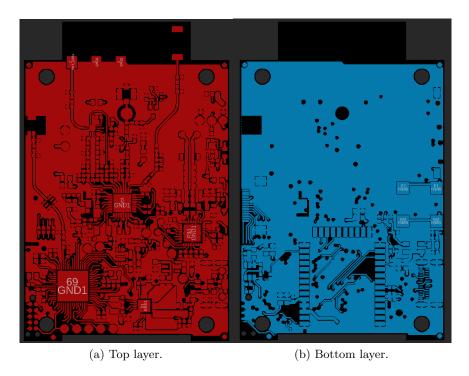


Figura 4.9: Top layer e Bottom Layer.

Creati tutti i collegamenti con il comando "Ratsnest" tutti i layer sono metallizzati, lasciando un certo spazio previsto dalla clearance dalle piste e pads, ma connettendo tutti i pads GND della scheda. Sono stati poi aggiunti dei vias, per collegare i piani di massa. Infine sul layer 4 (Figura 4.11) le varie alimentazioni sono state organizzate in poligoni con gerarchia diversa, e quello con gerarchia maggiore è la +3.3V in uscita dall'LDO che alimenta tutta la scheda. Ciascun componente ha il suo poligono di alimentazione al di sotto, collegato al piano +3.3V tramite una resistenza 0R e una capacità di 10uF, posizionate a cavallo tra i poligoni. Una volta terminato il routing del PCB il progetto è completato, e occorre generare i file di produzione che saranno mandati a chi realizzerà fisicamente il PCB ed assemblerà i componenti su di esso. Per fare ciò si generano i file Gerber, che sono di fatto lo standard utilizzato per la produzione di PCB, e contengono tutte le istruzioni per la produzione della scheda. I file possono essere generati in automatico dal tool "Cam Processor" messo a disposizione da Eagle, che crea un modello geometrico per ogni layer della scheda. Essendo la scheda composta da 6 layer dovranno essere presenti almeno 14 file per la produzione: 1 file con l'outline della scheda corrispondente al layer tDimension di Eagle, 2 file conterranno le informazioni sul rame presente sul top e sul bottom che corrisponderanno quindi ai layer Top e Bottom su Eagle (Figura 4.9), ed analogamente altri 4 per il rame sui layers interni, (Figura 4.11) e (Figura 4.10). Poi saranno presenti due file contenenti il layer tCream di Eagle, sia per il top che per il bottom, dove sono riportati tuti i pads dei componenti e che costituisce il file per la produzione della lamina che servirà per applicare la pasta

saldante sul PCB (Figura 4.12).

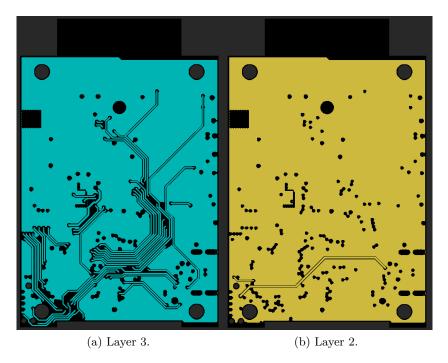


Figura 4.10: Layer 3 (segnali) e 2 (GND).

Serviranno poi altri due file, sempre per il top e bottom, che specificano le zone in cui non andrà applicata la maschera protettiva della scheda (pads e fori), che corrisponde al layer tStop di Eagle(Figura 4.12). Infine due files che corrispondono alla serigrafia della scheda sulle due facce (layer tPlace)(Figura 4.13), ed uno per le posizioni di tutti i fori presenti sulla board (tDrills). In aggiunta possono essere generati anche due files utili per chi assembla il PCB, che riportino il tDocu, e tPlace.

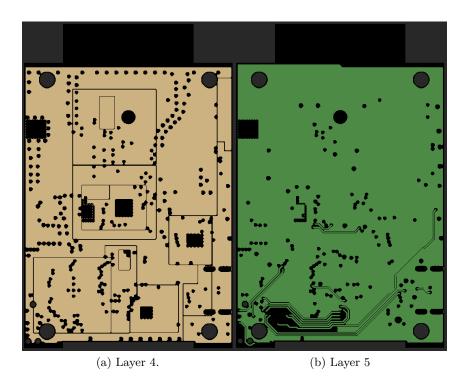


Figura 4.11: Layer 4 e layer 5.

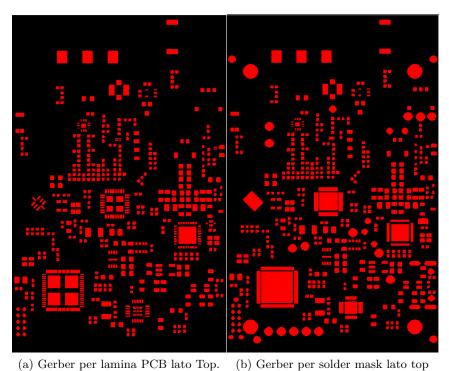


Figura 4.12: File gerber per lamina e solder mask.



Figura 4.13: File gerber per la serigrafia.

Capitolo 5

Scheda per lo sviluppo del DWM1004C

Poichè i produttori del dispositivo DWM1004C, non forniscono una board per il test dell'integrato, si è realizzata una semplice developement board per facilitare lo sviluppo del software UWB, e per poter fare dei test sul DWM1004C anche senza dover utilizzare la board principale sviluppata fino a qui. In questo caso la board è molto semplice, ed è costituita da uno stack-up di 2 soli layer, che sono sufficienti allo sbroglio di tutti i segnali.

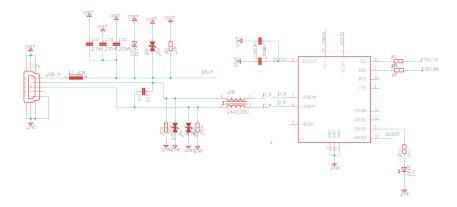


Figura 5.1: Schema elettrico board di sviluppo.

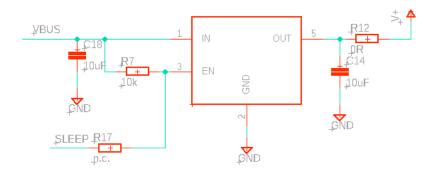


Figura 5.2: LDO utilizzato per alimentare la scheda.

Lo schema elettrico riportato nelle figure (Figura 5.1 e Figura 5.2) è molto semplice: la scheda è alimentata tramite un connettore USB mini, e la tensione è regolata a 3.3V da un LDO, lo stesso utilizzato nella board principale. Per fornire uno strumento aggiuntivo agli sviluppatori è stato montato un circuito integrato che permette di comunicare con la periferica UART interna al microcontrollore del modulo tramite USB, sfruttando le due linee dati della USB. In questo modo è possibile mandare dei comandi al micro direttamente tramite UART collegando il dispositivo tramite USB ad un computer. I pin del modulo sono stati tutti collegati ai tre connettori pinstrip, grazie ai quali si può accedere in modo semplice ad ogni periferica del modulo. La programmazione e il debug può essere fatta sfruttando un ST-Link da collegare ai pin VDD, GND, SWD e SCK.

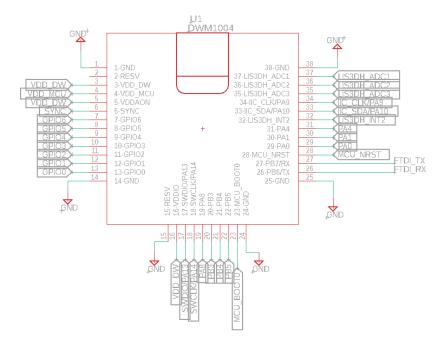
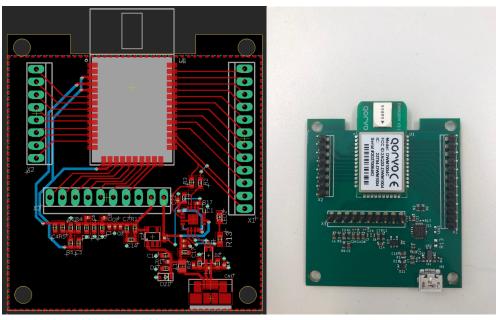


Figura 5.3: Segnali DWM1004C.

Anche in questo caso sono stati seguiti gli step di progetto precedentemente illustrati per la board principale. In questo caso è stata scelta una clearance di 0.2mm, ed una larghezza minima per le piste di 0.2mm e per i vias di 0.3mm, con le piste di alimentazione da 0.5mm. La forma del PCB si adatta alle specifiche richieste dal modulo DWM1004C, che prevede una keep-out area sotto e ai lati dell'antenna, ed in base a questo è stato sagomato il bordo superiore della scheda. I componenti sono tutti montati sul lato top della board, mentre il bottom è stato utilizzato per le piste di alimentazione del modulo.



(a) Progetto della board su Eagle.

(b) Immagine della board.

Figura 5.4: Board di sviluppo DWM1004C.

Capitolo 6

LR1110 drivers

6.1 Host-Controller Interface

L'LR1110 è dotato di drivers che permettono al dispositivo di comunicare con una MCU esterna e viceversa, tramite un set di comandi e risposte SPI. Il pin BUSY è utilizzato per segnalare se l'LR1110 è pronto ad accettare un comando oppure no, è quindi necessario controllarne lo stato prima di mandare un comando.

6.1.1 Write Commands

Con questo tipo di comandi, l'LR1110 ritorna all'host lo status register e l'interrupt register sul pin MOSI, a seconda della lunghezza dell'istruzione e dei suoi parametri. L'host manda un opcode di 16 bit che rappresentano il comando da eseguire, seguito dai parametri necessari. Il pin BUSY è automaticamente portato alto non appena il pin NSS diviene basso. Una volta che l'LR1110 finisce di processare il comando, il pin torna basso (Figura 6.1), indicando che il dispositivo è nuovamente pronto per ricevere un comando.

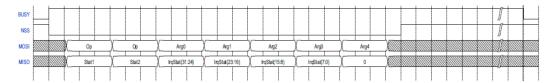


Figura 6.1: Timing delle linee BUSY e SPI durante un Write Command

6.1.2 Read Commands

Con questa tipologia di comandi è possibile ricevere dati dall'LR1110, come stato interno o risultati della geolocalizzazione ad esempio. Anche qui l'host manda una parola di 16 bit che indica l'operazione seguita da eventuali parametri, e il pin BUSY passa alto sulla caduta dell'NSS. Quando l'LR1110 ha preparato i dati richiesti, il pin BUSY passa nuovamente basso. A questo punto l'host può leggere i dati inviando un mumero di dummy bytes (0x00) pari al numero di bytes che si vogliono leggere per far shiftare i dati sulla SPI (Figura 6.2).

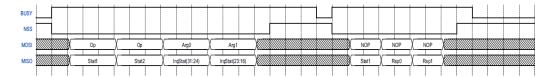


Figura 6.2: Timing delle linee BUSY e SPI durante un Read Command

6.2 Scansione GNSS

6.2.1 Introduzione generale al sistema GNSS

Le potenzialità del dispositivo LR1110 permettono di realizzare una geolocalizzazione outdoor veloce e energeticamente efficace. Il basso consumo energetico è ottenuto demandando le operazioni computazionalmente più complesse, e dispendiose in termini di tempo, ad altri componenti. In particolare per rendere funzionante il sistema di geolocalizzazione dell'LR1110 sono necessari i seguenti componenti di back-end:

- Componente per il calcolo della posizione: l'LR1110 non ha la funzionalità di calcolo della posizione, le misure dei segnali ottenuti con la scansione, sono combinati in un messaggio binario (messaggio NAV) che deve essere inviato tramite un canale di comunicazione al calcolatore della posizione. Questo componente è richiesto in ogni modalità operativa.
- Componente per l'aggiornamento delle effemeridi (richiesto nella modalità assistita): l'LR1110 è capace di ridurre il tempo di scansione tenendo conto dei parametri orbitali per le differenti costellazioni di satelliti. Congiuntamente ad una stima del tempo e della posizione, l'LR1110 utilizza queste informazioni per ottimizzare l'acquisizione dei segnali. Con il passare del tempo, la vera posizione dei satelliti diverge da quella fissa delle effemeridi, e quindi queste ultime devono essere aggiornate. Questa operazione può essere espletata da questo componente back-end che stima la qualità delle effemeridi presenti sul dispositivo e le aggiorna quando necessario.
- GNSS assistance component (richiesto in modalità assisita): per eseguire la scansione in modalità assistita, all'LR1110 sono necessarie una stima grossolana del tempo e della posizione attuale. Queste informazioni possono essere fornite in svariati modi: la posizione ad esempio può essere dedotta da scansioni precedenti.

La (Figura 6.3) mostra i componenti del sistema per una integrazione basata su LoRaWAN: in modalità di scansione GNSS, l'LR1110 ricerca segnali dipsponibili ed estrae un set minimo di informazioni dai satelliti, necessarie per il calcolo della posizione. Il segnale acquisito dalla scansione (NAV message) è poi trasmesso al

componente che si occupa del calcolo della posizione. Per l'aggiornamento delle effemeridi il componente relativo prepara un appropriato messaggio.

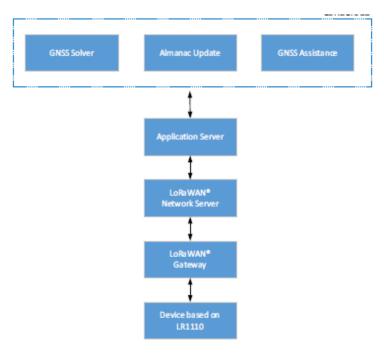


Figura 6.3: Overview del sistema GNSS

6.2.2 Principi di funzionamento

Le modalità implementate sono due:

- La modalità di scansione autonoma non richiede aluna posizione di assistenza o effemeridi dei satelliti, e punta ad acquisire i segnali più potenti dei satelliti. Quindi essa è adatta per scansioni all'aperto con una buona visibilità del cielo.
- La modalità assistita invece è più efficiente. Le informazioni aggiuntive permettono di costruire una lista dei satelliti in vista nel momento della scansione per ridurre lo spazio di ricerca dei stalliti e ottimizzare il tempo e l'energia spesi per le operazioni. queste informazioni consistono in:
 - La posizione approsimativa del dispositivo (in un range di 150km).
 - L'informazione di tempo (nel range di 1s).
 - Effemeridi aggiornate (non più vecchie di tre mesi).

L'LR1110 supporta sia segnali GPS L1 che segnali BeiDouB1, inoltre è capace di effettuare sia una singola scansione in una sola o entrambe le costellazioni, oppure una scansione doppia in una o entrambe le costellazioni. Durante la scansione il segnale BUSY è alto, indicando che il dispositivo non è pronto ad accettare messaggi

tramite SPI. Esso torna al livello basso quando la procedura è completata. Se è stato attivato l'interrupt *GNSSDone*, il pin IRQ va alto alla fine della procedura. Obbligatoria la presenza di un oscillatore per qualsiasi operazione di scansione.

6.2.3 Driver GNSS

Di seguito sono dettagliate le funzioni usate per configurare e inizializzare l'acquisizione del segnale GNSS che poi potrà essere condiviso per richiedere una geolocalizzazione. Ciascun componente è basato su questi file:

- lr1110_component.c: implementazione delle funzioni
- lr1110_component.h: dichiarazioni delle funzioni
- lr1110_component_types.h: definizione dei tipi

Di seguitono riportate le funzioni disponibili. Inoltre per quelle principali, sono evidenziati anche i bit che costituiscono il codice dell'operazione e degli argomenti che vengono trasmessi all'LR1110 tramite SPI.

SetGNSSConstellationToUse Con questo comando è possibile configurare la scansione per la costellazione di satelliti scelta (GPS,BEIDOU). La funzione utilizzata è:

```
• lr1110_status_t lr1110_gnss_set_constellations_to_use(
    const void* context,
    const lr1110 gnss constellation mask t constellation mask);
```

@param [in] context Chip implementation context

@param [in] constellation_mask Bit mask of the constellations to use.

@returns Operation status

Tabella 6.1: SetGNSSConstellationToUse.

Byte	0	1	2
Data from Host	0x04	0x00	ConstellationBitMask(7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)

Con la ConstellationBitMask si sceglie la costellazione tra GPS e BEIDOU oppure entrambe:

- bit0=1:GPS selezionato
- bit1=1:BeiDou selezionato
- Gli altri valori sono RFU

E' disponibile un'altra funzione che permette di leggere la costellazione utilizzata:

lr1110_status_t lr1110_gnss_read_used_constellations (
 const void* context,
 lr1110_gnss_constellation_mask_t* constellations_used)

@param [in] context Chip implementation context

@param [out] constellations_used bit che individuano la costellazione utilizzata.

@returns Operation status

Ed infine una funzione che legge le costellazioni supportate:

lr1110_status_t lr1110_gnss_read_supported_constellations (
 const void* context,
 lr1110_gnss_constellation_mask_t* supported_constellations)

Funzione per leggere le costellazioni supportate.

@param [in] context Chip implementation context

@param [out] supported_constellations bit che individuano la costellazione utilizzata. @returns Operation status

GnssSetMode Questo comando permette di configurare per una singola o doppia scansione per la costellazione selezionata in precedenza.

• lr1110_status_t lr11110_gnss_set_scan_mode(
 const void* context,
 const lr1110_gnss_scan_mode_t scan_mode,
 uint8_t* inter_capture_delay_second);

@param [in] context Chip implementation context

@param [in] scan_mode configura scansione singola o doppia (0 or 1)

@param [out] inter_capture_delay_second ritardo temporale tra le due scansioni @returns Operation status

Tabella 6.2: GnssSetMode.

Byte	0	1	2
Data from Host Data to Host	0x04 Stat1	0x08 Stat2	GnssMode IrqStatus(31:24)

Con i bit GnssMode si imposta il tipo di scansione:

• 0x00:scansione singola

• 0x01: scansione doppia

• Gli altri valori sono RFU

Capitolo 6 LR1110 drivers

E' disponibile anche una funzione che avvia la seconda scansione:

 lr1110_status_t lr1110_gnss_scan_continuous(const void* context)

@param [in] context Chip implementation context@returns Operation status

GnssAutonomous Questo comando permette di entrare in modalità scansione autonoma, utile se non si hanno a disposizione informazioni aggiuntive, o per una veloce outdoor/indoor detection.

• lr1110_status_t lr1110_gnss_scan_autonomous(
 const void* context,
 const lr1110_gnss_date_t date, c
 onst lr1110_gnss_search_mode_t effort_mode,
 const uint8_t gnss_input_parameters, const uint8_t nb_sat);

@param [in] context Chip implementation context

@param [in] date Data dell'attuale scansione. Il suo formato è in secondi trascorsi dal 6 Gennaio 1980 00:00:00 (GPS time).

@param [in] effort_mode Effort mode @ref lr1110_gnss_search_mode_t

@param [in] gnss_input_parameters bit indicanti quale informazione aggiuntiva è contenuta nell'output.

