POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Elettronica

Politecnico
di Torino

Tesi di Laurea Magistrale

Progettazione e sviluppo di un sistema di collaudo automatizzato per dispositivi di sicurezza antintrusione in radiofrequenza

Relatori
Prof. Massimo RUO ROCH
Ing. Simone ANTONIETTA

Candidato
Kevin PICCATI

Dicembre 2024

Abstract

La presente tesi si concentra sulla progettazione e realizzazione di una macchina di collaudo per periferiche di sistemi di sicurezza antintrusione, che svolgono un importante ruolo nella protezione di ambienti sia privati che pubblici. Lo svolgimento di procedure di test accurate e rigorose sui singoli dispositivi e sensori è di fondamentale importanza per garantire la solidità e l'affidabilità dell'intero sistema.

La macchina finale sarà in grado di testare le periferiche radio attraverso la misurazione e la caratterizzazione di alcune grandezze fisiche e mediante lo svolgimento di test elettrici e funzionali, simulando ed emulando condizioni operative reali. Dati e risultati, oltre a fornire l'esito finale, vengono raccolti e memorizzati in modo sistematico, rendendoli facilmente consultabili grazie alla generazione di una reportistica dettagliata e mediante un meccanismo di backup su cloud.

La progettazione si è orientata sia sulla ricerca e l'ottimizzazione del processo di collaudo, al fine di ridurre il tempo necessario per completare ogni ciclo senza trascurare l'accuratezza, sia sulla flessibilità adottata nell'implementazione dell'hardware, del firmware e del software, consentendo alla macchina di avere un unico ecosistema in grado di adattarsi a diversi dispositivi.

Attraverso un processo di validazione e di analisi, i risultati ottenuti sono stati verificati per mezzo di prove sperimentali. Esse hanno permesso di accertare le performance delle periferiche, assicurando che i risultati e le misure fossero coerenti e compatibili con quelli riportati dalla macchina.

Questa tesi evidenzia come una progettazione rigorosa possa ottimizzare l'automazione del processo di collaudo, contribuendo a rilevare con precisione e velocità difetti e anomalie nelle periferiche, migliorando la qualità dei dispositivi. Future evoluzioni del progetto potrebbero includere l'implementazione di ulteriori funzionalità e ottimizzazioni, incrementando l'efficacia e la versatilità.

Indice

1 Introduzione **4**

2 Contesto aziendale e stato dell'arte **6**
2.1 Cenni storici . . . 7
2.2 Metodi di test per circuiti stampati . . . 8

3 Progettazione della macchina di collaudo **11**
3.1 Descrizione della periferica da testare . . . 11
3.2 Descrizione del dispositivo SC8R . . . 14
3.3 Specifiche tecniche . . . 16
3.3.1 Specifiche funzionali . . . 17
3.3.2 Specifiche hardware . . . 17
3.3.3 Specifiche firmware e software . . . 18
3.3.4 Sicurezza, affidabilità e manutenibilità . . . 18
3.3.5 Reportistica . . . 18
3.3.6 Documentazione . . . 19

4 Implementazione e sviluppo **20**
4.1 Implementazione hardware . . . 20
4.1.1 Assorbimenti di corrente . . . 20
4.1.2 Tamper, aux e reed bypass . . . 24
4.1.3 LED . . . 24
4.1.4 Reed . . . 26
4.1.5 Accelerometro . . . 28
4.1.6 Alimentazioni e μC . . . 30
4.2 Architettura completa . . . 33
4.3 Parti meccaniche e struttura esterna . . . 42
4.3.1 Valutazione dei rischi elettrici e meccanici . . . 45
4.4 Implementazione firmware . . . 46
4.4.1 Protocollo seriale . . . 48
4.4.2 Assorbimenti di corrente . . . 50
4.4.3 Calibrazione . . . 60
4.4.4 Verifica ADC tramite controllo batteria . . . 63
4.4.5 Tamper, aux e reed bypass . . . 64
4.4.6 LED . . . 66
4.4.7 Reed . . . 67
4.4.8 Accelerometro . . . 70
4.4.9 Update ID periferica . . . 72

4.5 Implementazione software e pagina web 76
4.6 User interface e backend 76
4.6.1 Struttura del codice della pagina web 79
4.6.2 Codice del backend 83
4.6.3 Ulteriori funzionalità di contorno 90

5 Collaudo e validazione **93**
5.1 Assorbimenti di corrente 93
5.2 Calibrazione 96
5.3 Potenza in trasmissione ed RSSI 98
5.4 Tamper, aux, reed bypass e LED di segnalazione 101
5.5 Reed e accelerometro 101

6 Future implementazioni **102**
6.1 Periferiche da testare 102
6.2 Sviluppo hardware e firmware 103
6.3 Struttura meccanica componibile 104
6.4 Gestione interfacciamento e prevenzione errori di montaggio 107

1 Introduzione

Nel contesto della sicurezza delle abitazioni e degli edifici pubblici e commerciali, i sistemi antintrusione ricoprono un ruolo fondamentale finalizzato al monitoraggio e alla protezione delle strutture, nonché alla salvaguardia e alla tutela delle persone.

Tali sistemi sono costituiti da una vasta gamma di dispositivi elettronici quali sensori magnetici per porte e finestre, sensori di movimento, telecamere di videosorveglianza, controllo degli accessi e dispositivi di allarme come sirene ed emettitori acustici, volti a rilevare e segnalare situazioni di emergenza.

La loro efficacia è correlata all'affidabilità e alla qualità delle periferiche impiegate. La presenza di malfunzionamenti o errori potrebbe significare ritardi nelle segnalazioni, falsi allarmi o mancati rilevamenti di situazioni di reale emergenza, esponendo strutture e persone a rischi potenziali.

Per evitare che alcune criticità possano compromettere l'intero sistema, è necessario un rigoroso processo di collaudo dei vari dispositivi. Questa operazione ha lo scopo di individuare le periferiche che presentano difetti o anomalie, identificare lo specifico tipo di problematica e analizzare le possibili cause.

La seguente tesi è svolta in collaborazione con l'azienda SAET I.S., specializzata nel settore. In questo contesto si colloca il presente lavoro, affrontando questa sfida nella progettazione e realizzazione di una macchina di test dedicata alle periferiche dei sistemi di sicurezza antintrusione. Lo scopo di questa macchina sarà di automatizzare e standardizzare il processo di collaudo aziendale, effettuando test approfonditi e affidabili dei dispositivi, al fine di garantire un alto livello di qualità del prodotto e tranquillità per il cliente finale.

Il documento è organizzato in capitoli, ognuno dei quali affronta un aspetto specifico del progetto di sviluppo dell'attrezzo di collaudo. Gran parte della descrizione verte principalmente sulla realizzazione della prima apparecchiatura di test destinata al prodotto con i volumi di vendita più elevati, e si conclude con una breve introduzione alla realizzazione delle successive macchine.

La tesi si apre con una descrizione del contesto aziendale, illustrando il ruolo dell'azienda nel settore della sicurezza antintrusione, controllo accessi, domotica ed impianti speciali. Viene fornita una breve panoramica sul settore in cui SAET I.S. si colloca e sui principali prodotti che sviluppa e commercializza, sull'esigenza di dover testare e collaudare i sensori per garantire un alto livello di affidabilità e su come il presente lavoro si integri in questo processo. Viene effettuata anche una panoramica sui principali sistemi di collaudo e sugli approcci più utilizzati, nonché una rappresentazione delle tecnologie, dei protocolli e degli standard attuali per i dispositivi di sicurezza radio.

Successivamente si approfondisce il funzionamento della periferica da testare, evidenziando la varietà di logiche e componenti che devono essere verificati, sia a livello funzionale

che a livello elettrico; si inizia con analisi delle specifiche fino ad arrivare a descrivere il flusso di progetto adottato.

In primo luogo si descrive la realizzazione del principale circuito stampato (citato più volte mediante il suo codice CS3981) che rappresenta il cuore del sistema; attraverso questa scheda vengono effettuate tutte le stimolazioni necessarie per simulare condizioni operative reali, le misure di assorbimento in diverse situazioni e la stima della potenza della comunicazione radio del Device Under Test (DUT).

L'argomentazione si sposta poi sull'implementazione a livello di sistema, attraverso una descrizione di quali dispositivi e componenti vengono utilizzati e di come si interfacciano con il CS3981. In questa sezione, si descrivono il firmware, le logiche e gli algoritmi utilizzati per compiere gli step del processo di collaudo e si conclude illustrando lo sviluppo e il funzionamento del frontend, costituito dall'interfaccia utente, e del backend, la parte nascosta dell'applicazione che funge da ponte tra frontend e firmware. Si è prestata particolare attenzione a motivare in modo dettagliato la metodologia e le scelte progettuali adottate per le diverse parti, sia hardware che firmware.

Terminata la realizzazione di tutto il sistema, si illustrano le metriche di valutazione al fine di validare la macchina di collaudo. Sono riportati alcuni dei risultati ottenuti dall'apparecchiatura di test e questi vengono confrontati attraverso prove sperimentali, svolte in campo, per verificare l'accuratezza dei dati e degli esiti al fine di valutare l'affidabilità e l'efficacia della macchina.

