

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**EggLog Motion Core: progettazione e sviluppo
di un'applicazione mobile per l'interazione con
un dispositivo embedded ESP32**

Tesi di Laurea

Relatore

Prof. De Giovanni Luigi

Laureando

Agnese Carraro

Matricola 2068226

ANNO ACCADEMICO 2025-2026

Abstract

Il presente lavoro descrive l'attività di tirocinio svolta presso l'azienda Eggon S.r.l. all'interno del progetto EggLog Motion Core, un sistema di telemetria per motoveicoli. Tale sistema, sviluppato in più cicli di tirocinio, è basato sul data logger EggLog Track, un dispositivo embedded, basato sul microcontrollore ESP32, in grado di acquisire dati di posizione GPS, movimento inerziale (accelerazione, giroscopio) e altitudine, registrandoli su scheda microSD in formato binario. Il lavoro si è focalizzato sulla progettazione e lo sviluppo di un'applicazione mobile in React Native per la configurazione, il monitoraggio e la gestione delle sessioni di acquisizione dati del dispositivo embedded. L'applicazione comunica con tale dispositivo tramite il protocollo BLE (Bluetooth Low Energy) per il pairing, l'invio di comandi di controllo sessione e il provisioning della rete Wi-Fi. La connessione tramite rete Wi-Fi, invece, viene utilizzata per il download a chunk dei dati registrati su scheda microSD e include verifica di integrità, decompressione e persistenza locale. Il progetto ha previsto l'analisi dei requisiti, la progettazione e lo sviluppo dell'interfaccia utente per la gestione delle sessioni, la visualizzazione dei dati acquisiti e il loro download.

Indice

1	Introduzione	1
1.1	L'azienda Eggon S.r.l.	1
1.2	Il progetto	2
1.3	Organizzazione del testo	2
1.4	Notazione utilizzata	3
1.4.1	Terminologia	3
1.4.2	Riferimenti a figure e tabelle	4
1.4.3	Riferimenti bibliografici	4
2	Descrizione del tirocinio	5
2.1	Contenuti formativi previsti	5
2.2	Obiettivi fissati	6
2.3	Pianificazione	8
2.4	Analisi preventiva dei rischi	9
2.5	Organizzazione del lavoro	10
3	Analisi dei requisiti	12
3.1	Casi d'uso	12
3.1.1	Casi d'uso sul lato applicazione	13
3.1.2	Casi d'uso sul lato firmware	16
3.1.3	Descrizione casi d'uso	17
3.1.3.1	Descrizione casi d'uso sul lato applicazione	17
3.1.3.2	Descrizione casi d'uso sul lato firmware	20
3.2	Tracciamento dei requisiti	22
3.2.1	Requisiti funzionali	23

3.2.2	Requisiti non funzionali	28
3.2.3	Requisiti di vincolo	29
4	Progettazione e codifica	31
4.1	Tecnologie e strumenti	31
4.1.1	Applicazione mobile	31
4.1.1.1	Tecnologie già presenti nel progetto	31
4.1.1.2	Tecnologie introdotte nell'ambito di questo tiro- cinio	32
4.1.1.3	Motivazioni	32
4.1.2	Firmware	33
4.1.2.1	Tecnologie già presenti nel progetto	33
4.1.2.2	Tecnologie introdotte nell'ambito di questo tiro- cinio	34
4.1.2.3	Motivazioni	34
4.1.3	Altri componenti	35
4.2	Progettazione	36
4.2.1	Architettura generale del sistema	36
4.2.2	Modello dati	37
4.2.3	Comunicazione BLE	38
4.2.3.1	Ruolo del BLE nell'architettura	38
4.2.3.2	Connessione	39
4.2.3.3	Struttura GATT	39
4.2.3.4	Provisioning e connessione Wi-Fi	42
4.2.3.5	Gestione sessione (START / STOP / RESUME)	45
4.2.3.6	Streaming telemetria in tempo reale	49
4.2.4	Download via Wi-Fi	49
4.2.4.1	Download lato firmware	49
4.2.4.2	Download sul lato applicazione	52
4.2.5	UI e navigazione	55
4.2.5.1	Menù	55
4.2.5.2	Home	57

4.2.5.3	Devices	57
4.2.5.4	Record	58
4.2.5.5	Downloads	59
4.2.5.6	Activities	62
4.3	Design Pattern	63
4.3.1	Design Pattern lato firmware	64
4.3.1.1	State Machine	64
4.3.1.2	Producer-Consumer	65
4.3.1.3	Command [36]	66
4.3.2	Design Pattern sul lato applicazione	66
4.3.2.1	Repository	66
4.3.2.2	MVVM	66
4.3.2.3	Dependency Injection	67
4.3.3	Design Pattern misti	68
4.3.3.1	Observer	68
4.4	Sviluppo	68
5	Verifica e validazione	69
5.1	Verifica del codice	69
5.1.1	Verifica dei requisiti	70
5.1.2	Limiti della verifica	74
5.2	Validazione	75
6	Conclusioni	78
6.1	Consuntivo delle attività	78
6.2	Consuntivo degli obiettivi	79
6.3	Consuntivo dei rischi	80
6.4	Miglioramenti futuri	81
6.5	Conoscenze acquisite	82
	Glossario	83
	Riferimenti Bibliografici e Sitografici	85

Elenco delle figure

3.1	Diagramma dei casi d'uso lato applicazione.	15
3.2	Diagramma dei casi d'uso lato firmware.	17
3.3	Diagramma caso d'uso UC-A12.	20
3.4	Diagramma caso d'uso UC-F3.	22
4.1	Architettura generale del sistema, con i due canali di comunicazione tra applicazione e firmware.	36
4.2	Diagramma delle classi del modello dati di dispositivo e attività.	37
4.3	Diagramma di sequenza del flusso di connessione alla rete Wi-Fi esposta dal dispositivo.	43
4.4	Diagramma di sequenza del flusso di disconnessione alla rete Wi-Fi esposta dal dispositivo.	45
4.5	Diagramma di sequenza del controllo della sessione di attività (avvio).	46
4.6	Diagramma di sequenza del controllo della sessione di attività (termine).	47
4.7	Diagramma di sequenza del controllo della sessione di attività (riconnessione).	48
4.8	Diagramma di sequenza del flusso di download lato firmware.	50
4.9	Diagramma di sequenza del flusso di download lato applicazione.	53
4.10	Evoluzione della struttura di navigazione a tab.	56
4.11	Ristrutturazione della schermata Home, diventata una dashboard.	57
4.12	Evoluzione della schermata Devices, con l'aggiunta della modifica del nome e della selezione come preferito.	58

4.13	UI principale della schermata Record.	60
4.14	Schermata Record per la gestione della funzionalità di resume. .	61
4.15	Schermata Downloads.	62
4.16	Schermata Activities, con elenco delle attività scaricate.	63
4.17	Diagramma design pattern Producer-Consumer.	65
4.18	Diagramma design pattern MVVM.	67
5.1	Posizionamento del dispositivo e dello smartphone sulla moto. .	77

Elenco delle tabelle

2.1	Tabella obiettivi fissati prima del tirocinio.	6
2.2	Tabella obiettivi aggiornati nelle fasi iniziali del tirocinio.	7
2.3	Tabella pianificazione.	9
3.1	Tabella dei requisiti funzionali obbligatori lato applicazione. . .	23
3.2	Tabella dei requisiti funzionali obbligatori lato firmware.	26
3.3	Tabella dei requisiti funzionali desiderabili lato applicazione. . .	27
3.4	Tabella dei requisiti funzionali facoltativi lato applicazione. . . .	27
3.5	Tabella dei requisiti non funzionali desiderabili lato applicazione.	28
3.6	Tabella dei requisiti non funzionali desiderabili lato firmware. . .	29
3.7	Tabella dei requisiti di vincolo obbligatori lato applicazione. . .	29
3.8	Tabella dei requisiti di vincolo obbligatori lato firmware.	30
4.1	Service e Characteristic BLE implementati.	40
4.2	Transizioni dello stato globale del dispositivo.	64
5.1	Verifica dei requisiti funzionali obbligatori lato applicazione. . .	71
5.2	Verifica dei requisiti funzionali obbligatori lato firmware.	73

5.3	Verifica dei requisiti desiderabili, facoltativi, non funzionali e di vincolo.	73
6.1	Tabella consuntivo delle attività.	78
6.2	Tabella consuntivo degli obiettivi.	79

Capitolo 1

Introduzione

Questo capitolo descrive l'azienda presso cui è stato svolto il tirocinio e l'idea alla base del progetto. Vengono inoltre illustrate l'organizzazione e la notazione utilizzata in questo testo.

1.1 L'azienda Eggon S.r.l.

L'attività di tirocinio è stata svolta presso Eggon S.r.l., azienda informatica con sede a Padova nata con l'obiettivo di sviluppare soluzioni software innovative unendo concretezza e rapidità operativa [21]. Attualmente l'azienda conta circa trenta collaboratori e opera principalmente in tre settori:

- **Digital Health:** con la piattaforma MioPEDIATRA per la gestione digitale del rapporto tra pediatri e famiglie;
- **Internet of Things (IoT) e Industria 5.0:** per la connessione e l'analisi dati di dispositivi distribuiti;
- **Insurtech:** con la piattaforma MyWide per la gestione digitale delle polizze assicurative.

Il progetto EggLog, oggetto del presente tirocinio, si inserisce nell'ambito IoT dell'azienda, applicando le competenze di connettività e raccolta dati distribuita al contesto del motorsport motociclistico.

1.2 Il progetto

Il presente lavoro di tesi descrive l'attività di tirocinio svolta nell'ambito del progetto EggLog Track Intelligence & Motion Core, nato con l'obiettivo di sviluppare una piattaforma evoluta di data logging per il motorsport motociclistico, in grado di raccogliere, sincronizzare e analizzare i dati provenienti dal veicolo durante il suo utilizzo. Il progetto parte da un prodotto già esistente, costituito da:

- **dispositivo embedded** basato su microcontrollore ESP32 [12], sviluppato in un precedente ciclo di tirocini;
- **applicazione mobile** con delle funzionalità di base come il pairing con il dispositivo e la visualizzazione dei dati acquisiti in tempo reale.

Il tirocinio qui descritto si è concentrato sulla progettazione e sullo sviluppo, in React Native [38], di un'applicazione mobile completa che estendesse quella preesistente. Questo è stato fatto non solo evolvendo l'interfaccia utente, ma anche gestendo la comunicazione con il dispositivo e l'elaborazione e l'analisi dei dati ricevuti.

L'applicazione comunica con il dispositivo tramite i protocolli *BLE_G* (Bluetooth Low Energy) e *Wi-Fi_G* e offre all'utente funzionalità come l'invio di comandi di controllo per l'avvio e la conclusione di una nuova attività, il download a *Chunk_G* dei dati registrati sulla scheda *SD_G* (Secure Digital) con supporto alla ripresa di un trasferimento interrotto e alla verifica di integrità tramite codice *CRC_G* (Cyclic Redundancy Check), e la visualizzazione dei dati acquisiti e delle relative metriche di trasferimento. Tali funzionalità, e i requisiti da cui discendono, vengono descritti nel dettaglio nel Capitolo 3.

1.3 Organizzazione del testo

Il documento è organizzato nei seguenti capitoli:

Capitolo 1 - Introduzione. Presenta l'idea alla base del progetto, il contesto aziendale in cui si è svolto il tirocinio e l'organizzazione complessiva del testo;

Capitolo 2 - Descrizione del tirocinio. Elenca gli obiettivi formativi e descrive l'organizzazione delle attività e la pianificazione temporale;

Capitolo 3 - Analisi dei requisiti. Descrive i casi d'uso principali individuati, delineando le interazioni tra utente, applicazione e firmware, e definisce i requisiti tratti a partire da essi;

Capitolo 4 - Progettazione e codifica. Descrive la progettazione del sistema e gli aspetti relativi alla sua codifica;

Capitolo 5 - Verifica e validazione. Descrive le metodologie di test adottate per verificare il corretto funzionamento del prodotto;

Capitolo 6 - Conclusioni. Presenta i consuntivi delle attività e degli obiettivi e propone possibili sviluppi futuri.

1.4 Notazione utilizzata

In questa sezione vengono descritte la notazione e le convenzioni tipografiche utilizzate nel testo.

1.4.1 Terminologia

Per garantire la massima chiarezza ed evitare ambiguità, il testo integra una gestione formale della terminologia tecnica e degli acronimi attraverso un glossario, in cui ciascuna sigla o termine specialistico viene riportato per esteso e definito. Ad esempio, la sigla BLE viene definita come segue:

BLE (*Bluetooth Low Energy*): protocollo di comunicazione wireless a basso consumo energetico, ottimizzato per lo scambio di piccole quantità di dati tra dispositivi.

I termini presenti nel glossario sono contrassegnati nel testo alla loro prima comparsa con la codifica *termine_G*. Le occorrenze successive dello stesso termine non riportano più tale codifica, per non appesantire la lettura.

1.4.2 Riferimenti a figure e tabelle

Ogni figura e ogni tabella presente nel testo è numerata e corredata di didascalia, e viene sempre richiamata esplicitamente nel testo, ad esempio con "come riportato nella Figura 3.1". L'elenco delle figure e l'elenco delle tabelle sono consultabili all'inizio del documento.

1.4.3 Riferimenti bibliografici

Le fonti esterne richiamate nel testo sono indicate attraverso un numero racchiuso tra parentesi quadre. L'elenco completo delle fonti citate è riportato nella sezione di *Riferimenti Bibliografici e Sitografici* in fondo al documento.

Capitolo 2

Descrizione del tirocinio

Questo capitolo descrive gli obiettivi formativi del tirocinio e le modalità in cui è stato svolto.

2.1 Contenuti formativi previsti

Al fine di acquisire le competenze necessarie alla progettazione e all'implementazione dell'applicazione, il tirocinio si è aperto con una fase propedeutica di formazione ed esplorazione tecnologica. I contenuti formativi principali hanno riguardato:

- fondamenti dello sviluppo in **React Native** [38] (componenti, navigazione, gestione dello stato);
- utilizzo del protocollo **BLE** in ambiente React Native tramite librerie dedicate (es. *react-native-ble-manager* [24]) per la scansione, la connessione e la gestione del profilo **GATT_G** (Generic Attribute Profile) e delle notifiche;
- comunicazione client **Wi-Fi/HTTP_G** (Hypertext Transfer Protocol) tra dispositivo e applicazione;
- protocollo dati e comandi previsti dal firmware per la comunicazione con l'applicazione;
- gestione dello storage locale sul dispositivo mobile.

Queste competenze sono state individuate come propedeutiche in quanto strettamente necessarie allo sviluppo dell'applicazione mobile e alla sua comunicazione con il dispositivo embedded, così come richiesto dal progetto.

2.2 Obiettivi fissati

Gli obiettivi fissati per il tirocinio nel piano di lavoro sono descritti nella Tabella 2.1.

La codifica dei codici obiettivo segue la seguente logica:

- **O** rappresenta gli obiettivi obbligatori, quindi obiettivi primari richiesti dall'azienda;
- **D** rappresentandoli gli obiettivi desiderabili, non vincolanti o strettamente necessari, ma che danno un valore aggiunto significativo al prodotto;
- **F** rappresenta gli obiettivi facoltativi, il cui scopo è solo quello di dare valore aggiunto al prodotto.

Tabella 2.1: Tabella obiettivi fissati prima del tirocinio.

Codice	Descrizione
O1	Connessione e pairing BLE al device
O2	Invio dei comandi azione via BLE
O3	Download dei chunk dati via BLE (con resume e verifica CRC)
O4	Visualizzazione dei dati e delle metriche di trasferimento
O5	Stesura della documentazione tecnica
D1	Download dei chunk dati via Wi-Fi
D2	Decompressione dei chunk lato app
D3	Persistenza locale e gestione dei dati offline
F1	Streaming realtime dei dati in app
F2	Monitoraggio in tempo reale dei parametri del device
Continua nella prossima pagina...	

Codice	Descrizione
F3	Vista sullo stato del cloud / dati su EggLogCore

Durante lo svolgimento del tirocinio sono state apportate alcune modifiche al piano di lavoro, sulla base di considerazioni emerse nella fase iniziale di studio e analisi del sistema esistente. Tali modifiche, pur comportando una riorganizzazione degli obiettivi, non ne alterano in modo sostanziale i contenuti né le finalità formative: le attività svolte rimangono infatti coerenti con quelle originariamente pianificate, come mostrato nel confronto che segue. Nella Tabella 2.2 vengono riportati gli obiettivi aggiornati.

Tabella 2.2: Tabella obiettivi aggiornati nelle fasi iniziali del tirocinio.

Codice	Descrizione
O1	Invio dei comandi azione via BLE
O2	Download dei chunk dati via Wi-Fi (con resume e verifica CRC)
O3	Visualizzazione dei dati e delle metriche di trasferimento
O4	Gestione delle sessioni lato app
O5	Stesura della documentazione tecnica
D1	Decompressione dei chunk lato app
D2	Persistenza locale e gestione dei dati offline
D3	Streaming real time dei dati in app
F1	Monitoraggio in tempo reale dei parametri del device
F2	Vista sullo stato del cloud / dati su EggLogCore

Come si può osservare, le modifiche riguardano:

1. **connessione e pairing BLE al device:** diversamente da quanto previsto, i tirocinanti precedenti, che hanno completato il loro percorso tre

giorni prima dell'inizio del ciclo successivo, sono riusciti a sviluppare questa funzionalità, che è stata pertanto rimossa dal nuovo piano di lavoro in quanto già completata;

2. **download dei chunk:** è stato valutato che il BLE non offrisse banda sufficiente per il download dei dati e che fosse quindi necessario utilizzare esclusivamente il Wi-Fi. Poiché il download è stato considerato una funzionalità essenziale del prodotto, l'obiettivo è stato riassegnato tra gli obiettivi obbligatori;
3. **gestione delle sessioni:** la rimozione del download via BLE ha lasciato spazio a questo obiettivo, inserito tra quelli obbligatori vista la sua importanza per l'utente finale;
4. **streaming real time dei dati in app:** da obiettivo facoltativo è diventato un obiettivo desiderabile, a seguito di una rivalutazione della sua importanza.

Come si può notare, le modifiche introdotte riguardano principalmente una diversa distribuzione delle priorità e delle tecniche di realizzazione. Le attività svolte e gli obiettivi formativi restano compatibili con quanto definito nel piano originario. Per questo motivo, il piano aggiornato viene utilizzato come riferimento nella redazione del consuntivo delle attività (§6.1).

2.3 Pianificazione

È stato definito il seguente piano delle attività, articolato su un monte ore complessivo di 320 ore ripartite in otto settimane. A differenza degli obiettivi, la ripartizione delle ore non ha subito variazioni durante il tirocinio, e la Tabella 2.3 corrisponde pertanto alla pianificazione definita in fase iniziale.

Tabella 2.3: Tabella pianificazione.

Descrizione attività	Ore pianificate
Studio dello stato attuale del progetto lato app e lato firmware	40
Studio delle tecnologie	40
Analisi dei requisiti e progettazione	40
Sviluppo	120
Testing e validazione	40
Documentazione	40

2.4 Analisi preventiva dei rischi

Durante la fase iniziale del progetto è stata svolta un'analisi preventiva dei rischi, volta a identificare i potenziali ostacoli che avrebbero potuto compromettere il rispetto delle tempistiche o la qualità del prodotto finale. Per ciascun rischio individuato è stato stimato il livello di probabilità e impatto, definendo preventivamente le relative strategie di mitigazione. Di seguito riportiamo i rischi rilevati con le relative informazioni.

1. Difficoltà nell'estendere un sistema preesistente con nuove funzionalità, senza comprometterne il funzionamento originale.
 - Probabilità: Alta
 - Impatto: Medio
 - Mitigazione: Sviluppo di mini-progetti isolati per testare l'integrazione di ogni nuova componente prima di inserirla nel sistema principale, riducendo il rischio di regressioni.
2. Differenze di comportamento BLE/Wi-Fi tra Android e iOS.
 - Probabilità: Media

- Impatto: Medio
 - Mitigazione: Esecuzione di test continui su dispositivi fisici di entrambi gli ecosistemi.
3. Disallineamento tra interfaccia firmware e app durante lo sviluppo parallelo.
- Probabilità: Alta
 - Impatto: Alto
 - Mitigazione: Confronto costante tra i due tirocinanti, definizione anticipata del protocollo dati/comandi.
4. Instabilità della connessione Wi-Fi/BLE durante il trasferimento dati, con conseguente perdita di sincronizzazione tra stato app e stato reale del dispositivo.
- Probabilità: Media
 - Impatto: Alto
 - Mitigazione: Implementazione di meccanismi di resume e retry lato app, gestione esplicita degli stati di errore/timeout nella *UI_G* (User Interface).

2.5 Organizzazione del lavoro

Le attività di tirocinio sono state organizzate secondo la metodologia Agile Scrum [20], con sprint di durata bi-settimanale, stand-up meeting giornalieri tra tutor e tirocinanti, e momenti dedicati di pianificazione e revisione per monitorare l'avanzamento del lavoro e adattare le attività pianificate.