@param [in] nb_sat Numero massimo di satelliti.Valore compreso tra [0:128] @returns Operation status

Tabella	6.3:	GNSS	Autonomous.

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x04	0x06	Time (31:24)	Time (23:16)	Time (15:8)	Time (7:0)	Effort Mode	Result Mask	NbSv Max
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0	0

- Time: tempo GPS (GPST), in numero di secondi passati dal 6 gennaio 1980.
- EffortMode=0x00
- ResultMask: bit indicanti quale informazione è aggiunta al payload in output:
 - bit 0: presenza di un timestamp nell'output

- bit 1: presenza di effetto doppler in uscita
- bit 2: presenza di un cambio di bit in uscita
- NbSvMax definisce il numero massimo di satelliti voluti come risultato della scansione. Se sono rilevati più satelliti allora sono ritornati solo quelli con C/N più alto. Se NbSvMax=0 allora sono ritornati tutti i sattelliti rilevati.

Chiamando questo comando si resettano i risultati precedenti.

GnssAssisted Il comando permette di eseguire una scansione con dati aggiuntivi (tempo corrente, posizione approssimativa, effemeridi).

```
• lr1110_status_t lr1110_gnss_scan_assisted
          ( const void* context,
          const lr1110_gnss_date_t date,
          const lr1110_gnss_search_mode_t effort_mode
          const uint8_t gnss_input_parameters, const uint8_t nb_sat );
```

@param [in] context Chip implementation context

@param [in] date Data dell'attuale scansione. Il suo formato è in secondi trascorsi dal 6 Gennaio 1980

@param [in] effort_mode Effort mode

@param [in] gnss_input_parameters bit indicanti quale informazione aggiuntiva è contenuta nell'output

@param [in] nb_sat Numero massimo di satelliti.Valore compreso tra [0:128] @returns Operation status

3 4 5 7 Byte 6 0 1 8 Data from 0x0ATime Time Effort NbSv0x04Time Time Result Host (31:24)(23:16)(15:8)(7:0)Mode Mask Max Data IrqStatus IrqStatus IrqStatus IrqStatus Stat1 Stat2 to Host (31:24)(23:16)(15:8)(7:0)

Tabella 6.4: GNSS Assisted.

- Time: tempo GPS (GPST), in numero di secondi passati dal 6 gennaio 1980.
- EffortMode
 - 0x00: modalità low power. La scansione si ferma se non sono rilevati dei segnali forti.
 - 0x01: Best Effort Mode, la scansione continua anche se non sono rilevati segnali forti,

Tabella 6.5: GnssAssistancePosition.

Byte	0	1	2	3	4	5
Data from Host	0x04	0x10	Latitude (15-8)	Latitude (7:0)	Longitude(15-8)	Longitude (7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus(7:0)

- ResultMask: bit indicanti quale informazione è aggiunta al payload in output:
 - bit 0: presenza di un timestamp nell'output
 - bit 1: presenza di effetto doppler in uscita
 - bit 2: presenza di un cambio di bit in uscita
- ResultMask: bit indicanti le informazioni aggiunte nel payload di output. Settato a 0x03.
- NbSvMax definisce il numero massimo di satelliti voluti come risultato della scansione.

GnssSetAssistancePosition Comando per configurare la posizione approssimativa nella modalità assistita.

```
    lr1110_status_t lr1110_gnss_set_assistance_position
        (const void* context,
        const lr1110_gnss_solver_assistance_position_t* assistance_position );
```

@param [in] context Chip implementation context

@param [in] assistance_position, latitudine e longitudine in 12 bits

@returns Operation status

- Latitude: latitudine codificata in 12 bits (risoluzione di 0.044 gradi). latitude= latitudine in gradi (valore decimale)*2048/90. Es. 47.006-> latitude=47.006*2048/90=1070=0x042E
- Longitude: longitudine codificata in 12 bits (risoluzione di 0.088 gradi)
 Longitude= longitudine in gradi (valore decimale)*2048/180.

Inoltre vi è una funzione che ritorna la posizione approssimativa utilizzata:

lr1110_status_t lr1110_gnss_read_assistance_position(
 const void*context,
 lr1110_gnss_solver_assistance_position_t* assistance_position);

Funzione per leggere la posizione di assistenza. @param [in] context Chip implementation context @param [in] assistance_position, latitude 12 bits and longitude 12 bits @returns Operation status



Figura 6.4: Struttura NAV message.

I risultati delle scansioni sono formattati come NAV messages, di lunghezza variabile a seconda di quanti satelliti sono stati rilevati e alla modalità di scansione singola o doppia. Questo messaggio può essere mandato all'host (per informazioni di stato) al Solver (per il calcolo della posizione su cloud) o al DMC (per il management delle effemeridi). Per leggere i risultati occorre conoscere la dimensione dello stream di bit da leggere usando l'apposito comando.

Descrizione del messaggio NAV Il formato del messaggio è mostrato in (Figura 6.4). Esso è composto da un campo DestinationID, seguito da un payload di lunghezza variabile:

- DestinationID=0x00:NAV message all'host
- DestinationID=0x01:NAV message al GNSS solver
- DestinationID=0x02:NAV message al DMC

I NAV messages con DestinationID=0x01 e 0x02 devono essere inviati al GNSS solver e DMC dall'host. Per quanto riguarda il NAV message diretto all'host, esso è composto da un payload di un singolo byte così codificato:

- 0x00: OK
- 0x01: comando inaspettato
- 0x02: comando non implementato
- 0x03: parametri del comando non validi
- 0x04: messaggio di errore sul check
- 0x05: scansione fallita
- 0x06: mancanza dell'informazione di tempo
- 0x07: nessun satellite rilevato
- 0x08: effemeridi troppo vecchie
- -0x09: fallimento aggiornamento delle effemeridi per errore di CRC
- -0x0A: fallimento aggiornamento delle effemeridi per errore di integrità della flash

Capitolo 6 LR1110 drivers

- 0x0B: fallimento aggiornamento effemeridi perchè troppo vecchie
- 0x0C: aggiornamento effemeridi non consentito
- 0x0D: errore CRC delle effemeridi
- 0x0E: versione effemeridi non supportata

GnssGetResultSize Questo comando permette di leggere la dimensione in bytes dello stream contenente i risultati della scansione. Deve obbligatoriamente essere chiamato prima di leggere i risultati.

```
    lr1110_status_t lr11110_gnss_get_result_size(
        const void* context,
        uint16_t* result_size )
```

@param [in] context Chip implementation context

@param [out] result_size Result size

GnssReadResults Comando che permette di recuperare i risultati dell'ultima scansione. Deve essere chiamato subito dopo la lettura delle dimensioni del risultato.

```
    lr1110_status_t lr1110_gnss_read_results(
        const void* context,
        uint8_t* result_buffer, const uint16_t result_buffer_size );
```

I risultati GNSS sono inseriti direttamente in un buffer. Il buffer è passato a questa funzione e deve essere lungo abbastanza da contenere almeno result_buffer_size bytes. Non è fatto nessun controllo sul result_buffer_size, e se è troppo piccolo ci sarà un bug di overflow.

@param [in] context Chip implementation context

@param [out] result_buffer Application provided buffer to be filled with result @param [in] result_buffer_size The number of bytes to read from the LR1110. result_buffer must at least contains result_buffer_size bytes.

GnssGetNbSvDetected Comando che fornisce il numero di satelliti rilevati nell'ultima scansione:

```
    lr1110_status_t lr1110_gnss_get_nb_detected_satellites(
        const void* context,
        uint8_t* nb_detected_satellites);
```

```
    @param [in] context Chip implementation context
    @param [out] nb_detected_satellites Number of satellites detected
    @returns Operation status
```

GnssGetSvDetected Comando che permette di ritornare l'ID e il C/N dei satelliti rilevati durante l'ultima scansione.

GnssGetConsumption Comando che permette di ottenere il tempo speso nell'acquisizione e analisi del segnale e che permettono di determinare il consumo di corrente dell'ultima scansione. Il tempo è fornito in millisecondi.

```
const void* context,
lr1110_gnss_timings_t* timings);

@param [in] context Chip implementation context
@param [out] tempi dell'ultima scansione
@returns Operation status
```

• lr1110_status_t lr1110_gnss_get_timings(

I dati delle effemeridi sono delle informazioni sullo stato della intera costellazione di satelliti e sulla orbita approssimativa di ciascun satellite. C'è un almanacco specifico per ogni costellazione. Questi dati sono validi fino a 90 giorni. L'uso delle effemeridi permette di ottimizzare significativamente la durata della scansione: in questo modo l'LR1110 cerca solo i satelliti visibili, data la posizione e il tempo dell'utente, e questo permette di ridurre l'energia richiesta per una scansione. L'aggiornamento delle effemeridi è disponibile nel Device Managemente Center (DMC) server. Le effemeridi sono conservate nella memoria flash e sono quindi memorizzate anche dopo un power off.

Almanc Full Update Comando per aggiornare tutte le effemeridi. Poichè il payload dell'aggiornamento occupa 20bytes(header)+128(numero di satelliti)*20bytes=2580 bytes e il numero massimo di bytes che si possono mandare all'host sono 1020,occorrerà effettuare l'aggiornamento con transizioni SPI multiple. Le soluzioni implementate sono due:

```
    lr1110_status_t lr1110_gnss_one_satellite_almanac_update(
        const void* context,
        const lr1110_gnss_almanac_single_satellite_update_bytestram_t
        bytestream);
```

Funzione usata per aggiornare le effemeridi di un satellite. Tutte le effemeridi dei satelliti devono essere aggiornate in fila quindi questa funzione dovrà essere invocata 128 volte di fila senza altre chiamate tra di esse. In questo modo si realizzano 129 transazioni SPI ciascuna di 20 bytes.

@param [in] context Chip implementation context

@param [in] almanac_bytestream buffer per aggiornare le effemeridi di un satellite dell'LR1110. Deve essere lungo almeno 20 bytes.

```
    lr1110_status_t lr1110_gnss_almanac_full_update(
        const void* context,
        const lr1110_gnss_almanac_full_update_bytestream_t almanac_bytestream );
```

@param [in] context Chip implementation context

@param [in] almanac_bytestream buffer per aggiornare tutte le effemeridi dei satelliti dell'LR1110. Deve essere lungo almeno 20 bytes.

Returns Operation status.

Aggiornamento di tutte le effemeridi per tutti i satelliti. Funzione simile alla precedente, ma permette di ridurre il numero di accessi SPI per quelle applicazioni che hanno a disposizione molta memoria per almanac_bytestream. Con questa funzione si realizzano 2 transizioni SPI di 1020 bytes ciascuna, più una da 540 bytes.

```
    lr1110_status_t lr1110_gnss_set_almanac_update(
        const void*context,
        const lr1110_gnss_constellation_mask_t constellations_to_update);
```

Attiva l'aggiornamento delle effemeridi per una data costellazione.

@param [in] context Chip implementation context

@param [in] constellations_to_update bit indicanti la costellazione da aggiornare.

@returns Operation status

```
• lr1110_status_t lr1110_gnss_read_almanac_update(
      const void* context,
      lr1110_gnss_constellation_mask_t* constellations_to_update );
  Funzione per leggere le configurazioni di aggiornamento delle effemeridi.
  @param [in] context Chip implementation context
  @param [out] constellations_to_update Bit indicanti la costellazione da ag-
  giornare.
• lr1110_status_t lr1110_gnss_read_almanac(
      const void* context,
      lr1110_gnss_almanac_full_read_bytestream_t almanac_bytestream );
  Funzione per la lettura delle effemeridi.
  @param [in] context Chip implementation context
  @param [out] almanac bytestream bytestream delle effemeridi.
  @returns Operation status
• lr1110_status_t lr1110_gnss_get_almanac_age_for_satellite(
      const void* context,
      const lr1110_gnss_satellite_id_t sv_id, uint16_t* almanac_age );
  Fornisce la data di aggiornamento delle effemeridi di un satellite.
  @param [in] context Chip implementation context
  @param [in] sv_id ID del satellite corrispondente alle effemeridi da controllare
  @param [out] almanac_age Età delle effemeridi in giorni
  @returns Operation status
• lr1110_status_t lr1110_gnss_get_almanac_crc(
      const void* context,
      uint32_t* almanac_crc );
  Ritorna un CRC delle effemeridi.
  @param [in] context Chip implementation context
  @param [out] almanac_crc Almanac CRC
  @returns Operation status
```

Infine sono presenti alcune funzioni per altre funzionalità come leggere la versione del firmware, settare e leggere l'errore dell'oscillatore, o mandare messaggi al modulo.

```
• lr1110_status_t lr1110_gnss_push_solver_msg(
      const void* context,
      const uint8_t* payload, const uint16_t payload_size );
  Manda i dati ricevuti dal solver all'LR1110.
  @param [in] context Chip implementation context
  @param [in] payload Payload ricevuto dal calcolatore della posizione
  @param [in] payload size Dimensione del payload ricevuto dal calcolatore (in
  bytes)
• lr1110_status_t lr1110_gnss_read_firmware_version(
      const void* context,
      lr1110_gnss_version_t* version );
  Funzione che ritorna la versione del firmware GNSS
  @param [in] context Chip implementation context
  @param [in] version Versione del firmware GNSS corrente
  @returns Operation status
• void lr1110_gnss_parse_context_status_buffer(
    const lr1110 gnss context status bytestream t constext status bytestream,
    lr1110_gnss_context_status_t* context_status );
  Analizza il buffer del context status.
  @param [in] context status bytestream buffer del context status da analizzare.
  Lungo almeno LR1110_GNSS_CONTEXT_STATUS_LENGTH
  @param [out] context status puntatore a struttura lr1110 gnss context status t
  da riempire con le informazioni da context status bytestream
• lr1110 status t lr1110 gnss set xtal error( const void* context,
    const float xtal_error_in_ppm );
  Funzione per impostare l'errore del cristallo.
  @param [in] context Chip implementation context
  @param [in] xtal_error, value in +/-40ppm
  @returns Operation status
• lr1110_status_t lr1110_gnss_read_xtal_error(
      const void* context,
      float* xtal_error_in_ppm );
```

Funzione per leggere l'errore del cristallo. @param [in] context Chip implementation context @param [in] xtal_error, Valore tra +/-30ppm @returns Operation status

```
    lr1110_status_t lr1110_gnss_push_dmc_msg(
        const void* context,
        uint8_t* dmc_msg, uint16_t dmc_msg_len );
```

L'host riceve un aggiornamento dalla rete o assembla esso stesso un messaggio di aggiornamento e lo manda all'LR1110.

@param [in] context Chip implementation context

@param [in] dmc_msg buffer contenente l'aggiornamento dalla rete

@param [in] dmc_msg_len lunghezza del buffer

@returns Operation status

```
    lr1110_status_t lr1110_gnss_get_context_status(
        const void* context,
        lr1110_gnss_context_status_bytestream_t context_status_buffer );
```

Ritorna il GNSS context status in un buffer.

@param [in] context Chip implementation context.

@param [out] context_status_buffer puntatore a un buffer da riempire con le informazioni del context status. Deve esse lungo almeno 7 bytes.

@returns Operation status.

Per poter ottenere delle scansioni GNSS di successo, è obbligatorio impostare un TCXO (Temperature Compensated Crystal Oscillator) che permette di ridurre il consumo di potenza per performare una scansione GNSS in ambiente outdoor. Senza la scansione fallirà e verrà ritornato un messaggio di errore. Questo può essere fatto mediante la funzione inclusa nei driver di sistema:

```
• lr1110_status_t lr1110_system_set_tcxo_mode(
    const lr1110_system_tcxo_supply_voltage_t tune,
    const uint32_t timeout)

@param [in] context Chip implementation context
@param [in] tune valore di tensione di alimentazione
@param [in] timeout tempo atteso dal firmware affinche il cristallo sia pronto.
@returns Operation status
```

Capitolo 6 LR1110 drivers

Tabella 6.6: Comando SetTcxoMode.

Byte	0	1	2	3	4	5
Data from Host	0x01	0x17	RegTcxoTune	Delay (23:16)	Delay (15:8)	Delay (7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus(7:0)

La funzione implementa il comando SetTcxoMode che può essere analizzato in Tabella 6.6.

- RegTcxoTune permette di regolare la tensione in uscita sul pin VTCXO da un valore di 1.6V a 3.3V (è importante che la tensione di alimentazione sia superiore a quella impostata di almeno 200mV per il corretto funzionamento)
- Delay rappresenta il tempo massimo di inizializzazione e stabilizzazione dell'oscillatore. Se alla fine di questo tempo non è rilevato viene triggerato un errore di inizializzazione del cristallo. Se impostato a 0 viene disabilitata la modalità con TCXO.

In Figura 6.5 è mostrato un diagramma di flusso delle funzioni da richiamare per eseguire delle scansioni assistite o autonome con successo.

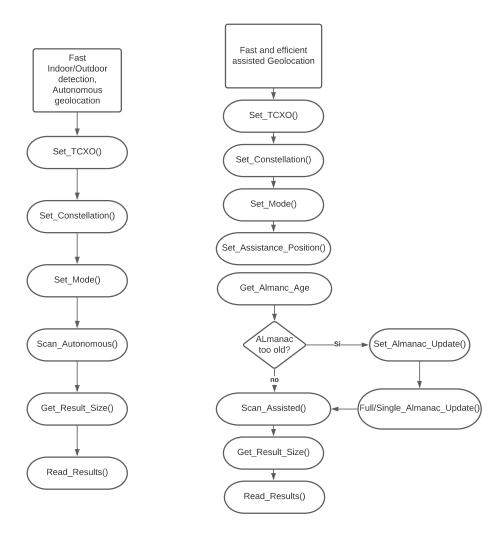


Figura 6.5: Possibile diagramma di utilizzo dei driver.

6.3 Wi-Fi Passive scanning

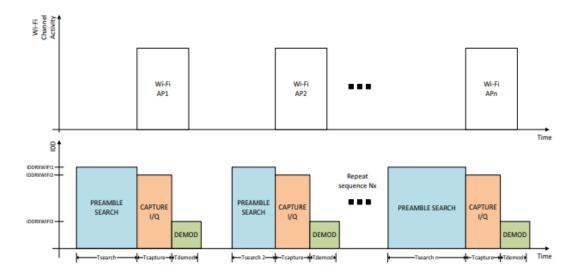


Figura 6.6: Sequenza Wi-Fi passive scanning.