Infine, come anticipato in precedenza, si fornisce una visione d'insieme sullo sviluppo delle successive macchine di collaudo, destinate a testare altri dispositivi e sensori della stessa linea di produzione, che l'azienda lancerà sul mercato all'inizio del 2025. Queste nuove macchine presenteranno caratteristiche simili a quella descritta in questo documento, mantenendo il focus sull'efficacia dei test e sulla validazione delle performance delle periferiche.

La tesi rappresenta un'analisi dettagliata del processo di progettazione e implementazione della macchina di collaudo, evidenziando l'importanza di rendere i test più efficienti e ripetibili. Questo approccio permette di superare le limitazioni dei metodi manuali attualmente in uso nell'azienda, velocizzando il processo di verifica e garantendo contemporaneamente risultati più consistenti.

2 Contesto aziendale e stato dell'arte

SAET I.S. è un'azienda italiana specializzata nella progettazione e commercializzazione di sistemi di sicurezza, in particolare anti-intrusione e controllo accessi. Fondata nel 1968, SAET è una realtà di riferimento nel settore grazie alla sua esperienza decennale, che offre soluzioni avanzate per la protezione di siti ad alta criticità, come banche, strutture industriali, grandi infrastrutture e siti governativi. Tra i principali clienti si possono citare ING Bank, Intesa San Paolo, CREDEM, Ministero della Difesa, TIM e Vodafone, Musei Milano, Pinacoteca Brera, Palazzo Reale, INPS, Ferrovie dello stato e numerose altre aziende, società ed enti. Sono anche disponibili delle linee di prodotti pensate appositamente per utilizzatori privati.

La sede principale è situata a San Maurizio Canavese, in provincia di Torino, centro di progettazione e sviluppo di nuove soluzioni. Per quanto riguarda i prodotti commercializzati, tra le periferiche e i dispositivi antintrusione troviamo sensori perimetrali (ad esempio sensori magnetici), che monitorano i confini di una proprietà, e sensori a doppia tecnologia, che combinano la tecnologia a microonda con quella a infrarosso. Vengono poi sviluppati gli altri elementi facenti parte del sistema di sicurezza come sirene, telecomandi e centrali di controllo. Inoltre, sono presenti implementazioni di soluzioni di videosorveglianza, con tecnologie per l'analisi video, il riconoscimento di anomalie e la registrazione di eventi. Infine, SAET progetta anche sistemi per il controllo degli accessi che monitorano l'accesso a edifici e aree sensibili attraverso badge e tecnologie di autenticazione avanzate.

Uno dei principali punti di forza è sicuramente l'integrazione di diversi sottosistemi di sicurezza (anti-intrusione, videosorveglianza, controllo accessi, ecc.) in un'unica piattaforma centralizzata, facilitando la gestione e il monitoraggio. Pertanto, un impianto già installato da un concorrente dell'azienda può essere integrato senza interventi importanti, che rappresenta un vantaggio notevole per il cliente finale.

Ad inizio 2025, è previsto il lancio sul mercato di una nuova linea di prodotti di sicurezza in radiofrequenza. Siccome SAET si occupa anche della produzione dei dispositivi, è necessario garantire un alto livello di qualità dei prodotti. Ogni periferica, prima di essere venduta, viene quindi testata e collaudata al fine di garantirne il corretto funzionamento.

Il progetto ha come obiettivo quello di automatizzare e rendere rapido il ciclo di test, migliorando l'efficienza produttiva. In particolare, è necessario realizzare un unico sistema che garantisca l'affidabilità e l'accuratezza dei risultati e delle misure, la ripetibilità delle procedure, la flessibilità di adattarsi a diversi tipi di test e la velocità, riducendo al minimo l'intervento manuale di un operatore. Le periferiche devono essere calibrate in frequenza affinché possano comunicare efficacemente, devono essere caratterizzati e misurati gli assorbimenti di corrente, sia in condizioni di riposo che durante comunicazioni radio e devono essere svolti test funzionali per assicurarsi che componenti e sensori presenti sulle schede funzionino come da protocollo.

Il progetto esposto in questo documento descrive la prima macchina sviluppata e rea-

lizzata, destinata a dei contatti magnetici, che appartengono alla famiglia dei sensori perimetrali, approfonditi successivamente.

2.1 Cenni storici

Le prime forme di sistemi di sicurezza risalgono all'antichità. Sin dai tempi più remoti, gli esseri umani hanno cercato modi per proteggere i loro beni e le loro famiglie da intrusioni esterne. Ad esempio, gli antichi egizi utilizzavano serrature di legno e chiavi rudimentali per proteggere i loro tesori. Anche i romani svilupparono sistemi di chiusura per porte e finestre. In epoca medievale, meccanismi di protezione erano rappresentati da torri di guardia, fossati e mura fortificate. Si deduce che le misure di sicurezza erano prevalentemente fisiche e basate sulla solidità delle strutture, lontane dagli attuali sistemi che ci circondano.

Agli inizi del '700, un inventore di nome Mr. Tildsley, creò un modello di un sistema di allarme, collegando meccanicamente un insieme di catene a porte e finestre. In questo modo, un potenziale intruso, nel tentativo di accedere nella proprietà, faceva muovere le catene, generando un forte rumore. Questo meccanismo serviva non solo a segnalare l'intrusione, ma anche a dissuadere i malintenzionati dal proseguire nel loro intento.

Il primo sistema antifurto reale fu brevettato dal reverendo Augustus Russell Pope nel 1853. Costituito da un circuito elettrico a cui porte e finestre erano collegate, quando queste venivano aperte il circuito si chiueva e la corrente che scorreva attraverso avrebbe fatto vibrare un magnete a cui era collegato un martelletto. Percuotendo un campanello in ottone, si produceva un suono ripetuto e continuo. La principale particolarità del sistema risiedeva nel fatto che, una volta azionato, non smetteva di suonare, nemmeno chiudendo porte o finestre, grazie ad una molla di attivazione che manteneva chiuso il circuito. Il progetto di Pope venne acquistato da un certo Edwin Holmes, un uomo di affari che con la sua azienda, la Holmes Protection Company, iniziò a produrre il sistema di allarme su larga scala [Alf19; Umb24].

Negli anni '70, vennero introdotti sul mercato sensori di movimento dotati di tecnologia degli ultrasuoni, mentre negli anni '80 il progresso tecnologico portò all'introduzione degli infrarossi, che permettevano ai sensori di eliminare i falsi allarmi [Sic24].

Gli standard di sicurezza attuali si sono evoluti notevolmente e i sistemi sono spesso progettati in conformità con normative internazionali come la EN50131. I moderni impianti sono dotati di sensori a infrarossi passivi (PIR) e sensori a microonde (oppure con entrambe le tecnologie), e hanno la possibilità di integrarsi con sistemi smart home e domotici. Dispongono inoltre di sistemi di monitoraggio 24/7 con opzioni di comunicazione tramite app, SMS o chiamate e sono dotati di tecnologie di crittografia per garantire la sicurezza dei dati trasmessi, soprattutto per i sistemi in radiofrequenza.

2.2 Metodi di test per circuiti stampati

Nel contesto della produzione di circuiti stampati, è fondamentale garantire la qualità e l'affidabilità dei PCB. I metodi di test sono progettati per individuare difetti di fabbricazione, errori di assemblaggio e problemi di affidabilità prima che i dispositivi entrino in uso. Esistono diversi tipi di test che consentono di verificare vari aspetti dei PCB, sia a livello di componenti singoli che di sistema completo. Tra i principali, possiamo trovare:

- **AOI**: L'ispezione ottica automatizzata utilizza telecamere per monitorare e scansionare schede elettroniche. Le immagini ad alta risoluzione vengono acquisite e confrontate con un campione di riferimento (detto anche "golden board") o con immagini di schede difettose. Tra i principali punti di forza troviamo l'automazione del processo, la risoluzione, che può arrivare anche a qualche μm e la velocità di esecuzione. Con questa tecnica si possono individuare anche errori di posizionamento dei componenti e non occorre alimentare il circuito. Tuttavia, l'AOI consente di individuare anomalie solo dove piste, saldature e connessioni sono visibili otticamente, e quindi solo sulla superficie del PCB. Non è quindi possibile verificare BGA, difetti interni di circuiti integrati o svolgere test funzionali.

- **AXI**: Automated X-ray Inspection è un controllo che sfrutta i raggi X. A differenza dell'AOI, consente di individuare problematiche non visibili ad occhio nudo. Infatti, i raggi X possono penetrare i layer del PCB, il package di circuiti integrati, rilevare problemi a pad nascosti e a componenti come BGA. Si possono individuare saldature fredde o difettose, cortocircuiti, disconnessioni, disallineamenti o componenti danneggiati internamente. Nonostante sia una tecnica automatizzata e molto efficace, è comunque necessaria la presenza di un operatore specializzato, in particolare in fase di configurazione e di interpretazione avanzata. Inoltre, trattandosi di un processo lungo e costoso, è necessario bilanciare il rilevamento dei difetti con i tempi della scansione, individuando i punti più critici da esaminare.

- **In-Circuit Test**: Questa tecnica utilizza un letto di aghi (bed of nails) sul quale viene appoggiato il PCB; gli aghi contattano punti specifici della board, consentendo di rilevare parametri (come l'impedenza), difetti di saldatura, orientamento errato di componenti, circuiti aperti o cortocircuiti e altro ancora. Tra i principali svantaggi, si ha l'elevato costo di questa tecnica, poiché è necessario realizzare un letto di aghi ad-hoc per ogni scheda da testare e solo in corrispondenza dei test point è possibile effettuare controlli e misure. Inoltre, è possibile svolgere solo test elettrici, senza verificare le prestazioni funzionali del PCB [Pro24].