Oltre agli incontri giornalieri, sono stati previsti incontri bi-settimanali (a fine sprint) con il tutor aziendale, finalizzati a monitorare lo stato di avanzamento delle attività, a fare retrospettiva e a chiarire eventuali dubbi tecnici.

Il team di progetto era composto da due tirocinanti, ciascuno con un obiettivo distinto: l'evoluzione del firmware del dispositivo embedded da un lato, e

lo sviluppo dell'applicazione mobile dall'altro. Sebbene le due attività fossero assegnate individualmente, il lavoro non poteva essere svolto in isolamento in quanto ogni scelta progettuale da un lato si rifletteva direttamente sull'altro.

Per questo motivo, gran parte delle decisioni tecniche sono state prese attraverso un confronto costante tra i due tirocinanti. Questo approccio ha permesso di individuare soluzioni coerenti sin dalle fasi iniziali di progettazione, evitando disallineamenti che sarebbero potuti emergere lavorando separatamente.

Capitolo 3

Analisi dei requisiti

Questo capitolo descrive i casi d'uso identificati e i requisiti tratti da essi.

3.1 Casi d'uso

L'analisi dei requisiti costituisce il punto di partenza per la progettazione del sistema, poiché consente di individuare le funzionalità che l'applicazione e il firmware devono offrire per soddisfare le esigenze dell'utente finale. Da questa analisi sono stati ricavati diversi casi d'uso, relativi a due sistemi distinti ma strettamente interconnessi tra loro:

- **l'applicazione** si interfaccia con l'utente (attore principale) tramite l'UI e con il firmware (attore esterno) tramite il protocollo BLE o Wi-Fi;
- **il firmware** si interfaccia a sua volta con l'applicazione, che, in questo caso, assume il ruolo di attore principale.

Nei paragrafi seguenti vengono descritti i principali casi d'uso individuati, distinguendo quelli relativi all'applicazione da quelli relativi al firmware, e definiti univocamente tramite la codifica **[UC]-[Sistema][Numero]**, in cui:

UC sta per *use case*;

Sistema indica dove si applica il caso d'uso (**A** per l'Applicazione mobile, **F** per il Firmware);

Numero è un numero progressivo che parte da 1 per ogni sistema.

Questi casi d'uso vengono prima elencati in forma sintetica, per fornire una visione d'insieme del funzionamento del sistema, e successivamente vengono approfonditi nel dettaglio quelli ritenuti più significativi ai fini della comprensione dell'interazione tra i due sottosistemi.

3.1.1 Casi d'uso sul lato applicazione

In questo primo gruppo di casi d'uso, l'applicazione è il sistema e l'utente è l'attore principale. Il percorso tipico dell'utente attraverso l'applicazione parte dalla consultazione della schermata principale (UC-A1), per poi proseguire alla schermata Record (UC-A2), avviando la registrazione di una nuova attività (UC-A3) con la visualizzazione in tempo reale dei dati (UC-A4), fino alla sua conclusione (UC-A5). L'utente può in qualsiasi momento consultare lo storico delle attività scaricate tramite la schermata Activities (UC-A6), analizzandone il dettaglio (UC-A7). La schermata di Downloads (UC-A8) permette all'utente di connettersi alla rete Wi-Fi esposta dal dispositivo (UC-A9), per visualizzare la lista delle attività disponibili per il download (UC-A10) e selezionarne una (UC-A11). Il sistema riceve quindi il file richiesto, mentre l'utente ne monitora l'avanzamento, venendo informato di eventuali problemi (UC-A12). Questi 12 casi d'uso sono rappresentati nella Figura 3.1 e riportati in forma sintetica nell'elenco seguente, con l'indicazione del relativo codice identificativo:

UC-A1 - Visualizzazione Home. L'utente visualizza la schermata principale dell'applicazione, che mostra la lista dei dispositivi segnati come preferiti e i dati riassuntivi dell'ultima attività registrata;

UC-A2 - Visualizzazione Record. L'utente visualizza la schermata dedicata alla gestione dei pulsanti per l'inizio e la fine di una nuova attività, la calibrazione e la visualizzazione dei dati in tempo reale;

- UC-A3 - Avvio attività.** L'utente avvia una nuova attività e il sistema, tramite BLE, invia una notifica di inizio attività al firmware per sincronizzare la raccolta dati;
- UC-A4 - Visualizzazione attività in tempo reale.** Durante l'attività, l'utente visualizza dinamicamente dati come la durata, la velocità attuale, l'altitudine e la pendenza laterale e frontale;
- UC-A5 - Termine attività.** L'utente conclude l'attività tramite un pulsante dedicato. Il sistema, tramite BLE, notifica la conclusione dell'attività al firmware;
- UC-A6 - Visualizzazione Activities.** L'utente visualizza la schermata dedicata alla consultazione dello storico di tutte le attività scaricate dall'utente;
- UC-A7 - Visualizzazione dettaglio attività.** L'utente visualizza la schermata che mostra i dati finali acquisiti durante l'attività selezionata. L'utente ha, inoltre, la possibilità di modificare il nome dell'attività e di rimuovere l'attività dallo storico;
- UC-A8 - Visualizzazione Downloads.** L'utente visualizza la schermata per avviare il download locale di un'attività svolta;
- UC-A9 - Connessione Wi-Fi al dispositivo.** L'utente avvia, dalla schermata Downloads, la connessione alla rete Wi-Fi esposta dal dispositivo;
- UC-A10 - Visualizzazione attività disponibili.** L'utente visualizza la lista delle attività registrate sul dispositivo e disponibili per il download, fornite dal dispositivo;
- UC-A11 - Selezione attività da scaricare.** L'utente seleziona, dall'elenco delle attività disponibili, quella da scaricare e il sistema invia la richiesta al firmware;

UC-A12 - Monitoraggio avanzamento download. L'utente visualizza lo stato di avanzamento del download in corso ed eventuali notifiche relative a problemi riscontrati.

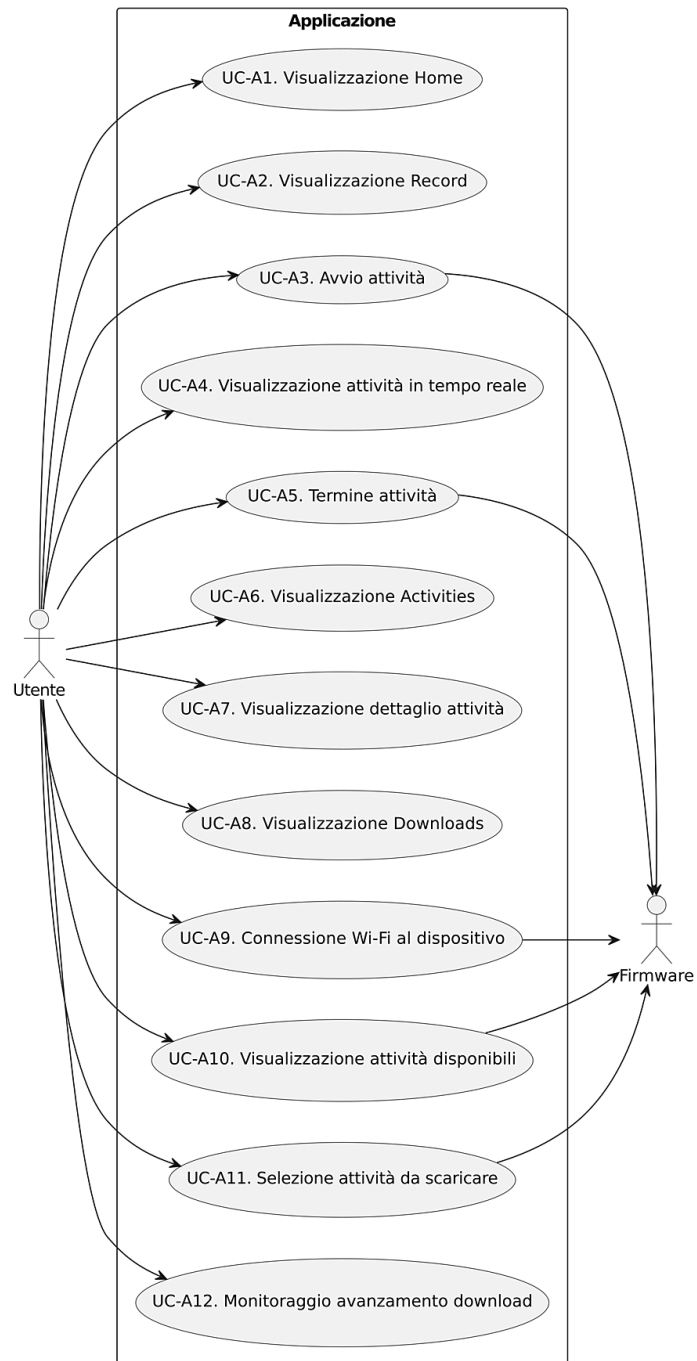


Figura 3.1: Diagramma dei casi d'uso lato applicazione.

3.1.2 Casi d'uso sul lato firmware

In questo gruppo di casi d'uso, invece, è il firmware stesso a rivestire il ruolo di sistema, mentre l'applicazione assume il ruolo di attore principale. I casi d'uso individuati sul lato firmware (Figura 3.2) riguardano strettamente il processo di download di un'attività: su richiesta dell'applicazione, il firmware espone la propria rete Wi-Fi in modalità access point (UC-F1). Una volta stabilita la connessione, fornisce all'applicazione l'elenco delle attività disponibili sulla scheda SD (UC-F2). Infine, alla selezione di un'attività da parte dell'utente, prepara e trasmette i relativi dati in modalità streaming continuo (UC-F3).

UC-F1 - Esposizione Wi-Fi. Il sistema riceve la richiesta di connessione dall'applicazione e inizia a esporre il suo Wi-Fi in modalità *SoftAP_G* (Access Point);

UC-F2 - Invio lista attività. Il sistema riceve dall'applicazione la richiesta dell'elenco delle attività disponibili per il download, consulta la scheda SD e restituisce il risultato;

UC-F3 - Invio dati attività. Il sistema riceve la richiesta di download di un'attività dall'applicazione, prepara i pacchetti da inviare in modalità streaming continuo e li trasmette via Wi-Fi.

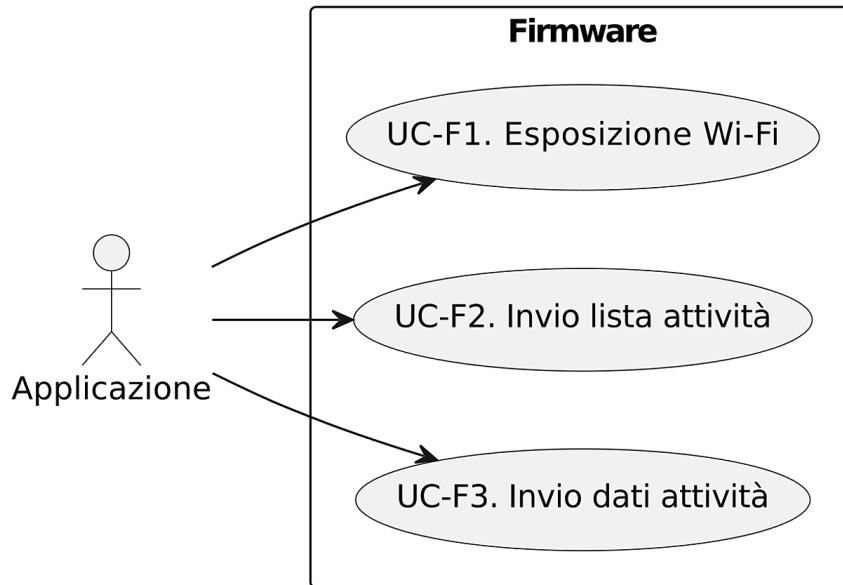


Figura 3.2: Diagramma dei casi d'uso lato firmware.

3.1.3 Descrizione casi d'uso

Tra i casi d'uso complessivamente individuati, vengono descritti in maggior dettaglio quelli ritenuti più significativi ai fini della comprensione del funzionamento del sistema, in quanto rappresentano i momenti di interazione più critici tra applicazione e firmware. Ogni caso d'uso verrà descritto indicando codice, nome, attore principale e attore esterno, precondizioni, postcondizioni e la sequenza dei passaggi che ne descrivono lo svolgimento.

3.1.3.1 Descrizione casi d'uso sul lato applicazione

Il primo caso d'uso approfondito riguarda l'avvio di una nuova attività da parte dell'utente.

UC-A3: Avvio attività

Attore Principale: Utente

Attore Esterno: Firmware

Precondizioni: L'applicazione è avviata e il dispositivo è correttamente connesso tramite BLE.

Descrizione:

1. L'utente seleziona il pulsante di avvio sull'applicazione;
2. L'applicazione riceve il segnale di avvio;
3. L'applicazione invia una notifica di inizio attività al firmware per avviare la raccolta dati.

Postcondizioni: L'attività è stata avviata sul lato applicazione e la notifica di sincronizzazione è stata inviata correttamente al firmware.

Una volta avviata, l'attività può essere conclusa dall'utente in qualsiasi momento, come descritto nel seguente caso d'uso.

UC-A5: Termine attività

Attore Principale: Utente

Attore Esterno: Firmware

Precondizioni: L'attività è stata correttamente avviata.

Descrizione:

1. L'utente seleziona il pulsante di termine sull'applicazione;
2. L'applicazione riceve il segnale di termine;
3. L'applicazione invia una notifica di termine attività al firmware per terminare la raccolta dati.

Postcondizioni: L'attività è stata terminata sul lato applicazione e la notifica di sincronizzazione è stata inviata correttamente al firmware.

Un'ulteriore funzionalità rilevante riguarda la possibilità, per l'utente, di scaricare in locale i dati completi di un'attività già svolta. Questo processo si articola in due casi d'uso: la selezione dell'attività da scaricare, con il conseguente invio della richiesta al firmware (UC-A11), e il monitoraggio dell'avanzamento del download fino al suo completamento (UC-A12). Quest'ultimo caso d'uso viene rappresentato nella Figura 3.3.

UC-A11: Selezione attività da scaricare

Attore Principale: Utente

Attore Esterno: Firmware

Precondizioni: L'utente ha avviato la connessione Wi-Fi al dispositivo e sta visualizzando l'elenco delle attività disponibili nella schermata Downloads.

Descrizione:

1. L'utente seleziona, dall'elenco delle attività disponibili, quella da scaricare;
2. L'applicazione invia al firmware la richiesta di download dell'attività selezionata.

Postcondizioni: La richiesta di download è stata inviata correttamente al firmware.

UC-A12: Monitoraggio avanzamento download

Attore Principale: Utente

Attore Esterno: Firmware

Precondizioni: La richiesta di download è stata inviata al firmware.

Descrizione:

1. Il sistema riceve i pacchetti di dati che compongono il file dell'attività;
2. Il sistema aggiorna in tempo reale lo stato di avanzamento del download;
3. Il sistema segnala all'utente eventuali problemi riscontrati durante il trasferimento;
4. Il termine, il sistema salva il file dell'attività in locale.

Postcondizioni: Il file dell'attività è stato ricevuto e salvato correttamente, oppure l'utente è stato informato di un eventuale errore.

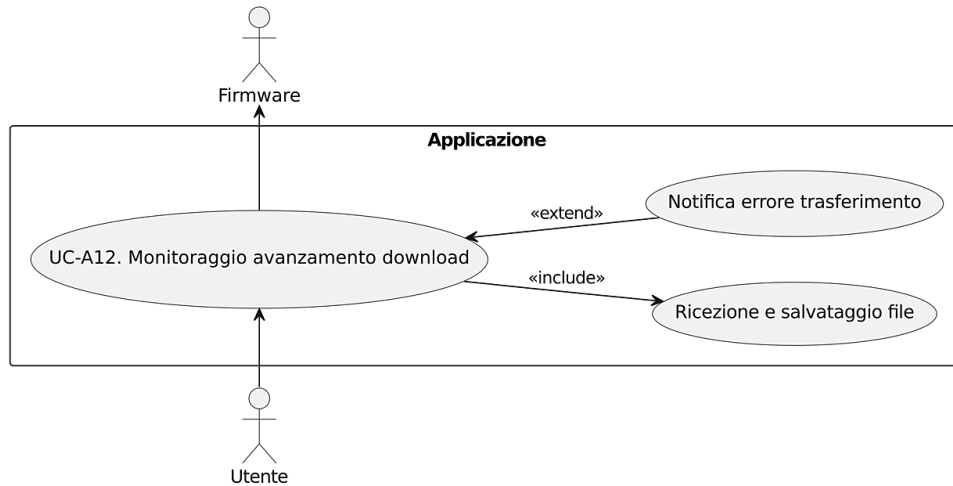


Figura 3.3: Diagramma caso d’uso UC-A12.

3.1.3.2 Descrizione casi d’uso sul lato firmware

Sul lato firmware, il primo caso d’uso descritto riguarda l’esposizione della rete Wi-Fi del dispositivo, su richiesta dell’applicazione.

UC-F1: Esposizione Wi-Fi

Attore Principale: Applicazione

Precondizioni: Il dispositivo è connesso via BLE e l’utente ha avviato la connessione Wi-Fi dall’applicazione.

Descrizione:

1. Il sistema riceve la richiesta di connessione Wi-Fi dall’applicazione;
2. Il sistema attiva l’interfaccia Wi-Fi in modalità SoftAP;
3. Il sistema espone la sua rete, rendendola visibile e disponibile per la connessione.

Postcondizioni: La rete Wi-Fi del dispositivo è attiva e pronta ad accettare la connessione dell’applicazione.

Una volta stabilita la connessione, il secondo caso d’uso riguarda la fornitura dell’elenco delle attività disponibili per il download.

UC-F2: Invio lista attività

Attore Principale: Applicazione

Precondizioni: La connessione Wi-Fi tra applicazione e dispositivo è stata stabilita.

Descrizione:

1. Il sistema riceve dall'applicazione la richiesta dell'elenco delle attività disponibili;
2. Il sistema consulta la scheda SD per individuare le attività registrate;
3. Il sistema invia all'applicazione l'elenco delle attività disponibili per il download.

Postcondizioni: L'applicazione ha ricevuto correttamente l'elenco delle attività disponibili.

Infine, l'ultimo caso d'uso sul lato firmware, mostrato nella Figura 3.4, descrive l'invio dei dati dell'attività selezionata, su richiesta dell'applicazione.

UC-F3: Invio dati attività

Attore Principale: Applicazione

Precondizioni: Il sistema ha ricevuto dall'applicazione la richiesta di download di un'attività.

Descrizione:

1. Il sistema preleva dalla scheda SD i dati dell'attività richiesta;
2. Il sistema prepara i pacchetti da inviare;
3. Il sistema trasmette i pacchetti all'applicazione via Wi-Fi in modalità streaming continuo.

Postcondizioni: I dati dell'attività sono stati trasmessi correttamente all'applicazione.

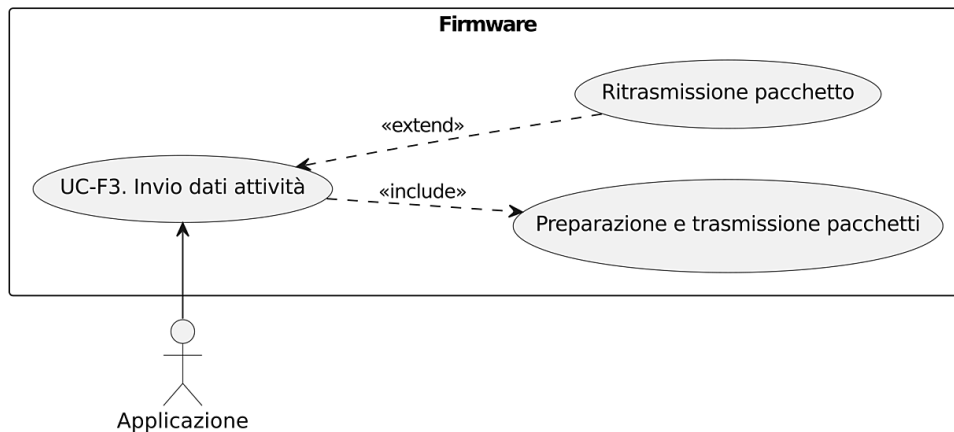


Figura 3.4: Diagramma caso d’uso UC-F3.

3.2 Tracciamento dei requisiti

A partire dalla definizione dei casi d’uso appena illustrati, sono stati identificati e definiti i requisiti del sistema, che vengono codificati tramite la sigla [Tipologia][Priorità]-[Sistema][Codice], in cui:

- **Tipologia** indica la categoria di appartenenza (**RF** per Requisito Funzionale, **RN** per Requisito Non Funzionale e **RV** per Requisito Di Vincolo);
- **Priorità** indica l’importanza del requisito (**O** per Obbligatorio, **D** per Desiderabile, **F** per Facoltativo);
- **Sistema** definisce dove si applica il requisito (**A** per l’Applicazione mobile, **F** per il Firmware);
- **Codice** numero progressivo che riparte da 1 per ciascuna combinazione di tipologia e sistema.

Nei paragrafi successivi, i requisiti vengono presentati suddivisi in base alla loro tipologia e, all’interno di ciascuna tipologia, distinti in base alla priorità e al sottosistema a cui si riferiscono.