Il dispositivo LR1110 permette di eseguire la sua geolocalizzazione tramite una scansione energeticamente efficiente di segnali Wi-Fi 802.11b/g/n. Il comando WifiScan() permette di rilevare i segnali Wi-Fi sul pin RFIO HF su un dato canale, per un segnale 802.11 definito (802.11b/g/n). Dal segnale sono estratti gli indirizzi MAC degli access points Wi-Fi con il corrispondente RSSI, che possono essere letti tramite il comando WifiReadResults(). Gli indirizzi MAC devono poi essere mandati ad un server di geolocalizzazione (ad esempio tramite una rete LPWAN), che calcola la posizione del device. Il numero di reti rilevate deve essere letto prima di leggere i risultati utilizzando il comando WifiGetNbResults(). La Figura 6.6 mostra la sequenza di una scansione Wi-Fi su un canale. Dopo la chiamata alla scansione, il dispositivo apre una finestra di ricezione (Preamble Search Window) su un certo canale, finchè non è rilevato un pacchetto Wi-Fi (T_{search}). Il pacchetto Wi-Fi è poi demodulato, e viene estratto l'indirizzo MAC dell'access point. Questa sequenza è ripetuta finchè non viene raggiunto NbSearchAttempt (numero di scansioni su un determinato canale) o NbMaxRes (numero totale di indirizzi MAC su tutti i canali Wi-Fi).

Statisticamente il tempo di durata della finestra di ricezione varia tra i 0us (pacchetto rilevato immediatamente), fino a un Beacon Interval dell'access point. Si può definire un parametro di Timeout per l'LR1110 che limiti il tempo di durata della finestra di ricezione, nel caso che non sia rilevata attività Wi-Fi su un canale. La cattura di un pacchetto Wi-Fi può essere fatta solo se un preambolo Wi-Fi è rilevato durante la finestra di ricezione. I risultati delle scansioni sono accumulati nella memoria fino alla scansione successiva. Possono essere salvati fino a 32 indirizzi MAC nella RAM che possono essere letti fino a che il dispositivo non entra in modalità Sleep o spento. Anche se è necessario 1 indirizzo MAC per determinare la

geolocalizzazione del dispositivo, un buon approccio è quello di utilizzare almeno 3 o più indirizzi, per assicurare una geolocalizzazione più precisa.

6.3.1 Driver Wi-Fi

Di seguito sono riportate le principali funzioni necessarie per l'implementazione dell'applicazione della scansione passiva Wi-Fi. Esse sono organizzate su tre differenti file:

- lr1110_component.c: implementazione delle funzioni
- lr1110_component.h: dichiarazioni delle funzioni
- lr1110_component_types.h: definizione dei tipi

Per le funzioni principali, sono evidenziati anche i bit che costituiscono il codice dell'operazione e degli argomenti che vengono trasmessi all'LR1110 tramite SPI.

WiFiScan Funzione che implementa l'avvio della scansione Wi-Fi:

Durante tutta la scansione Wi-Fi il pin BUSY è alto e il dispositivo non può ricevere altri comandi. Facendo partire la scansione si eliminano i risultati di quella precedente.

I risultati possono essere letti alla fine della scansione con il comando $lr1110_wifi$ $_get_nb_results$ (per ottenere il numero dei risultati da leggere) e $lr1110_wifi_read$ $_basic_complete_results$ o $lr1110_wifi_read_basic_mac_type_channel_results$ per ottenere i bytes del risultato.

- @param [in] context Chip Implementation context
- @param [in] signal_type Tipo di segnale Wi-Fi per cui eseguire la scansione
- @param [in] channels Canale Wi-Fi su cui eseguire la scansione
- @param [in] scan_mode Modalità scansione da eseguire
- @param [in] max_results Massimo numero di risultati. Raggiunto il limite la scansione si ferma automaticamente. Deve essere nel range [1:32].
- @param [in] nb_scan_per_channel Numero di scansioni all'interno di ogni canale, deve essere nel range [1:255].

Capitolo 6 LR1110 drivers

@param [in] timeout_in_ms Massima durata della finestra di ricerca espressa in ms. Nel range di [1:65535].

@param [in] abort_on_timeout Se attivo, al raggiungimento del timeout si passa alla scansione del canale Wi-Fi successivo.

@returns Operation status

Tabella 6.7: WiFiScan Command.

Byte	0	1	2	3	4	5	6	7	8	9	10
Data from Host	0x03	0x00	Wi-Fi Type	Chan Mask (15:8)	Chan Mask (7:0)	Acq Mode	Nb Max Res	Nb Scan Per Chan	Time Out (15:8)	Time Out (7:0)	Abort On Time Out
Data to Host	Stat1	Stat2	IrqStatu (31:24)	ıs IrqStatı (23:16)	ıs IrqStatı (15:8)	us IrqStatı (7:0)	ıs 0	0	0	0	0

L'operation code è costituito dai seguneti campi:

• WiFi Type definisce il tipo del segnale 802.11:

- 0x01: Wi-Fi 802.11b

-0x02: Wi-Fi 802.11g

-0x03: Wi-Fi 802.11n

- 0x04: Tutti i segnali, prima b e poi g/n sullo stesso canale

- ChanMask definisce i canali da scansionare:
 - [00ch14ch13ch12ch11ch10ch9ch8ch7ch6ch5ch4ch3ch2ch1]
 - Se il bit che corrisponde al canale è 1 allora la scansione sul canale è attiva.
- Acq Mode indica la modalità di di acquisizione:
 - -0x
01: Beacon Search Mode, usa solo i beacon Wi-Fi per estrarre gli indirizzi MAC
 - 0x02: Usa sia Beacon che pacchetti di dati per estrarre gli indirizzi MAC
 - Gli altri valori RFU
- NbMaxRes numero totale di indirizzi MAC desiderato come risultato della scansione sui vari canali (al massimo 32). Se raggiunto la scansione è fermata. Se un MAC è già presente nei risultati ed è rilevato una seconda volta con un RSSI diverso viene ignorato.
- NbScanPerChan: numero di scansioni da eseguire per canale (da 1 a 255)

- *Timeout*: timeout di 16 bit per la finestra di ricerca. Espresso in ms. Ad esempio per un periodo di beacon di 102.4ms, un timeout di 105ms assicura che la finestra copra l'intero periodo.
- AbortOnTimeout: se 1, la scansione è terminata e il dispositivo passa al canale successivo da scansionare.

Nei driver è presente anche un'altra funzione che permette di eseguire una scansione solo su Wi-Fi di tipo B e da usare se si vuole estrarre il campo Country Code:

```
    lr1110_status_t lr1110_wifi_search_country_code(
        const lr1110_wifi_channel_mask_t channels_mask,
        const uint8_t nb_max_results,
        const uint8_t nb_scan_per_channel,
        const uint16_t timeout_in_ms,
        const bool abort_on_timeout );
```

Durante la scansione i risultati sono filtrati e viene mantenuto solo un singolo indirizzo MAC.

@param [in] context Chip implementation context

@param [in] channels_mask maschera dei canali su cui eseguire la scansione

@param [in] nb_max_results Numero massimo di risultati da trattenere (il massimo è 32)

@param [in] nb_scan_per_channel numero di tentativi massimi di scansione per canale (il massimo è 255)

@param [in] timeout_in_ms massima durata di una ricerca beacon, espressa in ms. Il valore massimo è 65536ms.

@param [in] abort_on_timeout se vero, la ricerca passa al canale successivo, se non sono rilevati segnali wi-fi.

@returns Operation status

Sono presenti anche delle funzioni simili alle due precedenti, con le quali però è possibile impostare una durata temporale della scansione, e non è necessario il numero di scansioni da effettuare per ogni canale, così la ricerca di segnali Wi-Fi durerà fino a quando non ne viene trovato uno, oppure finchè non saranno trascorsi gli intervalli di tempo timeout_per_scan_ms o timeout_per_channel_ms. La massima durata della scansione è determinata dal numero di canali totali da coprire.

```
    lr1110_status_t lr1110_wifi_scan_time_limit(
        const lr1110_wifi_signal_type_scan_t signal_type,
        const lr1110_wifi_channel_mask_t channels,
        const lr1110_wifi_mode_t scan_mode,
        const uint8_t max_results,
```

```
const uint16_t timeout_per_channel_ms,
const uint16_t timeout_per_scan_ms );
```

```
    lr1110_status_t lr1110_wifi_search_country_code_time_limit(
        const lr1110_wifi_channel_mask_t channels_mask,
        const uint8_t nb_max_results,
        const uint16_t timeout_per_channel_ms,
        const uint16_t timeout_per_scan_ms);
```

@param [in] context Chip implementation context

@param [in] signal_type Tipi di segnali Wi-Fi su cui effettuare la scansione

@param [in] channels Maschera dei canali da scansionare

@param [in] scan_mode Modalità di scansione

@param [in] max_results Numero massimo di risultati da ottenere.

@param [in] timeout_per_channel_ms Tempo di scansione sul singolo canale.

@param [in] timeout_per_scan_ms Tempo massimo speso nella ricerca del preambolo in ms. Compreso nel range [0:65535]. Se 0, il comando terrà il dispositivo in ascolto fino a timeout_per_channel_ms o finchè sarà raggiunto until nb_max_results. @returns Operation status

Durante le scansioni il BUSY diventa alto e il dispositivo non può accettare altri comandi. Questa operazione potrebbe impiegare centinaia di millisecondi a seconda dei parametri impostati. Se è abilitato l'interrupt WifiScanDone, il pin IRQ va alto alla fine della scansione.

Sono implementati due tipi di risulati che permettono all'utente di ottenere o il minimo set di informazioni per la geolocalizzazione (per ottimizzare il consumo di potenza dell'applicazione) o un set di informazioni completo.

Formato dei risultati Il formato basilare dei risultati è organizzato come una serie di risultati base codificati in 9 bytes, come mostrato in Tabella 6.8

Tabella 6.8: Formato risultati base per indirizzi MAC

Byte	0	1	2	3	4	5	6	7	8
Content	Wi-Fi type	Channel Info	RSSI	MAC6	MAC5	MAC4	MAC3	MAC2	MAC1

- Wi-Fi type: codificato in 8 bit. Indicano il tipo di segnale Wi-Fi b/g/n.
- ChannelInfo: 8 bit, di cui i primi tre indicano il ChannelID ovvero i canali configurati per la scansione, e i successivi costituiscono il campo MacValidation

che indica se il MAC appartiene a un gateway, a un telefono o è indeterminato poichè è estratto da un pacchetto di dati.

- RSSI: valore di RSSI misurato codificato in 8 bit.
- MAC: indirizzo MAC codificato in 6 bytes.

Per i risultati completi, essi saranno codificati in 22 bytes, e conterranno oltre alle informazioni sopra dettagliate dei campi aggiuntivi, come un FrameCtrl, un TimeStamp, PhiOffset e BeaconPeriod.

6.3.2 WifiGetNbResults

Comando che permette di ottenere il numero di risultati ottenuti con la scansione. Questo numero è codificato in 8 bit, e deve essere ottenuto e passato poi alla funzione per la lettura dei bytes dei risultati. La funzione che implementa il comando è:

```
@param [in] context Chip implementation context
```

[@]returns Operation status

Tabella 6.9:	Comando	WifiGet	NbResults
--------------	---------	---------	-----------

Byte	0	1
Data from Host	0x03	0x05
Data to Host	Stat1	Stat2

6.3.3 WifiReadResults

WifiReadResults() permette di leggere il bytestream contenente un numero definito di risultati ottenuti dalla scansione, del formato richiesto. Prima di eseguire questo comando è necessario ottenere il numero dei risultati tramite il comando precedentemente illustrato. Per leggere i bytes occorre mandare dei dummy byte (0x00) per far scorrere il registro SPI.

- Index: contiente l'indice dei risultati da leggere, da 0 a 31.
- NbResults: contiene il numero di indirizzi MAC da leggere da 1 a 32.
- Format: formato dei risultati da leggere:

[@]param [out] nb_results numero di risultati diponibili nel LR1110

Tabella	6 10.	Comando	WifiR	eadResults
Labouta	0.10.	Comando	A A TITIT (Caurosuros

Byte 0 1		2	3	4	
Data from Host	0x03	0x06	Index	Index NbResults	
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)

- 1: risultati completi
- 4: risultati base

Il numero massimo di byte che può essere letto da una WifiReadResults() è 1020. Se la dimensione dei risultati da leggere supera questa soglia, l'operazione di lettura deve essere separata in due.

Permette di ottenere i risultati in formato completo, in una riga oppure uno dopo l'altro. Essa può essere utilizzata solo se la chiamata alla scansione è stata fatta con la modalità $LR1110_WIFI_SCAN_MODE_BEACON$ oppure $LR1110_WIFI_SCAN_MODE_BEACON_AND_PKT$.

@param [in] radio Radio abstraction.

@param [in] start_result_index Indice da cui partire per la lettura dei risultati.

@param [in] nb results Numero di risultati da leggere.

@param [out] puntatore alla struttura dei risultati da popolare. Deve poter contenere almeno nb_results elementi.

@returns Operation status.

Con questa funzione si possono ottenere solo gli indirizzi MAC, il tipo di Wi-Fi e canali. Essa può essere utilizzata solo se la chiamata alla scansione è stata fatta con la modalità $LR1110\ WIFI\ SCAN\ MODE\ BEACON$ oppure

```
LR1110 WIFI SCAN MODE BEACON AND PKT.
```

- @param [in] radio Radio abstraction
- @param [in] start_result_ Indice dal quale far partire la lettura.
- @param [in] nb_results Numero di risultati da leggere.
- @param [out] results puntatore alla struttura dei risultati da popolare. Deve poter contenere almeno nb_results elementi.

Funzione che permette di ritornare i risultati completi in forma estesa. Può essere invocata solo se la scansione è stata eseguita in modalità $LR1110_WIFI_SCAN_MODE$ $FULL\ BEACON.$

- @param [in] radio Radio abstraction
- @param [in] start_result_Indice dal quale far partire la lettura.
- @param [in] nb_results Numero di risultati da leggere.
- @param [out] results Puntatore alla struttura dei risultati da popolare. Deve poter contenere almeno nb results elementi.
- @returns Operation status

Di seguito invece le funzioni di lettura dei risultati dopo la chiamata della scansione per la lettura del Country Code:

```
    lr1110_status_t lr1110_wifi_get_nb_country_code_results(
    uint8_t* nb_country_code_results );
```

@param [in] context Chip implementation context

@param [out] nb_country_code_results Numero di risultati da leggere

```
    lr1110_status_t lr1110_wifi_read_country_code_results(
        const uint8_t start_result_index,
        const uint8_t nb_country_results,
        lr1110_wifi_country_code_t* country_code_results);
```

```
@param [in] context Chip implementation context
```

- @param [in] start result index Indice dal quale far partire la lettura
- @param [in] nb country results Numero di country code da leggere

Capitolo 6 LR1110 drivers

@param [out] country_code_results Array lr1110_wifi_country_code_t da riempire.
Deve poter contenere nb_country_results elements
@returns Operation status

WifiResetCumulTimings Tramite questo comando l'utente può resettare i tempi delle scansioni, e questa operazione può essere fatta prima di inizializzare una nuova scansione, se i tempi vogliono essere letti alla fine di questa.

lr1110_status_t lr1110_wifi_reset_cumulative_timing(
 const void* context);

Con la funzione si resettano i contatori che tengono i tempi complessivi della scansione @param [in] context Chip implementation context @returns Operation status

WifiReadCumulTimings Si può ottenere il tempo complessivo della scansione Wi-Fi, codificato in 16 bytes come mostrato in tabella 6.11. Il tempo rappresenta la somma nelle varie modalità durante la scansione e possono essere utilizzati per il calcolo del consumo energetico dell'operazione. L'operation code è mostrato in 6.12.

Tabella 6.11: Descrizione dei bytes del tempo della scansione

Byte0:34:78:1112:15MeaningTotal dura- tion in Rx mode with ADC onTotal dura- tion in pre- tion in cap- ture modeTotal dura- ration in demo- dulation mode			v	1	
tion in Rx tion in pre- tion in cap- ration mode with amble detec- ture mode in demo- ADC on tion mode dulation	Byte	0:3	4:7	8:11	12:15
	Meaning	tion in Rx mode with	tion in pre- amble detec-	tion in cap-	ration in demo- dulation

Tabella 6.12: Comando WifiCumulTimings

Byte	0	1
Data from Host	0x03	0x08
Data to Host	Stat1	Stat2

• lr1110_status_t lr1110_wifi_read_cumulative_timing(const void* context,

Ritorna il tempo impiegato nella scansione Wi-Fi @param [in] context Chip implementation context @param [out] timing puntatore alla struttura contenente i tempi @returns Operation status

6.4 Sub GHz Radio

L'LR1110 è un ricetrasmettitore RF half-duplex, capace di gestire due schemi di modulazione: LoRa e (G)FSK.

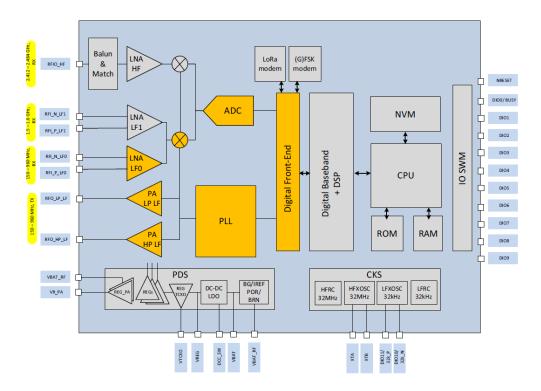


Figura 6.7: Schema della parte radio.

Il sistema radio è mostrato in Figura 6.7. Esso è composto da un PLL, due path radio (High Power e Low Power) per trasmettitore e ricevitore, seguiti da un ADC a banda larga. Sia ADC che PLL sono collegati al digital front-end e ai modem. Il PLL permette al dispositivo di operare nel range di frequenze 150MHz - 2700MHz, ed è condiviso dalla parte radio, GNSS e Wi-Fi, non rendendo quindi possibile delle operazioni contemporanee. Il riferimento in frequenza a 32MHz può essere ottenuto o da un oscillatore o da un TCXO.

6.4.1 Driver Radio

Di seguito sono riportate le principali funzioni necessarie per l'implementazione della parte radio del dispositivo. Esse sono organizzate su tre differenti file:

• lr1110 component.c: implementazione delle funzioni

Capitolo 6 LR1110 drivers

- lr1110_component.h: dichiarazioni delle funzioni
- lr1110_component_types.h: definizione dei tipi

Per le funzioni principali, sono evidenziati anche i bit che costituiscono il codice dell'operazione e degli argomenti che vengono trasmessi all'LR1110 tramite SPI.

SetRfFrequency Funzione utilizzata per impostare la frequenza delle operazioni radio:

 lr1110_status_t lr1110_radio_set_rf_freq(const void* context, const uint32_t freq_in_hz);

@param [in] context Chip implementation

@param [in] freq_in_hz Frequenza in Hz per le operazi

@returns Operation status

Tabella 6.13: Comando SetRfFrequency.