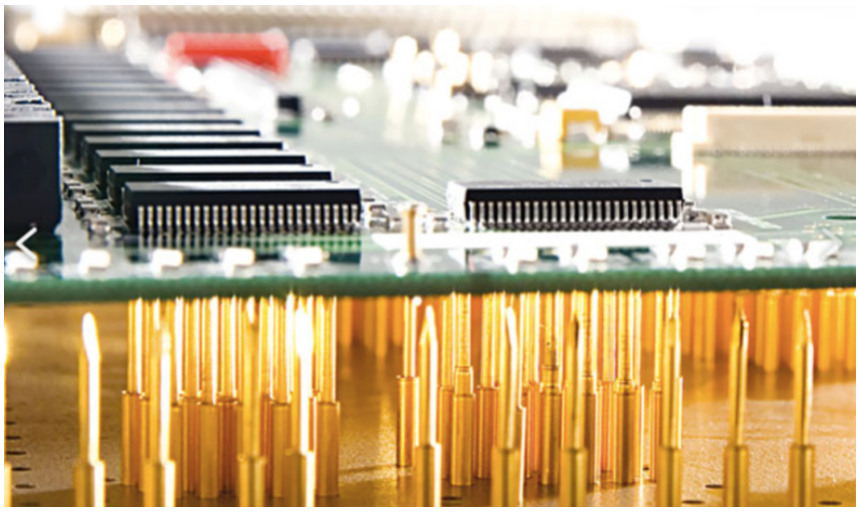

Figura 1: Esempio di test con letto d'aghi (bed of nails), ©Mirotech.

- **Flying Probe Test**: Si tratta di una tecnica simile al bed of nails, ma si utilizzano delle sonde mobili al posto di una matrice fissa di aghi. Si deduce che questa tecnica risulta più flessibile, può contattare diversi punti in una scheda (e non solo i test point previsti dal progettista) e può essere utilizzata per diversi PCB, programmando la macchina opportunamente. Tuttavia, siccome le sonde devono spostarsi e muoversi fisicamente nello spazio, le tempistiche necessarie per un test si dilatano. Pertanto, non è una tecnica indicata per la produzione su larga scala per via della bassa velocità di scansione [Pro24].

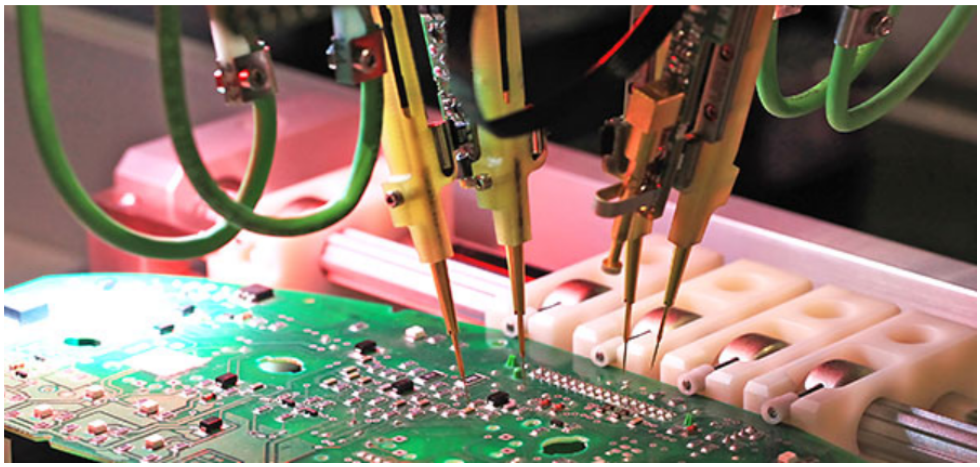

Figura 2: Esempio di flying probe test, ©Doxals.

- **Test Funzionali**: è una tecnica per i PCB assemblati, progettata per verificare le funzionalità e il comportamento di un circuito in condizioni di reale operatività. L'obiettivo è verificare che il dispositivo fornisca gli output previsti in risposta a specifici input. Si possono eseguire test sull'alimentazione, sui protocolli di comunicazione (seriale o radio) e sulla gestione di periferiche e sensori esterni. Nonostante rappresenti la tipologia di test più vicina al caso di utilizzo reale, la sua efficacia richiede un'attenta progettazione delle procedure di test da svolgere, al fine di garantire un'ampia copertura di casistiche.

- **Burn-In Test**: Si tratta di una sorta di test funzionale, in cui vengono verificate le prestazioni di un circuito in condizioni estreme, agendo su parametri ambientali come temperatura, umidità e pressione, o parametri elettrici come tensioni e correnti. La durata dei test può variare da qualche ora a diversi giorni e può provocare uno stress eccessivo che potrebbe influire negativamente sull'affidabilità e sulla durata se non vengono svolti in modo appropriato.

La macchina di test progettata utilizza un ibrido di due delle tecniche menzionate: l'ICT (con bed of nails) e il test funzionale. Attraverso il letto di aghi si contattano punti strategici dove vengono effettuate stimolazioni, verifica di potenziali cortocircuiti sull'alimentazione e misure di tensione e corrente, mentre attraverso i test funzionali si verifica il comportamento a fronte degli stimoli ricevuti. Nei capitoli successivi verrà approfondito il ruolo di queste tecniche all'interno del processo di test.

3 Progettazione della macchina di collaudo

3.1 Descrizione della periferica da testare

La macchina di collaudo ha il compito di effettuare un test completo di un dispositivo in particolare dell'azienda. Si tratta di un componente essenziale all'interno di un sistema di sicurezza antintrusione e fa parte della nuova linea che l'azienda ha intenzione di lanciare sul mercato all'inizio del 2025.

La periferica in questione è un sensore perimetrale wireless ad alta sicurezza, comunemente chiamato "contatto magnetico", "contatto 3V" o indicato con il suo codice CS3965. Il sofisticato circuito di analisi consente di rilevare l'apertura del serramento, lo scasso e l'accelerazione e decelerazione dell'anta mobile. La comunicazione avviene in radiofrequenza, con portante a 868MHz, ed è supervisionata da un ricevitore (l'SC8R, descritto nella Sezione 3.2). Le Figure 3 e 4 mostrano il circuito del device (fronte e retro), senza il case di protezione. Lo scopo e la funzione delle varie parti riportate nell'immagine sono discusse in seguito.

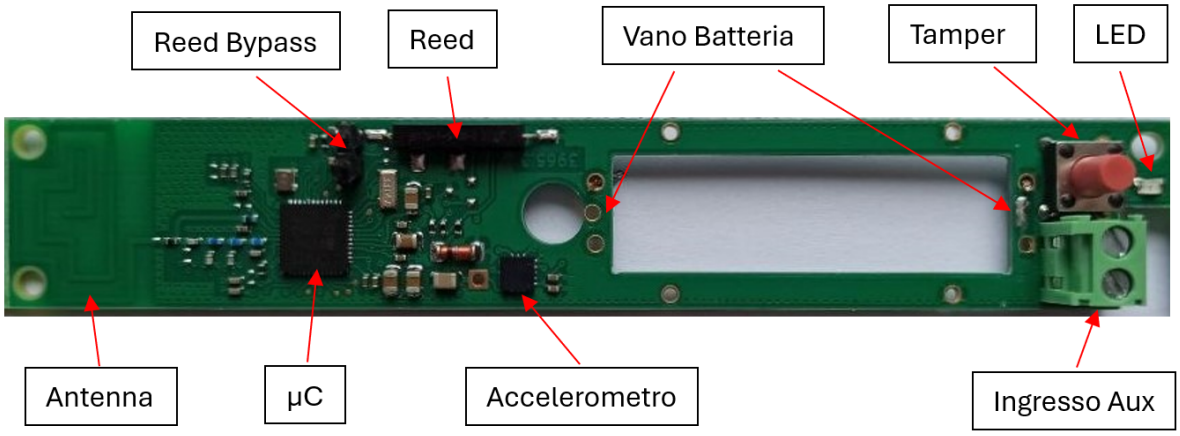

Figura 3: Fronte del circuito del contatto 3V linea 868 da interno.

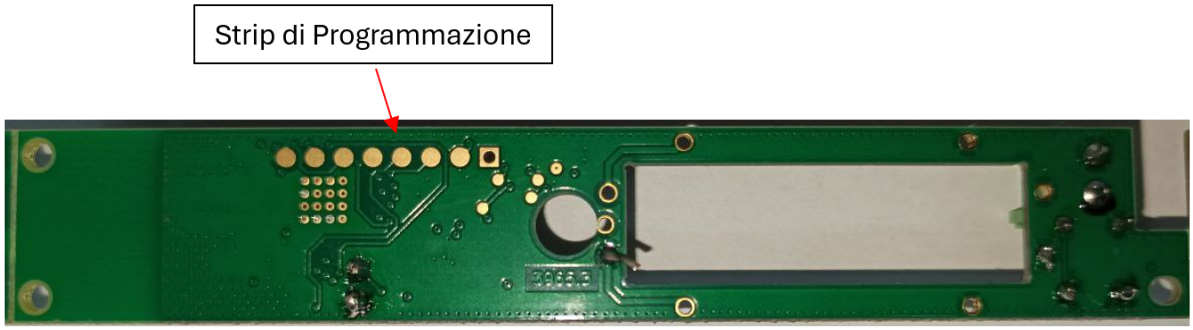

Figura 4: Retro del circuito del contatto 3V linea 868 da interno.