3.2.1 Requisiti funzionali

I requisiti funzionali descrivono le funzionalità che l'applicazione e il firmware devono offrire, e vengono tratti in modo pressoché diretto dai casi d'uso descritti nella sezione precedente. Sul lato applicazione sono stati individuati requisiti a tutti e tre i livelli di priorità, mentre sul lato firmware, data la minore estensione delle funzionalità coinvolte, sono stati definiti unicamente requisiti obbligatori.

I requisiti funzionali obbligatori sul lato applicazione, riportati nella Tabella 3.1 con indicazione dei casi d'uso di riferimento, coprono l'intero percorso dell'utente attraverso l'applicazione. Partendo dalla consultazione della schermata Home, si passa alla registrazione di una nuova attività, poi alla consultazione dello storico e, infine, al download in locale di un'attività. In quest'ultimo ambito rientrano anche i requisiti relativi alla verifica di integrità dei dati ricevuti tramite codice CRC e alla possibilità di riprendere un download interrotto.

Tabella 3.1: Tabella dei requisiti funzionali obbligatori lato applicazione.

Requisito	Descrizione	Caso d'uso
RFO-A1	L'applicazione deve mostrare nella schermata Home la lista dei dispositivi preferiti e i dati riassuntivi dell'ultima attività registrata.	UC-A1
RFO-A2	L'applicazione deve fornire una schermata Record con i pulsanti per l'avvio e la fine di una nuova attività e per la calibrazione.	UC-A2
RFO-A3	L'applicazione deve inviare al firmware, tramite BLE, una notifica di inizio attività alla selezione del pulsante di avvio.	UC-A3
RFO-A4	L'applicazione deve mostrare in tempo reale, durante l'attività, la sua durata, la velocità attuale, l'altitudine e la pendenza laterale e frontale.	UC-A4
Continua nella prossima pagina...		

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Caso d'uso
RFO-A5	L'applicazione deve inviare al firmware, tramite BLE, una notifica di termine attività alla selezione del pulsante di termine.	UC-A5
RFO-A6	L'applicazione deve mostrare nella schermata Activities lo storico di tutte le attività scaricate dall'utente.	UC-A6
RFO-A7	L'applicazione deve consentire, per ogni attività, la modifica del nome e la rimozione dallo storico.	UC-A7
RFO-A8	L'applicazione deve fornire una schermata Downloads da cui avviare il download locale di un'attività svolta.	UC-A8
RFO-A9	L'applicazione deve consentire all'utente di avviare, dalla schermata Downloads, la connessione alla rete Wi-Fi esposta dal dispositivo.	UC-A9
RFO-A10	L'applicazione deve richiedere e mostrare la lista delle attività disponibili per il download fornita dal dispositivo.	UC-A10
RFO-A11	L'applicazione deve inviare al firmware la richiesta di download dell'attività selezionata dall'utente.	UC-A11
RFO-A12	L'applicazione deve mostrare in tempo reale lo stato di avanzamento del download in corso, segnalando all'utente eventuali problemi riscontrati.	UC-A12
RFO-A13	L'applicazione deve salvare in locale il file dell'attività al termine del download.	UC-A12

Continua nella prossima pagina...

Tabella 3.1 – Continuo della tabella

Requisito	Descrizione	Caso d'uso
RFO-A14	L'applicazione deve calcolare il codice CRC di ogni chunk ricevuto e confrontarlo con quello fornito dal firmware, segnalando l'errore e richiedendo la sua ritrasmissione in caso di mancata corrispondenza.	UC-A12
RFO-A15	L'applicazione deve poter riprendere un download interrotto a partire dall'ultimo chunk correttamente ricevuto, senza dover riavviare il trasferimento dall'inizio.	UC-A12
RFO-A16	L'applicazione deve mantenere lo stato della sessione attiva con il dispositivo, conservandolo durante la navigazione tra le schermate.	UC-A3, UC-A5
RFO-A17	L'applicazione deve gestire la chiusura o l'interruzione della sessione con il dispositivo, notificando l'utente e riportando l'interfaccia a uno stato coerente.	UC-A3, UC-A5

Sul lato firmware, i requisiti funzionali obbligatori, elencati nella Tabella 3.2 con indicazione dei casi d'uso di riferimento, riguardano invece l'intero processo di download, dall'esposizione della rete Wi-Fi fino alla trasmissione dei dati dell'attività. Vengono inclusi anche i due requisiti complementari a RFO-A14 e RFO-A15: il calcolo del codice CRC di ogni chunk trasmesso e il supporto alla ripresa di un download interrotto.

Tabella 3.2: Tabella dei requisiti funzionali obbligatori lato firmware.

Requisito	Descrizione	Caso d'uso
RFO-F1	Il firmware deve, su richiesta dell'applicazione, attivare l'interfaccia Wi-Fi in modalità SoftAP ed esporre la rete del dispositivo.	UC-F1
RFO-F2	Il firmware deve fornire all'applicazione l'elenco delle attività disponibili per il download, consultando la scheda SD.	UC-F2
RFO-F3	Il firmware deve preparare i pacchetti dei dati di un'attività alla ricezione di una richiesta di download dall'applicazione.	UC-F3
RFO-F4	Il firmware deve trasmettere i pacchetti dei dati dell'attività all'applicazione via Wi-Fi in modalità streaming continuo.	UC-F3
RFO-F5	Il firmware deve calcolare e allegare a ciascun chunk di dati trasmesso un codice CRC, per consentirne la verifica di integrità da parte dell'applicazione.	UC-F3
RFO-F6	Il firmware deve supportare la ripresa di un download interrotto, riprendendo la trasmissione dal chunk richiesto dall'applicazione senza ritrasmettere l'intero file.	UC-F3

Oltre ai requisiti obbligatori appena descritti, sono stati individuati dei requisiti funzionali desiderabili sul lato applicazione, riportati nella Tabella 3.3. Si tratta di funzionalità che, pur non essendo indispensabili al corretto funzionamento del sistema, ne migliorano l'esperienza d'uso.

Tabella 3.3: Tabella dei requisiti funzionali desiderabili lato applicazione.

Requisito	Descrizione	Caso d'uso
RFD-A1	L'applicazione dovrebbe decomprimere i chunk di dati ricevuti dal firmware prima di salvarli o elaborarli.	UC-A12
RFD-A2	L'applicazione dovrebbe rendere disponibile la consultazione dello storico delle attività già scaricate anche in assenza di connessione di rete, gestendone la persistenza in locale.	UC-A6, UC-A7
RFD-A3	L'applicazione dovrebbe ricevere e visualizzare in streaming i dati relativi all'attività in corso, rappresentandoli mediante grafici illustrativi.	UC-A4

Infine, sono stati individuati due requisiti funzionali facoltativi sul lato applicazione, mostrati nella Tabella 3.4. Trattandosi di funzionalità aggiuntive non ancora formalizzate in un caso d'uso specifico, la relativa colonna non riporta alcun riferimento.

Tabella 3.4: Tabella dei requisiti funzionali facoltativi lato applicazione.

Requisito	Descrizione	Caso d'uso
RFF-A1	L'applicazione dovrebbe poter visualizzare in tempo reale parametri di stato del dispositivo, quali livello di batteria, spazio di archiviazione residuo e stato della connessione.	—
Continua nella prossima pagina...		

Tabella 3.4 – Continuo della tabella

Requisito	Descrizione	Caso d'uso
RFF-A2	L'applicazione dovrebbe fornire una vista sullo stato di sincronizzazione delle attività con la piattaforma cloud EggLogCore.	–

3.2.2 Requisiti non funzionali

I requisiti non funzionali non descrivono una funzionalità specifica, ma la modalità in cui il sistema deve operare, ad esempio in termini di affidabilità rispetto a condizioni di funzionamento non ideali.

Sul lato applicazione, è stato individuato il requisito riportato nella Tabella 3.5, relativo alla gestione delle interruzioni temporanee della connessione BLE durante lo svolgimento di un'attività.

Tabella 3.5: Tabella dei requisiti non funzionali desiderabili lato applicazione.

Requisito	Descrizione	Caso d'uso
RND-A1	L'applicazione dovrebbe gestire in modo controllato la perdita temporanea della connessione BLE, informando l'utente senza terminare la sessione.	UC-A3, UC-A5

Sul lato firmware, è stato individuato il requisito mostrato nella Tabella 3.6, riguardante la stabilità della rete Wi-Fi esposta dal dispositivo.

Tabella 3.6: Tabella dei requisiti non funzionali desiderabili lato firmware.

Requisito	Descrizione	Caso d'uso
RND-F1	Il firmware dovrebbe mantenere stabile la connessione Wi-Fi in modalità SoftAP per l'intera durata del download di un'attività.	UC-F1, UC-F3

3.2.3 Requisiti di vincolo

I requisiti di vincolo, infine, rappresentano le limitazioni o le condizioni che devono essere rispettate nello sviluppo o nell'utilizzo del sistema. In questo caso derivano dalle tecnologie di comunicazione adottate tra applicazione e firmware.

Sul lato applicazione, i vincoli individuati, riportati nella Tabella 3.7, riguardano i protocolli di comunicazione con il firmware.

Tabella 3.7: Tabella dei requisiti di vincolo obbligatori lato applicazione.

Requisito	Descrizione	Caso d'uso
RVO-A1	L'applicazione deve comunicare con il firmware tramite il protocollo BLE per lo scambio delle notifiche di sincronizzazione.	UC-A3, UC-A5
RVO-A2	L'applicazione deve comunicare con il firmware tramite il protocollo Wi-Fi per il download delle attività.	UC-A9, UC-A11

Sul lato firmware, il vincolo individuato, riportato nella Tabella 3.8, riguarda la modalità con cui il dispositivo deve esporre la propria rete Wi-Fi.

Tabella 3.8: Tabella dei requisiti di vincolo obbligatori lato firmware.

Requisito	Descrizione	Caso d'uso
RVO-F1	Il firmware deve esporre la rete Wi-Fi del dispositivo esclusivamente in modalità access point (SoftAP), su richiesta dell'applicazione.	UC-F1

Capitolo 4

Progettazione e codifica

Questo capitolo descrive la progettazione del sistema (architettura, modello dei dati, comunicazione, elaborazione dei dati e design pattern) e gli aspetti relativi alla sua codifica (ambiente e strumenti di sviluppo utilizzati).

4.1 Tecnologie e strumenti

Questa sezione presenta le principali tecnologie e librerie impiegate nello sviluppo del sistema, distinguendo tra componenti già presenti nel progetto ereditati dal precedente ciclo di tirocinio e componenti introdotti durante il tirocinio. Per ogni componente introdotto viene specificato brevemente il suo scopo all'interno del progetto e le motivazioni alla base della scelta effettuata.

4.1.1 Applicazione mobile

Lo sviluppo dell'applicazione si è appoggiato principalmente a librerie già presenti nel progetto per la comunicazione con il dispositivo e la persistenza locale dei dati. L'unica nuova tecnologia introdotta riguarda la decompressione dei dati ricevuti durante il download di un'attività.

4.1.1.1 Tecnologie già presenti nel progetto

Questo elenco mostra le tecnologie già integrate nel progetto e riutilizzate senza modifiche.

- React Native [38], framework di sviluppo dell'applicazione mobile;
- TypeScript [29], linguaggio di programmazione utilizzato per lo sviluppo dell'applicazione mobile;
- react-native-ble-manager [24], libreria per la comunicazione BLE con il dispositivo (§4.2.3);
- HTTP, protocollo per lo scambio dati con il webserver esposto dal firmware durante il download (§4.2.4);
- AsyncStorage [35], sistema di persistenza locale per i dati applicativi, come i dispositivi associati, attività scaricate (§4.2.2).

4.1.1.2 Tecnologie introdotte nell'ambito di questo tirocinio

L'unica tecnologia introdotta è:

- fflate [32], libreria per la decompressione dei chunk ricevuti dal dispositivo (§4.2.4).

4.1.1.3 Motivazioni

La libreria `fflate` [32] presenta le seguenti funzioni:

- `inflateSync` e `inflateRawSync` sono funzioni che permettono di decomprimere correttamente il flusso raw-DEFLATE prodotto da `uzlib`;
- `zipSync` è una funzione che consente di generare l'archivio `.zip` richiesto per l'esportazione.

Sono state valutate anche soluzioni alternative:

- JSZip [39], libreria focalizzata sulla creazione e gestione di archivi zip, ma non in grado di gestire autonomamente la decompressione raw-DEFLATE;
- pako [31], libreria solida per la sola fase di decompressione raw-DEFLATE, ma priva di funzionalità per la creazione di archivi zip, rendendo comunque necessaria una libreria aggiuntiva (ad esempio JSZip) per la fase di esportazione.

A fronte di queste alternative, `fflate` è risultata la soluzione più adatta poiché l'unica in grado di coprire entrambe le esigenze (compressione e decompressione) con una singola dipendenza, senza richiedere l'introduzione di pacchetti aggiuntivi, e mantenendo allo stesso tempo un peso ridotto.

4.1.2 Firmware

Rispetto all'applicazione, il firmware ha richiesto un numero maggiore di componenti nuovi, necessari a implementare le funzionalità di esposizione della rete Wi-Fi e di trasferimento dei dati introdotte nel tirocinio, che si affiancano alle librerie preesistenti per la gestione delle task concorrenti, lo stack BLE e l'accesso alla scheda microSD.

4.1.2.1 Tecnologie già presenti nel progetto

Le seguenti tecnologie elencate erano già integrate nel progetto e sono state riutilizzate senza modifiche:

- ESP32-S3 [12] (framework Arduino), il microcontrollore su cui è eseguito il firmware;
- C++ [16], linguaggio di programmazione in cui è stato scritto il firmware;
- FreeRTOS [1], sistema operativo in tempo reale per la gestione delle task concorrenti (§4.3);
- NimBLE-Arduino [18], libreria per lo stack BLE lato dispositivo (§4.2.3);
- WebServer.h [14], libreria per il webserver HTTP esposto durante il download via Wi-Fi (§4.2.4);
- SdFat [19], libreria per l'accesso alla scheda microSD su cui vengono salvate le attività registrate (§4.2.4.1);
- JSON, formato dei file per la serializzazione delle risposte del webserver, come l'elenco delle attività disponibili (§4.2.4.1);

- `uzlib` [17], libreria per la compressione dei dati prima della trasmissione, controparte sul lato firmware della decompressione sul lato applicazione (§4.2.4.1).

4.1.2.2 Tecnologie introdotte nell’ambito di questo tirocinio

Le tecnologie introdotte sono:

- `WiFiAP.h` [15], libreria per l’esposizione della rete Wi-Fi in modalità SoftAP (§4.2.3.4);
- `esp_rom_crc` [10], libreria per il calcolo del codice di controllo CRC allegato a ciascun chunk trasmesso (§4.2.4.1).

4.1.2.3 Motivazioni

Le motivazioni di queste scelte sono le seguenti:

- per **WiFiAP.h**, la scelta di esporre la rete Wi-Fi in modalità SoftAP, anziché in modalità *STAG* (Station), deriva da un vincolo di funzionamento del sistema (requisito RVO-F1), in particolare, EggLog Track deve poter essere utilizzato ovunque si trovi il veicolo (garage, pista, o altro), senza alcuna garanzia che nelle vicinanze sia disponibile una rete Wi-Fi a cui connettersi.

Tramite la modalità SoftAP, il dispositivo stesso crea ed espone la propria rete, rendendo il download possibile ovunque, senza alcuna dipendenza dalla configurazione dell’ambiente dell’utente. Il costo di questa scelta è che, una volta connesso alla rete del dispositivo, lo smartphone perde l’accesso a Internet, ma per un flusso di download di breve durata si tratta di un costo accettabile;

- per `esp_rom_crc`, la scelta di `esp_rom_crc32_le`, rispetto a un’implementazione software del CRC-32, proviene dal fatto che questa funzione risiede nella ROM del chip anziché nell’immagine flash compilata dell’ESP32 [12], il che comporta i seguenti vantaggi:

- **costo nullo in termini di flash:** la memoria flash del dispositivo ha dimensioni limitate (8 MB) e va utilizzata con prudenza. La versione in ROM non comporta alcun costo aggiuntivo in essa;
- **disponibilità immediata all'avvio:** essendo parte della ROM di boot e non del codice residente in flash, la funzione è utilizzabile fin dalle prime fasi di avvio, ancor prima che l'inizializzazione del firmware sia completa;
- **prestazioni:** l'algoritmo di base è lo stesso, ma l'esecuzione da ROM evita la latenza di lettura e i cache-miss legati al recupero di tabelle o codice dalla flash *SPI_G* (Serial Peripheral Interface), un aspetto rilevante su un chip come l'ESP32 [12], dove l'accesso alla flash esterna è più lento rispetto alla memoria ROM/SRAM interna al chip.

4.1.3 Altri componenti

I componenti hardware, già presenti nel progetto e non oggetto di questo tirocinio, sono:

- STM32duino LSM6DSOX [5], accelerometro e giroscopio;
- Adafruit DPS310 [4], barometro;
- Adafruit Ultimate GPS [2], modulo GPS;
- Adafruit MicroSD [3], modulo microSD.

Sul lato applicazione, è stata introdotta una nuova libreria per la visualizzazione 3D del veicolo, sviluppata dall'altro membro del team di tirocinio e non approfondita in questo elaborato:

- react-three-fiber [34]: visualizzazione 3D del veicolo nella schermata Record (§4.2.5.4).

4.2 Progettazione

In questa sezione viene descritta l'architettura generale del sistema, che ne definisce i componenti principali, il modello dati su cui si basano le funzionalità sviluppate e i meccanismi di comunicazione tra applicazione e firmware, approfonditi rispettivamente via BLE e Wi-Fi.

4.2.1 Architettura generale del sistema

Il sistema si articola in due componenti principali, sviluppati separatamente ma strettamente interconnessi. Da una parte l'applicazione mobile, realizzata in React Native [38], dall'altra il firmware del dispositivo embedded, basato su ESP32 [12]. Questi due componenti, come mostrato nella Figura 4.1, comunicano attraverso due canali distinti:

- **BLE:** sempre disponibile e utilizzato per lo scambio di comandi di controllo e per la notifica in tempo reale dello stato del device e della telemetria;
- **Wi-Fi:** attivato su richiesta e riservato per il download delle attività registrate, per cui il BLE offre una banda insufficiente.

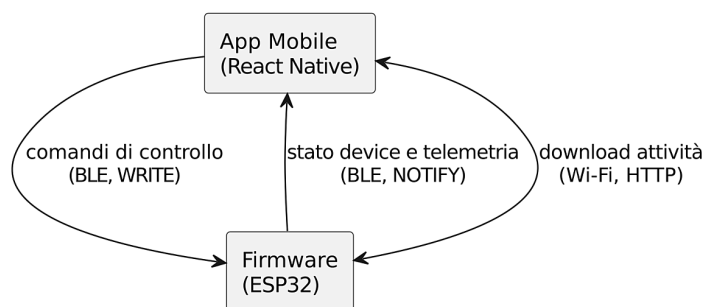


Figura 4.1: Architettura generale del sistema, con i due canali di comunicazione tra applicazione e firmware.

4.2.2 Modello dati

Il modello dati dell'applicazione si articola attorno a tre classi principali, `Device`, `ZipFile` e `Activity`, rappresentate nel diagramma delle classi di Figura 4.2.

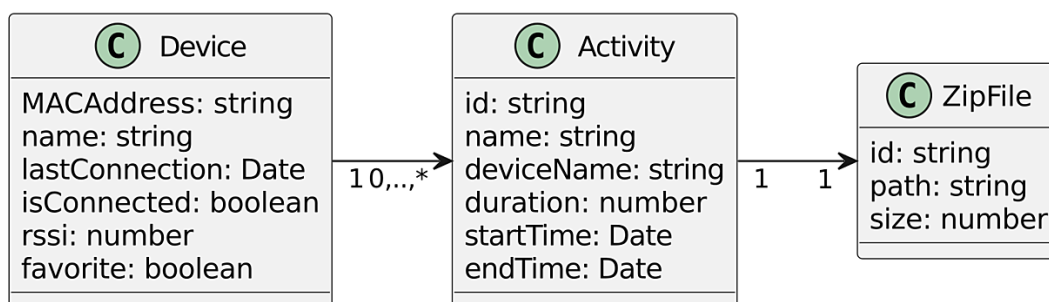


Figura 4.2: Diagramma delle classi del modello dati di dispositivo e attività.

Un `Device` rappresenta un dispositivo con cui l'applicazione ha effettuato il pairing, identificato dal campo `MACAddress`. Questa classe era inizialmente già presente, e a essa sono state aggiunte due funzionalità per l'utente:

- modifica del nome tramite il campo `name`;
- selezione del device come preferito, tramite il campo booleano `favorite`.

`ZipFile` è una nuova classe che rappresenta un riferimento a una particolare sessione scaricata. Ogni istanza contiene i seguenti campi:

- `id`: corrisponde al nome della cartella della sessione salvata su microSD;
- `path`: contiene il percorso che porta al file scaricato;
- `size`: rappresenta la dimensione in byte del file.

`Activity` è una nuova classe che rappresenta una sessione di registrazione già scaricata dal dispositivo. Appena un download viene completato, un'attività viene creata, calcolando i suoi campi (`startTime`, `endTime`, `duration` e `deviceName`) in base all'id del file zip corrispondente. Il nome dell'attività,

invece, viene assegnato di default a "New Activity", e può essere modificato successivamente dall'utente.