Byte	0	1	2	3	4	5
Data from Host	0x02	0x0B	RfFreq $(31:24)$	RfFreq (23:16)	RfFreq $(15:8)$	RfFreq (7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus(7:0)

SetRx Funzione per impostare la modalità ricevitore (RX). Se non è ricevuto nessun pacchetto prima dello scadere di un timeout impostato, il dispositivo torna in modalità Standby.

 lr1110_status_t lr1110_radio_set_rx(const void* context, const uint32_t timeout_in_ms);

@param [in] context Chip implementation context

@param [in] timeout_in_ms Timeout per le operazioni radio

@returns Operation status

 @param [in] context Chip implementation context

@param [in] timeout_in_rtc_step Configurazione del timeout (0x000000 per RX classica o 0xFFFFFF per RX continuo)

@returns Operation status

Tabella 6.14: Comando SetRx.

Byte	0	1	2	3	4
Data from Host	0x02	0x0B	RxTimeout(23:16)	RxTimeout(15:8)	RxTimeout(7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)

- RxTimeout è espresso in periodi dell'oscillatore RTC a 32.768KHz. Il tempo massimo è 512s.
 - 0x000000 imposta il dispositivo come ricevitore fino a quando non si riceve un pacchetto, dopo la ricezione il dispositivo torna in Standby.
 - 0xFFFFF imposta il dispositivo in ricezione finchè non si manda un comando per cambiare modalità. Si possono ricevere diversi pacchetti.

Se il timer è attivo, la modalità terminerà alla fine del timeout impostato, oppure finchè non sarà rilevato un preambolo e parola di sinronizzazione ((G)FSK) o un header (LoRa), in dipendenza allo StopOnPreamble. Il pin BUSY va basso dopo che si è settata la modalità RX.

SetTx Funzione per impostare la modalita trasmettitore (TX):

 lr1110_status_t lr1110_radio_set_tx(const void* context, const uint32_t timeout_in_ms);

@param [in] context Chip implementation context

@param [in] timeout_in_ms Configurazione del timeout

@returns Operation status

lr1110_status_t lr1110_radio_set_tx_with_timeout_in_rtc_step(
 const void* context, const uint32_t timeout_in_rtc_step);

@param [in] context Chip implementation context

@param [in] timeout_in_rtc_step The timeout configuration for TX operation (0
oppure 0xFFFF)

@returns Operation status

Capitolo 6 LR1110 drivers

Queste funzioni, triggerano la trasmissione di pacchetti RF e fanno partire l'RTC, con il TxTimeout impostato. Se il tempo finisce prima della trasmissione sarà lanciato un TIMEOUT IRQ, e la trasmissione sarà bloccata, viceversa dopo la trasmissione di un pacchetto sarà generato un TX_DONE.

Tabella 6.15: Comando SetTx.

Byte	0	1	2	3	4
Data from Host	0x02	0x0A	TxTimeout(23:16)	TxTimeout(15:8)	TxTimeout(7:0)
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)

• TxTimeout è espresso in funzione del periodo dell'RTC. Il valore massimo è 512s, mentre il valore 0x000000 disabilita il timeout.

AutoTxRx Permette di gestire in modo automatico il passaggio tra modalità Tx e Rx.

 lr1110_status_t lr1110_radio_auto_tx_rx(const void* context, const uint32_t delay, const lr1110_radio_intermediary_mode_t intermediary_mode, const uint32_t timeout);

@param [in] context Chip implementation context

@param [in] delay Time to spend in Intermediary mode

@param [in] intermediary_mode The mode the LR1110 enters after first mode completion during delay time

@param [in] timeout The timeout duration of the automatic RX or TX @returns Operation status

Se si usa il comando SetTx, dopo questa funzione, il dispositivo entrerà in modalità TX, poi attenderà il delay corrispondente alla IntermediaryMode per poi passare alla modalità RX. Analogamente per la transizione inversa. Il comando AutoTxRx() esegue automaticamente la transizione alla modalità RX dopo una tramissione o a quella TX dopo la ricezione di un pacchetto.

Tabella 6.16: Comando AutoTxRx.

Byte	0	1	2	3	4	5	6	7	8
Data from Host	0x02	0x0C	Delay (23:16)	Delay (15:8)	Delay (7:0)	Intermediary Mode	Timeout (23:16)	Timeout (15:8)	Timeout (7:0)
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0	0

I bytes *Delay* definiscono i tempi di transizione tra TX ed RX, il valore massimo è 512s.

- 0x000000 opera una transizione diretta da TX a RX e viceversa, senza passare per la IntermediaryMode
- 0xFFFFFF disabilita la funzione.

Il byte Intermediary Mode definisce la modalità del dispositivo nelle transizioni:

- 0x00: Sleep Mode
- 0x01: Standby RC mode
- 0x02: Standby Xosc mode
- 0x03: FS mode

I bytes di Timeout definiscono il tempo di rimanenza nella seconda modalità dopo la transizione automatica, il valore massimo è 512s mentre con il valore 0x000000 si disabilita.

SetRxTxFallbackMode Questa funzione imposta la modalità in cui il dispositivo deve entrare a seguito di una trasmissione o ricezione di un pacchetto:

```
    lr1110_status_t lr1110_radio_set_rx_tx_fallback_mode(
        const void* context,
        const lr1110 radio_fallback_modes_t fallback_mode);
```

@param [in] context Chip implementation context

@param [in] fallback_mode Modalità del dispositivo dopo l'uscita da quella TX o RX.

@returns Operation status

Tabella 6.17: Comando SetRxTxFallbackMode.

Byte	0	1	2
Data from Host	0x02	0x013	FallbackMode
Data to Host	Stat1	Stat2	IrqStatus(31:24)

Valori per il byte FallbackMode:

• 0x00: Sleep Mode

• 0x01: Standby RC mode

Capitolo 6 LR1110 drivers

- 0x02: Standby Xosc mode
- 0x03: FS mode

Se si sta usando la modalità RxDutyCycle o AutoRxTx questa modalità non è utilizzata.

SetRxDutyCycle Funzione che apre periodicamente una finestra di ricezione, e manda in sleep mode il dispositivo tra di esse:

```
    lr1110_status_t lr1110_radio_set_rx_duty_cycle( const void* context,
        const uint32_t rx_period_in_ms,
        const uint32_t sleep_period_in_ms,
        const lr1110_radio_rx_duty_cycle_mode_t mode );
```

@param [in] context Chip implementation context

@param [in] rx_period_in_ms Durata modalità RX

@param [in] sleep_period_in_ms Durata modalità sleep

@param [in] Modalità operativa durante le operazioni RX

@returns Operation status

Byte 0 1 2 3 4 5 6 7 8 0x14Data from 0x02RxRxRxSleep Sleep Sleep Mode Host Period Period Period Period Period Period (23:16)(15:8)(7:0)(23:16)(15:8)(7:0) ${\bf IrqStatus}$ 0 Data Stat1 Stat2 IrqStatus IrqStatus IrqStatus0 0 to Host (23:16)(15:8)(7:0)(31:24)

Tabella 6.18: Comando SetRxDutyCycle.

- RxPeriod definisce la durata della finestra di ricezione, il cui valore massimo corrisponde a 512s.
- SleepPeriod definisce la durata della modalità sleep tra due finestre di ricezione, e la durata massima è sempre di 512s.
- Mode seleziona la modalità durante le finestre di ricezione:
 - Mode=0 corrisponde alla modalità Rx.
 - Mode=1 corrisponde alla modalita CAD (Channel Activity Detection).

Quando viene lanciato questo comando, il dispositivo entra in un loop caratterizzato dalle seguenti operazioni (Figura 6.8): entra in modalità RX per un periodo definito dal byte RxPeriod e aspetta pacchetti in arrivo; quando rileva un preambolo il

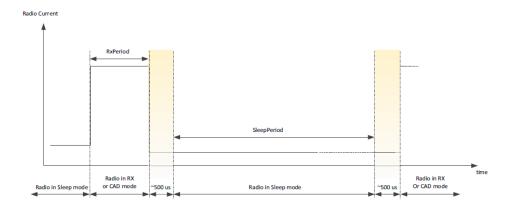


Figura 6.8: Rx Duty cycle.

timeout viene interrotto e reinizializzato al valore 2*RxPeriod + SleepPeriod (Figura 6.9). Se nessun pacchetto è ricevuto durante la finestra RX, il dispositivo va in sleep. Alla fine del periodo di sleep il processo ricomincia. Il loop termina quando un pacchetto arriva durante la finestra di ricezione, e viene generato un interrupt RX_DONE, oppure l'host manda un comando di SetStandby durante la finestra RX o con un fronte di discesa sul pin NSS durante la sleep mode.

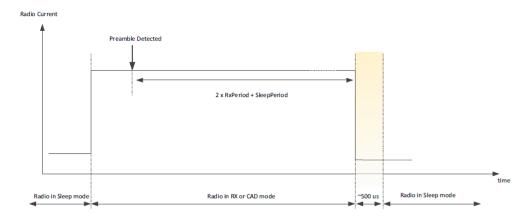


Figura 6.9: Rx duty cycle con rilevamento di preambolo.

StopTimeoutOnPreamble Funzione che definisce se il timeout della modalità RX deve essere fermato alla ricezione di un header/syncword.

```
    lr1110_status_t lr1110_radio_stop_timeout_on_preamble(
        const void* context,
        const bool stop_timeout_on_preamble);
```

@param [in] context Chip implementation context

@param [in] stop_timeout_on_preamble Attiva o disattiva la funzionalità di stop

Capitolo 6 LR1110 drivers

del timeout
@returns Operation status

Tabella 6.19: Comando StopTimeoutOnPreamble.

Byte	0	1	2
Data from Host	0x02	0x17	StopOnPreamble
Data to Host	Stat1	Stat2	IrqStatus(31:24)

Valori di StopOnPreamble:

- 0x00: Stop alla ricezione di una Synchword/Header.
- 0x01: Stop alla ricezione di un preambolo.

GetRssilnst La funzione ritorna il valore istantaneo dell'RSSI. Deve essere chiamata durante la ricezione di un pacchetto, altrimenti ritorna un valore che corrisponde all'RF noise.

@param [in] context Chip implementation context

@param [out] rssi_in_dbm RSSI istantaneo.

@returns Operation status

GetStats Funzioni che permettono di ottenere le statistiche sui pacchetti ricevuti:

```
    lr1110_status_t lr1110_radio_get_gfsk_stats(
    const void* context, lr1110_radio_stats_gfsk_t* stats);
```

Funzione per le statistiche dei pacchetti GFSK.

@param [in] context Chip implementation context

@param [out] stats struttura con le statistiche sui pacchetti ricevuti

@returns Operation status

```
    lr1110_status_t lr1110_radio_get_lora_stats(
    const void* context, lr1110_radio_stats_lora_t* stats);
```

ResetStats Funzione che permette di resettare le statistiche sui pacchetti ricevuti.

• lr1110_status_t lr1110_radio_reset_stats(const void* context);

```
@param [in] context Chip implementation context@param [out] stats struttura da resettare@returns Operation status
```

GetRxBufferStatus Comando che permette di ottenere la lunghezza dell'ultimo pacchetto ricevuto, e dell'offfset nel buffer (posizione dalla quale cominciare a leggere il pacchetto).

SetRxBoosted Mette il dispositivo in una modalità che permette di aumentare la sensibilità di 2dB con un consumo di corrente più alto di 2mA rispetto alla modalità RX classica.

lr1110_status_t lr1110_radio_cfg_rx_boosted(
 const void* context, const bool enable_boost_mode);

@param [in] context Chip implementation context

@param [in] enable_boost_mode attivazione della modalità boost

@returns Operation status

Valore di RxBoosted:

Tabella 6.20: Comando SetRxBoosted.

Byte	0	1	2
Data from Host	0x02	0x27	RxBoosted
Data to Host	Stat1	Stat2	IrqStatus(31:24)

- RxBoosted=0: modalità boost attivata.
- RxBoosted=1: RX Boosted disattivato.

6.4.2 Modem Configuration

L'LR1110 contiene due diversi modem caratterizzati da due modulazioni differenti la (G)FSK e LoRa. Inizialmente si deve scegliere il tipo di modulazione con il comando SetPacketType(), successivamente con SetModulationParam() è possibile impostare i parametri della modulazione e con SetPacketParam() si definiscono i parametri dei pacchetti. Successivamente si deve configurare l'amplificatore con il comando SetPaConfig() ed infine i suoi parametri con il comando SetTxParams(). In Figura 6.10 è mostrato il diagramma di flusso delle configurazioni da eseguire. In particolare si descriverà il modem LoRa, che è quello che verrà utilizzato in questo progetto.

SetPacketType Funzione che definisce il modem da utilizzare:

 lr1110_status_t lr1110_radio_set_pkt_type(const void* context, const lr1110_radio_pkt_type_t pkt_type);

@param [in] context Chip implementation context
@param [in] pkt_type tipo di pacchetto impostato
@returns Operation status

Valori per PacketType:

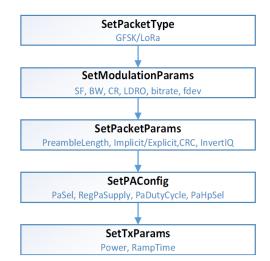


Figura 6.10: Ordine delle istruzioni da eseguire per configurare il modem.

Tabella 6.21: Comando SetPacketType.

Byte	0	1	2
Data from Host	0x02	0x0E	PacketType
Data to Host	Stat1	Stat2	IrqStatus(31:24)

• 0x00: None

• 0x01: (G)FSK

• 0x02: LoRa

Questo è il primo comando che deve essere chiamato prima di andare in modalità TX/RX e di definire parametri di modulazione e pacchetti.

GetPacketType Legge il tipo di modem attualmente utilizzato.

```
    lr1110_status_t lr1110_radio_get_pkt_type(
    const void* context, lr1110_radio_pkt_type_t* pkt_type );
```

@param [in] context Chip implementation context

@param [out] pkt_type Pacchetto attualmente configurato

@returns Operation status

6.4.3 LoRa Modem

Il modem LoRa utilizza una modulazione a spettro espanso, che permette di incrementare il link budget e di migliorare l'immunità alle interferenze se confrontato

con le modulazioni classiche. La modulazione LoRa consiste nel rappresentare ciascun bit del payload in chips di informazione. Il rate al quale l'informazione è mandata è detto symbol rate (Rs). Il rapporto tra il symbol rate e il chip rate è il fattore di spreading (SF) e rappresenta il numero di simboli mandati per ogni bit di informazione. Il fattore di spreading deve essere noto a priori al trasmettitore e ricevitore, poichè differenti SF sono ortogonali tra di loro. La larghezza di banda (BWL) è configurabile attorno ad una frequenza centrale fRF, solitamente ci si riferisce ad una double side band (DSB) come mostrato in figura.

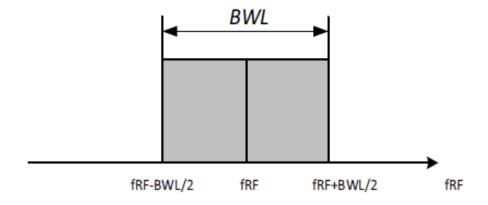


Figura 6.11: Banda di un segnale LoRa.

Una banda maggiore permette di avere un data rate effettivo più grande, riducendo il tempo di trasmissione a scapito di una sensibilità ridotta. Per avere una maggiore robustezza del collegamento, si utilizza una codifica ciclica, che permette di rilevare e correggere errori. Il symbol rate è definito come $Rs = \frac{BWL}{2^{SF}}$ dove BWL è la larghezza di banda scelta e SF lo spreading factor. Il modem LoRa impiega due tipi di messaggi: esplicito e implicito. Quello esplicito include un breve header che contiene informazioni sul numero di bytes, ed è incluso un CRC (Figura 6.12).

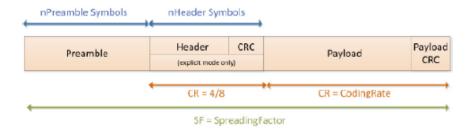


Figura 6.12: Formato pacchetto LoRa.

Il preambolo è usato per sincronizzare il ricevitore con il messaggio in arrivo, la sua lunghezza può variare da 1 a 65535 simboli. Questo permette di poter utilizzare

in principio una sequenza di lunghezza arbitraria. Per ottimizzare la ricezione è consigliato di utilizzare un preambolo lungo almeno 12 con uno SF di 5 o 6, oppure di lunghezza 8 per altri SF.

L'header fornisce informazioni sul payload del messaggio:

- la lunghezza del payload in bytes
- il rate del codice per la correzione d'errori
- la presenza di un CRC di 16 bit opzionale del payload

L'header è trasmesso con un rate massimo per la correzione d'errori (4/8). Esso ha anche il suo CRC per permettere al ricevitore di scartare header non validi. In alcuni scenari il rate del codice e la presenza del CRC sono fisse o note a priori, e può essere in alcuni casi vantaggioso ridurre il tempo di trasmissione utilizzando un header implicito. In questo caso l'header è rimosso dal pacchetto. Se si utilizza la modalità implicita la lunghezza del payload, rate del codice e presenza del CRC devono essere configurati manualmente ad entrambi i lati del collegamento radio.

Il pacchetto è di lunghezza variabile, e contiente i dati veri e propri codificati con il rate appropriato. Può essere seguito in coda da un eventuale CRC.

Channel Activity Detection La modalità *Channel Activity Detection* è usata solo per pacchetti LoRa. Il dispositivo cerca la presenza di preamboli di segnali LoRa. Dopo che la ricerca è completata, il dispositivo torna in standby. La lunghezza della ricerca è configurata con il comando SetCadParams().

Per configurare il modem LoRa, sono presenti sempre nei driver radio, delle funzioni che implementano i comandi necessari, e sono di seguito riportate.

SetModulationParam Funzione per configurare i paramentri del modem LoRa:

```
    lr1110_status_t lr1110_radio_set_lora_mod_params(
        const void* context,
        const lr1110_radio_mod_params_lora_t* mod_params);
```

@param [in] context Chip implementation context

@param [in] mod_params Struttura contentente i parametri della modulazione @returns Operation status

- SF definisce lo spreading factor utilizzato
- BWL definisce la larghezza di banda

Tabella 6.22: Comando SetModulationParam.

Byte	0	1	2	3	4	5
Data from Host	0x02	0x0F	SF	BWL	CR	LowDataRateOptimize
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus(7:0)

- CR configura il rate del codice
- Low Data Rate Optimize riduce il numero di bit per simbolo se attivo

SetPacketParam Questa funzione serve per configurare i parametri dei pacchetti RF.

 lr1110_status_t lr1110_radio_set_lora_pkt_params(const void context, const lr1110_radio_pkt_params_lora_t* pkt_params);

@param [in] context Chip implementation context

@param [in] pkt_params Struttura con i parametri dei pacchetti radio

@returns Operation status

Tabella 6.23: Comando SetPacketParam.