Il sensore reed consiste essenzialmente in due lamine di materiale ferroso poste all'interno di un contenitore plastico (o di vetro) riempito con un gas non reattivo, solitamente argon o azoto. Quest'ambiente stagno permette di ottenere un elevato numero di commutazioni. Le lamelle sono distanziate di qualche centinaio di micrometri e si attraggono se viene applicato un campo magnetico esterno [Wik22]. Questo componente ha lo scopo di identificare lo stato dell'infisso su cui è stato montato il CS3965. Infatti, se l'anta mobile risulta essere chiusa, il magnete montato su di essa sollecita il reed che chiude il circuito. Viceversa, se l'anta viene aperta, la forza di attrazione delle lamine non è più sufficiente e il contatto magnetico interrompe il collegamento della pista.

Il CS3965, trattandosi di un dispositivo versatile, potrebbe non essere installato su un infisso. Pertanto, sono stati previsti due terminali (reed bypass) progettati per essere cortocircuitati in base all'applicazione. In questo modo è possibile escludere il reed e non essere più sensibile ad esso.

Il tamper è un pulsante che ha lo scopo di rilevare eventuali manomissioni. Una volta installato, il sensore ad alta sicurezza viene inserito all'interno di un contenitore plastico, visibile in Figura 5. Il case mantiene il tamper premuto mediante una molla. Qualora un malintenzionato cercasse di rimuovere l'involucro, il pulsante risulterebbe aperto e il sistema, rilevando tale situazione, entrerebbe in allarme. Tuttavia, se il sensore fosse rimosso direttamente dal punto d'installazione, il tamper non rileverebbe alcuna manomissione. In questa circostanza, l'accelerometro entra in gioco. Il compito di questo componente e quello di rilevare variazioni nell'inclinazione del contatto magnetico riconducibili allo smontaggio del dispositivo, oppure vibrazioni dovute ad un tentativo di danneggiamento.

L'ingresso ausiliario (aux) è un ingresso supplementare per lo scambio dei dati. Permette di collegare dispositivi aggiuntivi, quali tapparelle o telecamere. Questo consente di estendere le potenzialità della periferica attraverso l'integrazione con altri componenti del sistema domestico o aziendale. L'aux distingue automaticamente due modalità di funzionamento: la prima è essenzialmente un ON/OFF, mentre la seconda è un conta impulsi (utile nel caso delle tapparelle) per discriminare situazioni di allarme dal normale

funzionamento. Pertanto, grazie alla comunicazione radio del contatto 3V, l'ingresso aux consente la gestione di sensori e dispositivi esterni privi di comunicazioni wireless.

Accanto al tamper è presente un led di segnalazione. Ad ogni rilevazione e cambio di stato di un qualsiasi sensore, la periferica invia una trasmissione verso il dispositivo SC8R, con conseguente accensione del led. Questo serve per avere un riscontro immediato circa il corretto funzionamento della periferica.

Nella parte sinistra è presente un'antenna realizzata direttamente su circuito stampato. Questa consente di comunicare in radiofrequenza con il ricevitore SC8R, collegato direttamente alla centrale.

Il microcontrollore è un SoC della famiglia EFR32FG23 [Lab23a], della Silicon Labs. Si tratta di una soluzione progettata per applicazioni sub-GHz quali smart homes, security, illuminazione, automazione di edifici e misurazioni. Le funzionalità radio disponibili del μC garantiscono la robustezza alle interferenze con tecnologie come Wi-Fi e comunicazioni 2.4 GHz.

Lo strip di programmazione sul retro del contatto 3V è dotato di 8 piazzole. Due di queste sevono per VCC e GND, mentra un'altra coppia è utilizzata come predisposizione per la comunicazione seriale (TX e RX). Infine, sono ancora presenti un ingresso per il reset della periferica e SWDIO e SWCLK per il caricamento e aggiornamento del firmware.

Infine, tutto il sistema viene alimentato da un batteria al litio da 3V, che può essere inserita nell'apposito vano (il porta batteria non è presente nelle immagini 3 e 4).

Figura 5: Contatto 3V linea 868 con case esterno, configurazione finale, variante bianca.

3.2 Descrizione del dispositivo SC8R

La base di partenza per la progettazione e realizzazione della scheda responsabile dei test è l'SC8R. Questo dispositivo consente lo scambio dei dati tra i dispositivi radio, come il contatto 3V, e la centrale. Si tratta essenzialmente di un ricevitore che rimane principalmente in ascolto: quando arriva una comunicazione radio da parte di un sensore (come una possibile manomissione o un allarme), l'SC8R risponde alla periferica con un ACK e poi segnala il cambiamento di stato o l'informazione alla centrale via cavo. In Figura 6 si mostra il dispositivo SC8R. Siccome viene montato vicino alla centrale con la quale comunica via cavo, non ha problemi di spazio e prende l'alimentazione tramite il proprio morsetto. A differenza del CS3965, è dotato di un'antenna decisamente più grande, capace di garantire una portata estesa.

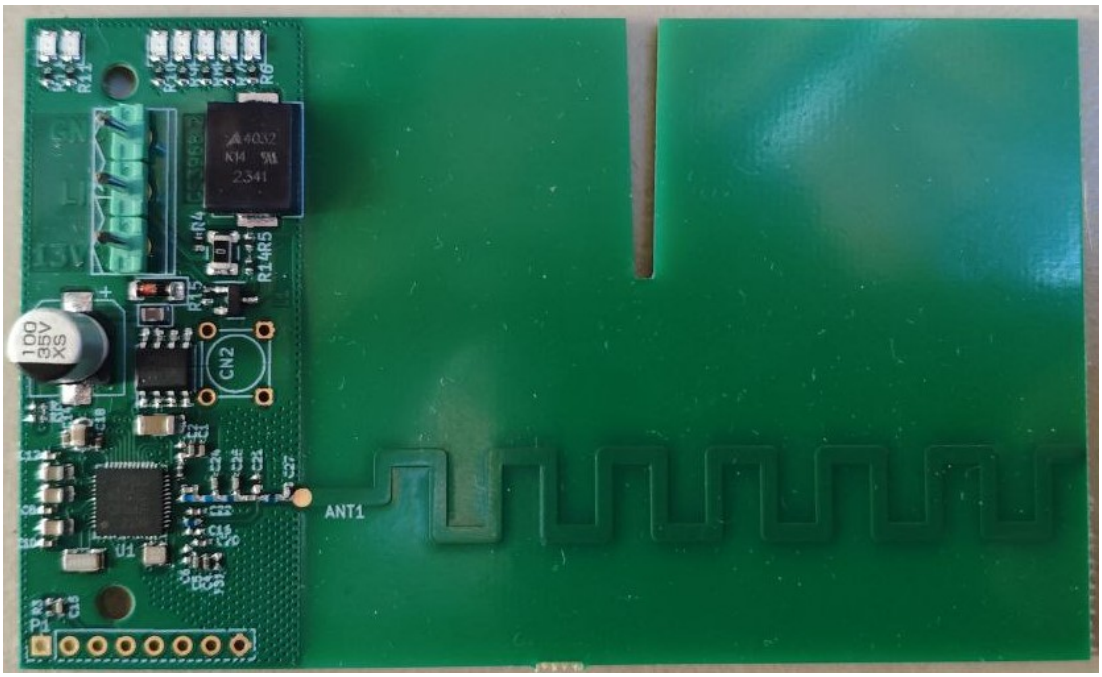

Figura 6: Circuito dell'SC8R.

La comunicazione radio avviene tramite modulazione Direct Sequence Spread Spectrum (DSSS). Si tratta di un tipo di modulazione spread spectrum che distribuisce il segnale originale su una banda di frequenza molto più ampia rispetto a quella richiesta dal contenuto del segnale stesso. Infatti, il segnale originale viene combinato con una sequenza pseudo-randomica di valori (spreading sequence o chip sequence). Tale combinazione genera un segnale a banda estesa, conferendo alla trasmissione una maggiore robustezza contro disturbi, interferenze e tentativi di intercettazione. Nello specifico, il segnale trasmesso è solitamente una XOR tra i bit che contengono informazione e la chip sequence. In Figura 7 è mostrato un esempio di come il segnale viene costruito.

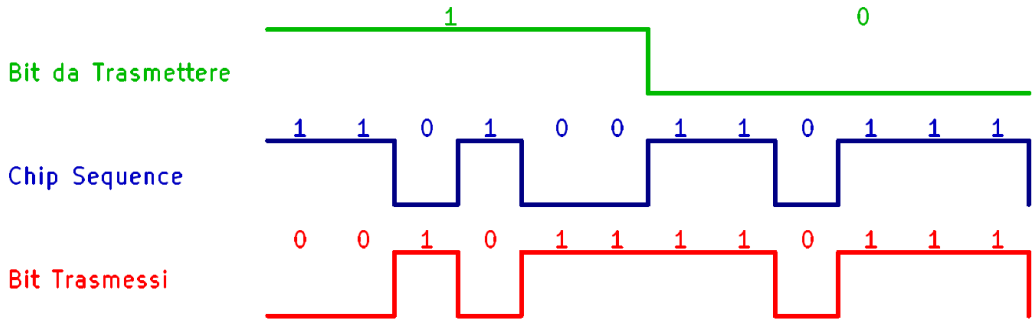

Figura 7: Esempio di costruzione del segnale in una modulazione DSSS.