A differenza di Device, Activity e ZipFile, lo stato di una sessione di registrazione non viene modellato come entità persistita perché riflette in tempo reale lo stato esposto dal firmware tramite BLE e viene descritto nel dettaglio in §4.2.3.5.

4.2.3 Comunicazione BLE

In questa sezione viene descritta la comunicazione via BLE tra applicazione e firmware, a partire dal ruolo che il protocollo BLE ricopre nell'architettura del sistema, per poi approfondirne la struttura GATT [9] e le funzionalità che si appoggiano a essa (gestione della sessione di registrazione, provisioning Wi-Fi e streaming della telemetria in tempo reale).

4.2.3.1 Ruolo del BLE nell'architettura

Come introdotto in §4.2.1, il BLE costituisce il canale di comunicazione sempre disponibile tra applicazione e firmware (requisito RVO-A1) e svolge tre funzioni distinte:

- controllo del ciclo di vita di un'attività tramite i comandi di avvio, termine e sincronizzazione dello stato descritti in §4.2.3.5;
- provisioning e attivazione della rete Wi-Fi per il successivo download dei dati registrati, descritto in §4.2.3.4;
- trasmissione in tempo reale dei dati di telemetria durante lo svolgimento di un'attività, descritta in §4.2.3.6.

Queste tre funzioni condividono la stessa struttura di comunicazione, basata sul protocollo GATT [9] (§4.2.3.3), ma si differenziano nella natura dello scambio. Le prime due, infatti, sono di tipo comando-risposta, innescate da un'azione dell'utente sull'applicazione, mentre la terza è di tipo continuativo, gestita dal firmware in autonomia una volta attivata.

4.2.3.2 Connessione

Questa funzionalità viene descritta brevemente per fornire il contesto necessario a comprendere quanto segue, ma non viene approfondita nel dettaglio, essendo stata sviluppata durante il precedente ciclo di tirocinio.

L'instaurazione della connessione BLE segue il modello *Central-Peripheral* tipico del profilo *GAP_C* [8] (Generic Access Profile). L'applicazione, nel ruolo di *Central*, avvia una scansione dei dispositivi nelle vicinanze in stato di *advertising*, filtrando i risultati sulla base del nome esposto dal dispositivo, e ne mostra l'elenco all'utente per la selezione.

Una volta selezionato un dispositivo (*Peripheral*), l'applicazione avvia la connessione vera e propria e, una volta stabilita, procede alla scoperta dei Service e delle Characteristic esposti dal dispositivo, descritti nella sottosezione seguente.

4.2.3.3 Struttura GATT

La comunicazione BLE tra applicazione e firmware si basa sul profilo GATT [9], che organizza i dati scambiati in *Service* e *Characteristic*. Il dispositivo assume il ruolo di *GATT Server*, esponendo i dati, mentre l'applicazione assume il ruolo di *GATT Client*, interrogandoli o sottoscrivendovi.

Tra i Service definiti nel firmware, tre risultano centrali ai fini della comunicazione tra applicazione e firmware che ha fatto parte del lavoro di tirocinio:

- **WIFI** raggruppa le Characteristic dedicate al provisioning e al controllo della rete Wi-Fi esposta dal dispositivo in modalità SoftAP, permettendo all'applicazione di configurarne i parametri, avviarne l'attivazione e monitorarne lo stato;
- **SESSION** raggruppa le Characteristic dedicate al controllo del ciclo di vita di un'attività, consentendo all'applicazione di inviare comandi di avvio, termine o ripresa e di ricevere lo stato corrente della sessione;
- un insieme di Characteristic distribuite tra i Service relativi ai singoli sensori (accelerometro, giroscopio e barometro), utilizzate per la trasmissione in tempo reale dei dati di telemetria durante lo svolgimento di un'attività.

Per ciascuna Characteristic sono definite le operazioni GATT [9] supportate: *Write*, con cui l'applicazione invia un comando o un valore di configurazione al firmware, e *Notify*, con cui il firmware invia in modo asincrono un aggiornamento all'applicazione, senza che questa debba interrogarlo esplicitamente. L'uso della notifica risulta particolarmente indicato per i dati che variano nel tempo, come lo stato della connessione Wi-Fi, lo stato della sessione e i valori di telemetria in tempo reale, in quanto evita un'interrogazione continua (*polling*) da parte dell'applicazione.

La Tabella 4.1 riporta l'elenco completo delle Characteristic implementate, organizzate per Service di appartenenza, con il relativo *UUID_G* (Universally Unique Identifier), il tipo di operazione supportata (segnata come Op. nella tabella) e una breve descrizione funzionale.

Tabella 4.1: Service e Characteristic BLE implementati.

Service	Characteristic	UUID	Op.	Descrizione
WIFI (0x1863)	WIFI_SSID	0x3000	Write	<i>SSID_G</i> (Service Set Identifier) da assegnare alla rete SoftAP.
	WIFI_PASS	0x3001	Write	Password da assegnare alla rete SoftAP.
	WIFI_COMMIT	0x3002	Write	Comando di attivazione della rete con i parametri impostati.
Continua nella prossima pagina...				

Service	Characteristic	UUID	Op.	Descrizione
	WIFI_STATUS	0x3003	Notify	Stato corrente della connessione Wi-Fi.
	WIFI_IP	0x3004	Notify	Indirizzo IP assegnato al dispositivo una volta attiva la rete.
	WIFI_DISCONNECT	0x3005	Write	Comando di disattivazione della rete SoftAP.
SESSION (0x1864)	SESSION_COMMAND	0x5001	Write	Comando di avvio, termine o ripresa dell'attività (START / STOP / RESUME).
	SESSION_STATUS	0x5002	Notify	Stato corrente della sessione di attività.
IMU (0x181A)	ACCELEROMETER	0x2C06	Notify	Dati dell'accelerometro in tempo reale.
	GYROSCOPE	0x2C07	Notify	Dati del giroscopio in tempo reale.
Continua nella prossima pagina...				

Service	Characteristic	UUID	Op.	Descrizione
BAROMETER (0x1860)	ALTITUDE	0x2AB3	Notify	Altitudine in tempo reale.
GPS (0x1819)	SPEED	0x2A67	Notify	Velocità in tempo reale.

4.2.3.4 Provisioning e connessione Wi-Fi

Il download di un'attività richiede che il dispositivo esponga una rete Wi-Fi in modalità SoftAP, attivata su richiesta dell'applicazione tramite BLE (requisito RFO-F1). Il Service `WIFI` espone le Characteristic necessarie a configurare, attivare, monitorare e disattivare la rete.

Nell'implementazione sono stati individuati due flussi distinti, che condividono lo stesso Service ma si differenziano per lo scopo e per l'insieme di Characteristic effettivamente coinvolte:

1. Flusso di provisioning iniziale, eseguito la prima volta che un dispositivo viene associato all'applicazione, in cui vengono scritte le Characteristic `WIFI_SSID` e `WIFI_PASS` per definire i parametri della rete che il dispositivo dovrà esporre;
2. Flusso di connessione, eseguito ogni volta che l'utente desidera scaricare un'attività su un dispositivo già configurato, che si appoggia ai parametri di rete già salvati sul firmware e richiede pertanto la sola scrittura della Characteristic `WIFI_COMMIT`.

Flusso di connessione

Il flusso di connessione, rappresentato in Figura 4.3, è composto dai passaggi descritti di seguito.

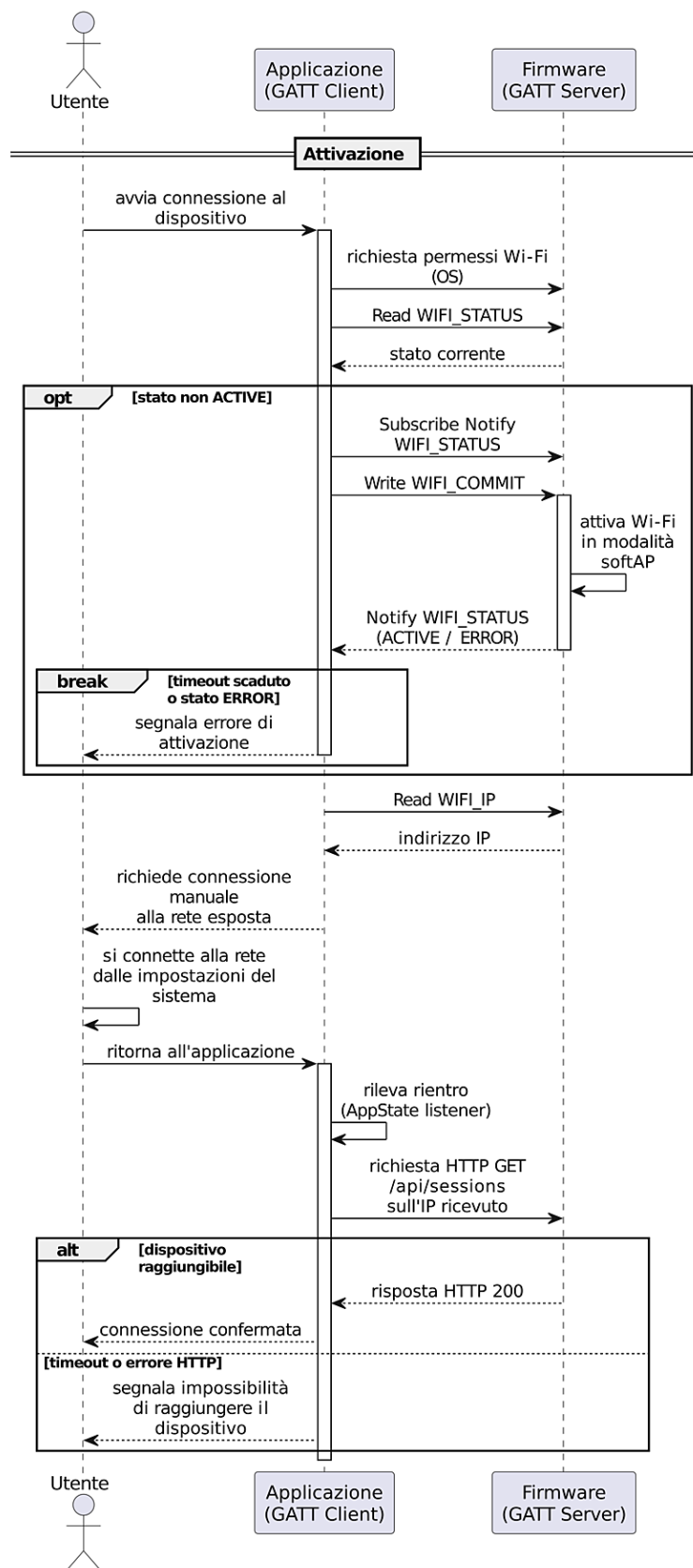


Figura 4.3: Diagramma di sequenza del flusso di connessione alla rete Wi-Fi esposta dal dispositivo.

1. L'applicazione legge il valore corrente di `WIFI_STATUS`:
 - (a) Se la rete risulta già attiva, la richiesta di attivazione viene omessa e si procede direttamente al punto 2b;
 - (b) Altrimenti, l'applicazione si sottoscrive alle notifiche di `WIFI_STATUS` e scrive un comando di attivazione sulla Characteristic `WIFI_COMMIT`.
2. Il firmware attiva il Wi-Fi in modalità SoftAP, utilizzando i parametri di rete precedentemente configurati, e notifica l'esito all'applicazione tramite `WIFI_STATUS`:
 - (a) Se lo stato notificato indica un errore, o se non viene ricevuta alcuna notifica entro un intervallo di tempo prestabilito (15 secondi), l'operazione viene interrotta e l'utente informato;
 - (b) Altrimenti, l'attivazione è avvenuta con successo e l'applicazione legge l'indirizzo IP assegnato tramite `WIFI_IP`.
3. Poiché la connessione alla rete esposta dal dispositivo non può essere stabilita direttamente via BLE, all'utente viene richiesto di connettersi manualmente alla rete dalle impostazioni del sistema operativo. L'applicazione rileva il rientro dell'utente nell'applicazione stessa per riprendere l'esecuzione del flusso;
4. Una volta rilevato il rientro, l'applicazione verifica l'effettiva raggiungibilità del dispositivo inviando una richiesta HTTP diretta all'indirizzo IP ricevuto:
 - (a) Se la risposta del firmware è positiva (HTTP 200), viene confermata la connessione al dispositivo;
 - (b) In caso di errore HTTP o di timeout, l'utente viene notificato.

Flusso di disconnessione

La disattivazione della rete, rappresentata in Figura 4.4, viene avviata in tre casi: l'utente la richiede direttamente, l'attivazione della rete fallisce oppure

la connessione non viene stabilita con successo. Per semplicità, nel diagramma viene rappresentato solo il primo caso, ma tutti e tre seguono lo stesso flusso: l'applicazione scrive la Characteristic `WIFI_DISCONNECT` e il firmware disattiva l'interfaccia Wi-Fi.

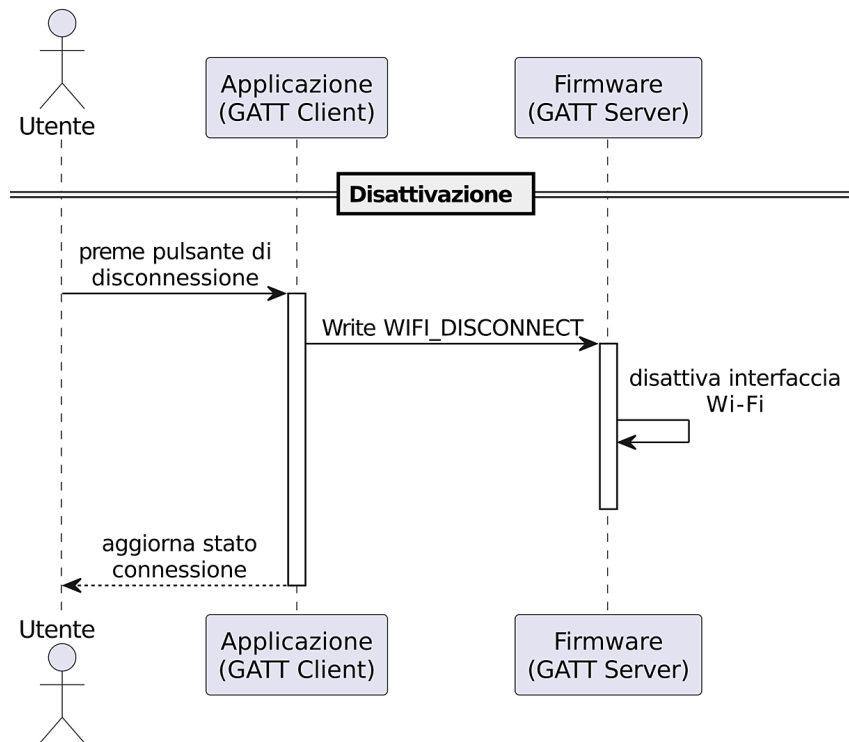


Figura 4.4: Diagramma di sequenza del flusso di disconnessione alla rete Wi-Fi esposta dal dispositivo.

4.2.3.5 Gestione sessione (START / STOP / RESUME)

Il Service `SESSION` consente all'applicazione di controllare il ciclo di vita di un'attività di registrazione sul dispositivo, attraverso la Characteristic `SESSION_COMMAND` (Write) e di riceverne lo stato aggiornato tramite `SESSION_STATUS` (Notify). I comandi supportati sono `START`, `STOP` e `RESUME`, corrispondenti rispettivamente all'avvio, al termine e alla sincronizzazione dello stato dopo una riconnessione al dispositivo.

Il flusso di avvio (requisito RFO-A3), rappresentato in Figura 4.5, prevede che l'applicazione si sottoscriva alle notifiche di `SESSION_STATUS` prima di inviare il comando `START`, in modo da non perdere la notifica di conferma. Alla

ricezione del comando, il firmware verifica innanzitutto che il dispositivo non sia occupato in un'altra operazione, come un download o l'eliminazione di un'attività, dato che queste funzionalità condividono lo stesso stato del dispositivo e risultano pertanto mutualmente esclusive. Se invece il dispositivo è libero, il firmware predispone la sessione di registrazione e notifica l'applicazione del corretto avvio.

Una volta avviata l'attività, l'applicazione continua a ricevere aggiornamenti periodici tramite `SESSION_STATUS`, indipendentemente da ulteriori comandi, contenenti la durata trascorsa e il numero di campioni acquisiti.

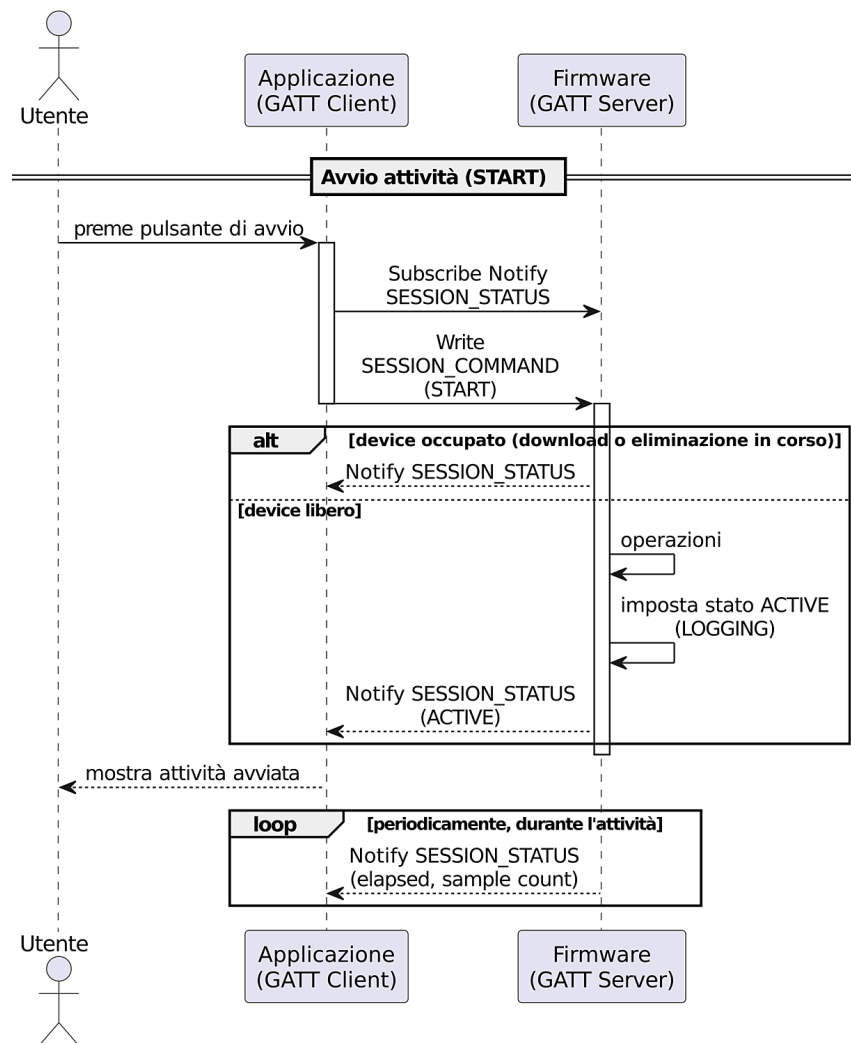


Figura 4.5: Diagramma di sequenza del controllo della sessione di attività (avvio).

Il flusso di termine (requisito RFO-A5), rappresentato in Figura 4.6, è simmetrico: l'applicazione invia il comando `STOP` e il firmware, una volta che lo

riceve, conclude la registrazione, finalizza i dati raccolti sul dispositivo e notifica lo stato aggiornato. Infine, l'applicazione annulla la propria sottoscrizione alle notifiche.

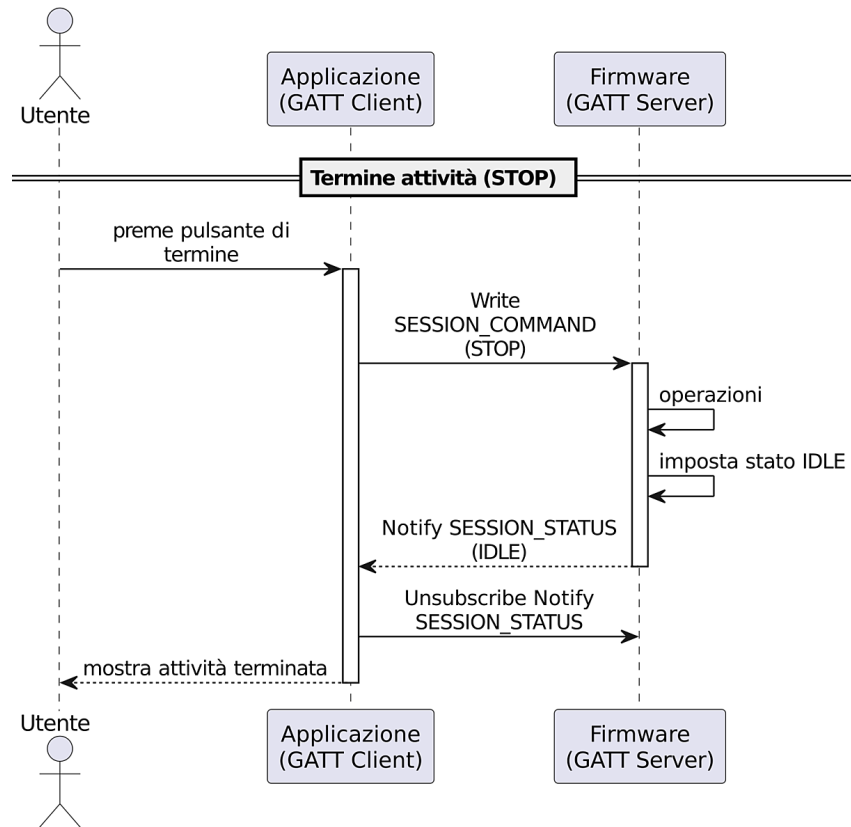


Figura 4.6: Diagramma di sequenza del controllo della sessione di attività (termine).