Byte	0	1	2	3	4	5	6	7
Data from Host	0x02	0x10	PbLength Tx(15:8)	PbLength Tx(7:0)	Header Type	Payload Len	CRC	InvertIQ
Data to Host	Stat1	Stat2	IrqStatus (31:24)	IrqStatus (23:16)	IrqStatus (15:8)	IrqStatus (7:0)	0	0

- *PbLenghtTX* definisce la lunghezza del preambolo del messaggio, codificato su 2 bytes, da 1 a 65535.
- HeaderType definisce se l'header è implicito o esplicito
- PayloadLen definisce la misura del payload (in bytes) da trasmettere, o la lunghezza massima che il ricevitore può accettare. Con header esplicito, se questo byte vale 0 allora è consentita la ricezione di qualsiasi lunghezza da 0 a 255 bytes. Se invece vale N allora sono accettati payloads con lunghezza compresa tra 0 ed N. Con header implicito invece il byte indica la lunghezza esatta del messaggio da trasmettere.
- CRC definisce se è attivo oppure no il CRC
- InvertIQ definisce se i segnali I e Q sono invertiti

SetCad Attiva la modalità Channel activity detection

• lr1110_status_t lr1110_radio_set_cad(const void* context);

@param [in] context Chip implementation context@returns Operation status

SetCadParam Per configurare i parametri della modalità precedente occorre chiamare la seguente funzione:

 lr1110_status_t lr1110_radio_set_cad_params(const void* context, const lr1110_radio_cad_params_t* cad_params);

@param [in] context Chip implementation context
 @param [in] cad_params Struttura che contiene la configurazione della CAD
 @returns Operation status

Byte 0 1 3 5 6 Data from 0x0DSymbol CadExit Timeout Timeout Timeout 0x02 ${\it DetPeak}$ DetMin Host Num Mode (23:16)(15:8)(7:0)0 Data Stat1Stat2 IrqStatus IrqStatus IrqStatus IrqStatus 0 0 to Host (31:24)(23:16)(15:8)(7:0)

Tabella 6.24: Comando SetCadParam.

- SymbolNum definisce il numero di simboli utilizzati in questa modalità
- DetPeak e DetMin definiscono la sensibilità del modem LoRa.
- CadExitMode definisce cosa fare una volta che si esce dalla modalità CAD (passaggio alla modalità RX, TX, oppure standby)
- Timeout utilizzato solo se CadExitMode è impostato a CAD_RX e in tal caso definisce quanto tempo trascorrere in nella modalità RX

LoRaSynchTimeout Comando che configura il modem per lanciare un timeout dopo un certo numero di simboli se non viene rilevato nessun pacchetto

```
    lr1110_status_t lr1110_radio_set_lora_sync_timeout(
        const void* context,
        const uint8_t nb_symbol );
```

```
@param [in] context Chip implementation context
@param [in] nb_symbol numero di simboli da configurare
@returns Operation status
```

SetLoRaPublicNetwork Funzione utilizzata per impostare una rete LoRa pubblica o privata.

```
    lr1110_status_t lr1110_radio_set_lora_public_network(
        const void* context,
        const lr1110_radio_lora_network_type_t network_type );
    @param [in] context Chip implementation context
    @param [in] network_type tipo di rete da configurare
    @returns Operation status
    lr1110_status_t lr1110_radio_set_lora_sync_timeout( const void* context,
        const uint8_t nb_symbol );
    @param [in] context Chip implementation context
    @param [in] sync_word tipo di rete da configurare
    @returns Operation status
```

GetPacketStatus Funzione necessaria a ritornare lo stato dell'ultimo pacchetto ricevuto.

```
    lr1110_status_t lr1110_radio_get_lora_stats( const void* context,
lr1110_radio_stats_lora_t* stats );
```

```
@param [in] context Chip implementation context
@param[out] pkt_status Stato dell'ultimo pacchetto
@returns Operation status
```

Grazie a questa è possibile ottenere i valori dell'RSSI e SNR dell'ultimo messaggio ricevuto.

Come mostrato precedentemente in Figura 6.10, dopo aver impostato il tipo di modem ed i vari parametri di pacchetto e modulazione, è necessario mandare due comandi al dispositivo per configurare gli amplificatori necessari alle operazioni radio. In particolare due sono quelli utilizzati come si può vedere dallo schema in Figura 6.7, uno high power, ottimizzato per operazioni fino a +22dBm, e uno low

power ottimizzato per operazioni +14 dB. I parametri che impattano sulla potenza in uscita sono la potenza in uscita programmata, il duty cycle e la tensione di alimentazione (solo per l'high power poichè il low power è regolato internamente), e ciascuno va opportunamente scelto tenendo conto degli altri. Questi sono configurati con due comandi: SetPaConfig() e SetTxParams() da essere invocati in questo ordine. Il primo serve per selezionare quale dei due amplificatori utilizzare, selezionare l'alimentazione per l'amplificatore e selezionare il duty cycle. Con il secondo invece, si controlla la tensione di alimentazione e la potenza in uscita, e si sceglie il tempo di rampa.

SetPaConfig

 lr1110_status_t lr1110_radio_set_pa_cfg(const void* context, const lr1110_radio_pa_cfg_t* pa_cfg);

@param [in] context Chip implementation context

@param [in] pa_cfg Struttura con le configurazioni dell'amplificatore

@returns Operation status

Tabella 6.25: Comando SetPaConfig.

Byte	0	1	2	3	4	5
Data from Host	0x02	0x15	PaSel	RegPaSupply	PaDutyCycle	PaHPSel
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	IrqStatus(15:8)	IrqStatus(7:0)

- PaSel=0x00 seleziona l'amplificatore low power, PaSel=0x01 seleziona quello high power.
- RegPaSupply=0x00 alimenta attraverso il regolatore interno, RegPaSupply=0x01
 dal pin VBAT. Si deve utilizzare sempre questa seconda opzione quando
 TxPower>14.
- PaDutyCycle controlla il duty cycle: i valori ammessi per il low power sono 20% < DutyCycle < 48% (che corrispondono a 0 < PaDutyCycle < 7) e 20% < DutyCycle < 36% per l'high power (0 < PaDutyCycle < 6). Il DutyCycle può essere ottenuto dalla relazione DutyCycle = 20% + 4% * PaDutyCycle
- PaHPSel controlla un parametro dell'amplificatore high power, per raggiungere i 22dB, di potenza in uscita deve essere impostato a 7.

SetTxParams

```
    lr1110_status_t lr1110_radio_set_tx_params( const void* context,
const int8_t pwr_in_dbm,
const lr1110_radio_ramp_time_t ramp_time );
```

@param [in] context Chip implementation context

@param [in] pwr_in_dbm potenza in uscita in dBm

@param [in] ramp time Tempo di rampa

Tabella 6.26: Comando SetTxParams.

Byte	Byte 0 1		2	3	
Data from Host	0x02	0x11	TxPower	RampTime	
Data to Host	Stat1	Stat2	IrqStatus(31:24)	IrqStatus(23:16)	

- TxPower definisce la potenza in uscita in dBm, in un range che va da -17dBm (0xEF) a +14dBm (0x0E) con passo 1 dB per il low power, mentre in un range da -9dBm (0xF7) a +22dBm (0x16) per quello high power.
- RampTime definisce il tempo di rampa, può essere impostato da 10us a 3400us. Un valore di 40us è il miglior trade-off tra una stabilizzazione veloce della potenza e minime componenti armoniche indesiderate.

WriteBuffer8 e ReadBuffer8 Per scrivere e leggere nei registri TX ed RX è possibile utilizzare le funzioni:

 lr1110_status_t lr1110_regmem_write_buffer8(const void* context, const uint8_t* buffer, const uint8_t length);

Scrive le informazioni da inviare nel buffer RX.

@param [in] context Chip implementation context

@param [in] data Dati da scrivere nel buffer RX che si vogliono inviare

@param [in] length numero di bytes che si vogliono inviare

```
    lr1110_status_t lr1110_regmem_read_buffer8( const void* context,
        uint8_t* buffer,
        const uint8_t offset,
        const uint8_t length );
```

Legge i bytes ricevuti dall'RX buffer. Da chiamare dopo il comando GetRxBuffer-Status, che ritorna il numero di bytes da leggere e la posizione da cui partire.

@param [in] context Chip implementation context

@param [in] Posizione dalla quale cominciare a leggere

@param [in] length Numero di bytes da leggere

@param [in] data Puntatore ad un array nel quale si trasferiranno i dati letti.

@returns Operation status

Capitolo 7

Test LR1110

Per il test delle funzionalità del dispositivo LR1110, è stata utilizzata una delle due boards fornite nell'evaluation kit, in particolare la PCBE516V02B, che ha un LNA integrato per poter operare con una antenna GPS passiva, come descritto nell'Application Note [13]. Il PCB si collega ad una board Nucleo STM32L476RG, che lo alimenta e comunica con l'LR1110 tramite SPI per impartire al chip i comandi. Nello sviluppo dell'applicazione sono stati utilizzati i driver sopra descritti, reperibili nella libreria fornita da Semtech [14], e dei quali sono stati modificati i files lr1110_hal.c/h, per implementare la comunicazione SPI, con la Nucleo STM32L476RG, utilizzando le librerie HAL fornite da STMicroelectronics [15]. Per l'elaborazione dei dati acquisiti dal dispositivo, è stato realizzato uno script Python che riceve tramite UART i risultati e manda le richieste di geolocalizzazione al server, stampando a schermo i risultati delle scansioni. Inoltre se vengono ottenute delle cordinate corrispondenti alla posizione del dispositivo, viene generato un file KML, che può essere visualizzato ad esempio su Google Earth.

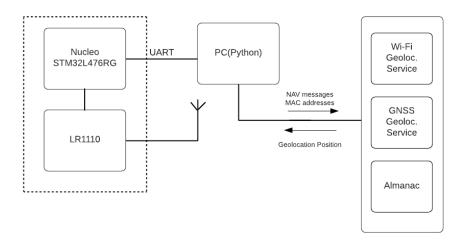


Figura 7.1: Schema a blocchi dell'applicazione realizzata.

7.1 Codice C

Il software è stato sviluppato con STM32CubeIDE, ambiente di sviluppo fornito da STMicroelectronics. Questo fornisce un tool grafico, STM32CubeMX che permette di impostare tramite un'interfaccia grafica le periferiche del microcontrollore come si desidera, e di generare in automatico il codice per l'inizializzazione. Il programma è stato organizzato gerarchicamente su più files: due file gnss_scan.c/h, che contengono delle funzioni implementate con i driver gnss dell'LR1110 che consentono l'aggiornamento delle effemeridi, l'inizializzazione del dispositivo per le scansioni e il lancio di scansione autonoma e assistita; inoltre è presente una funzione per la trasmissione dei risultati allo script python. Nei due file wifi_scan.c/h è stata la fatta la stessa cosa per la parte Wi-Fi: essi contengono l'implementazione delle scansioni wifi e la trasmissione dei risultati. Infine nei file application.c/h è presente il loop del programma che viene inserito nel main, e richiama le funzioni dei file gerarchicamente inferiori.

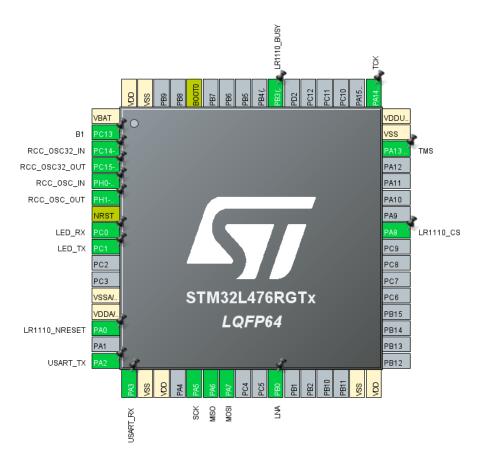


Figura 7.2: Impostazione dei pin del microcontrollore con STMCubeMx.

Per simulare quello che poi sarà il funzionamento del dispositivo, nel loop principale la board lancia le scansioni gnss e Wi-Fi, trasmette i risultati e manda in sleep l'LR1110; successivamente entra in modalità standby anche il micro della scheda, che

è la modalità low power con minor consumo di corrente, e viene risvegliato da un interrupt generato dall'RTC timer. I pins utilizzati del microcontrollore sono i PA5, PA6 e PA7 per le linee dati e clock dell'SPI, per la comunicazione con l'LR1110, il chip select è controllato via software ed è collegato al PA8, sul PB3 è collegato il BUSY per la lettura dello stato del dispositivo. Sui PC0 e PC1 sono collegati i due led Tx e Rx della board, per eventuali segnali. Infine sono necessari il PA13 e PA14 per il debug in modalità SWD e i PA2 e PA3 per la comunicazione UART con la porta COM, con un Baud Rate settato a 9600Bit/s.

7.1.1 gnss_scan.c/h

Di seguito è mostrata la routine per l'aggiornamento delle effemeridi: come precedentemente descritto il file è grande 129 bytes, di cui il primo è un header e a seguire le effemeridi vere e proprie. La funzione trasmette tramite UART il comando allo script python per ottenere il file aggiornato e successivamente riceve i 129 bytes. A questo punto vengono settate le costellazioni satellitari da aggiornare, e con un ciclo for viene richiamata 129 volte la funzione $lr1110_gnss_one_satellite_almanac_update()$, che tramite SPI manda i 129 bytes dal microcontrollore all'LR1110.

```
uint32_t almanac_update(){
lr1110_gnss_almanac_single_satellite_update_bytestram_t bytestream[129]={0};
char str[4] = \{0\};
lr1110_gnss_constellation_mask_t constellation_mask=
                  LR1110_GNSS_GPS_MASK+LR1110_GNSS_BEIDOU_MASK;
lr1110_status_t gnss_status = 0;
uint32_t almanac_crc = 0;
lr1110_gnss_satellite_id_t sv_id=0x01;
strcpy(str, "almn");
HAL_UART_Transmit(&huart2, str, 4, 100);
HAL_Delay(2000);
strcpy(str, "copy");
HAL_UART_Transmit(&huart2, str, 4, 100);
for (int i = 0; i<129; i++)
{
HAL_UART_Receive(&huart2, &bytestream[i], 20, 100);
}
gnss_status = lr1110_gnss_set_almanac_update(constellation_mask);
//gnss_status = lr1110_gnss_almanac_full_update(bytestream);
for (int i = 0; i < 129; i + +)
{
```

```
gnss_status = lr1110_gnss_one_satellite_almanac_update( bytestream );
}
lr1110_gnss_get_almanac_crc( &almanac_crc );
}
```

Nella gnss_scan_init() viene inizalizzato il chip per la scansione gnss: viene attivato il TCXO, senza il quale le scansioni non possono essere eseguite, settata la posizione di assistenza, la costellazione di satelliti su cui effettuare la ricerca, ed infine la modalità di scansione singola o doppia.

```
lr1110_status_t gnss_scan_init(
    lr1110_gnss_solver_assistance_position_t* assistance_position,
    lr1110_gnss_constellation_mask_t constellation_mask){
 lr1110_status_t gnss_status = 0;
 uint8 t inter capture delay second = 0;
 while (HAL_GPIO_ReadPin(LR1110_BUSY_GPIO_Port, LR1110_BUSY_Pin) == 1){}
 lr1110_system_set_tcxo_mode( LR1110_SYSTEM_TCXO_CTRL_3_OV, 500 );
 gnss_status= lr1110_gnss_set_assistance_position(assistance_position);
 //gnss status= lr1110 gnss read assistance position(&read assistance position);
 gnss_status = lr1110_gnss_set_constellations_to_use(constellation_mask);
 //leggo costellazione gps
 //gnss_status = lr1110_gnss_read_used_constellations(&read_used_constellation);
 HAL_Delay(100);
 gnss status = lr1110 gnss set scan mode(LR1110 GNSS SINGLE SCAN MODE,
                                &inter_capture_delay_second);
}
```

Nelle funzioni gnss_assisted_scan() e gnss_autonomous_scan() avvengono le scansioni vere e proprie: vengono settate le variabili per il salvataggio dei risultati, lanciata la scansione, che può essere "best effort" o "low power", e una volta finita, leggendo lo stato del BUSY, viene letta la lunghezza del buffer contenente il Nav message e successivamente tutti gli altri risultati come numero di satelliti rilevati, ID dei satelliti e relativo C/N, ed infine il tempo impiegato nella scansione. Successivamente viene invocata la funzione per la trasmissione dei dati allo script Python. Nella scansione assistita, dal controllo del secondo byte del nav message, se risulta che le effemeridi non sono aggiornate, viene richiamata la funzione di aggiornamento.

```
lr1110_status_t gnss_assisted_scan(lr1110_gnss_date_t gpstimestamp,
uint8_t gnss_input_parameters_mask, uint8_t nb_sat){
```

```
lr1110_status_t gnss_status = 0;
uint8_t nb_detected_assisted_satellites=0;
uint16_t gnss_scanassisted_result_size = 0;
uint8_t scan_type = 1; //1 for assisted scan
lr1110_gnss_timings_t timings;
gnss_status = lr1110_gnss_scan_assisted( gpstimestamp,
LR1110_GNSS_OPTION_BEST_EFFORT,
      gnss_input_parameters_mask,
   nb sat);
    while (HAL_GPIO_ReadPin(LR1110_BUSY_GPIO_Port, LR1110_BUSY_Pin) == 1){}
    if (gnss_status == LR1110_STATUS_ERROR){
     return (gnss_status);
    gnss_status = lr1110_gnss_get_result_size( &gnss_scanassisted_result_size );
    uint8_t result_assisted_buffer[gnss_scanassisted_result_size];
    gnss_status = lr1110_gnss_read_results( result_assisted_buffer,
    gnss_scanassisted_result_size );
    lr1110_gnss_get_nb_detected_satellites(&nb_detected_assisted_satellites);
    lr1110_gnss_detected_satellite_t id_snr[nb_detected_assisted_satellites];
    if (result_assisted_buffer[1] == 0x8){
     almanac_update();
        gnss status=lr1110 gnss get detected satellites(
        nb_detected_assisted_satellites,
        detected_assisted_satellite_id_snr);
        gnss_status = lr1110_gnss_get_timings (&timings);
        HAL_Delay(1000);
        transmit_gnss_scan_results(&nb_detected_assisted_satellites, &scan_type,
                 &gnss_scanassisted_result_size, timings,
                      detected_assisted_satellite_id_snr,
                                 result_assisted_buffer);
}
    lr1110_status_t gnss_autonomous_scan(lr1110_gnss_date_t gpstimestamp,
          uint8_t gnss_input_parameters_mask, uint8_t nb_sat){
lr1110_status_t gnss_status = 0;
uint8_t nb_detected_satellites=0;
```

```
uint16_t gnss_scan_result_size = 0;
uint8_t scan_type = 0; //0 for autonomous scan
lr1110_gnss_timings_t timings;
  gnss_status = lr1110_gnss_scan_autonomous(gpstimestamp,
  LR1110_GNSS_OPTION_BEST_EFFORT,
    gnss_input_parameters_mask, nb_sat);
  while (HAL_GPIO_ReadPin(LR1110_BUSY_GPIO_Port, LR1110_BUSY_Pin) == 1){}
    if (gnss_status == LR1110_STATUS_ERROR){
       return (gnss_status);
      }
    gnss_status = lr1110_gnss_get_result_size( &gnss_scan_result_size );
    uint8_t result_buffer[gnss_scan_result_size];
    HAL_Delay(100);
    gnss_status = lr1110_gnss_read_results(result_buffer, gnss_scan_result_size);
    gnss_status = lr1110_gnss_get_timings (&timings);
    gnss_status=lr1110_gnss_get_nb_detected_satellites(&nb_detected_satellites);
    lr1110 gnss_detected satellite_t id_snr[nb_detected_satellites];
    gnss_status=lr1110_gnss_get_detected_satellites(nb_detected_satellites,
     detected_satellite_id_snr);
    transmit_gnss_scan_results(&nb_detected_satellites, &scan_type,
        &gnss_scan_result_size, timings,
          detected_satellite_id_snr,
     result_buffer);
}
```

Infine sono presenti altre tre funzioni per la comunicazione con lo script Python: una già citata per la trasmissione dei risultati delle scansioni; una per lo scambio del tempo GPS, parametro in ingresso necessario per il lancio delle scansioni, ed infine una funzione che viene lanciata alla fine delle tre scansioni, che serve per dare il comando a Python di aggiornare il file KML contenente i risultati.