A seguito dell'elaborazione, il segnale viene effettivamente trasmesso tramite una modulazione più tradizione, ad esempio una FSK o una PSK. Per recuperare le informazioni originali, il ricevitore utilizza la stessa chip sequence per moltiplicare il segnale ricevuto, eseguendo il processo di despreading durante la demodulazione. Poiché la sequenza di chip è significativamente più lunga rispetto ai bit del messaggio da trasmettere, si introduce una notevole ridondanza nella trasmissione. Questa ridondanza rende il sistema più robusto poiché spalma il segnale su un banda di frequenze molto ampia; inoltre, grazie alla correlazione con la chip sequence, il ricevitore è in grado di ricostruire il segnale originale anche se alcuni bit sono corrotti o influenzati da interferenze. Infine, il codice di spreading pseudo-casuale rende difficile intercettare e decodificare il segnale, garantendo una trasmissione più riservata.

La macchina di collaudo che si andrà a sviluppare avrà come base di partenza proprio l'SC8R, al fine di analizzare le risposte radio fornite dal contatto magnetico a seguito degli stimoli. La progettazione verrà approfondita nella Sezione 4.

3.3 Specifiche tecniche

Il seguente elenco puntato rappresenta le specifiche tecniche che la macchina di collaudo deve soddisfare. In seguito verranno approfondite e dettagliate.

- Specifiche funzionali:
  - Funzionalità di test supportate
  - Capacità di eseguire test automatizzati
  - Precisione e accuratezza delle misurazioni
- Hardware:
  - Dimensioni e peso ridotte
  - Riduzione dei costi sui componenti elettronici principali
  - Sviluppo di interfacce di input/output
  - Sviluppo di interfacce di comunicazione con i dispositivi in test
  - Requisiti di alimentazione elettrica
- Firmware e software:
  - Struttura modulare e mantenibile del firmware
  - Efficienza degli algoritmi di controllo
  - Interfaccia utente intuitiva
- Sicurezza, affidabilità e manutenibilità:
  - Caratteristiche di sicurezza implementate per prevenire infortuni
  - Robustezza della macchina
  - Semplicità di sostituzione dei componenti difettosi
- Reportistica:
  - Tipologie di report generati (non solo relativi ai test)
  - Formato e frequenza di generazione dei report
  - Backup dei risultati
- Documentazione:
  - Manuale dell'utente
  - Manuale di installazione
  - Specifiche tecniche
  - Documentazione di manutenzione

3.3.1 Specifiche funzionali

La macchina di collaudo ha lo scopo di testare autonomamente e interamente il CS3965. Le parti principali sono state esposte nella Sezione 3.1. Tutti i dispositivi sono inizialmente vergini; questo significa che non sono mai stati programmati precedentemente. Il primo step funzionale consiste nel caricare il corretto firmware, il programma essenziale che consente di eseguire le routine legate alla periferica. Il secondo step consiste nella misura degli assorbimenti sia quando il DUT si trova a riposo, sia quando avviene una trasmissione, in cui il consumo di corrente è massimo. La fase successiva riguarda la calibrazione (per ridurre il più possibile l'offset di frequenza) e la verifica della comunicazione radio attraverso una stima della Received Signal Strength Indication (RSSI). Viene svolto un'ulteriore verifica riguardante il funzionamento dell'ADC, tramite la lettura della tensione di alimentazione (il controllo batteria). Terminate queste procedure iniziali, si passa al collaudo vero e proprio. In particolare, devono essere sottoposti controlli al tamper, all'ingresso aux, al reed bypass, al led di segnalazione, al sensore reed e infine all'accelerometro.

Qualora tutti gli step fossero andati a buon fine, alla periferica deve essere assegnato un codice identificativo univoco. L'ultima fase consiste nella scrittura della nuova etichetta e nella verifica dell'effettivo update.

Oltre alla procedura completa (quella appena descritta), ogni test dovrebbe essere eseguibile singolarmente: questa soluzione, non essenziale in fase di produzione poiché il dispositivo deve superare tutti i controlli per essere commercializzato, potrebbe risultare utile per una fase di debug.

Si sottolinea che ogni test svolto deve essere ripetibile, accurato ed affidabile. I risultati prodotti devono essere simili (se non uguali) se svolti nelle stesse condizioni. Inoltre, i dati ottenuti devono corrispondere ai valori acquisiti tramite l'utilizzo di altri strumenti di misura o altre procedure, pertanto devono rappresentare ciò che effettivamente si propongono di misurare. Dunque, i risultati devono essere verificabili e privi di errori significativi.

3.3.2 Specifiche hardware

La macchina di collaudo deve avere dimensioni e peso ridotti per facilitarne la portabilità, permettendo all'operatore di poter lavorare anche da casa. La compattezza rappresenta quindi un aspetto fondamentale.

Il punto di partenza della nuova scheda è l'SC8R. Esso è stato rivisitato al fine di ottenere nuovo circuito stampato con l'elettronica necessaria per poter svolgere un test completo con tutti gli step descritti nelle specifiche funzionali. Inoltre, si deve prevedere tutta la componentistica necessaria per l'alimentazione del sistema e la gestione della comunicazione da e verso il mondo esterno.

Poiché la parte elettronica è inserita all'interno di un contenitore plastico, devono essere previste tutte le interfacce di input e output necessarie per garantire la massima flessibilità e facilità d'uso per l'utente. Sono pertanto necessarie delle porte per il collegamento di un

monitor esterno dal quale è possibile avviare il test e visionare i risultati, mouse e tastiera, un barcode reader per la lettura dei codici da applicare ai dispositivi e una porta ethernet per il collegamento alla rete. Maggiori dettagli saranno approfonditi nella sezione relativa all'implementazione e allo sviluppo.

3.3.3 Specifiche firmware e software

Il firmware di partenza risulta quello dell'SC8R. Si devono integrare tutte le funzioni necessarie per poter controllare opportunamente gli elementi introdotti per soddisfare specifiche funzionali e meccaniche. Il codice dovrebbe essere stabile e robusto, senza la presenza di crash o interruzioni frequenti, modulare e flessibile per permettere l'integrazione di ulteriori funzionalità, mantenibile ed efficiente. Il tempo di esecuzione dell'intero test dovrebbe essere inferiore rispetto a quello dell'attuale linea, permettendo una maggiore produzione.

Per quanto riguarda la parte software e interfaccia utente, i risultati dovrebbero essere chiari, ben visibili ed inequivocabili. Un aspetto importante è la facilità d'uso, poiché anche un operatore non specializzato dovrebbe essere in grado di interpretare i dati ottenuti dal ciclo di collaudo.

3.3.4 Sicurezza, affidabilità e manutenibilità

La macchina di test non deve rappresentare un pericolo per l'operatore, sia dal punto di vista elettrico che meccanico. Siccome deve essere trasportabile, anche la robustezza ad urti e polvere è un aspetto chiave, così come la protezione da disturbi provenienti da altri macchinari al fine di garantire le corrette comunicazioni via radio.

Infine, poiché saranno presenti componenti soggetti ad usura, l'aspetto della manutenzione non va trascurato. Dispositivi danneggiati o guasti dovrebbero essere sostituiti facilmente, senza complicazioni e senza necessità di lunghi interventi anche per problemi di piccola entità. Con questa accortezza, si riduce al minimo il tempo di inattività del macchinario e si ottimizza maggiormente la produzione.

3.3.5 Reportistica

Al termine del ciclo di collaudo, i risultati ottenuti devono essere mostrati a video e salvati in locale. Ogni periferica dovrà avere un file associato che certifica i controlli effettuati su quel dispositivo. I report generati devono riassumere i test svolti e il relativo esito per ogni singolo step, nonché le misure ottenute. Anche i test falliti dovrebbero essere tracciati, al fine di ottenere una statistica degli scarti.

Oltre al salvataggio in locale, i risultati devono essere memorizzati anche nell'archivio dell'azienda. Questo permette di rendere i file consultabili facilmente e velocemente, e inoltre garantisce che in caso di guasto alla macchina di collaudo, tutti i test non presenti altrove non vengano persi definitivamente.

Deve essere prevista un'ulteriore reportistica per le operazioni di contorno gestite dalla macchina, per monitorare e tracciare lo svolgimento di backup o l'aggiornamento dei firmware per i DUT.

3.3.6 Documentazione

Uno degli ultimi aspetti da tenere in considerazione e la documentazione relativa all'intero progetto. Deve essere prevista una guida utente che indichi chiaramente e dettagliatamente come utilizzare il dispositivo passo passo e quali sono le funzionalità disponibili.

Devono essere previsti documenti relativi al cablaggio, assemblaggio e installazione, ai componenti utilizzati per facilitare l'approvvigionamento e la sostituzione in caso di guasto e un manuale di manutenzione che indichi quali potrebbero essere le cause per cui è necessario un intervento. Infine, deve essere valutato l'intervallo di taratura/ricalibrazione per evitare il degradamento dei risultati causati dalle derive temporali della componentistica di macchina.