Il comando **RESUME**, rappresentato in Figura 4.7, non riavvia né modifica l'attività in corso. Si tratta di una richiesta di sincronizzazione dello stato, utilizzata dall'applicazione per allineare la propria interfaccia allo stato reale del dispositivo dopo aver perso e ristabilito la connessione BLE (requisito RND-A1), garantendo la continuità della visualizzazione (requisiti RFO-A16, RFO-A17) senza necessità di riavviare la registrazione dall'inizio.

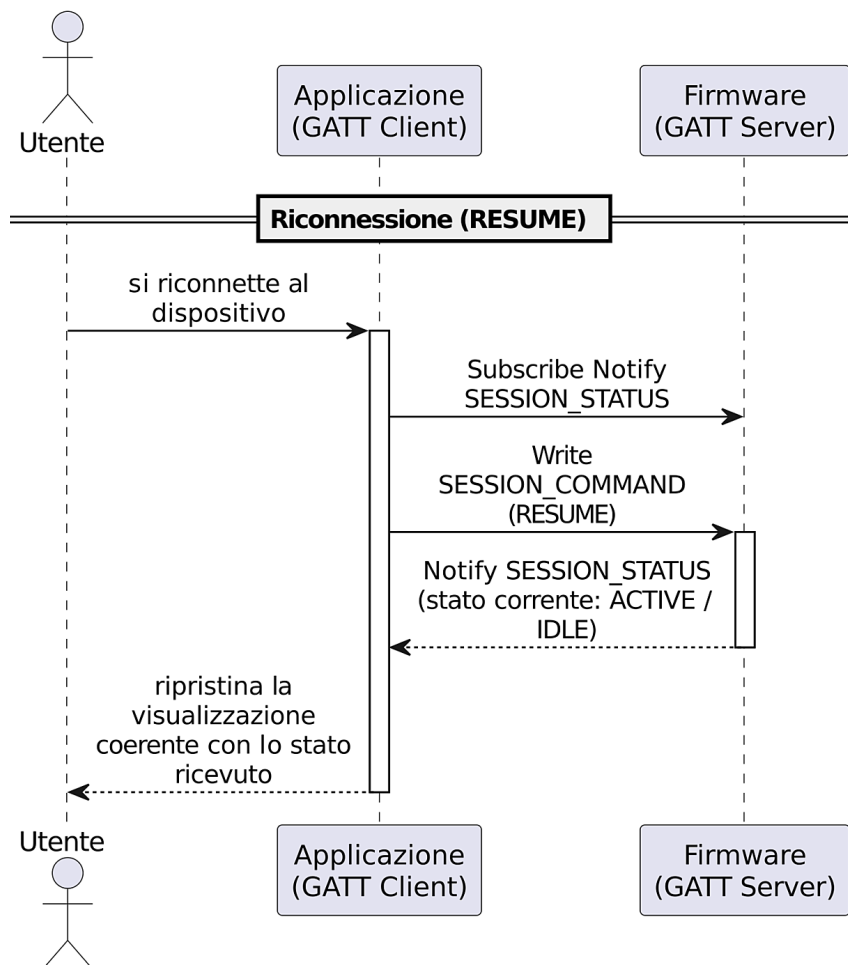


Figura 4.7: Diagramma di sequenza del controllo della sessione di attività (riconnessione).

4.2.3.6 Streaming telemetria in tempo reale

Durante lo svolgimento di un'attività, l'applicazione si sottoscrive alle notifiche di un insieme di Characteristic distribuite tra i Service dei singoli sensori: accelerometro e giroscopio (Service IMU), velocità (Service GPS) e altitudine (Service BAROMETER). Una volta conclusa la sessione, la sottoscrizione viene annullata automaticamente. A differenza del controllo di sessione e del provisioning Wi-Fi, di tipo comando-risposta, questo scambio è di tipo continuativo: una volta stabilita la sottoscrizione, il firmware notifica autonomamente ogni nuovo campione, senza ulteriori richieste da parte dell'applicazione.

La generazione delle notifiche sul lato firmware non viene approfondita in questa sede in quanto non è stata oggetto del tirocinio. In §4.3, si può comunque trovare la descrizione del design pattern *Producer-Consumer*, strettamente collegato a questo ambito.

Alla ricezione di ciascuna notifica, l'applicazione ne risolve il tipo di dato a partire dall'UUID della Characteristic di provenienza e decodifica il valore secondo il formato previsto (valore singolo o vettoriale a seconda del sensore), per poi renderlo disponibile alla schermata di visualizzazione in tempo reale.

4.2.4 Download via Wi-Fi

Una volta stabilita la connessione alla rete Wi-Fi esposta dal dispositivo (§4.2.3.4), l'applicazione comunica con un webserver HTTP esposto dal firmware sulla stessa rete per recuperare in locale i dati di un'attività registrata. L'intero scambio avviene su questo canale (requisito RVO-A2) anziché su BLE, che non offrirebbe banda sufficiente per il volume di dati coinvolto.

Questa funzionalità viene approfondita sia sul lato firmware che sul lato applicazione, visto che sono state sviluppate entrambe.

4.2.4.1 Download lato firmware

Sul lato firmware, come si può osservare nella Figura 4.8, il flusso segue i passaggi descritti di seguito.

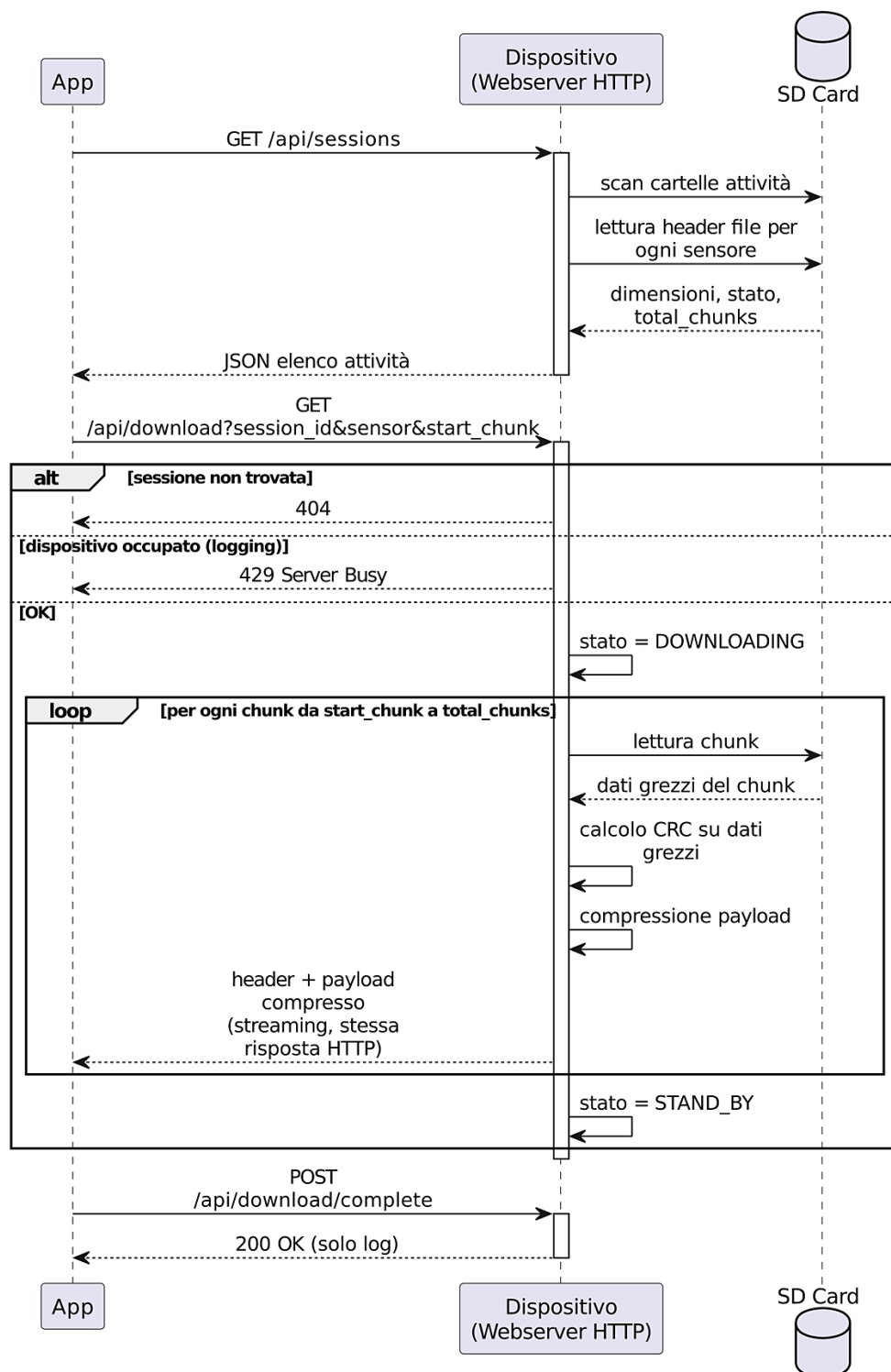


Figura 4.8: Diagramma di sequenza del flusso di download lato firmware.

1. Ricezione richiesta dell'elenco delle attività disponibili

Alla ricezione di una richiesta GET /api/sessions sul webserver HTTP, il firmware avvia la scansione delle attività registrate presenti sulla scheda microSD, rispondendo con l'elenco richiesto dall'applicazione (requisito RFO-F2).

2. Interrogazione della memoria microSD

Nella scheda microSD le attività vengono salvate in cartelle, ciascuna contenente un file binario per ogni sensore coinvolto, preceduto da un header che ne riporta, tra le altre informazioni, il numero di record validi presenti. Il firmware scandisce le cartelle presenti sulla microSD e, per ciascuna, legge gli header dei rispettivi file binari per calcolarne la dimensione complessiva. Da questa dimensione ricava anche il numero di chunk in cui i dati verranno suddivisi in fase di download, dividendo il numero di record per la capacità, in numero di record, di un chunk di dimensione prefissata (16 KB). Il firmware determina, inoltre, lo stato dell'attività (completata, interrotta o ancora in corso), informazione che l'applicazione userà per decidere quali attività proporre all'utente. Infine, invia all'applicazione un file *JSON_G* (JavaScript Object Notation) con l'elenco delle attività trovate.

3. Creazione dei pacchetti con CRC e compressione

Su richiesta di download di un sensore (GET /api/download, con identificativo di attività, sensore e indice del chunk richiesto), il firmware costruisce i chunk richiesti a partire da quell'indice. Ogni chunk è formato da un header a dimensione fissa (16 B) contenente:

- identificativo del chunk;
- numero totale di chunk;
- lunghezza del payload;
- codice CRC (requisito RFO-F5).

In particolare, il codice CRC viene calcolato sui dati grezzi (prima della compressione), in modo da permettere all'applicazione di validare con un unico controllo l'intera catena di operazioni. Questo header viene seguito dal payload compresso (requisito RFO-F3).

Per tutta la durata del trasferimento, il dispositivo entra nello stato **DOWNLOADING**, che ne impedisce l'uso per altre operazioni. Infatti, se una richiesta di download arriva mentre è già in corso una registrazione, il firmware la rifiuta segnalando che il dispositivo è occupato, coerentemente con la mutua esclusione già discussa per la gestione della sessione (§4.2.3.5).

4. Invio dei pacchetti all'applicazione

I chunk vengono trasmessi in streaming continuo sulla stessa risposta HTTP, senza che il firmware mantenga alcuno stato relativo al download tra una richiesta e l'altra (requisito RFO-F4). Per questo motivo, il firmware è in grado di rispondere a partire da un punto qualsiasi della sequenza, il che consente all'applicazione di riprendere un download interrotto specificando semplicemente sensore e indice di chunk da cui ripartire, senza dover riscaricare l'intera attività (requisito RFO-F6).

4.2.4.2 Download sul lato applicazione

Sul lato applicazione, come si può osservare nella Figura 4.9, il flusso segue i passaggi descritti di seguito.

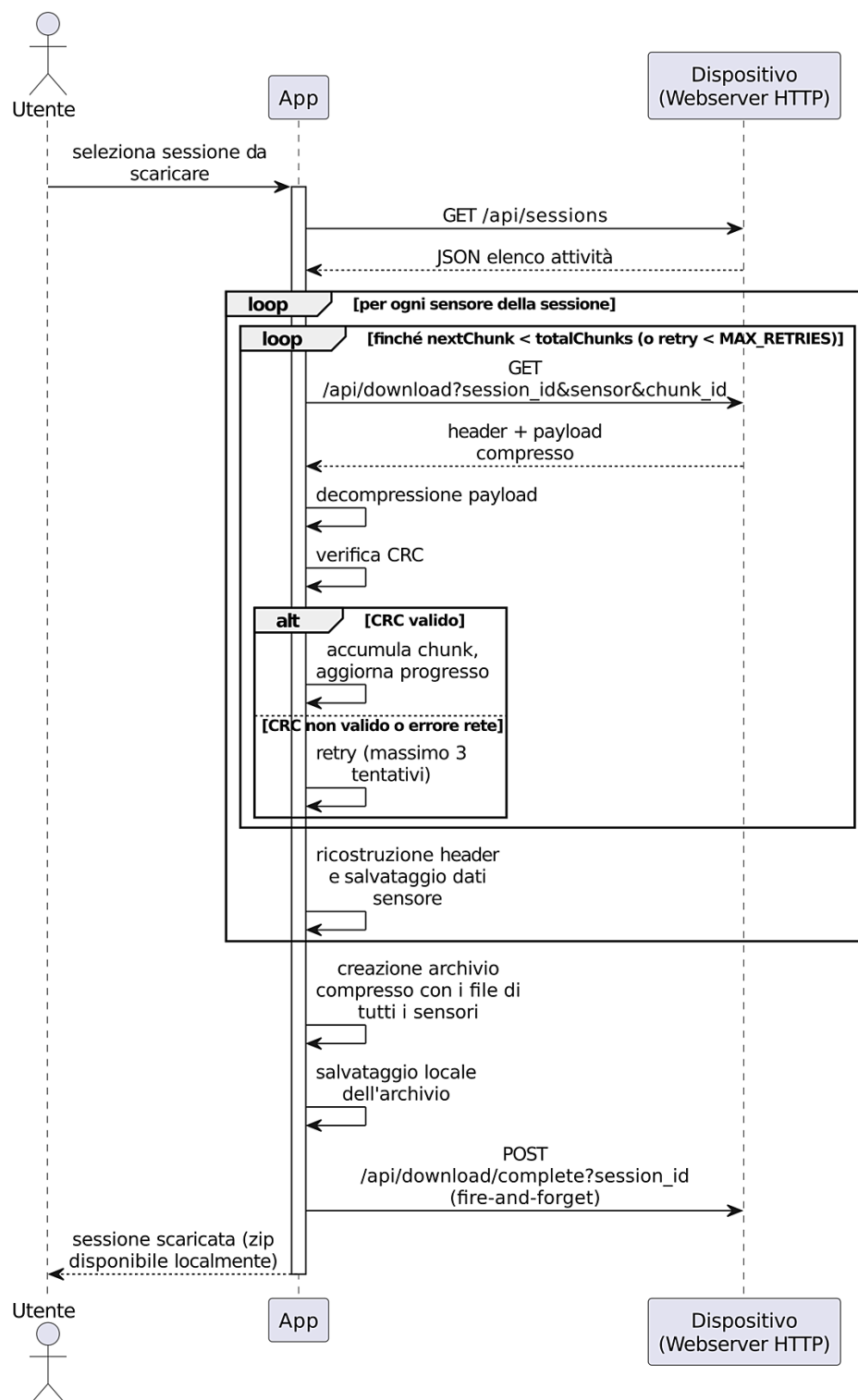


Figura 4.9: Diagramma di sequenza del flusso di download lato applicazione.

1. Richiesta dell'elenco delle attività disponibili

Una volta stabilita la connessione alla rete Wi-Fi del dispositivo, l'applicazione richiede l'elenco delle attività scaricabili tramite GET /api/sessions (requisito RFO-A10), escludendo dalla visualizzazione le attività ancora in corso o interrotte in base allo stato ricevuto.

2. Download dei dati sensore per sensore

Per l'attività selezionata, l'applicazione avvia il download richiedendo i dati di ciascun sensore coinvolto uno alla volta (requisito RFO-A11), anziché come unico flusso esteso a tutta l'attività. Questa scelta semplifica sia la logica di trasmissione sul lato firmware, che non deve gestire la transizione da un sensore al successivo a metà di un chunk, sia quella di ripresa sul lato applicazione, per la quale il punto di interruzione è identificato dalla semplice coppia sensore/indice di chunk. Per ogni sensore l'applicazione effettua una serie di richieste GET /api/download, specificando l'indice del chunk da cui proseguire. Durante l'intero processo tiene traccia dell'avanzamento complessivo, sommando i chunk ricevuti su tutti i sensori, e lo mostra all'utente in tempo reale, segnalando eventuali problemi riscontrati durante il trasferimento (requisito RFO-A12).

3. Verifica e decompressione

Ogni chunk ricevuto viene innanzitutto decompresso (requisito RFD-A1), dato che il payload viene trasmesso in forma compressa per ridurre la quantità di dati scambiati sulla rete. Il dato decompresso viene poi confrontato, tramite il proprio CRC calcolato sul lato applicazione, con quello fornito dal firmware nell'header del chunk. Una mancata corrispondenza o un errore di rete verificatosi durante la richiesta comporta la ritrasmissione automatica del chunk interessato, ripetuta fino a un numero massimo di 3 tentativi, oltre i quali il download viene interrotto e l'utente informato (requisito RFO-A14). Applicare questa verifica a ogni singolo chunk, anziché all'intero file scaricato, consente di individuare e correggere un errore di trasmissione localmente, senza dover ripe-

tere l'intero download del sensore coinvolto, e rende possibile la ripresa puntuale di un download interrotto (requisito RFO-A15).

4. Salvataggio locale

Una volta scaricati e verificati tutti i chunk di un sensore, i relativi payload vengono concatenati a formare il file dei dati grezzi di quel sensore, a cui viene anteposto un header ricostruito sul lato applicazione per renderlo compatibile con il formato usato dal resto del sistema. Completato il download di tutti i sensori dell'attività, i file ottenuti vengono raccolti in un unico archivio compresso, salvato in locale sul dispositivo mobile (requisito RFO-A13). L'applicazione notifica infine al firmware, tramite POST /api/download/complete, il completamento dell'operazione, una notifica priva di effetti sullo stato del dispositivo, usata solo a scopo di log.

4.2.5 UI e navigazione

In questa sezione viene illustrato lo stato finale dell'applicazione, seguendo il percorso tipico dell'utente attraverso di essa e mostrando, nei casi più significativi, l'evoluzione rispetto allo stato iniziale.

Oltre alle modifiche funzionali e strutturali descritte nel seguito, l'interfaccia ha subito anche un aggiornamento stilistico (palette cromatica, tipografia, iconografia) volto a rendere l'interfaccia più ordinata e di più immediata comprensione rispetto alla versione preesistente. Questi cambiamenti stilistici potranno essere notati tramite le figure presenti in questa sezione, ma non saranno descritti in dettaglio in quanto riguardano un aspetto secondario rispetto agli obiettivi del tirocinio.

4.2.5.1 Menù

La prima versione dell'applicazione era organizzata in un numero limitato di schermate, sufficienti a coprire il pairing con il dispositivo, la visualizzazione dei dati in tempo reale e il download del file zip dei dati registrati. Nel corso del tirocinio la struttura di navigazione è stata estesa per accogliere nuove funzio-

nalità, come la visualizzazione dello storico e il download delle attività, avendo come risultato la struttura a tab mostrata nella Figura 4.10.