7.1.2 Wi-Fi_scan.c/h

Nella funzione wifi_scan(), viene lanciata la scansione wi-fi, vengono prima inizializzate le variabili necessarie, come il numero massimo di risultati e quelle per il salvataggio dei risultati; poi viene lanciata la scansione scegliendo il tipo (B,G,N) di Wi-Fi per cui si effettua la ricerca, i canali, la modalità di scansione, il numero di scansioni per canale e un timeout (variabili che saranno scelte nel file application.c). Terminata la scansione i risultati vengono salvati e trasmessi a Python.

```
lr1110_status_t wifi_scan(lr1110_wifi_channel_mask_t channels,
uint8_t nb_scan_per_channel,
```

```
lr1110_status_t wifi_status = 0;
uint8_t max_results = 0x0A;
uint8_t nbwifi_results = 0;
wifi_status = lr1110_wifi_scan(LR1110_WIFI_TYPE_SCAN_B_G_N,
    channels, LR1110_WIFI_SCAN_MODE_BEACON_AND_PKT,
                max_results, nb_scan_per_channel,
                             timeout_in_ms, 0x00);
if (wifi status == LR1110 STATUS ERROR){
     return (wifi_status);
    while (HAL GPIO ReadPin(LR1110 BUSY GPIO Port, LR1110 BUSY Pin) == 1){}
    wifi_status = lr1110_wifi_get_nb_results(&nbwifi_results);
    lr1110_wifi_basic_complete_result_t all_results[nbwifi_results];
    lr1110 wifi read basic complete results(0, nbwifi results, all results);
    transmit_wifi_scan_results(&nbwifi_results, all_results);
}
Si è realizzata anche una funzione per la ricerca del country code, che è analoga alla
precedente, solo che effettua delle scansioni solo su Wi-Fi di tipo B. La struttura del
codice è analoga alla funzione precedente.
    lr1110_status_t wifi_scan_country_code(lr1110_wifi_channel_mask_t channels,
uint8_t nb_scan_per_channel, uint16_t timeout_in_ms){
lr1110_status_t wifi_status = 0;
uint8_t max_results = 0x0A;
uint8_t nb_country_code_results = 0;
wifi_status =lr1110_wifi_search_country_code(channels, max_results,
nb_scan_per_channel, timeout_in_ms, 0x00 );
 if (wifi_status == LR1110_STATUS_ERROR){
   return (wifi_status);
    }
 while (HAL_GPIO_ReadPin(LR1110_BUSY_GPIO_Port, LR1110_BUSY_Pin) == 1){}
lr1110 wifi get nb country code results(&nb country code results );
lr1110_wifi_basic_complete_result_t results[nb_country_code_results];
```

uint16_t timeout_in_ms){

```
lr1110_wifi_read_country_code_results( 0,
nb_country_code_results, all_country_code_results );
```

Infine è presente la solita funzione per la trasmissione dei risultati a Python.

7.1.3 application.c/h

In questo file è stata creata la funzione applicationloop() che viene richiamata nel main() del programma principale.

```
void application_loop(){
//wake-up
HAL_GPIO_WritePin(LR1110_CS_GPIO_Port, LR1110_CS_Pin, GPIO_PIN_RESET);
HAL_Delay(1);
HAL_GPIO_WritePin(LR1110_CS_GPIO_Port, LR1110_CS_Pin, GPIO_PIN_SET);
while (HAL_GPIO_ReadPin(LR1110_BUSY_GPIO_Port, LR1110_BUSY_Pin) == 1){}
lr1110_gnss_constellation_mask_t constellation_mask =
LR1110_GNSS_GPS_MASK+LR1110_GNSS_BEIDOU_MASK;
lr1110_wifi_channel_mask_t channels =
LR1110 WIFI CHANNEL 1 | LR1110 WIFI CHANNEL 2 | LR1110 WIFI CHANNEL 3 |
LR1110_WIFI_CHANNEL_4|LR1110_WIFI_CHANNEL_5|LR1110_WIFI_CHANNEL_6|
LR1110_WIFI_CHANNEL_7|LR1110_WIFI_CHANNEL_8|LR1110_WIFI_CHANNEL_9|
LR1110_WIFI_CHANNEL_10|LR1110_WIFI_CHANNEL_11|LR1110_WIFI_CHANNEL_12;
lr1110_gnss_solver_assistance_position_t assistance_position;
assistance_position.latitude = 43.438;
assistance_position.longitude = 13.613;
lr1110_gnss_date_t gpstimestamp;
uint8_t gnss_input_parameters_mask=0x03;
uint8_t nb_sat = 0;
uint8_t nb_scan_per_channel = 0x06;
uint16_t timeout_in_ms = 0x0046;
lr1110_system_sleep_cfg_t is_warm_start;
gnss_scan_init(&assistance_position, constellation_mask);
gps timestamp(&gpstimestamp);
gnss_autonomous_scan(gpstimestamp, gnss_input_parameters_mask, nb_sat);
gps_timestamp(&gpstimestamp);
gnss_assisted_scan( gpstimestamp, gnss_input_parameters_mask, nb_sat);
wifi_scan( channels, nb_scan_per_channel, timeout_in_ms);
//wifi_scan_country_code( channels, nb_scan_per_channel, timeout_in_ms);
lr1110_system_set_sleep(is_warm_start, 0 );
    prepare kml results(); //send command to python to update kml file
}
```

La funzione comincia con il risveglio del dispositivo, mandando basso il pin CS, attendendo che l'LR1110 completi la procedura di calibrazione inizale, con una lettura del pin BUSY si controlla lo stato del dispositivo. Una volta pronto vengono inizializzate le variabili che saranno passate alle funzioni gerarchicamente inferiori, come costellazioni e canali da usare, posizione di assistenza, ed i vari parametri delle scansioni precedentemente nominati. In seguito viene inizializzato il dispositivo e vengono lanciate le scansioni, dopo uno scambio di GPS time con lo script python. Terminate queste operazioni il dispositivo viene posto in modalità sleep, la modalità con il minor consumo di corrente del circuito integrato (1.6uA come indicato in [1]).

7.1.4 main()

Il fucro del programma, grazie ai tools forniti da STM è già pronto, incluse le funzioni per l'inizializzazione delle periferiche del microcontrollare, impostate con il tool grafico CubeMX, e la loro chiamata.

```
/* Private variables ------*/
RTC_HandleTypeDef hrtc;

SPI_HandleTypeDef hspi1;

UART_HandleTypeDef huart2;
    /* Private function prototypes ------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_RTC_Init(void);
static void MX_NVIC_Init(void);
```

In particolare si possono notare la funzione per l'inizializzazione del clock, dei pins GPIO, l'SPI per la comunicazione con l'LR1110, l'USART per la comunicazione con lo script Python, l'RTC necessario per il risveglio del microcontrollore dalla modalità standby e l'inizializzazione degli interrupts. Di seguito invece sono riportate le funzioni per la gestione dello standby: se il micro va in standby viene settato il flag PWR_FLAG_SB che ad ogni risveglio deve essere controllato ed eventualmente azzerato. Successivamente nella procedura consigliata nel manuale della scheda [16] occorre disabilitare il timer per il risveglio, azzerare il PWR_FLAG_WU ed infine reimpostare con HAL_RTCEx_SetWakeUpTimer_IT() il timer di wake-up, in questo caso particolare sono stati scelti i parametri in modo che il dispositivo si risvegli ogni 30s. Successivamente viene lanciato l'application_loop() con le scansioni, ed infine si entra nella modalità standby.

```
if(_HAL_PWR_GET_FLAG(PWR_FLAG_SB) != RESET)
```

```
{
  /* Clear Standby flag */
  __HAL_PWR_CLEAR_FLAG(PWR_FLAG_SB);
/* The Following Wakeup sequence is highly recommended prior to
each Standby mode entry
  mainly when using more than one wakeup source this is
  to not miss any wakeup event.
  - Disable all used wakeup sources,
  - Clear all related wakeup flags,
  - Re-enable all used wakeup sources,
  - Enter the Standby mode.
*/
/* Disable all used wakeup sources*/
HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);
/* Clear all related wakeup flags */
__HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);
HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, OxOFFFF, RTC_WAKEUPCLOCK_RTCCLK_DIV16);
application_loop();
HAL_PWR_EnterSTANDBYMode();
```

7.2 Script Python

Lo script in Python è stato realizzato per avere un mezzo rapido per poter ricevere i risultati delle scansioni dalla scheda, poterli analizzare e poterli inoltrare al server, nonchè poter ricevere ed inoltrare alla scheda le effemeridi aggiornate, come illustrato nello schema della Figura 7.1. I principali pacchetti utilizzati per la scrittura del programma sono:

- *serial*, che permette di realizzare la comunicazione UART sulla porta COM selezionata.
- astropy, per poter disporre rapidamente dell'informazione di tempo GPS, senza dover effettuare delle conversioni da altri formati.
- requests e json, per inoltrare al server i dati con delle POST requests, contenenti file .json con i dati necessari.
- simplekml per la creazione dei file KML finali, per poter visualizzare i risultati delle scansioni su Google Earth.

Nella prima parte di codice vengono importati i vari pacchetti, settate alcune variabili che saranno utili in seguito come ad esempio dei contatori per le scansioni; viene aperto il file .kml e stabilita la connessione sulla porta COM3, utilizzata per la comunicazione con la scheda.

```
import serial
import time
import numpy as np
from astropy.time import Time
import os
import sys
import json
import requests
import pkg_resources
from base64 import decodebytes
import simplekml
authentication_token = 'AQEA/Cu6HvwqRuvoTdxWZZTOwdTGAN/vCOQIL4qZ8L9IFbnbyYd6'
DEFAULT_PORT = 443
almanac_single_satellyte_bytes = 20
SRVURI="https://gls.loracloud.com"
path = "C:\\Users\\falle\\Desktop\\test"
kml = simplekml.Kml()
wifi_count = 0
gnss_auto_count = 0
gnss_assisted_count = 0
ser = serial.Serial(
    port='COM3',\
    baudrate=9600,\
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
        timeout=1)
print("connected to: " + ser.portstr)
```

Il programma consiste in un loop infinito che legge di continuo 4 bit dalla porta COM, e li confronta con delle stringhe prestabilite che costituiscono i codici dei comandi inviati dalla scheda allo script. Ad esempio per le scansioni gps, alla ricezione della stringa "gpsd", la scansione GNSS è terminata e comincia lo scambio di dati: la prima informazione ricevuta è il tipo di scansione effettuato, se autonomo o assistito, inoltre viene incrementato il contatore delle scansioni, per tenere traccia del numero di scansioni effettuate. Successivamente segue lo scambio di dati, partendo dalla dimensione del nav message e numero di satelliti rilevati, per poi proseguire con il

nav message vero e proprio, tutti i dati dei satelliti (ID, C/N), tempi e consumi della scansione.

```
if (x == bytes(str("gpsd"), 'UTF-8')):
   print ("Scansione effettuata")
    scan type = ser.read(1)
   nb_sat_det = ser.read(1)
    if ((int.from_bytes(scan_type, "big") == 0)):
         gnss_auto_count = gnss_auto_count + 1
         print("Scansione di tipo autonomo")
    elif((int.from bytes(scan type, "big") == 1)):
        gnss_assisted_count = gnss_assisted_count + 1
        print("Scansione di tipo assistito")
    print ("numero satelliti rilevati:"
                        + str(int.from_bytes(nb_sat_det, "big")))
    scan_result_size = int.from_bytes(ser.read(1), "big")
    print ("dimensione del risultato in bytes:" + str(scan_result_size))
    sat_id = list(range(int.from_bytes(nb_sat_det, "big")))
    cnr = list(range(int.from_bytes(nb_sat_det, "big")))
    cnr_int = list(range(int.from_bytes(nb_sat_det, "big")))
    sat_id_int = list(range(int.from_bytes(nb_sat_det, "big")))
if (x == bytes(str("rslt"), 'UTF-8')):
   print("Il nav message è:")
   nav_message = ser.read(scan_result_size)
   nav_message_hex = nav_message.hex()
    nav_message_payload=nav_message[1:] #scarto il primo byte
   nav_payload=nav_message_payload.hex()
    print (nav_message_hex)
if (x == bytes(str("satd"), 'UTF-8')):
   print("I satelliti trovati sono:")
    for i in range(int.from_bytes(nb_sat_det, "big")):
        sat id[i] = ser.read(1)
        cnr[i] = ser.read(1)
        sat_id_int[i] = int.from_bytes(sat_id[i], "big")
        cnr_int[i] = int.from_bytes(cnr[i], "big")
        print ("ID satellite" +str(i+1)+ "=" + str(sat_id_int[i]))
       print ("C/N="+str(cnr_int[i]))
if (x == bytes(str("time"), 'UTF-8')):
        radio_ms= ser.read(4)
```

Analogamente per le scansioni Wi-Fi, lo script riceve delle stringhe che danno inizio alla trasmissione dei risultati. Al termine delle operazioni lo script torna nel loop principale a leggere 4 bit sulla porta in attesa di ulteriori istruzioni.

Richieste al server All'arrivo della stringa "rqst" lo script manda i risultati al server. Come descrito nella documentazione del Cloud Semtech [17], tutte le richieste devono essere strutturate nel seguente modo:

- HTTP POST requests, con i dati in formato JSON, (con content-type = application-json).
- Nell' HTTP header deve essere presente un Ocp-Apim-Subscription-Key token.
- Se l'autenticazione del token fallisce il server risponde con il codice 401 (Unauthorized)
- Se la risposta del server coincide con il codice 429 (Too Many Requests), bisogna rimandare la richiesta dopo un tempo pari al campo X-Retry-After-Millis contenuto nell'header.
- I risultati del cloud sono anch'essi in formato JSON

Le richieste quindi devono avere due campi nell'header, uno "Content-type"— application/json ed uno "Ocp-Apim-Subscription-Key" che andrà riempito con l'access token. Il corpo della richiesta deve avere la seguente struttura:

```
{
  "payload": HEX,
  "gnss_capture_time": FLOAT,
  "gnss_capture_time_accuracy": FLOAT,
  "gnss_assist_position": [FLOAT, FLOAT],
  "gnss_assist_altitude": FLOAT,
  "gnss_use_2D_solver": BOOL
}
```

Nel campo payload va inserito il NAV message in formato esadecimale e per come è formattato il primo byte deve essere scartato perchè indica solo la destinazione (se 0x00 il messaggio è destinato all'host, se 0x01 il messaggio è destinato al solver). Il campo gnss_capture_time è opzionale, e deve contenere eventualmente il tempo al quale la sdcansione è stata eseguita in formato GPS time. Il campo

gnss_capture_time_accuracy è opzionale e indica una stima dell'accuratezza del tempo fornito. gnss_assist_position è opzionale e contiente latitudine e longitudine del punto di partenza per la procedura iterativa per il calcolo della posizione. Infine gnss_assist_altitude e gnss_use_2D_solver contiene l'informazione in metri dell'altitudine, obbligatoria se il campo successivo è impostato a True, che riduce il problema di localizzazione a un problema 2D. La risposta del server è formattata nel seguente modo:

```
{
    "result": OBJECT, // Optional. LocationSolverResult or null
    "errors": STRING[], // Optional. Array of error messages
    "warnings": STRING[] // Optional. Array of warning messages
}
```

Dove il primo campo contiene il risultato della localizzazione, mentre gli altri due eventuali errori o warnings. Il codice di seguito riportato implementa queste procedure come richiesto nella documentazione, grazie al pacchetto requests che permette di realizzare in modo semplice ed intuitivo le richieste al server: si possono notare la definizione del payload nel dizionario "data", poi convertito in formato json, la definizione degli header necessari, e la richiesta effettuata con il metodo request.post(). La risposta sarà salvata nella variabile "response", dalla quale poi vengono estratti i risultati. Il server da contattare è: https://gls.loracloud.com/api/v3/almanac/full, https://gls.loracloud.com/api/v3/solve/gnss_lr1110_singleframe e https://gls.loracloud.com/api/v2/loraWifi rispettivamente per effemeridi, localizzazione gnss e Wi-Fi.

```
if (x == bytes(str("rqst"), 'UTF-8')):
  print("Mando i dati al solver ... ")
   data = {
                                      #Json body request
       'payload': str(nav_payload),
       'gnss_capture_time': capture_time,
       'gnss_capture_time_accuracy': 15,
       'gnss_assist_position': [43.437772, 13.613738]
  }
  newHeaders = {"Ocp-Apim-Subscription-Key": authentication_token,
   "Content-Type": "application/json"}
  response = requests.post(SRVURI
                + '/api/v3/solve/gnss_lr1110_singleframe',
                 data=json.dumps(data),
                 headers=newHeaders)
   gnss_response = response.json()
  print("Status code: ", response.status_code)
```

```
print (gnss_response)
if (gnss_response['result'] != None):
  if ((int.from_bytes(scan_type, "big") == 0)):
     gnss_flag_autonomous_results = 1
     latitude_gnss_autonomous=
                     gnss_response['result']['llh'][0]
     longitude_gnss_autonomous=
                     gnss_response['result']['llh'][1]
     altitude_gnss_autonomous=
                     gnss_response['result']['llh'][2]
   else:
     gnss_flag_assisted_results = 1
     latitude_gnss_assisted=
                     gnss_response['result']['llh'][0]
     longitude_gnss_assisted=
                     gnss_response['result']['llh'][1]
     altitude_gnss_assisted=
                     gnss_response['result']['llh'][2]
else:
  if((int.from_bytes(scan_type, "big") == 0)):
     gnss_flag_autonomous_results = 0
  else:
     gnss_flag_assisted_results = 0
```

Contattando il server in modo analogo, l'applicazione riesce ad ottenere anche i risultati per le scansioni Wi-Fi e un file aggiornato per le effemeridi. Gli header presenti nella richiesta saranno sempre gli stessi descritti precedentemente; per la richiesta Wi-Fi, la richiesta da fare è la seguente:

```
{
    "lorawan": ARRAY, // Required. Array of UplinkRssi objects.
    "wifiAccessPoints": ARRAY // Required. Array of WiFiAccessPoint objects.
}
```

L'informazione dei MAC addresses e RSSI è inserita nel campo "wifiAccessPoints", non avendo a disposizione il gateway il primo campo è stato lasciato vuoto. Di seguito sono riportate le parti per la richiesta Wi-Fi.

```
wifi[i] = diz
print("Mando i dati al solver ... ")
 data = {
                                  #Json body request
     'lorawan': [
                  "gatewayId": "00",
                  "rssi": 0,
                  "snr": 0,
                  "toa": 0,
                  "antennaId": 0,
                  "antennaLocation":
                  {"latitude": 0,
                  "longitude": 0,
                   "altitude": 0,},
                },
                ],
     'wifiAccessPoints': wifi
}
newHeaders = {"Ocp-Apim-Subscription-Key": authentication_token,
                       "Content-Type": "application/json"}
response = requests.post(SRVURI + '/api/v2/loraWifi',
               data=json.dumps(data),
               headers=newHeaders)
 wifi_response = response.json()
print("Status code: ", response.status_code)
print (wifi_response)
 if (wifi_response['result'] != None):
     if (wifi_response['result']['latitude'] != 0.0):
         latitude_wifi = wifi_response['result']['latitude']
         longitude_wifi = wifi_response['result']['longitude']
         wifi_flag_results = 1
     else:
         wifi_flag_results = 0
```

Per le effemeridi non sarà necessario inserire alcun file json, e in risposta si otterrà direttamente il file, se la richiesta è andata a buon fine, in seguito è mostrato il codice per la richiesta delle effemeridi.