4 Implementazione e sviluppo

4.1 Implementazione hardware

Inizialmente l'attenzione si è concentrata sul collaudo delle componenti hardware del contatto magnetico. Lo sviluppo è partito da un SC8R (descritto nella Sezione 3.2), al quale sono state apportate diverse modifiche, tra cui lo spostamento di alcuni componenti già presenti (per motivi di spazio) e l'aggiunta dell'elettronica necessaria per svolgere il collaudo dei diversi sensori.

La nuova scheda si dovrà comportare come un SC8R in grado di verificare il corretto funzionamento delle varie parti del contatto 3V. In particolare, si deve prevedere della circuiteria per testare l'assorbimento in corrente (sia quando il microcontrollore si trova a riposo sia durante una comunicazione radio), il tamper, l'aux, reed e reed bypass , LED e infine l'accelerometro. In seguito verranno dettagliate le scelte progettuali adottate attraverso uno schema circuitale e una relativa descrizione, includendo le motivazioni che hanno portato a tali decisioni.

La scheda responsabile dello svolgimento degli step funzionali sarà identificata con il codice CS3981.

4.1.1 Assorbimenti di corrente

Il CS3965, come evidenziato precedentemente, si tratta di un dispositivo alimentato a batteria. Al fine di preservare la carica per il più lungo tempo possibile, la routine del μC include una fase di riposo a basso consumo energetico con un risveglio periodico ogni 15,625ms. Ad ogni risveglio, la periferica controlla lo stato di tutti i suoi sensori e se ci fosse un cambio di stato, avvia una trasmissione, al termine della quale torna nella fase di riposo.

Nella Figura 8 è rappresentata la corrente assorbita durante un normale ciclo di risveglio del microcontrollore. La misura è stata effettuata con una resistenza di shunt da 153,04Ω. Inoltre, l'assorbimento medio è stato acquisito anche tramite un amperometro, la cui misura ha fornito una corrente media a riposo che ricade nell'intervallo 60-80μA. I picchi veri e propri corrispondono al passaggio da uno stato di riposo ad una condizione di pieno regime. L'andamento esponenziale è associato alle curva di scarica dei componenti passivi presenti, quali condensatori.

La Figura 9 evidenza invece una trasmissione del contatto 868. In questo caso, la resistenza di shunt ha un valore di 0,993Ω. Si possono notare tre principali fasi: la prima, in cui la periferica è effettivamente in trasmissione (il consumo è intorno a 80mA); la seconda, in cui il CS3965 entra in ricezione per un'eventuale risposta da parte dell'SC8R, con un consumo intorno a 10mA; la terza ed ultima fase, in cui il microcontrollore torna nuovamente a riposo e la routine riparte dall'inizio.

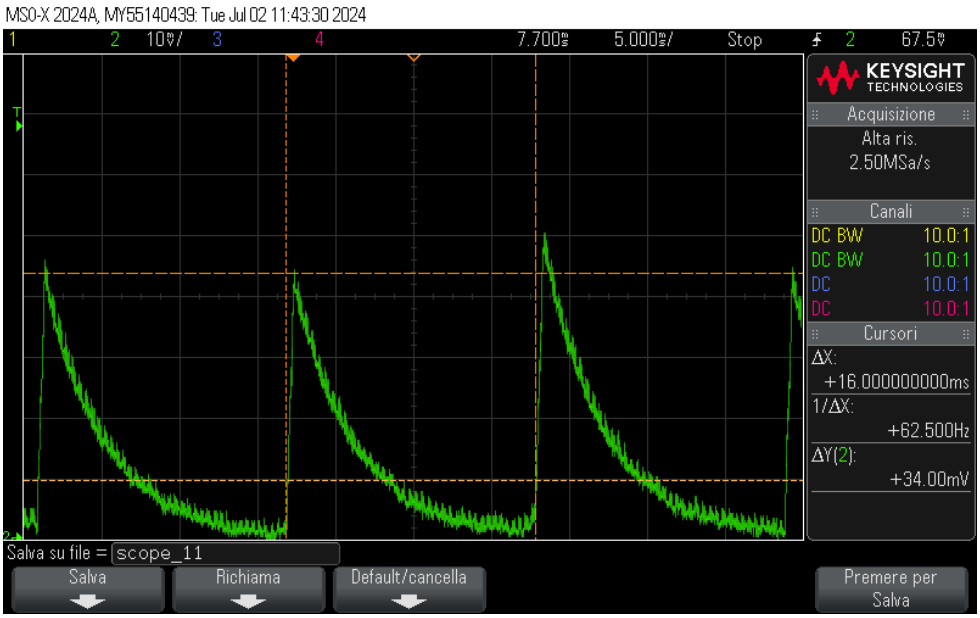

Figura 8: Routine di risveglio del microcontrollore.

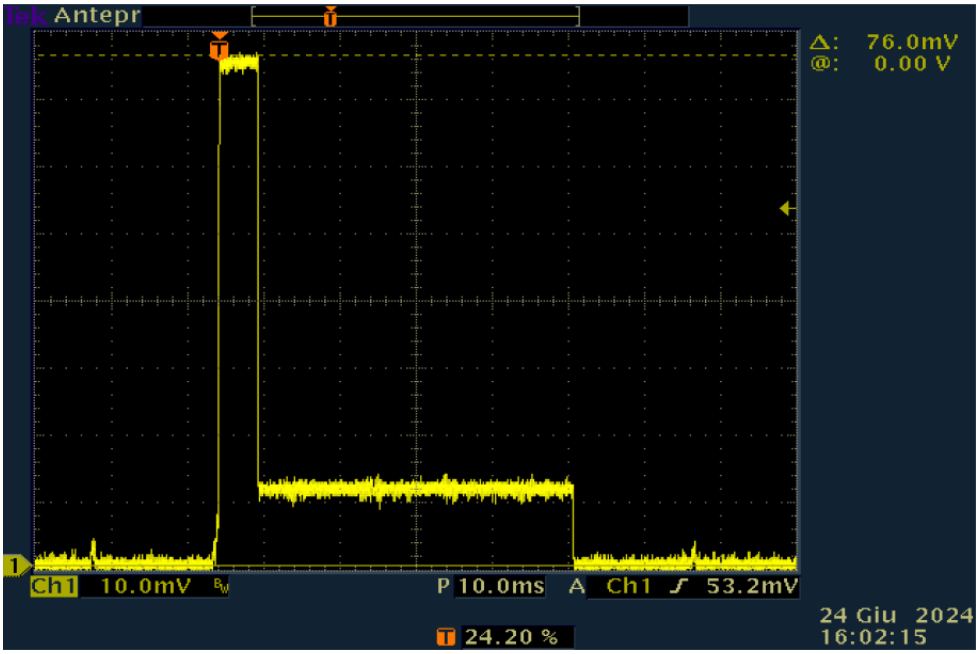

Figura 9: Fase di trasmissione del contatto 868.

Al fine di acquisire le misure relative ai consumi, è stato utilizzato lo schema circuitale mostrato in Figura 10. La macchina di collaudo deve risultare affidabile nel fornire i risultati, ma non si tratta di uno strumento di misura. Pertanto, è stata progettata una soluzione composta da componenti le cui performance sono in linea con le specifiche di costo.

Nel ramo di controllo, un MOS pilota un relè che fornisce (o rimuove) alimentazione al DUT. La corrente che scorre all'interno della periferica in collaudo si richiude sulla resistenza R16, la cui caduta di tensione verrà acquisita da un ADC.

Il MOS è controllato da un GPIO del microcontrollore. Sono presenti due resistori in uscita dal pin di controllo del transistor, uno in serie al gate, mentre l'altro tra gate e GND. Il primo ha lo scopo di limitare la corrente di carica del condensatore del MOS, alleggerendo la porta logica di pilotaggio. Infatti, il gate del MOS è un condensatore: senza una resistenza in serie, il picco di corrente potrebbe risultare elevato durante la commutazione, causando stress sul driver. Con la resistenza in serie, la corrente si limita a $I_{peak} = \frac{V_{gs}}{R}$. Il principale svantaggio sarebbe la dilatazione delle costanti di tempo $\tau = R \cdot C$ per via del valore pari a 10kΩ; in questo caso (come anche per altre applicazioni descritte in seguito), dato che non passano segnali di dato ad alta frequenza (il transistor è usato come interruttore ON/OFF), questo problema non si pone. Il secondo resistore ha la funzione di evitare che il gate possa trovarsi flottante, ad esempio in caso di malfunzionamento del circuito di controllo. In tale situazione, il gate verrebbe collegato a massa, pertanto il transistor risulterebbe interdetto. Il valore è stato dimensionato per avere una $V_{gs} > V_{th}$ in ogni condizione. Da datasheet, il worst case (upper bound) è pari a $V_{th} = 1.5V$. Quando il GPIO è a livello alto, la tensione è pari a 3.3V. Considerando il partitore, $V_{gs} = 1.98$V e pertanto risulta ampiamente marginata.