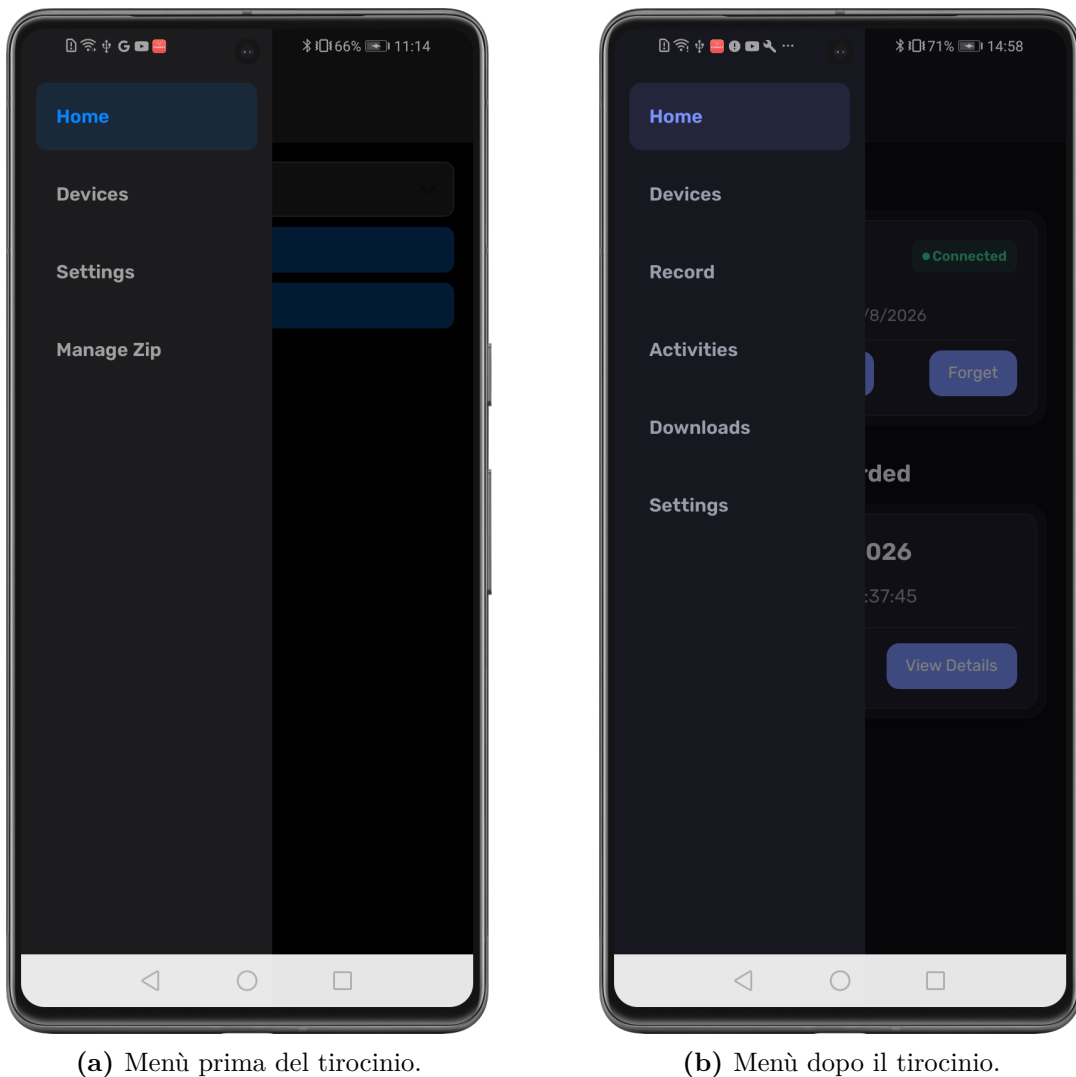


Figura 4.10: Evoluzione della struttura di navigazione a tab.

Come si può osservare, rispetto alla versione precedente sono state aggiunte le schermate Record e Activities. La schermata Manage Zip, invece, è stata completamente ristrutturata e rinominata Downloads. Visto che lo scopo finale, il download di un'attività, non è cambiato tra le due versioni, esse verranno successivamente approfondite insieme.

La schermata Settings, invece, non ha subito modifiche sostanziali nel corso del tirocinio e non viene pertanto ulteriormente descritta nel presente elaborato.

4.2.5.2 Home

La schermata Home, mostrata a confronto con la sua versione precedente nella Figura 4.11, ha cambiato ruolo rispetto alla prima versione. La funzionalità di registrazione di una nuova attività, infatti, è stata spostata nella nuova schermata Record, mentre Home è diventata una semplice dashboard che mostra la lista dei dispositivi salvati come preferiti (funzionalità approfondita nella prossima sezione) e i dati riassuntivi dell'ultima attività registrata (requisito RFO-A1).

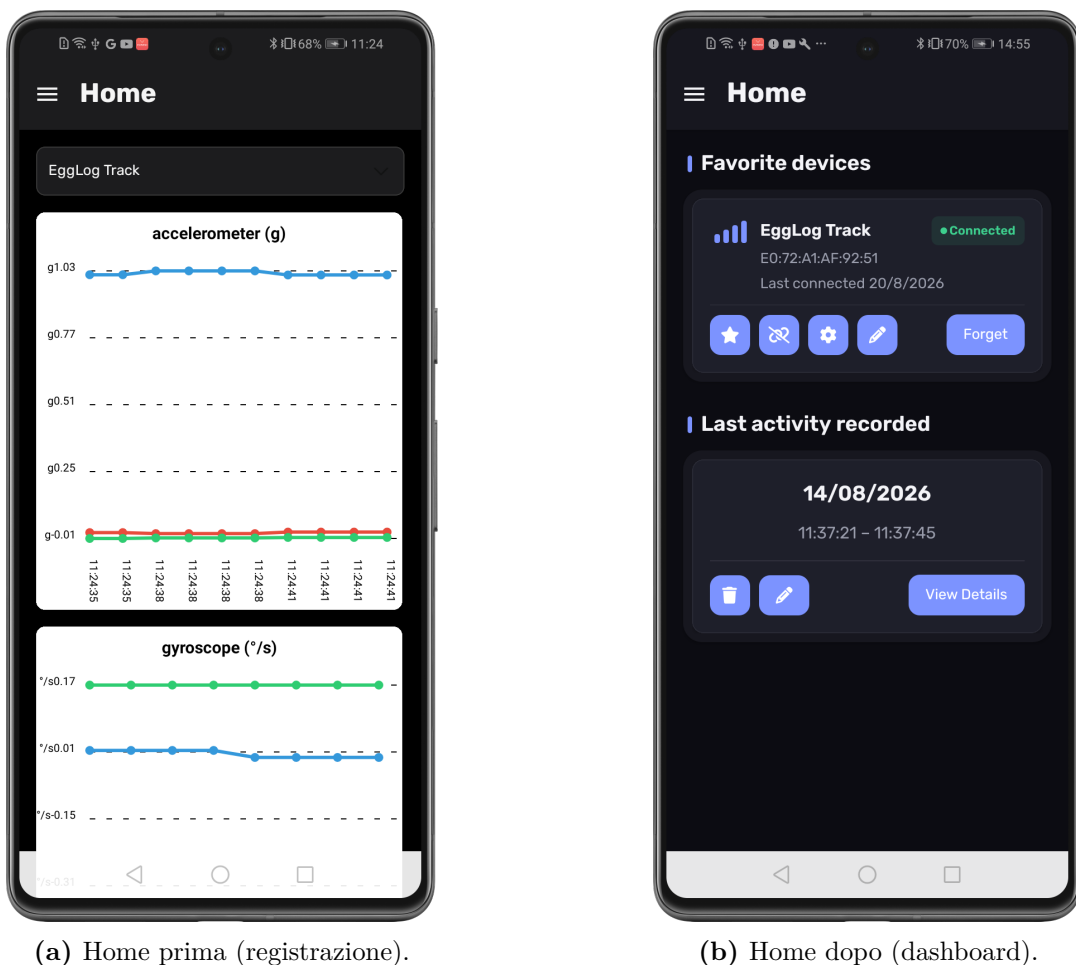
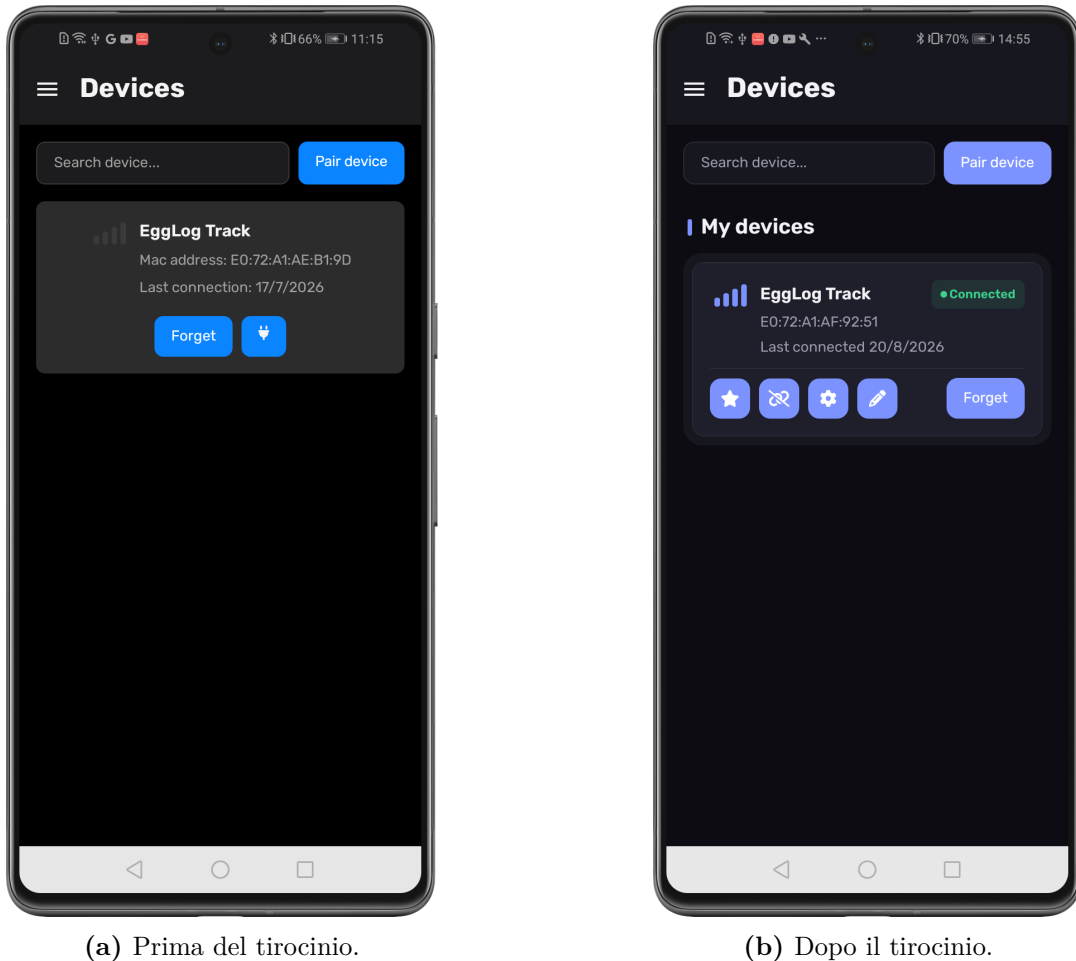


Figura 4.11: Ristrutturazione della schermata Home, diventata una dashboard.

4.2.5.3 Devices

Alla schermata Devices, già presente per la gestione del pairing, sono state aggiunte per ciascun dispositivo le funzionalità di modifica del nome e di

selezione come preferito, come mostrato nella Figura 4.12. L'implementazione consente di contrassegnare più dispositivi come preferiti contemporaneamente (si veda la Figura 4.2 per il relativo modello dati).



(a) Prima del tirocinio.

(b) Dopo il tirocinio.

Figura 4.12: Evoluzione della schermata Devices, con l'aggiunta della modifica del nome e della selezione come preferito.

4.2.5.4 Record

La schermata Record eredita la funzionalità di selezione del dispositivo su cui registrare dalla schermata Home preesistente e propone una gestione diversa della registrazione (spiegata in dettaglio in §4.2.3.5).

La coppia di Figure 4.13 mostra come viene visualizzata la schermata Record prima e durante l'avvio di una registrazione.

La Figura 4.13a mostra lo stato iniziale della schermata (requisito RFO-A2), che prevede:

- selezione del device da cui registrare;
- pulsante per la calibrazione del device (questa funzionalità è stata sviluppata da un altro membro del team di tirocinio, quindi non sarà approfondita in questo testo);
- pulsante per l'avvio di una sessione.

La Figura 4.13b, invece, mostra la visualizzazione di una sessione in corso (requisito RFO-A4), mantenuta anche durante la navigazione tra le schermate (requisito RFO-A16), che comprende:

- tempo trascorso;
- pulsante per il termine della registrazione;
- dati come la velocità attuale, l'altitudine, l'inclinazione laterale e frontale, e l'accelerazione;
- la visualizzazione in 3D del veicolo (questa funzionalità è stata sviluppata dall'altro membro del team di tirocinio, quindi non sarà approfondita in questo testo).

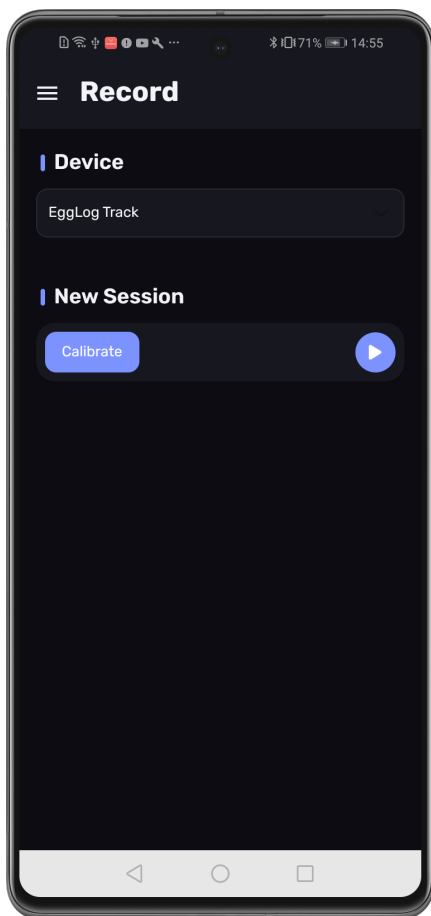
La coppia di Figure 4.14 mostra come viene visualizzata la schermata Record in caso di disconnessione e riconnessione dal device (requisito RFO-A17).

La Figura 4.14a mostra la gestione di una disconnessione. Se la connessione tramite BLE viene persa, l'applicazione non riceve più dati dal device, ma questo non termina la registrazione. L'utente viene informato di questa disconnessione tramite un messaggio.

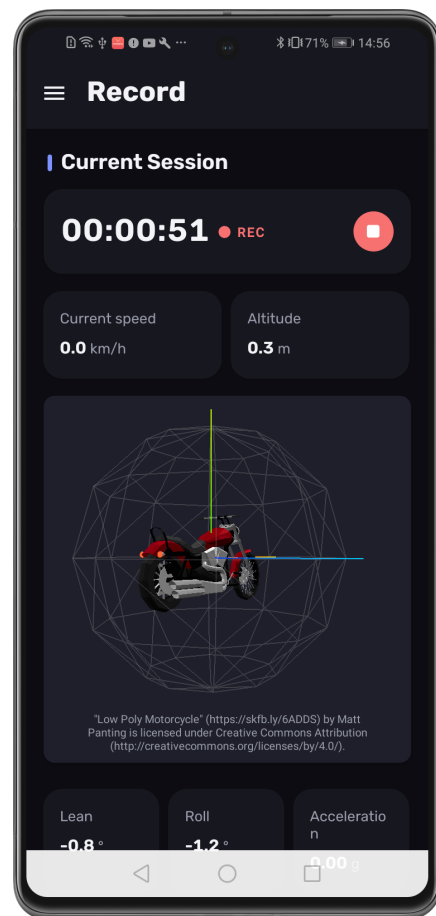
Nella Figura 4.14b si può osservare cosa vede l'utente quando si riconnette al device tramite BLE. Tramite il pulsante presente può ripristinare la visualizzazione della registrazione ancora in corso (Figura 4.13b).

4.2.5.5 Downloads

La schermata Downloads (denominata Manage Zip nella versione preesistente) dà all'utente la possibilità di avviare l'esposizione della rete Wi-Fi del device.

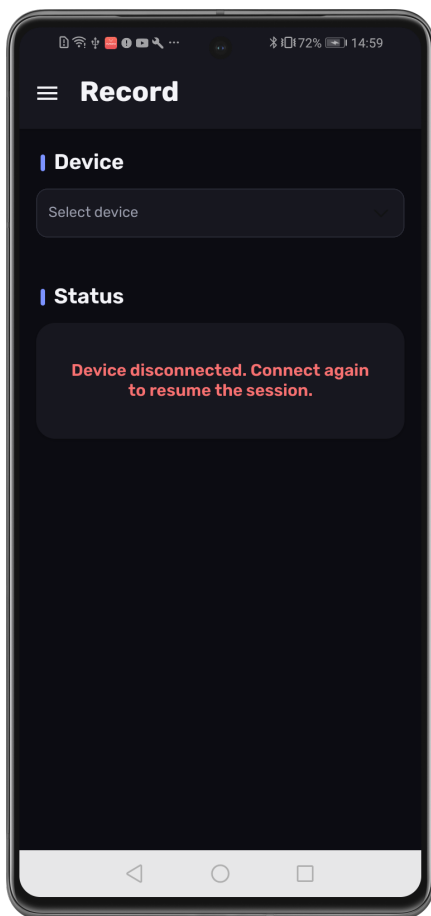


(a) Schermata Record prima di una registrazione.

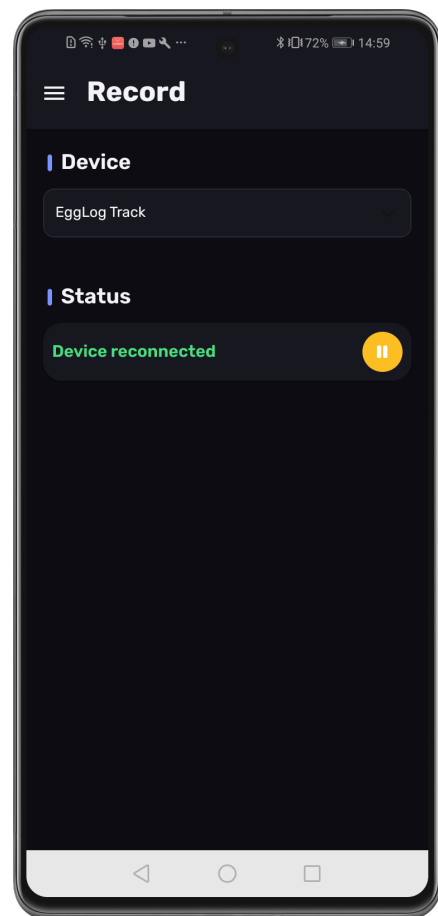


(b) Schermata Record durante una registrazione.

Figura 4.13: UI principale della schermata Record.



(a) Schermata Record in caso di disconnessione dal device.

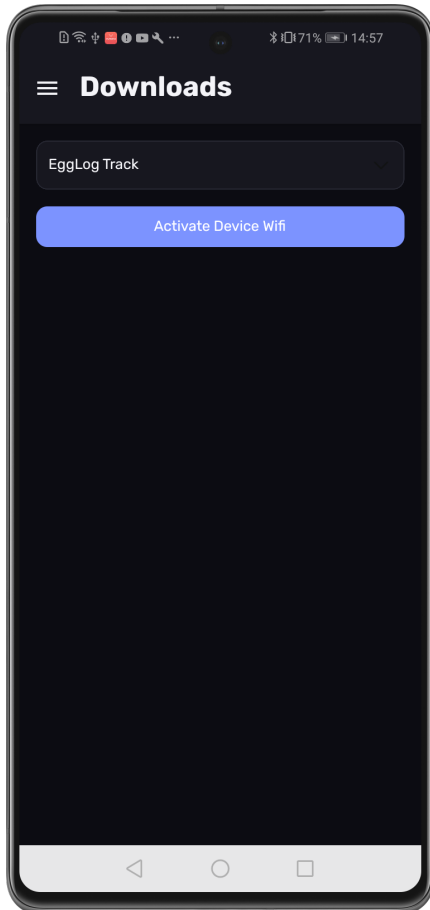


(b) Schermata Record in caso di riconnessione al device.

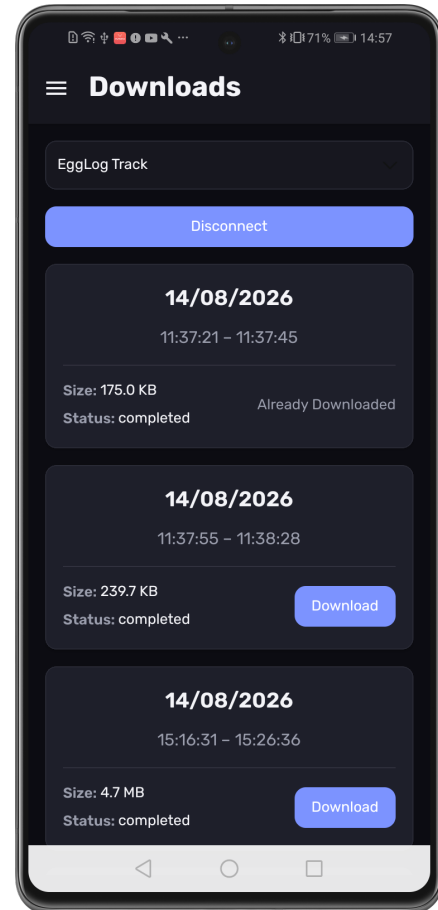
Figura 4.14: Schermata Record per la gestione della funzionalità di resume.