Le risposte ricevute anche in questo caso sono in formato JSON, e contegono per la richiesta Wi-Fi due campi "longitude" e "latitude", mentre quella delle effemeridi un campo "almanac_image" con una stringa codificata Base64 contenente le informazioni desiderate.

7.3 Misure effettuate

7.3.1 Indoor/outdoor detection

Come prima prova si sono effettuate delle misure per la classificazione indoor/outdoor sfruttando le scansioni GNSS, del dispositivo. Sono state eseguite delle prove sia della modalità scansione autonoma, che di quella assistita nelle sue due versioni, "low power" e "best effort". Come indicato nel datasheet del dispositivo [1], la versione assistita si dovrebbe rivelare più efficiente in termini di consumo, poichè limita la ricerca a un set di satelliti minore rispetto a quella autonoma. In questa modalità il dispositivo costruisce una lista di satelliti da ricercare grazie all'informazione delle effemeridi e nella "low power" termina la scansione se nessuno di questi satelliti viene rilevato. Nella "best effort" invece, anche se nessun satellite della lista viene trovato, si prosegue nella ricerca per satelliti con segnali meno forti. Le prime prove sono state eseguite con scansioni sia per la costellazione GPS che Beidou, mentre poi si è limitata la ricerca ad una sola delle due costellazioni. Si è stimato il consumo di corrente e di potenza per le varie prove effettuate inserendo una funzione all'interno del programma che calcolasse i uAh consumati durante ogni scansione. Grazie ai driver forniti è possibile avere accesso ai tempi in ms che il dispositivo trascorre nelle varie fasi della scansione GPS, grazie alla funzione $lr1110_gnss_get_timings()$. Moltiplicandoli per il consumo di corrente relativo a ciascuna fase riportato nel datasheet, rispettivamete 10mA nella fase di cattura e 5mA in quella di processamento dei dati, è possibile ottenere una stima in uAh della corrente consumata. Inoltre considerando che il circuito è alimentato a 3.3V, è possibile anche ricavare il consumo di potenza in uWh.

```
float power_cons_uAh (lr1110_gnss_timings_t timings){
uint32_t cons_uAs;
float cons_uAh;

cons_uAs = timings.radio_ms*CURRENT_CONSUMPTION_CAPTURE_PHASE
```

Figura 7.3: Log script Python della scansione autonoma per costellazione GPS e BeiDou.

```
GPS time:1295535688

Scansione effettuata
Scansione di tipo assistito
numero satelliti rilevati:0
dimensione del risultato in bytes:2
Il nav message è:
11 nav message è:
12 nav message è:
13 nav message è:
14 nav message è:
15 nav message à:
16 nav message à:
17 nav message à:
18 nav message à:
19 nav message à:
19 nav message à:
10 nav message à:
10 nav message à:
10 nav message à:
10 nav message à:
11 nav message à:
12 nav message à:
13 nav message à:
14 nav message à:
15 nav message à:
16 nav message à:
17 nav message à:
18 nav message à:
19 nav message à:
10 n
```

Figura 7.4: Log script Python della scansione assistita per costellazione GPS e BeiDou.

Si può notare come per entrambe le tipologie di scansione i satelliti trovati siano 0, ed il Nav messagge corrisponde a 0007 in esadecimale, ovvero due bytes di cui il primo 0x00 indica il destinatario del messagio (in questo caso l'host processor), e il secondo 0x07 indica che nessun satellite è stato rilevato. Confrontando i tempi impiegati nelle scansioni, risulta evidente che nel caso della scansione autonoma il dispositivo impiega un tempo maggiore (3428ms) rispetto a quello del caso assistito e di conseguenza anche il consumo è maggiore (5.54uAh-18.15uWh). Per quella assistita, nella modalità "best effort", il tempo risulta minore della scansione autonoma (2811ms) e così anche il consumo (3.9uAh-12.87uWh). La modalità più efficiente è quella "low power": se non vengono trovati satelliti nella lista costruita, al constrario della "best effort" che prosegue con una scansione per i segnali più deboli, la ricerca viene interrotta, ottenendo i tempi e consumi più bassi (2101ms, 3.6uAh-11.88uWh).

Come atteso, limitando la ricerca ad una sola delle due costellazioni, si riducono ulteriormente i tempi ed i consumi delle scansioni. Infatti il tempo di ricerca si

```
GPS time:1295537562
Scansione effettuata
Scansione di tipo autonomo
numero satelliti rilevati:0
dimensione del risultato in bytes:2
Il nav message è:
0007
I satelliti trovati sono:
Timing_ms 1208
consumption_uAh 2.0250000953674316
```

Figura 7.5: Log script Python della scansione autonoma per costellazione GPS.

```
GPS time:1295538100

Scansione effettuata
Scansione di tipo assistito
numero satelliti rilevati:0

dimensione del risultato in bytes:2

Il nav message è:

0007

I satelliti trovati sono:

Timing_ms 918

consumption_uAh 1.6146423941616333

(a) Scansione effettuata
Scansione effettuata
Scansione di tipo assistito
numero satelliti rilevati:0
dimensione del risultato in bytes:2
Il nav message è:
0007

I satelliti trovati sono:
Timing_ms 918

consumption_uAh 1.3444443941116333

(b) Scansione assistita low power.
```

Figura 7.6: Log script Python della scansione assistita per costellazione GPS.

dimezza, ed essendo la parte più dispendiosa a livello di consumo, anche questo viene notevolmente abbassato. Si nota come i tempi risultano più che dimezzati passando da un massimo per la autonoma di 2811ms a 1208ms, con un consumo da 5.4uAh (18.15uWh) a 2.0uAh (6,6uWh), mentre per la modalità "low power" si passa da 2101ms a 718ms, con una corrente assorbita che passa da 3.6uAh (11.22uWh) a 1.34uAh (4.4uWh). Se la scansione assistita ritorna un nav message pari a 0008, che corrisponde ai due bytes 0x00 (destination=host processor) e 0x08 (almanac too old) l'applicazione manda una word allo script python ed invoca il download delle effemeridi dal server. Muovendosi all'interno di un ambiente indoor, potrebbe capitare di avvicinarsi a delle finestre o a zone di visibilità in cui il dispositivo riesce a catturare dei segnali. In generale, osservando i risultati dei test eseguiti e per come sono definite, con la modalità assistita si riescono a catturare più satelliti rispetto a quella autonoma, con un dispendio energetico comunque minore. In generale si riescono ad ottenere massimo 2 o 3 satelliti, ma non la risposta con le coordinate dal server.

In questo caso (Figura 7.7) si nota come nella scansione assistita vengono rilevati 3 satelliti, mentre in quella autonoma solamente uno. Il consumo risulta comunque più elevato nel secondo caso con 5.56uAh (18.3uWh) contro i 3.95uAh (13.0uWh) del secondo. Osservando il nav message esadecimale si può notare come in caso di rilievo di segnali in prima posizione si trova 01 che corrispe al byte 0x01 (Destination solver). Mandando la richiesta al solver tuttavia non si ottiene una localizzazione

```
GPS time: 1295611113
GPS time:1295611097
Scansione effettuata
                                      Scansione di tipo assistito numero satelliti rilevati:3
Scansione di tipo autonomo
                                      dimensione del risultato in bytes:27
numero satelliti rilevati:1
Il nav message è:
                                      I satelliti trovati sono:
GPS#23
01014e97222ea2d99ee44e00
                                      C/N=40dB
I satelliti trovati sono:
                                      GPS#19
                                      C/N=39dB
GPS#23
                                      GPS#22
C/N=39dB
                                      C/N=36dB
Timing_ms 3508
                                      Timing_ms 2976
consumption uAh 5.566666603088379
                                      consumption_uAh 3.9566666984558105
        (a) Scansione autonoma.
                                                  (b) Scansione assistita.
```

Figura 7.7: Scansioni autonoma e assistita in ambiente indoor con alcuni satelliti rilevati.

```
Status code: 200 {'result': None, 'warnings': [], 'errors': ['GNSS solver error [1]: Not enough received or usable satellites, totalReceived=3, usableGPS=3, usableBDS=0']}
```

Figura 7.8: Risposta del server.

ma l'errore "Not enough received or usable satellites" (Figura 7.8).

7.3.2 Outdoor assisted vs. autonomous

Spostandosi in outdoor, sono state effettuate delle misure per cercare di confrontare le due modalità assistita e autonoma sia dal punto di vista energetico che da quello di precisione nel risultato finale ottenuto. Inoltre si è valutata l'incidenza del gps timestamp necessario all'avvio della scansione.

Se non si pongono dei limiti al numero massimo di segnali desiderati, con la modalità assistita si otterranno ovviamente un numero maggiore di satelliti rispetto che al caso autonomo. Le scansioni assistite sono state effettuate in modalità low power, poichè in ambiente outdoor il dispositivo riesce sempre a trovare dei satelliti nella lista costruita, e quindi in automatico prosegue per la ricerca dei satelliti con segnali più deboli. Come si può vedere dal log dello script (Figura 7.9), con la autonoma si sono rilevati 9 satelliti, contro 12 di quella assistita. Il tempo impiegato ed il consumo di corrente risulta minore per la modalità assistita, che ricercando in una lista precostruita ottimizza i tempi di scansione pur avendo dei tempi di computazione maggiori in quanto riceve più satelliti. Per confrontare in modo migliore i consumi è possibile tramite un parametro dato in ingresso alla funzione di lancio della scansione, impostare un numero massimo di satelliti da ricercare, che porta il dispositivo a terminare le operazioni una volta che questo limite viene raggiunto. Nelle seguenti prove si è utilizzato un numero massimo di 7 satelliti. A parità di satelliti rilevati la modalità assistita si rivela ancora la più efficiente in termini di tempo impiegato e quindi di consumo. Si va dai 3524ms e 5.6uAh (18,48uWh) della autonoma ai 2973ms e 3.9uAh (12.87uWh) della assistita (Figura 7.11). Se

```
GPS time: 1295690366
                                                           Scansione effettuata
Scansione di tipo assistito
numero satelliti rilevati:12
GPS time:1295690061
                                                            dimensione del risultato in bytes:81
Scansione effettuata
                                                           Il nav message è:
0101a8aab94713c218a02ce9d33d1040214708bcdf803
Scansione di tipo autonomo
numero satelliti rilevati:9
dimensione del risultato in bytes:62
                                                           6e631c3d009cd861f93de10caa28c143b6c90788f6255
d022573a23c6a031447a444c43e15c334297861ad0089
Il nav message è:
Il nav message è: a9f14e58028df9f3c3a3f00853a 01010000c21aa0bb71c12f3040071d177ed884dafceed I satelliti trovati sono:
14008a5874617bf136a430342bd41f4d4211d76993168 GPS#12
2c8a8e1155d0f3502e24a4a19c35ece903
                                                           GPS#4
I satelliti trovati sono:
                                                           C/N=46dB
BeiDou#36
                                                           GPS#27
C/N=48dB
GPS#13
                                                           BeiDou#22
C/N=48dB
GPS#12
                                                           BeiDou#36
C/N=47dB
                                                            C/N=45dB
BeiDou#22
                                                           GPS#29
                                                            C/N=44dB
C/N=46dB
                                                           BeiDou#20
GPS#27
                                                            C/N=43dB
C/N=44dB
                                                           BeiDou#19
GPS#4
                                                            C/N=42dB
C/N=44dB
                                                           GPS#6
GPS#29
                                                            C/N=41dB
                                                           BeiDou#10
C/N=44dB
                                                           C/N=38dB
BeiDou#37
BeiDou#19
C/N=42dB
                                                           C/N=36dB
GPS#6
                                                           GPS#48
C/N=40dB
                                                            C/N=33dB
Timing_ms 3762
                                                            Timing_ms 3013
consumption uAh 5.741666889190674
                                                           consumption_uAh 4.026388931274414
             (a) Scansione autonoma.
                                                                      (b) Scansione assistita.
```

Figura 7.9: Scansioni autonoma e assistita outdoor.

volontariamente viene introdotto un errore sul timestamp GPS, anche se vengono rilevati sette satelliti come nel caso precedente, mandando la richiesta al server non si riescono ad ottenere in risposta le coordinate, ma viene segnalato l'errore "Location fix dropped due to poor accuracy, caused most likely by big timestamp error." (Figura 7.12). Nello script Python è stata realizzata una routine per salvare i risultati della geolocalizzazione in un file KML per poter visualizzare i dati. Oltre a latitudine e longitudine sono stati inseriti satelliti trovati, relativi C/N e consumo della scansione. Il segnaposto rosso indica la posizione attuale mentre quelli verdi e gialli rispettivamente scansioni assistite ed autonome. Nella Tabella 7.1 sono riportati i dati mediati su 100 scansioni in modalità assistita e autonoma. Come si può notare quella assistita è più efficiente. Per quanto riguarda la precisione, non risultano delle differenze tra le due modalità.

Tabella 7.1: GnssAssisted vs GnssAutonomous.

	Average Time(ms)	$Average\ Consumption(uAh/uWh)$
GNSS Assisted	2932	3.8/12.2
GNSS Autonomous	3532	5.7/18.79

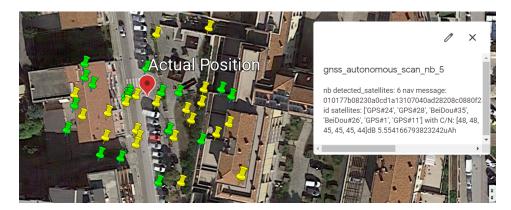


Figura 7.10: Visualizzazione dei risultati scansioni GNSS.

```
GPS time:1295714791
                                            GPS time:1295714992
Scansione effettuata
                                             Scansione effettuata
Scansione di tipo autonomo
                                             Scansione di tipo assistito
numero satelliti rilevati:7
                                            numero satelliti rilevati:7
dimensione del risultato in bytes:49
                                            dimensione del risultato in bytes:52
Il nav message è:
                                            Il nav message è:
01019eb08230a0338e910804425fc9a4b75f849ab4e 0101abb0b947138238a13e42da3960426370790c588
301ef09bd341fc442260b9a0e46c08038d45d448301 47efb01fc1e08cd4dcbc25e461c9a6b028e4036b488
6e28422f5ded
                                             4c1d816ba8b8f71a15
I satelliti trovati sono:
                                             I satelliti trovati sono:
GPS#24
                                             BeiDou#35
C/N=45dB
                                             C/N=44dB
BeiDou#26
                                             GPS#28
C/N=45dB
                                             C/N=43dB
GPS#1
                                             GPS#24
C/N=44dB
                                             C/N=43dB
GPS#11
                                             GPS#11
C/N=44dB
                                             C/N=43dB
BeiDou#35
                                             BeiDou#26
C/N=44dB
                                             C/N=42dB
BeiDou#29
                                             GPS#1
C/N=43dB
                                             C/N=42dB
GPS#30
                                             BeiDou#24
C/N=42dB
                                            C/N=41dB
Timing ms 3524
                                             Timing_ms 2973
consumption_uAh 5.588889122009277
                                             consumption uAh 3.969166603088379
          (a) Scansione autonoma.
                                                        (b) Scansione assistita.
```

Figura 7.11: Scansioni autonoma e assistita outdoor con numero massimo di satelliti da ricercare uguale a sette.

```
GPS time: 1295691825
Scansione effettuata
Scansione di tipo assistito
numero satelliti rilevati:7
dimensione del risultato in bytes:49
Il nav message è:
010103abb94713a23aa0d8dc01c86f422502d92e2084028bc5150f1bdc62e168f0cd
472263d0e3adb11ba4a163e892d203
 satelliti trovati sono:
BeiDou#36
C/N=47dB
GPS#29
C/N=46dB
GPS#27
C/N=44dB
GPS#4
C/N=44dB
BeiDou#19
C/N=42dB
GPS#48
C/N=34dB
GPS#35
C/N=34dB
Timing ms 2930
consumption uAh 3.91388931274414
Mando i dati al solver ...
Status code: 200
{'result': None, 'warnings': [], 'errors': ['GNSS solver error [2]:
Location fix dropped due to poor accuracy, caused most likely by big
timestamp error']}
```

Figura 7.12: Errore GPS Time.

7.3.3 Localizzazione Wi-Fi

Di seguito sono riportati i test eseguiti con le scansioni Wi-Fi. L'obiettivo è quello di ottenere almeno due indirizzi MAC che possono essere utilizzati per ottenere la posizione del dispositivo, unitamente ai valori dell'RSSI per migliorare la precisione della localizzazione. I risultati sono poi mandati in cloud ad un servizio Wi-Fi lookup. Come descritto la scansione si compone di tre fasi (Ricerca del preambolo, cattura e demodulazione), ripetute per ciascun canale impostato. Durante le prima delle due fasi il dispositivo attiva la parte a RF con un consumo maggiore, mentre nell'ultima fase di demodulazione questa viene dissattivata. La ricerca del preambolo dipende chiaramente dal traffico sul canale: se il canale è molto trafficato, questo viene subito rilevato e la durata è breve, in altri casi in cui la rete è poco trafficata la ricerca può essere lunga fino a un intervallo beacon. Per questi test sono state eseguite delle scansioni per Wi-Fi di tipo B G e N, sui canali 1, 6 e 11, che sono i più utlizzati in quanto gli unici che permettono di allocare il segnale Wi-Fi senza che sovrapposizioni. Anche qui si può decidere di terminare la scansione dopo un certo numero di MACs ricavati, potendo potenzialmente fermare l'applicazione dopo due MACs. Nei test si è impostato il valore massimo di 32 per vedere tutti gli access points possibili. Tipicamente gli access point trasmettono dei beacon ogni 102.4ms, quindi nella scansione è stata impostata una durata di 105ms per canale, per cercare di coprire tutto l'intervallo beacon, per un totale di 6 tentativi per canale, a meno che non sia rilevato un beacon.