Il relè funziona come un interruttore: quando si eccita la bobina, il contatto mobile viene attratto e il relativo circuito si chiude. Trattandosi di un carico induttivo, è stato previsto il diodo di ricircolo in controfase alla bobina, al fine di fornire un percorso alternativo all'energia immagazzinata nell'induttore, evitando le sovratensioni quando viene diseccitato. La corrente necessaria per eccitare il relè è intorno ai 30mA, pertanto è necessario un diodo che sopporti una forward current almeno di pari valore. Inoltre, bisogna tenere in conto la massima reverse voltage ai suoi capi. La scelta è ricaduta su un 1N4007, che presenta una massima corrente di 1A e una reverse voltage fino a 1000V (decisamente marginata per questa applicazione).

Il relè a riposo collega l'ingresso non invertente dell'amplificatore operazionale a GND: questo servirà in fase di acquisizione al fine di compensare l'offset dell'operazionale. Invece, quando si eccita il relè, la corrente nel DUT si richiude sulla R16, causando una caduta di tensione sull'ingresso non invertente dell'operazionale. La scelta del relè è motivata dal fatto che esso si comporta come interruttore ideale, mentre un MOS ha delle correnti di perdita anche quando è interdetto che potrebbero influenzare la misura, specialmente per correnti basse. L'uso dell'amplificatore operazionale consiste nel disaccoppiare il carico in

ingresso dall'ADC. Nonostante il convertitore abbia un'impedenza di ingresso sufficientemente alta (nell'ordine di qualche MΩ, come evidenziato sul datasheet), l'opAmp migliora tale caratteristica. Infine, l'inserimento di un connettore permetterà in fase di cablaggio di collegare correttamente il DUT alla porzione di circuito appena descritta.

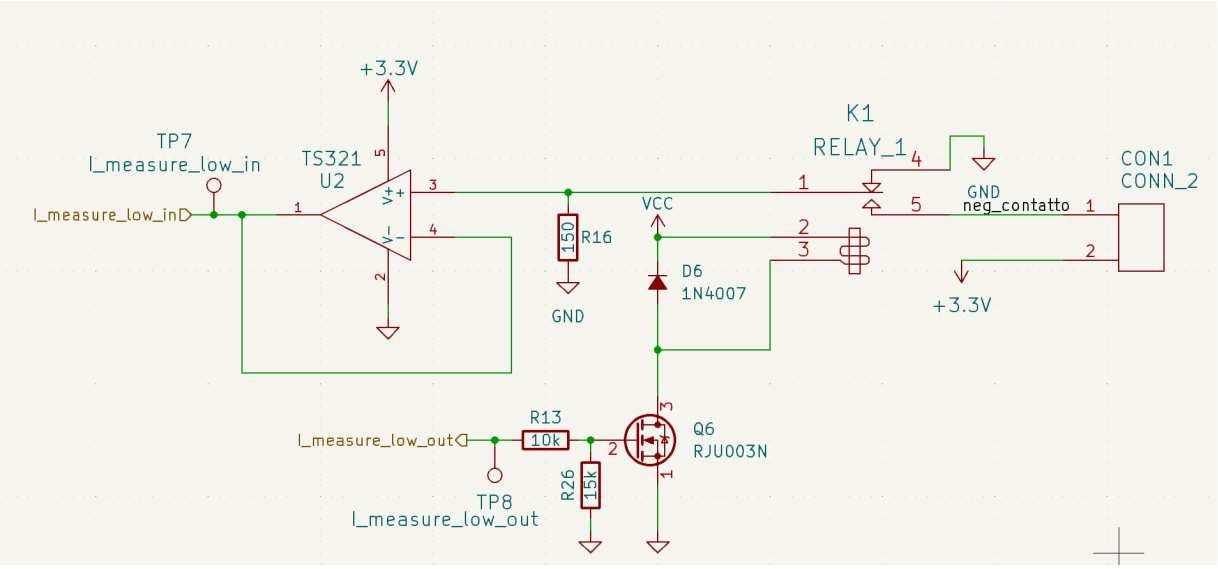

Figura 10: Schema circuitale per la misura di corrente in fase di riposo.

Il valore del componente R16 è stato scelto al fine di avere una massima tensione di circa 0,5V; questa caduta viene sottratta alla periferica, ma è stata dimensionata al fine di alimentare correttamente il dispositivo ed essere a metà della dinamica dell'ADC. Analizzando i grafici dei consumi riportati precedentemente, la scelta è ricaduta su una 150Ω.

Un analogo circuito è stato progettato per la misura della corrente durante una trasmissione. L'unica differenza risiede nel valore del resistore tra l'ingresso non invertente dell'operazionale e GND, pari a 4,7Ω.

Infine, si ha un ultimo circuito che serve a fornire alimentazione durante il normale funzionamento della periferica, senza la presenza di resistenze di shunt per la valutazione delle correnti. In questo caso, non viene sottratta alcuna tensione alla periferica. L'alternanza dei tre circuiti durante il collaudo completo è gestita opportunamente via firmware, al fine di garantire la corretta alimentazione ed evitare che la tensione effettivamente presente sul DUT sia superiore alla V_{min} del microcontrollore, per evitare che si resetti.

4.1.2 Tamper, aux e reed bypass

La Figura 11 mostra il circuito per il collaudo del tamper. La medesima configurazione è stata replicata anche per l'ingresso ausiliario e per il reed bypass, poiché tutti questi sensori funzionano in modo analogo.

La parte di pilotaggio è la stessa descritta nel paragrafo precedente. Si evidenzia invece la differenza sulla parte di attuazione. La macchina di collaudo dispone di aghi che vanno a contattare direttamente le piazzole dei componenti da testare quando viene armata. Questi aghi sono collegati ai connettori (in questo caso il CON3) tramite dei cavi. Quando il GPIO attiva il relè, il circuito viene chiuso, simulando la pressione del pulsante. Confrontando il valore trasmesso dal contatto 868 con quello che la macchina si aspetta, si riesce a fornire l'esito del test.

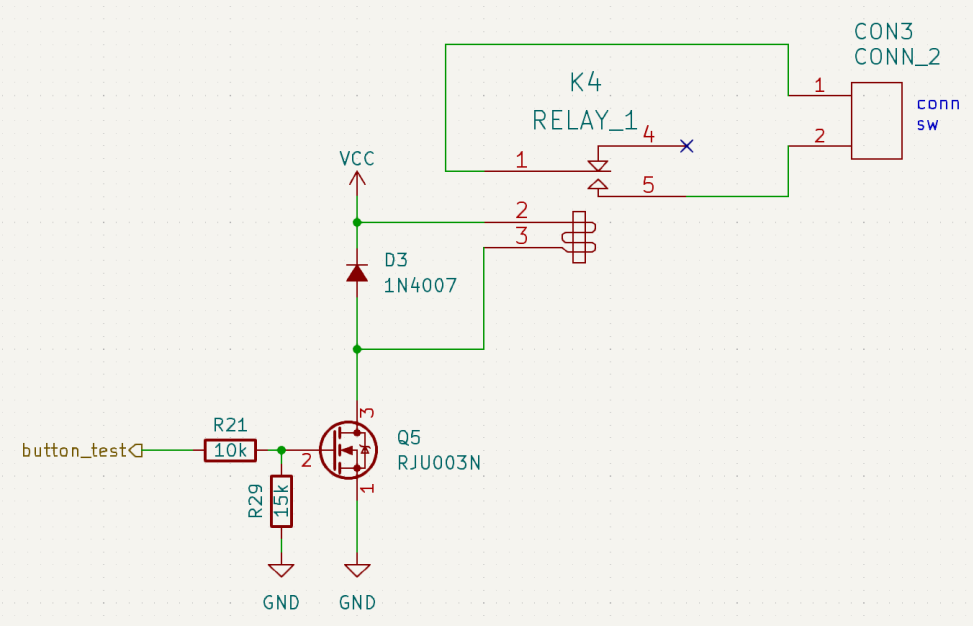

Figura 11: Schema circuitale per il test del tamper.

4.1.3 LED

Ad ogni stimolazione, il CS3965 effettua una trasmissione e il LED si accende per un breve intervallo. Quando si collauda il contatto magnetico, esso entra in una modalità differente, detta `TEST MODE`. In questa configurazione, il LED rimane acceso fisso. La verifica dell'indicatore luminoso può quindi essere effettuata senza la necessità di forzare una stimolazione. Il circuito progettato e adottato è mostrato in Figura 12.

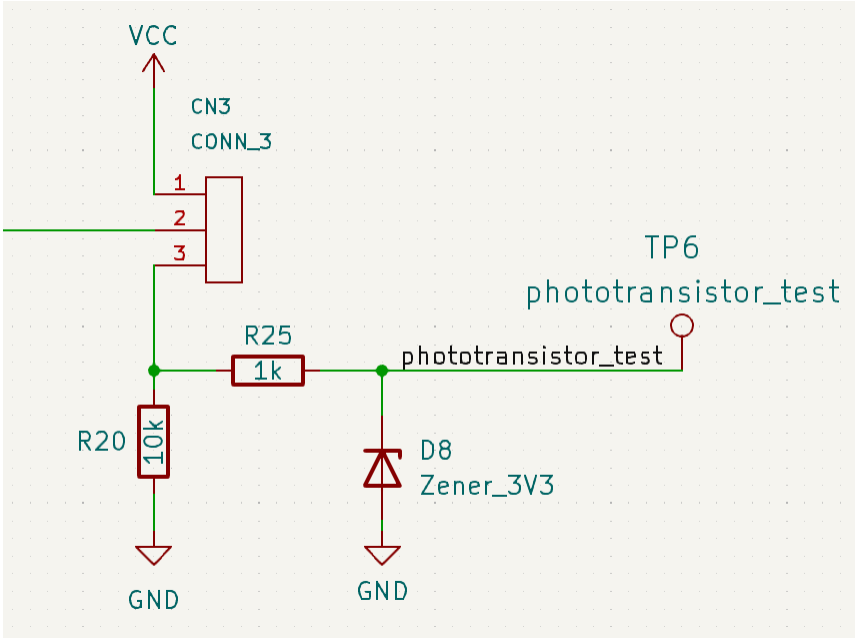

Figura 12: Schema circuitale per il test del LED di segnalazione.