Una volta che si connette a essa, viene mostrata la lista delle attività disponibili per il download in locale (requisiti RFO-A8, RFO-A9, RFO-A10).

La coppia di Figure 4.15 mostra la schermata Downloads prima e dopo la connessione al device tramite Wi-Fi.



(a) Schermata Downloads prima della connessione tramite Wi-Fi.



(b) Schermata Downloads durante la connessione tramite Wi-Fi.

Figura 4.15: Schermata Downloads.

4.2.5.6 Activities

La schermata Activities, mostrata nella Figura 4.16, presenta l'elenco delle attività scaricate (requisiti RFO-A6, RFD-A2), ciascuna corredata di tre pulsanti: il primo per la rimozione dell'attività stessa, il secondo per la modifica del suo nome e il terzo per accedere alla schermata di dettaglio (requisito RFO-A7).

Quest'ultima non è stata implementata nell'ambito di questo tirocinio. Il pulsante corrispondente è pertanto presente in interfaccia ma non ancora fun-

zionale.

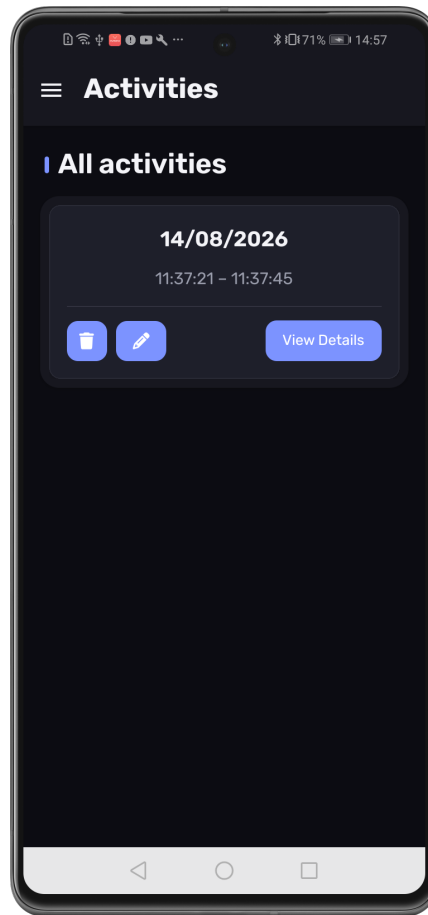


Figura 4.16: Schermata Activities, con elenco delle attività scaricate.

4.3 Design Pattern

Trattandosi dell'estensione di un progetto già avviato nei cicli di tirocinio precedenti, i pattern di progettazione descritti nel seguito non sono tutti stati introdotti in questo lavoro: per ciascuno viene pertanto precisato se si tratta di una struttura preesistente riutilizzata oppure di un pattern applicato direttamente nel codice sviluppato durante il tirocinio.

Anche in questo caso viene effettuata una suddivisione tra firmware e applicazione.

4.3.1 Design Pattern lato firmware

4.3.1.1 State Machine

La gestione dello stato del dispositivo è stata implementata dall'altro stagista e regola le transizioni tra i sei stati operativi mutuamente esclusivi del dispositivo (STAND_BY, LOGGING, SUSPENDED, DOWNLOADING, DELETING, ERROR), riportati in dettaglio nella Tabella 4.2. Il lavoro di tirocinio ha operato all'interno di questa struttura esistente, in particolare per la gestione della mutua esclusione tra registrazione e download di un'attività (§4.2.3.5, §4.2.4), rappresentata in tabella dall'impossibilità di transitare verso DOWNLOADING o DELETING se non a partire da STAND_BY.

Tabella 4.2: Transizioni dello stato globale del dispositivo.

Stato attuale	Stato successivo	Evento
STAND_BY	LOGGING	Comando START
STAND_BY	DOWNLOADING	Richiesta di download di una sessione
STAND_BY	DELETING	Richiesta di cancellazione di una o più sessioni
LOGGING	STAND_BY	Comando STOP
LOGGING	SUSPENDED	Inattività prolungata (oltre la soglia configurabile)
SUSPENDED	LOGGING	Ripresa del movimento
SUSPENDED	STAND_BY	Comando STOP
DOWNLOADING	STAND_BY	Download completato
DELETING	STAND_BY	Cancellazione completata
Qualsiasi stato	ERROR	microSD mancante o errore generico
ERROR	STAND_BY	Reinserimento microSD o reset

4.3.1.2 Producer-Consumer

Il pattern Producer-Consumer [6] era già adottato nel firmware prima dell'inizio del tirocinio, ed è stato esteso dall'altro stagista, sia per l'acquisizione dei dati dai sensori verso la scheda microSD, sia per la loro distribuzione verso l'applicazione via BLE: ciascun sensore accumula i propri campioni in un buffer dedicato a doppio slot (Producer), alternando la scrittura tra i due per garantire continuità di acquisizione durante il flush. Al riempimento del buffer attivo (o allo scadere di un timeout configurabile), il contenuto viene inoltrato tramite un'unica coda condivisa verso una task dedicata alla scrittura su microSD (Consumer). Lo stesso pattern regola anche la distribuzione dei dati in tempo reale verso l'applicazione: qui i Consumer sono due task distinte, con priorità differenziate tra i dati IMU e quelli degli altri sensori, che notificano i valori tramite le rispettive Characteristic (§4.2.3.6). La Figura 4.17 mostra schematicamente le due pipeline appena descritte, evidenziando la coda condivisa verso la scheda microSD e le due code separate, distinte per priorità, verso le rispettive Characteristic BLE.

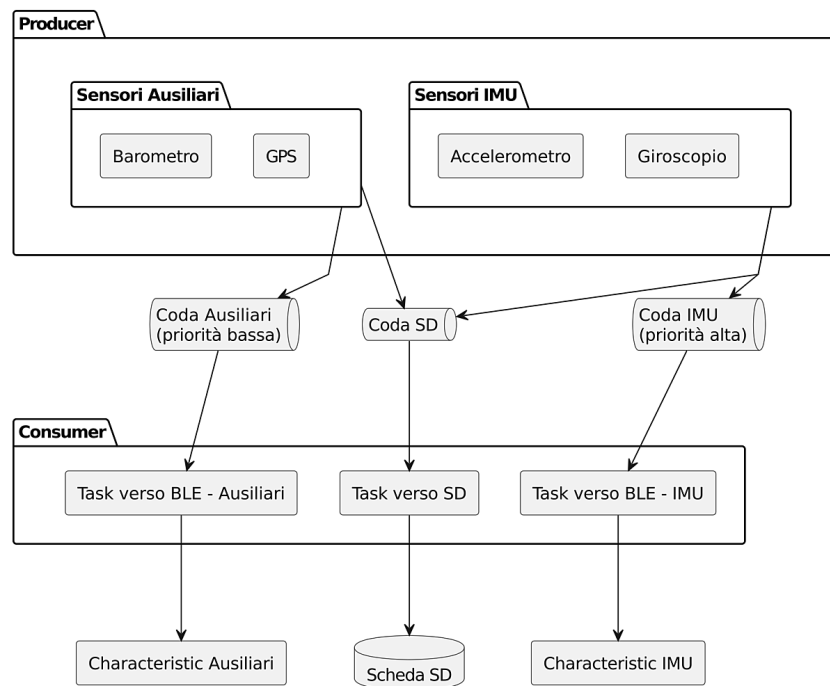


Figura 4.17: Diagramma design pattern Producer-Consumer.

4.3.1.3 Command [36]

Sul lato firmware, viene applicato il pattern Command [36]. Infatti, la ricezione dei comandi tramite Characteristic BLE segue una convenzione già stabilita nel progetto, imposta anche dall'*API_G* (Application Programming Interface) della libreria NimBLE-Arduino [18]. Ogni comando è gestito da una classe dedicata che eredita da `NimBLECharacteristicCallbacks` e ne implementa il metodo `onWrite`, incapsulando così la logica di un singolo comando (ad esempio `SessionCommandCallback`, già presente per il controllo della sessione). Nell'ambito di questo tirocinio tale convenzione è stata seguita per l'implementazione dei comandi relativi al provisioning Wi-Fi (`WiFiCommitCallback` e `WiFiDisconnectCallback`, §4.2.3.4).

4.3.2 Design Pattern sul lato applicazione

4.3.2.1 Repository

Il pattern Repository [27] era già utilizzato nel progetto per disaccoppiare la logica applicativa dalla persistenza dei dati, tramite interfacce dedicate (`logic/repository`) e relative implementazioni concrete (`logic/persistence`), basate su `AsyncStorage` [35] e sul filesystem locale. Nell'ambito di questo tirocinio tale struttura è stata riutilizzata ed estesa per la gestione dell'entità Attività (`ActivityRepository`, `ActivityPersistence`), introdotta insieme al relativo modello dati (§4.2.2).

4.3.2.2 MVVM

L'applicazione segue un'architettura MVVM (*Model-View-ViewModel*) [30], come mostrato in Figura 4.18. Ogni schermata (`app/`) usa uno o più `ViewModel` (`app/viewmodel`), ognuno dei quali è incaricato di orchestrare un insieme di servizi applicativi (`logic/services`), a loro volta responsabili della logica di comunicazione con il dispositivo. Questi servizi, infine, utilizzano le repository (`logic/repository`) per la gestione dei dati. Il flusso inverso segue la stessa catena in senso opposto: i dati letti dalle repository o gli aggiornamenti

di stato ricevuti dai servizi vengono propagati verso l'alto fino al ViewModel, che aggiorna reattivamente la schermata corrispondente. Service e Repository costituiscono insieme il Model dell'architettura MVVM. Anche questa struttura era già consolidata nel progetto. Le funzionalità sviluppate in questo tirocinio sono state realizzate seguendone le convenzioni, tramite l'aggiunta di nuovi ViewModel e servizi dedicati coerenti con quelli preesistenti.

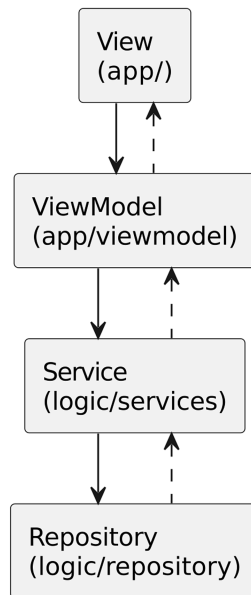


Figura 4.18: Diagramma design pattern MVVM.

4.3.2.3 Dependency Injection

I servizi applicativi vengono istanziati una sola volta e resi disponibili ai componenti tramite il pattern Dependency Injection [26] basato sul Context di React (`context/service.context.tsx`). Ciascun servizio, infatti, riceve le proprie dipendenze nel costruttore, mentre i ViewModel li ottengono tramite l'hook `useServices()`, senza istanziarli direttamente. Questa struttura era già presente nel progetto e nell'ambito del tirocinio è stata estesa per la registrazione dei servizi introdotti.

4.3.3 Design Pattern misti

4.3.3.1 Observer

Il pattern Observer [37] è stato impiegato secondo lo schema classico *Subject/Observer*. Sul firmware, la Characteristic BLE assume il ruolo di *Subject*. Il meccanismo di Notify previsto dal protocollo GATT (§4.2.3.3) permette alle Characteristic di notificare i propri *Observer* (l'applicazione sottoscritta) senza richiedere interrogazione continua da parte di quest'ultima. Questo meccanismo era già disponibile nel progetto ed è stato riutilizzato per le nuove Characteristic introdotte (WIFI_STATUS e WIFI_IP).

4.4 Sviluppo

Lo sviluppo del progetto è stato condotto interamente in Visual Studio Code, sia per la componente firmware che per quella applicativa. Per il firmware è stato utilizzato il linguaggio C++ [16], tramite l'estensione **PlatformIO** [33], che ne ha gestito compilazione, upload e debug sulla scheda ESP32 [12]. Per l'applicazione mobile, invece, è stato utilizzato il linguaggio TypeScript [29], tramite gli strumenti **Expo** [22] e **Metro** [28] per lo sviluppo e il bundling in ambiente React Native [38]. Il versionamento del codice è stato gestito tramite repository su **Bitbucket** [7]

Per gran parte delle funzionalità, lo sviluppo di app e firmware è proceduto in parallelo, poiché molte di esse coinvolgevano entrambi i lati del sistema ed erano verificabili solo disponendo contemporaneamente della relativa controparte (ad esempio, la comunicazione BLE o il download via Wi-Fi). Un numero più limitato di funzionalità è stato invece sviluppato separatamente sui due lati. In entrambi i casi, l'approccio adottato è stato quello di implementare una funzionalità alla volta, verificandone di volta in volta la compatibilità con il resto del sistema già sviluppato, prima di procedere con la successiva.

Capitolo 5

Verifica e validazione

Questo capitolo descrive le modalità con cui è stato verificato il corretto funzionamento delle funzionalità sviluppate.

5.1 Verifica del codice

La verifica del codice sviluppato durante il tirocinio è avvenuta principalmente in maniera manuale. Non sono stati scritti test automatici di unità o di integrazione.

Questa scelta è in parte dovuta all'assenza di una struttura di test preesistente nel progetto, in parte a causa della natura delle funzionalità sviluppate. Buona parte del codice riguarda infatti l'interazione diretta con protocolli di comunicazione (BLE, Wi-Fi) e con l'hardware del dispositivo (scheda microSD, interfaccia di rete), il cui comportamento dipende da fattori difficilmente riproducibili in un ambiente di test isolato, quali la stabilità della connessione, la latenza di rete e il comportamento del filesystem su hardware reale. Testare automaticamente queste componenti avrebbe richiesto la simulazione (mocking) di protocolli e periferiche hardware, riducendo però la verifica al comportamento del mock stesso anziché a quello del sistema reale, proprio sugli aspetti più critici (temporizzazione, concorrenza, gestione degli errori hardware).

Si è pertanto ritenuto che l'onere di predisposizione di un'infrastruttura di test automatici (comprensiva di mock per protocolli di comunicazione e periferiche hardware) non fosse giustificato dal valore aggiunto da essa apportato, data

la necessità comunque presente di una verifica manuale end-to-end sistematica, ripetuta a ogni singola modifica, per validare gli aspetti più critici del sistema.

L'approccio adottato si è quindi basato sul tracciamento dei requisiti definiti nel Capitolo 3. Per ciascun requisito è stata individuata e documentata una specifica modalità di verifica. Questo approccio, pur non fornendo le garanzie di ripetibilità e di regressione tipiche di una suite di test automatici, ha comunque permesso di individuare e correggere diversi problemi nel corso dello sviluppo.

Ad esempio, durante la verifica dell'elenco delle attività disponibili per il download (requisito RFO-F2), un confronto manuale tra i valori restituiti dall'API e quelli calcolati a partire dai file grezzi presenti sulla scheda microSD ha permesso di validare la correttezza del calcolo delle dimensioni riportate. Questa verifica, pur riguardando logica isolabile, richiedeva comunque il confronto con i dati reali letti dalla scheda SD per essere significativa. Allo stesso modo, la ripetizione di più cicli di attivazione e disattivazione della rete Wi-Fi ha portato all'individuazione di un comportamento anomalo del SoftAP, dovuto alla persistenza della configurazione di rete in memoria flash da parte dello stack Wi-Fi di ESP32 [12], risolto disabilitando esplicitamente tale persistenza.

5.1.1 Verifica dei requisiti

Per ciascun requisito individuato nel Capitolo 3 viene indicata la modalità con cui ne è stata verificata manualmente l'implementazione, insieme all'esito riscontrato. Questi requisiti sono suddivisi in tre tabelle che mostrano, in ordine, la verifica dei requisiti funzionali obbligatori che riguardano l'applicazione (Tabella 5.1), dei requisiti funzionali obbligatori che riguardano il firmware (Tabella 5.2) e di altri requisiti che non appartengono alle precedenti categorie (Tabella 5.3).

Tabella 5.1: Verifica dei requisiti funzionali obbligatori lato applicazione.

Requisito	Modalità di verifica	Esito
RFO-A1	Ispezione visiva della schermata Home dopo aver associato dispositivi preferiti e completato attività di test.	Verificato
RFO-A2	Ispezione visiva della schermata Record nel suo stato iniziale.	Verificato
RFO-A3	Osservazione del passaggio della schermata Record allo stato "in corso" alla pressione del pulsante di avvio, confermando la ricezione della notifica <code>SESSION_STATUS</code> da parte del firmware.	Verificato
RFO-A4	Osservazione dell'aggiornamento continuo dei valori (durata, velocità, altitudine, pendenza) durante una sessione di registrazione reale.	Verificato
RFO-A5	Osservazione dell'arresto della registrazione e del cambio di stato alla pressione del pulsante di termine.	Verificato
RFO-A6	Verifica della presenza in Activities delle attività scaricate durante i test di download.	Verificato
RFO-A7	Test manuale delle azioni di modifica nome e rimozione su attività di test.	Verificato
RFO-A8	Ispezione visiva della schermata Downloads.	Verificato
RFO-A9	Connessione manuale reale alla rete Wi-Fi esposta dal dispositivo tramite le impostazioni del sistema operativo.	Verificato
Continua nella prossima pagina...		

Tabella 5.1 – Continuo della tabella

Requisito	Modalità di verifica	Esito
RFO-A10	Confronto tra la lista restituita dall'applicazione e il contenuto reale della scheda SD, calcolato manualmente tramite script di supporto.	Verificato — corrispondenza esatta.
RFO-A11	Osservazione delle richieste HTTP inviate al firmware alla selezione di un'attività da scaricare.	Verificato
RFO-A12	Osservazione dell'avanzamento mostrato in UI durante un download reale, incluso il conteggio cumulativo su più sensori.	Verificato
RFO-A13	Verifica della presenza e correttezza del file salvato in locale sul dispositivo mobile al termine del download.	Verificato
RFO-A14	Osservazione della verifica CRC.	Verificato
RFO-A15	Interruzione manuale del download a metà trasferimento e riavvio dell'applicazione, verificando la ripresa dall'ultimo chunk correttamente ricevuto anziché dall'inizio.	Verificato
RFO-A16	Navigazione tra le schermate dell'applicazione durante una sessione di registrazione attiva, verificando la persistenza dello stato.	Verificato
RFO-A17	Disattivazione manuale del Bluetooth durante una sessione attiva, verificando la notifica all'utente e il comportamento coerente dell'interfaccia alla riconnessione.	Verificato

Tabella 5.2: Verifica dei requisiti funzionali obbligatori lato firmware.

Requisito	Modalità di verifica	Esito
RFO-F1	Scansione delle reti Wi-Fi disponibili da smartphone dopo l'invio del comando di attivazione, verificando la comparsa della rete SoftAP con SSID configurato.	Verificato
RFO-F2	Confronto tra la risposta di <code>GET /api/sessions</code> e il contenuto reale della scheda SD (vedi RFO-A10).	Verificato
RFO-F3, RFO-F4	Ricezione sul lato applicazione dei dati completi di un'attività, verificando l'assenza di dati mancanti o troncati al termine del download.	Verificato
RFO-F5	Confronto, sul lato applicazione, tra il CRC calcolato sul payload decompresso e quello ricevuto nell'header del chunk.	Verificato
RFO-F6	Richiesta di download a partire da un indice di chunk arbitrario (diverso da zero), verificando che il firmware risponda correttamente a partire da quel punto senza richiedere lo stato dei chunk precedenti.	Verificato

Tabella 5.3: Verifica dei requisiti desiderabili, facoltativi, non funzionali e di vincolo.

Requisito	Modalità di verifica	Esito
RFD-A1	Verifica che i dati salvati corrispondano ai dati grezzi attesi dopo la decompressione dei chunk ricevuti.	Verificato
Continua nella prossima pagina...		

Tabella 5.3 – Continuo della tabella

Requisito	Modalità di verifica	Esito
RFD-A2	Apertura della schermata Activities in assenza di connessione di rete, verificando la persistenza dello storico.	Verificato
RFD-A3	-	Requisito non implementato
RND-A1	Disattivazione manuale del Bluetooth durante un'attività, verificando che la sessione non venga terminata e che l'utente sia informato.	Verificato
RND-F1	-	Requisito non implementato
RVO-A1, RVO-A2	Verificati per costruzione: i canali di comunicazione BLE e Wi-Fi sono quelli effettivamente implementati per le rispettive funzionalità.	Verificato
RVO-F1	Verificato per costruzione: la rete Wi-Fi è esposta esclusivamente tramite l'API SoftAP (WiFiAP.h), senza modalità alternative implementate.	Verificato
RFF-A1, RFF-A2	-	Requisiti non implementati.

5.1.2 Limiti della verifica

La verifica manuale descritta nella sezione precedente, per quanto sistematica nel coprire ciascun requisito, presenta alcuni limiti che è opportuno evidenziare, per fornire un quadro onesto dell'affidabilità raggiunta dal sistema.

I test sono stati condotti prevalentemente in condizioni operative favorevoli, senza quindi una verifica del comportamento del sistema in condizioni di carico elevato. Ad esempio, con un numero elevato di attività memorizzate (che potrebbe incidere sui tempi di risposta di `GET /api/sessions`, dovendo il firmware scansionare l'intera scheda SD) o con attività di durata e dimensione molto superiori a quelle utilizzate nei test.