Come è possibile notare da Figura 7.13 con queste impostazioni vengono rilevate dagli 8 ai 10 access points con i loro rispettivi RSSI. Essendo random il tempo di

Capitolo 7 Test LR1110

```
Wifi detected: 8
MAC_Adresses: ['20:b0:01:b4:2b:9b', '58:7f:66:2c:6c:05', 'cc:32:e5:90:
98:ad', '54:83:3a:cb:80:8e', 'd4:b7:09:27:09:93', '80:02:9c:50:65:ef',
'e0:28:6d:a4:6f:c6', '15:e1:35:85:d9:d2']
Consumption in uAh: 3.8508334159851074
Mando i dati al solver ...
Status code: 200 {'result': {'latitude': 43.43846, 'longitude': 13.611077, 'altitude': 0.0, 'accuracy': 17, 'algorithmType': 'Wifi', 'numberOfGatewaysReceive d': 0, 'numberOfGatewaysUsed': 0}, 'warnings': [], 'errors': []}
Wifi detected: 9
MRSSI values: [-92, -82, -72, -75, -78, -84, -32, -76, -80]
MAC_Adresses: ['02:22:6c:0a:0a:1d', '20:b0:01:b4:2b:9b', '58:7f:66:2c:
6c:05', '54:83:3a:cb:80:8e', 'fa:8f:ca:34:d1:6e', 'e0:19:54:43:a4:7a',
'e0:cc:f8:ef:7a:1c', '80:02:9c:50:65:ef', 'e0:28:6d:a4:6f:c6']
Consumption in uAh: 2.4927778244018555
Mando i dati al solver ...
Status code: 200
{'result': {'latitude': 43.438522, 'longitude': 13.611069, 'altitude':
0.0, 'accuracy': 16, 'algorithmType': 'Wifi', 'numberOfGatewaysReceive
d': 0, 'numberOfGatewaysUsed': 0}, 'warnings': [], 'errors': []}
                                                                (b)
 Wifi detected: 10
MAC_Adresses: ['58:7f:66:2c:6c:05', '02:22:6c:0a:0a:1d', '20:b0:01:
 b4:2b:9b', 'd4:b7:09:27:09:93', '54:83:3a:cb:80:8e', 'fa:8f:ca:34:d
 1:6e', 'e0:19:54:43:a4:7a', 'e0:cc:f8:ef:7a:1c', '80:02:9c:50:65:ef
    'e0:28:6d:a4:6f:c6'
 Consumption in uAh: 1.5322222709655762
                                                                 (c)
```

Figura 7.13: Log scansioni Wi-Fi.

scansione, anche il consumo è variabile da un massimo di 3.8uAh ad un minimo di 1.5uAh. In Figura 7.14 sono riportati i risultati della localizzazione ottenuta dalle scansioni Wi-Fi. Come per le scansioni GNSS anche i risultati Wi-Fi sono inseriti in un file KML che contiente l'informazione dei MAC rilevati, RSSI relativi e consumo della scansione. Il segnaposto giallo indica la posizione attuale dalla quale sono state fatte le misure, mentre quelli rossi indicano le coordinate ottenute dal solver. Cliccando sul segnaposto è possibile vedere le informazioni di ciascuna scansione. Le posizioni calcolate dal solver sono comprese in un raggio di 20m dalla posizione attuale.

Se si vogliono abbattere i consumi delle scansioni Wi-Fi occorre impostare il numero massimo di risultati desiderati pari a due. In questo modo non appena sono ottenuti due indirizzi di access point la ricerca viene bloccata ed i dati sono mandati al server. Come si può vedere in Figura 7.16 i consumi risultano molto minori (dell'ordine di 0.02uAh), tuttavia con soli due access points la precisione peggiora, mentre un buon compromesso potenza consumata/precisione può essere ottenuta con quattro access points (Figura 7.13).



Figura 7.14: Visualizzazione delle coordinate ottenute su Google Earth.

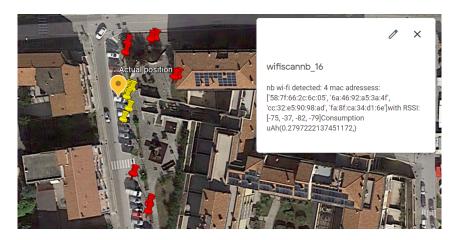


Figura 7.15: Visualizzazione delle coordinate ottenute con 4 MAC e 2 MAC.

```
MAC_Adresses: ['78:b4:6a:f8:76:94', '0c:b6:d2:5e:8c:74']
Consumption in uAh: 0.02527777850627899
Mando i dati al solver ...
Status code: 200
{'result': {'latitude': 43.438719, 'longitude': 13.611197, 'altitude': 0.0, 'accuracy': 53, 'algorithmType': 'Wifi', 'numberOfGatewaysReceived': 0, 'numberOfGatewaysUsed': 0}, 'warnings': [], 'errors': []}

(a) Scansione per 2 MAC

Wifi detected: 4
RSSI values: [-37, -81, -90, -64]
MAC_Adresses: ['6a:46:92:a5:3a:4f', 'cc:32:e5:90:98:ad', '7e:b4:6a:f8:76:94', '54:83:3a:cb:80:8e']
Consumption in uAh: 0.14888888597488403
Mando i dati al solver ...
Status code: 200
{'result': {'latitude': 43.438364, 'longitude': 13.611162, 'altitude': 0.0, 'accuracy': 16, 'algorithmType': 'Wifi', 'numberOfGatewaysReceived': 0, 'numberOfGatewaysUsed': 0}, 'warnings': [], 'errors': []}

(b) Scansione per 4 MAC
```

Figura 7.16: Log scansioni Wi-Fi.

7.3.4 Localizzazione GNSS vs. Wi-Fi

Indoor Detection Per la indoor detection è possibile solamente sfruttare le scansioni GNSS, poichè quella Wi-Fi rileva in ogni caso dei MAC address se sono presenti degli access points nelle vicinanze e non riesce a distinguere i due ambienti. Con le scansioni GNSS invece in indoor non sono rilevati satelliti o al più ne sono ottenuti 2/3. Ovviamente per ottenere un consumo più basso possibile è bene utilizzare la modalità assistita in low power, ed effettuare le scansioni per una sola costellazione di satelliti. Di seguito sono riportati i valori medi su 150 scansioni indoor, per la modalità low power assisted e autonomous.

Tabella 7.2: Valori medi dei consumi e tempi della indoor detection .

	Average Time(ms)	Current Consumption(uAh)	Power Consumption(uWh)
GNSS Assisted Low Power	753	1.44	5.08
GNSS Autonomous	1235	2.23	7.35

Wi-Fi vs GNSS Per la localizzazione outdoor si possono utilizzare entrambe le applicazioni in base alle esigenze. Se ci si trova in condizioni di buona visibilità si può sfruttare al meglio la parte GNSS, in particolare la modalità assistita che consente di ottenere una geolocalizzazione con un consumo di potenza minore. Le effemeridi sono valide per tre mesi, ma per avere una precisione migliore ed ottimizzare i consumi è bene aggiornarle una volta al mese, e quindi questo scambio di informazioni dovrà poi essere conteggiato nel consumo totale dell'applicazione, oltre all'uplink periodico del NAV message ottenuto. In ambienti più ostici come ambienti urbani, dove sono presenti degli ostacoli che peggiorano la visibilità, si può utilizzare la localizzazione Wi-Fi, che fornisce dei buoni risultati con consumi minori rispetto al GNSS, e senza la necessità di messaggi aggiuntivi oltre a quello di uplink contenente l'informazione dei MAC e RSSI.

Tabella 7.3: GnssAssisted vs Wi-Fi tempi.

	Average Time(ms)	Maximum Time(ms)	Minimum Time (ms)
GNSS Assisted	2932	3050	2875
Wi-Fi	507.2	2302	32

In Tabella 7.3, Tabella 7.4, e Tabella 7.5 sono riportati i tempi, i consumi medi minimi e massimi ed infine gli errori nella posizione finale determinata dal server. I valori sono stati ottenuti mediando i risultati su 150 scansioni eseguite nello stesso punto. Se i tempi e i consumi sono molto varibili per per le scansioni Wi-Fi, che

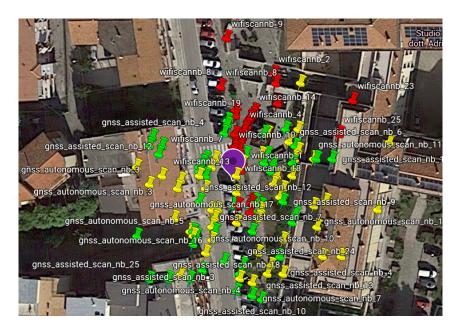


Figura 7.17: Visualizzazione delle coordinate ottenute con GNSS e Wi-Fi.

OD 1 11	_ 1	O 4 • 1		TT7. T7.	α .
Tabella	7 4.	GnssAssisted	VS	VV 1– H`1	Consumi
Laboria		CIIDDI IDDIDUCA	v D	* * 1 1 1	Combann.

	Average	Max	Min
	$\begin{array}{c} {\rm Consumptions} \\ {\rm (uAh/uWh)} \end{array}$	$\begin{array}{c} Consumption \\ (uAh/uWh) \end{array}$	$\begin{array}{c} Consumption \\ (uAh/uWh) \end{array}$
GNSS Assisted	3.8/12.4	4.1/13.5	3.73/12.0
Wi-Fi	1.43/4.75	6.65/21.9	0.09/0.32

risultano comunque minori della GNSS in media, per quest'ultima modalità essi sono essenzialmente fissi e poco variabili. Per quanto riguarda la precisione delle due, l'errore medio di entrambe le scansioni è vicino, leggermente superiore quello delle scansioni satellitari, rispettivamente 18.6m contro 15.2m. Con le scansioni GNSS in un caso di errore massimo si è ottenuto un valore di 400m, mentre l'errore massimo nel caso Wi-Fi è più basso e pari a 40.7m. D'altro canto le scansioni satellitari sono anche quelle che riescono a fornire in alcuni casi delle localizzazioni più precise con un errore minimo di 1.74m, contro 6m di quelle Wi-Fi. Per quanto riguarda i consumi se si confronta il valore medio, quello dell'applicazione Wi-Fi risulta più basso del

Tabella 7.5: GnssAssisted vs Wi-Fi errore.

	Average error(m)	Maximal error(m)	Minimal error(m)
GNSS Assisted	18.6	400	1.74
Wi-Fi	15.2	40.7	6.0

Capitolo 7 Test LR1110

GNSS con un valore di 4.75uWh. Per i valori massimi in alcuni casi si potrebbe avere che il consumo della Wi-Fi superi le scansioni satellitari, in quanto il tempo di ricerca è random e dipende dal traffico sul canale al momento della scansione. Nel caso migliore se i MAC vengono immediatamente rilvati, si ha un consumo di due ordini di grandezza inferiori rispetto alle scansioni Wi-Fi.

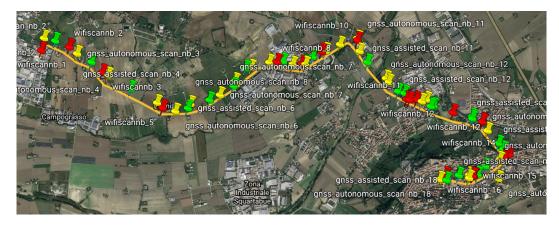


Figura 7.18: Tracking di un percorso ottenuto con GNSS e Wi-Fi.

Il dispositivo potrebbe essere sfruttato anche per realizzare delle applicazioni di tracking: in Figura 7.18 è stato tracciato un percorso in auto sfruttando tutte le scansioni disponibili del dispositivo: come si può vedere le posizioni ricavate approssimano di buon grado il percorso seguito (linea gialla). In assenza di centri abitati le scansioni Wi-Fi non funzionano e la posizione è ottenuta sfruttando le scansioni satellitari, mentre nei centri abitati vengono sfruttate anche queste in aggiunta a quelle satellitari.

Capitolo 8

Conclusioni

Il dispositivo è ancora in fase di sviluppo ed in questa tesi è stato descritto il progetto hardware della scheda, e le prime prove per il firmware (in particolare per il dispositivo LR1110), che saranno utili per realizzare il codice che andrà poi adattato ai nuovi prototipi, i quali presenteranno un microprocessore diverso da quello presente sulla developement board utilizzata. Nelle prossime fasi di progetto, da un lato occorrerà testare i prototipi realizzati e verificarne il funzionamento, effettuare le misure delle antenne presenti sulla scheda, ed inserire eventuali componenti, come ad esempio quelli che costituiscono le reti di adattamento delle varie antenne. Dall'altro lato, per quanto concerne il software del dispositivo, sono state effettuate solamente delle prove preliminari per il test del circuito integrato LR1110, e si procederà con la realizzazione del codice per i nuovi prototipi, per implementare anche le altre tecniche di localizzazione. Come si è potuto vedere dai risultati ottenuti, grazie all'LR1110 si possono combinare in un unico dispositivo le funzioni di ricetrasmettitore LoRa e ricevitore GNSS, con il valore aggiunto delle scansioni Wi-Fi. In questo modo si hanno a disposizione le funzionalità di localizzazione GNSS in buone condizioni di visibilità, ma anche localizzazione low power Wi-Fi per ambienti più complessi, con una BOM economica. Demandando le operazioni di computo della posizione al LoRa Cloud, si riescono ad ottenere dei consumi molto bassi, che contraddistinguono questo dispositivo, unico nel suo genere. Come si è visto la funzionalità Wi-Fi permette di avere dei consumi generalmente più bassi della GNSS, si adatta ad ambienti più difficili, e può essere tentata nella maggior parte dei casi. Se nei test effettuati la scheda trasmetteva i risultati via UART allo script Python e poi al server, ora si sta lavorando allo sviluppo della parte radio del dispositivo, che permette di inviare i dati ad un gateway tramite protocollo LoRaWAN, e da qui di inoltrarli ad un network e application server, che si occuperà di mandare le richieste al Cloud. Fino ad ora è stata sviluppata una comunicazione LoRa tra la developement board e il gateway SX1308, che riceve i dati ottenuti con le scansioni, e si sta lavorando ora all'implementazione del protocollo LoRaWAN per gestire il livello MAC. Nello sviluppo successivo si passerà alla parte UWB del dispositivo, che permette una localizzazione con precisione più elevata ai metodi precedenti. In particolare saranno implementate sia la tecnica del TWR per il check del distanziamento tra due dispositivi mobili (sfruttando anche l'interfaccia Bluetooth del microprocessore STM32WB55), che quella più complessa

Capitolo 8 Conclusioni

del TDoA per la localizzazione real time del device.

Bibliografia

- [1] Semtech. Datasheet lr1110, 2019. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/ 2R000000Q2YY/7hyal8gUewWREN8DSEk5R3Ee80pqEuVOdsHpiAHb3jo.
- [2] Decawave. Dwm1004c module datasheet, 2019. https://www.decawave.com/dwm1004/datasheet/.
- [3] STMicroelectronics. St25r95-datasheet, 2019. https://www.st.com/resource/en/datasheet/st25r95.pdf.
- [4] STMicroelectronics. Datasheet stm32wb55xx stm32wb35xx multiprotocol wireless 32-bit mcu arm®-based cortex®-m4 with fpu, bluetooth® 5 and 802.15.4 radio solution, 2019.
 - https://www.st.com/resource/en/datasheet/stm32wb55cc.pdf.
- [5] OKW. Enclosure technical drawings, 2019. https://www.okw.com/en/Soft-Case/A9050107.htm? ref=41d604b0-6960-11e5-b123-8eba63e66ed5&var= af314fc0-c2e5-11e2-8e2c-0050568225d7.
- [6] Smart Space. Smart safety, 2019. https://www.smartspace.it/smart-safety/.
- [7] Smart Space. Cantiere digitale: la sicurezza in cantiere è sempre più smart, 2019. https://www.smartspace.it/cantiere-digitale-la-sicurezza-in-cantiere-e-sempre-piu
- [8] Semtech. Lr1110 user manual, 2019. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/ 2R000000HpM8/OpM2oAJHQEpz6I_eL3t60wFr5wCmOtrkj2lfOuFZjPU.
- [9] Autodesk. Every layer in autodesk eagle. https://www.autodesk.com/products/eagle/blog/ every-layer-explained-autodesk-eagle/.
- [10] Semtech. Dev kit production folders, 2020. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/ 2R000000Hewj/jGl2KZP679Z46Yu40f.7opqpIenl_3NE.EQsIb9_YFk.
- [11] STMicroelectronics. Development of rf hardware using stm32wb microcontrollers application note, 2017.

Bibliografia

```
https://www.st.com/resource/en/application_note/dm00504903-development-of-rf-hardware-using-stm32wb-microcontrollers\-stmicroelectronics.pdf.
```

[12] STMicroelectronics. Getting started with x-nucleo-nfc03a1 nfc card reader board, 2019.

https://www.st.com/resource/en/user_manual/dm00544145-getting-started-with-xnucleonfc03a1-nfc-card-reader-board\-based-on-st25r95-for-stm32-nucleo-stmicroelectronics.pdf.

- [13] Semtech. Application note lr1110 evaluation kit, 2019. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/ 2R000000Q3Fh/FNVv6cMgSE0H80h5.q._eU.8BX6DbC8bcdW2tkG8UL4.
- [14] Semtech. Lr1110 tranceiver drivers, 2019. https://github.com/Lora-net/lr1110_driver.
- [15] STMicroelectronics. Description of stm32l4/l4+ hal and low-layer drivers, 2019. https://www.st.com/resource/en/user_manual/ dm00173145-description-of-stm32l4l4-hal-and-lowlayer-drivers-stmicroelectronics.pdf.
- [16] STMicroelectronics. Reference manual, 2019.

 https://www.st.com/resource/en/reference_manual/
 dm00083560-stm32147xxx-stm32148xxx-stm32149xxx-and-stm3214axxx-advanced\
 -armbased-32bit-mcus-stmicroelectronics.pdf.
- [17] Semtech. Api v3 (gnss), 2019.

 LoRa Cloud Documentation, https://www.loracloud.com/documentation/geolocation?url=gnss.html.