Infine, la mutua esclusione tra registrazione e download (§4.2.3.5) è stata verificata solo nei percorsi previsti dall'interfaccia utente, che di per sé impedisce all'utente di avviare le due operazioni contemporaneamente. Non sono stati eseguiti test volti a forzare condizioni di concorrenza al di fuori di questi percorsi, ad esempio, inviando comandi direttamente al firmware al di fuori del flusso previsto dall'applicazione.

Questi limiti non hanno impedito di raggiungere gli obiettivi del tirocinio, ma rappresentano aspetti da tenere in considerazione in un eventuale sviluppo successivo del progetto, specialmente nel caso in cui il dispositivo venga esposto a un utilizzo più prolungato o a condizioni di rete meno controllate rispetto a quelle dell'ambiente di test.

5.2 Validazione

La validazione ha l'obiettivo di verificare che il sistema, considerato nel suo complesso, soddisfi effettivamente lo scopo per cui è stato progettato.

Per questo, è stato eseguito più volte l'intero flusso applicativo in condizioni di laboratorio: avvio di un'attività di registrazione, conclusione di essa, connessione alla rete Wi-Fi esposta dal dispositivo, download dell'attività registrata e verifica dei dati salvati in locale. Questo test end-to-end ha permesso di validare la corretta integrazione tra le singole funzionalità sviluppate, verificate singolarmente in §5.1.1, e la coerenza del flusso utente nel suo complesso.

Una validazione più ampia è stata inoltre condotta tramite due test sul campo e una demo del sistema presentata ai tutor del progetto. I test sul campo sono stati svolti durante alcuni giri in moto effettuati dal tutor aziendale, con l'obiettivo di verificare il corretto funzionamento del sistema anche in condizioni

reali. In primo luogo, il dispositivo è stato fissato alla moto, come mostrato in Figura 5.1. È stata quindi verificata la corretta connessione al dispositivo tramite BLE e il flusso di calibrazione. Avviata la sessione, ha avuto inizio il giro, al cui termine, la sessione è stata conclusa ed è stato effettuato il download dei dati raccolti, che ha permesso di testare anche la connessione Wi-Fi.

Da questi due test è emerso un problema legato al rilascio dell'applicazione. La build di produzione per Android presentava un comportamento diverso rispetto alla versione utilizzata in fase di sviluppo, impedendo all'utente di attivare e connettersi alla rete Wi-Fi esposta dal dispositivo. Il problema è stato risolto aggiungendo alla build di rilascio alcuni permessi che, non essendo richiesti in modalità di sviluppo, non erano stati dichiarati.

La demo, effettuata a fine tirocinio dopo la risoluzione dei problemi riscontrati nei due test precedenti, ha avuto esito positivo, confermando la validità del prodotto sviluppato.

Si osserva, tuttavia, che l'assenza di una validazione in condizioni d'uso reale estensivo rappresenta un limite significativo. Infatti, il comportamento del sistema in scenari di utilizzo effettivo, caratterizzati da condizioni di rete meno controllate e da sessioni di durata potenzialmente molto maggiore rispetto a quelle testate, non è stato verificato, e costituisce un'attività raccomandata per un eventuale sviluppo successivo del progetto.



Figura 5.1: Posizionamento del dispositivo e dello smartphone sulla moto.

Capitolo 6

Conclusioni

Questo capitolo traccia un bilancio conclusivo del lavoro svolto durante il tirocinio, confrontando le attività e gli obiettivi pianificati con quanto effettivamente realizzato, e presenta possibili sviluppi futuri e le conoscenze acquisite durante l'esperienza.

6.1 Consuntivo delle attività

Nella Tabella 6.1 viene riportato il consuntivo delle ore effettivamente dedicate a ciascuna attività, a confronto con quanto pianificato inizialmente (§ 2.3).

Tabella 6.1: Tabella consuntivo delle attività.

Descrizione attività	Ore preventivate	Ore effettive
Studio dello stato attuale del progetto lato app e lato firmware	40	40
Studio delle tecnologie	40	40
Analisi dei requisiti e progettazione	40	40
Sviluppo	120	128
Testing e validazione	40	30
Documentazione	40	30
Totale	320	308

Come si può osservare, la distribuzione delle ore è rimasta sostanzialmente coerente con quanto pianificato, con uno scostamento di 12 ore rispetto a quelle totali preventivate (308 ore invece di 320). Le fasi di studio del progetto esistente, di studio delle tecnologie e di analisi dei requisiti e progettazione hanno richiesto esattamente il tempo previsto.

Lo sviluppo ha invece richiesto 8 ore in più (128 contro 120 preventivate), con una corrispondente riduzione delle ore dedicate a testing e validazione (30 contro 40) e a documentazione (30 contro 40). Questo scostamento è avvenuto in quanto parte delle attività di verifica e di stesura della documentazione tecnica sono state svolte in modo incrementale durante lo sviluppo stesso, anziché come fasi separate e successive, riducendo così il tempo dedicato a esse come attività a sé stanti. Nel complesso, non ci sono state differenze sostanziali rispetto a quanto concordato in fase di pianificazione.

6.2 Consuntivo degli obiettivi

Il confronto tra gli obiettivi definiti nel piano aggiornato (Tabella 2.2) e il lavoro effettivamente svolto è riportato nella Tabella 6.2.

Tabella 6.2: Tabella consuntivo degli obiettivi.

Codice	Descrizione	Stato
O1	Invio dei comandi azione via BLE	Soddisfatto
O2	Download dei chunk dati via Wi-Fi (con resume e verifica CRC)	Soddisfatto
O3	Visualizzazione dei dati e delle metriche di trasferimento	Soddisfatto
O4	Gestione delle sessioni lato app	Soddisfatto
O5	Stesura della documentazione tecnica	Soddisfatto
D1	Decompressione dei chunk lato app	Soddisfatto
Continua nella prossima pagina...		

Codice	Descrizione	Stato
D2	Persistenza locale e gestione dei dati offline	Soddisfatto
D3	Streaming real time dei dati in app	Soddisfatto
F1	Monitoraggio in tempo reale dei parametri del device	Non soddisfatto
F2	Vista sullo stato del cloud / dati su EggLogCore	Non soddisfatto

Come si può notare, tutti gli obiettivi obbligatori (da O1 a O5) e desiderabili (da D1 a D3) definiti nel piano aggiornato sono stati soddisfatti.

Gli obiettivi facoltativi F1 (monitoraggio in tempo reale dei parametri del device) e F2 (vista sullo stato del cloud e dei dati su EggLogCore) non sono stati affrontati perché si è preferito privilegiare la qualità e la robustezza delle funzionalità obbligatorie e desiderabili. Trattandosi di obiettivi facoltativi, pensati per dare valore aggiunto al prodotto senza costituire un requisito vincolante, la loro assenza non pregiudica il raggiungimento degli obiettivi formativi e di progetto del tirocinio. Le ore disponibili sono state destinate al completamento e al consolidamento degli obiettivi obbligatori e desiderabili, ritenuti maggiormente rilevanti sia per l'azienda sia per il percorso formativo.

6.3 Consuntivo dei rischi

Viene riportato il consuntivo dei rischi individuati in fase preventiva (§2.4), con l'indicazione di come ciascuno di essi si sia effettivamente manifestato durante lo svolgimento del tirocinio e dell'efficacia delle strategie di mitigazione adottate.

1. Difficoltà nell'estendere un sistema preesistente con nuove funzionalità, senza comprometterne il funzionamento originale.

- Esito: il rischio si è effettivamente manifestato, ma la fase di studio del sistema esistente e l'approccio basato su mini-progetti isolati per testare l'integrazione di ogni nuova componente si sono rivelati sufficienti a mitigarlo, senza introdurre regressioni nel sistema originale.
2. Differenze di comportamento BLE/Wi-Fi tra Android e iOS.
 - Esito: il rischio si è manifestato, come discusso anche nel Capitolo 5. Il test continuo su dispositivi fisici di entrambi gli ecosistemi si è rivelato non solo utile ma anche necessario per individuare e gestire tali differenze.
 3. Disallineamento tra interfaccia firmware e app durante lo sviluppo parallelo.
 - Esito: il rischio è stato evitato grazie al confronto costante tra i due tirocinanti. Ogni potenziale disallineamento è stato individuato e risolto tempestivamente.
 4. Instabilità della connessione Wi-Fi/BLE durante il trasferimento dati, con conseguente perdita di sincronizzazione tra stato app e stato reale del dispositivo.
 - Esito: non sono stati riscontrati problemi legati alla stabilità della connessione Wi-Fi durante i trasferimenti dati.

6.4 Miglioramenti futuri

Il lavoro svolto durante il tirocinio lascia margine per alcuni sviluppi futuri che non hanno trovato spazio all'interno del periodo di tirocinio ma che rappresentano possibili estensioni naturali del sistema realizzato.

In primo luogo, gli obiettivi facoltativi F1 (monitoraggio in tempo reale dei parametri del device) e F2 (vista sullo stato del cloud e dei dati su EggLogCore), restano validi come possibili sviluppi futuri per arricchire ulteriormente il prodotto.

Un secondo ambito di miglioramento riguarda la schermata di dettaglio di una sessione (Trip Detail), attualmente non esistente, per visualizzare informazioni aggregate sulla sessione (ad esempio velocità media, inclinazione massima, ecc.) e grafici che rappresentino l'andamento nel tempo dei dati raccolti durante la sessione (ad esempio velocità, altitudine, inclinazione).

Il completamento di questi elementi consentirebbe di sfruttare appieno i dati già raccolti e trasferiti dal dispositivo, offrendo all'utente finale una visualizzazione più ricca e immediata delle proprie sessioni.

6.5 Conoscenze acquisite

Le competenze acquisite nel corso di laurea, in particolare riguardanti la programmazione e le reti di comunicazione, hanno costituito una base fondamentale per affrontare il tirocinio, che ha rappresentato un'occasione di crescita sia sul piano tecnico sia su quello metodologico.

Dal punto di vista tecnico, l'esperienza ha permesso di acquisire competenze nello sviluppo di applicazioni mobile con React Native [38], nella comunicazione tramite il protocollo BLE e nella gestione del profilo GATT, e nell'implementazione di comunicazioni Wi-Fi/HTTP tra applicazione e dispositivo embedded. A queste si è affiancata una componente pratica: il debug su dispositivi fisici Android e iOS, necessario per individuare e risolvere le differenze di comportamento tra le due piattaforme.

Sul piano metodologico, il tirocinio ha permesso di confrontarsi con le difficoltà tipiche dello sviluppo di un sistema composto da due parti fortemente interdipendenti, quella firmware e quella applicativa, gestite da due persone diverse. Questo ha reso necessario un confronto costante e una definizione condivisa delle interfacce di comunicazione. Infine, il tirocinio ha offerto l'opportunità di lavorare sull'estensione di un sistema preesistente, imparando a introdurre nuove funzionalità senza comprometterne il funzionamento originale, un aspetto spesso più delicato rispetto allo sviluppo di un progetto da zero.

Glossario

Salvo dove diversamente indicato, le definizioni dei termini generali sono tratte da Wikipedia [40].

API (Application Programming Interface) Insieme di funzioni ed endpoint esposti da un sistema per consentire l'interazione con altri software. 66

BLE (Bluetooth Low Energy) Protocollo di comunicazione wireless a basso consumo energetico, ottimizzato per lo scambio di piccole quantità di dati tra dispositivi [25]. 2

Chunk Blocco di dimensione prefissata in cui vengono suddivisi i dati trasmessi durante il download di un'attività, per consentirne la verifica di integrità e la ripresa in caso di interruzione. 2

CRC (Cyclic Redundancy Check) Codice di controllo calcolato a partire da una sequenza di dati, utilizzato per verificarne l'integrità dopo la trasmissione. 2

GAP (Generic Access Profile) Profilo del protocollo BLE che definisce i ruoli e le modalità con cui i dispositivi si scoprono e stabiliscono una connessione [8]. 39

GATT (Generic Attribute Profile) Profilo del protocollo BLE che definisce la struttura con cui i dati vengono organizzati ed esposti tramite Service e Characteristic [9]. 5

HTTP (Hypertext Transfer Protocol) Protocollo di comunicazione utilizzato per lo scambio di richieste e risposte tra applicazione e webserver esposto dal dispositivo [23]. 5

JSON (JavaScript Object Notation) Formato testuale per la rappresentazione di dati strutturati, utilizzato per lo scambio di informazioni tra applicazione e firmware. 51

SD (Secure Digital) Scheda di memoria non volatile utilizzata dal dispositivo per la memorizzazione i dati registrati durante le sessioni. 2

SoftAP (Access Point) Modalità operativa in cui un dispositivo Wi-Fi crea ed espone una propria rete, a cui altri dispositivi possono connettersi direttamente, senza richiedere un access point esterno [11]. 16

SPI (Serial Peripheral Interface) Protocollo di comunicazione seriale sincrona, utilizzato per lo scambio di dati tra un microcontrollore e periferiche esterne [13]. 35

SSID (Service Set Identifier) Nome identificativo di una rete Wi-Fi, utilizzato dai dispositivi per individuarla e connettersi a essa. 40

STA (Station) Modalità operativa in cui un dispositivo Wi-Fi si connette come client a una rete esistente, anziché creandone una propria [11]. 34

UI (User Interface) Insieme degli elementi grafici tramite cui l'utente interagisce con l'applicazione. 10

UUID (Universally Unique Identifier) Identificativo univoco, utilizzato per identificare in modo non ambiguo Service e Characteristic BLE. 40

Wi-Fi Protocollo di comunicazione wireless per reti locali, utilizzato nel sistema per il trasferimento di dati che richiedono una banda maggiore rispetto a quella offerta dal BLE. 2

Riferimenti Bibliografici e Sitografici

Testi

- [1] Richard Barry. *Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide*. Vol. v1.1.0. Disponibile in PDF su FreeRTOS.org. Real Time Engineers Ltd., 2016 (cit. a p. 33).

Datasheet

- [2] Adafruit. *Ultimate GPS Datasheet*. URL: <https://learn.adafruit.com/adafruit-ultimate-gps?view=all> (visitato il giorno 28/08/2026) (cit. a p. 35).
- [3] Jeff Epler. *Adafruit MicroSD SPI or SDIO Card Breakout Board Guide*. Last edited: September 2025. Disponibile online su: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-microsd-spi-sdio.pdf>. Adafruit Industries (cit. a p. 35).
- [4] Infineon Technologies AG. *DPS310 Datasheet*. <https://files.seeedstudio.com/products/101020812/res/DPS310-datasheet.pdf>. V1.1 July 11th 2019 (cit. a p. 35).
- [5] STMicroelectronics. *LSM6DSOX Datasheet*. <https://www.st.com/resource/en/datasheet/lsm6dsox.pdf>. Rev 4 - June 2024 (cit. a p. 35).

Siti

- [6] AlgoMaster. *Producer-Consumer Design Pattern*. URL: <https://algomaster.io/learn/concurrency-interview/producer-consumer-pattern> (visitato il giorno 28/08/2026) (cit. a p. 65).
- [7] Atlassian. *Bitbucket*. URL: <https://bitbucket.org/product/> (visitato il giorno 28/08/2026) (cit. a p. 68).
- [8] Bluetooth. *GAP (Generic Access Profile)*. URL: <https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/generic-access-profile.html> (visitato il giorno 28/08/2026) (cit. alle pp. 39, 83).
- [9] Bluetooth. *GATT (Generic Attribute Profile)*. URL: <https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/generic-attribute-profile--gatt-.html> (visitato il giorno 28/08/2026) (cit. alle pp. 38–40, 83).
- [10] Espressif Systems Co. *esp_rom_crc*. URL: https://github.com/espressif/esp-idf/blob/master/components/esp_rom/linux/esp_rom_crc.c (visitato il giorno 28/08/2026) (cit. a p. 34).
- [11] Espressif Systems Co. *ESP32 WiFi API*. URL: <https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html> (visitato il giorno 13/09/2026) (cit. a p. 84).
- [12] Espressif Systems Co. *ESP32-S3-DevKitC-1-N8R8 Documentation*. URL: <https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitc-1/index.html> (visitato il giorno 28/08/2026) (cit. alle pp. 2, 33–36, 68, 70).
- [13] Espressif Systems Co. *SPI*. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/spi_master.html (visitato il giorno 13/09/2026) (cit. a p. 84).

- [14] Espressif Systems Co. *WebServer.h*. URL: <https://github.com/espressif/arduino-esp32/blob/master/libraries/WebServer/src/WebServer.h> (visitato il giorno 28/08/2026) (cit. a p. 33).
- [15] Espressif Systems Co. *WiFiAP.h*. URL: <https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFiAP.h> (visitato il giorno 28/08/2026) (cit. a p. 34).
- [16] cplusplus.com. *C++*. URL: <https://cplusplus.com/doc/tutorial/> (visitato il giorno 13/09/2026) (cit. alle pp. 33, 68).
- [17] Paul Sokolovsky Damien P. George. *uzlib*. URL: <https://docs.micropython.org/en/v1.15/library/uzlib.html> (visitato il giorno 28/08/2026) (cit. a p. 34).
- [18] Arduino Docs. *NimBLE-Arduino Documentation*. URL: <https://docs.arduino.cc/libraries/nimble-arduino/> (visitato il giorno 28/08/2026) (cit. alle pp. 33, 66).
- [19] Arduino Docs. *SdFat*. URL: <https://docs.arduino.cc/libraries/sdfat/> (visitato il giorno 28/08/2026) (cit. a p. 33).
- [20] Claire Drumond. *Agile Scrum Methodology*. URL: <https://www.atlassian.com/agile/scrum> (visitato il giorno 28/08/2026) (cit. a p. 10).
- [21] Eggon S.r.l. *Eggon*. URL: <https://start.eggon.co> (visitato il giorno 28/08/2026) (cit. a p. 1).
- [22] Expo. *Expo*. URL: <https://docs.expo.dev/guides/overview/> (visitato il giorno 28/08/2026) (cit. a p. 68).
- [23] Roy T. Fielding, Mark Nottingham e Julian Reschke. *HTTP*. URL: <https://httpwg.org/specs/rfc9110.html> (visitato il giorno 13/09/2026) (cit. a p. 84).
- [24] innoveit. *react-native-ble-manager Documentation*. URL: <https://innoveit.github.io/react-native-ble-manager/> (visitato il giorno 28/08/2026) (cit. alle pp. 5, 32).

- [25] Kevin Townsend. *Bluetooth Low Energy*. URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy?view=all> (visitato il giorno 13/09/2026) (cit. a p. 83).
- [26] Martin Fowler. *Dependency Injection Design Pattern*. URL: <https://martinfowler.com/articles/dependency-composition.html> (visitato il giorno 28/08/2026) (cit. a p. 67).
- [27] Martin Fowler. *Repository Design Pattern*. URL: <https://martinfowler.com/eaaCatalog/repository.html> (visitato il giorno 28/08/2026) (cit. a p. 66).
- [28] Meta Open Source. *Metro*. URL: <https://metrobundle.dev/docs/concepts> (visitato il giorno 28/08/2026) (cit. a p. 68).
- [29] Microsoft. *TypeScript*. URL: <https://www.typescriptlang.org/docs/> (visitato il giorno 13/09/2026) (cit. alle pp. 32, 68).
- [30] Microsoft Learn. *MVVM Design Pattern*. URL: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm> (visitato il giorno 28/08/2026) (cit. a p. 66).
- [31] Nodeca. *pako*. URL: <https://nodeca.github.io/pako/> (visitato il giorno 28/08/2026) (cit. a p. 32).
- [32] npmjs. *fflate library*. URL: <https://www.npmjs.com/package/fflate> (visitato il giorno 28/08/2026) (cit. a p. 32).
- [33] PlatformIO Labs. *PlatformIO*. URL: <https://platformio.org> (visitato il giorno 28/08/2026) (cit. a p. 68).
- [34] React Three Fiber. *react-three-fiber*. URL: <https://r3f.docs.pmnd.rs/getting-started/introduction> (visitato il giorno 28/08/2026) (cit. a p. 35).
- [35] react-native-async-storage. *AsyncStorage*. URL: <https://github.com/react-native-async-storage/async-storage> (visitato il giorno 28/08/2026) (cit. alle pp. 32, 66).

- [36] Refactoring.Guru. *Command Design Pattern*. URL: <https://refactoring.guru/design-patterns/command> (visitato il giorno 28/08/2026) (cit. a p. 66).
- [37] Refactoring.Guru. *Observer Design Pattern*. URL: <https://refactoring.guru/design-patterns/observer> (visitato il giorno 28/08/2026) (cit. a p. 68).
- [38] Meta Open Source. *React Native*. URL: <https://reactnative.dev/docs/getting-started> (visitato il giorno 28/08/2026) (cit. alle pp. 2, 5, 32, 36, 68, 82).
- [39] Stuart Knightley. *JSZip*. URL: <https://stuk.github.io/jszip/> (visitato il giorno 28/08/2026) (cit. a p. 32).
- [40] Wikimedia Foundation, Inc. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Main_Page (visitato il giorno 13/09/2026) (cit. a p. 83).