Tramite un phototransistor si può monitorare l'intensità luminosa di una sorgente. Il simbolo è raffigurato nell'immagine 13. I fotoni che incidono sulla base generano elettroni e la corrente prodotta viene poi amplificata. Attraverso il componente R20, si converte la corrente in tensione, che potrà essere acquisita dal microcontrollore.

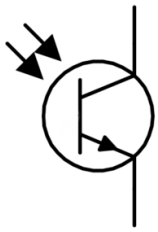

Figura 13: Simbolo del phototransistor.

Il diodo zener da 3,3V è necessario per proteggere il GPIO di ingresso. Siccome VCC corrisponde a circa 13V e la corrente generata dal phototransistor è proporzionale all'intensità luminosa, si potrebbe avere una tensione maggiore di quella sopportabile dal pin del μC. Per evitare il suo danneggiamento, qualora si salga sopra i 3,3V, lo zener entra in conduzione e la tensione viene mantenuta constante.

Il phototransistor non è presente nel circuito mostrato in Figura 12, ma è cablato esternamente alla scheda tramite dei fili, per via del fatto che deve essere posto esattamente sopra il LED del DUT in un apposito alloggiamento. Inoltre, in caso di malfunzionamento può essere sostituito con più facilità.

4.1.4 Reed

Il sensore magnetico permette di segnalare quando si ha un'apertura (o chiusura) di un infisso. Allontanando e avvicinando un magnete è quindi possibile stimolare il reed.

Il componente più appropriato per svolgere questo ruolo è un servomotore. Si tratta di un dispositivo che consente di controllare la posizione angolare. É costituito da un un motore (con un riduttore) e da un circuito di controllo. Quest'ultimo driver consiste essenzialmente in un potenziometro, un comparatore e un micro. Il segnale di controllo (in PWM) corrispondente al riferimento, viene convertito in tensione e confrontato con l'uscita attuale del potenziometro, che rappresenta la posizione corrente. A questo punto si fornisce l'alimentazione al motore al fine di raggiungere la posizione di riferimento. Una volta ottenuta, questa viene mantenuta. Introducendo un magnete sull'estremità del braccio pilotato è possibile stimolare il sensore reed, emulando l'apertura e la chiusura del serramento.

In Figura 14 si mostra il circuito di controllo del servomotore. Poiché il componente ha già l'elettronica necessaria al suo interno, è sufficiente fornire alimentazione e il segnale di controllo. Nonostante l'alimentazione debba essere compresa tra 4.8V e 7.2V, il controllo può essere tranquillamente a 3.3V senza avere problemi sui livelli logici. Infine, per facilitare la sostituzione, è stato previsto un connettore. Sarebbe stato possibile implementare un circuito di controllo di coppia, per prevenire il danneggiamento dei componenti meccanici in situazioni di stress eccessivo per il motore o per limitare la forza evitando infortuni agli operatori. Dalla valutazione dei potenziali rischi meccanici ed elettrici, è emerso che questo non costituisce un problema, pertanto è stato scelto di evitare di introdurre ulteriore elettronica.

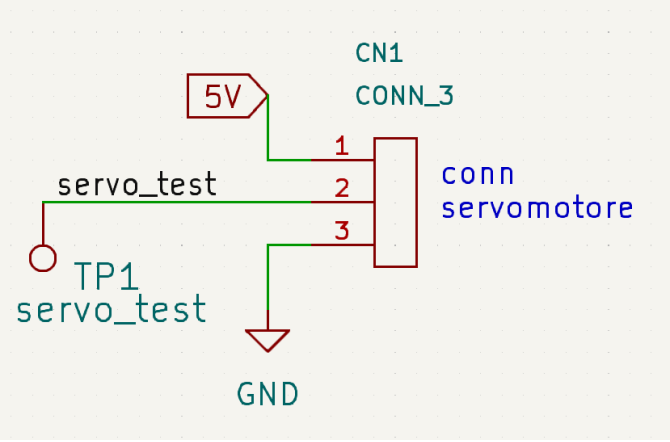

Figura 14: Schema circuitale del servomotore.

La forma d'onda del servomotore (Figura 15), durante le fasi di apertura e chiusura, è stata acquisita utilizzando una resistenza di shunt da 3,3Ω, al fine di valutare gli assorbimenti massimi. Essendo le due fasi vicine tra loro, il servomotore mantiene una certa

inerzia, giustificando così il picco di corrente più alto nella fase di chiusura. In Figura 16 si mostra un ingrandimento riguardante la fase di ritorno, su cui sono stati svolti i calcoli per la stima della corrente massima assorbita:

$$I_{spunto} = \frac{V_{picco}}{R_{shunt}} = \frac{1,74V}{3,3\Omega} = 527\text{mA}$$

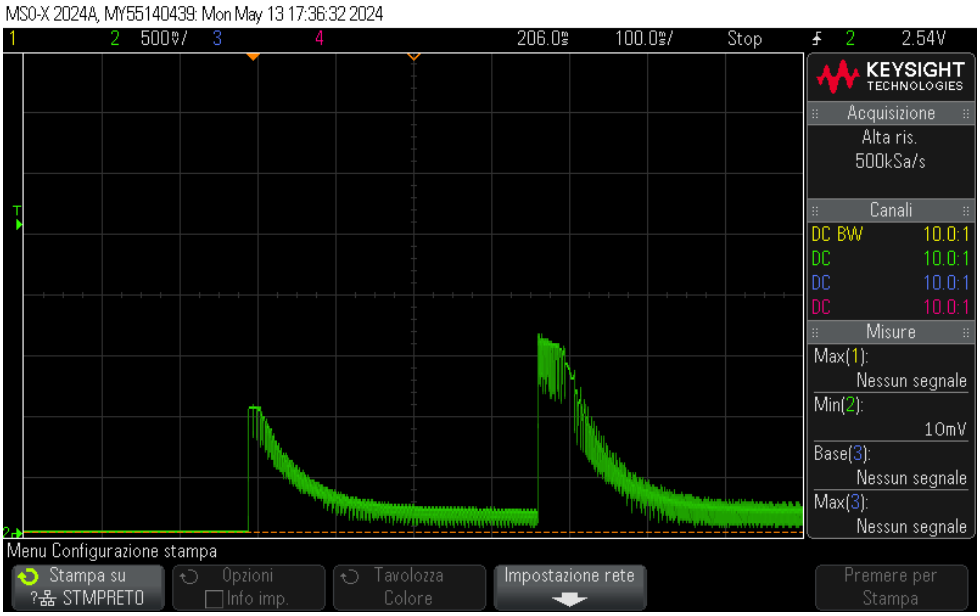

Figura 15: Forma d'onda durante fase di apertura e chiusura servomotore.

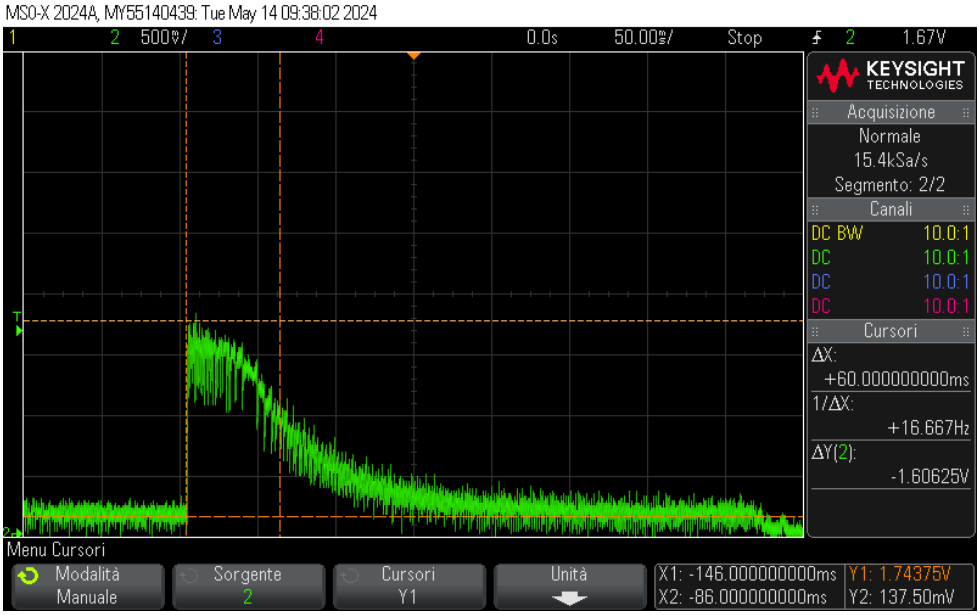

Figura 16: Zoom della fase di chiusura del servomotore.

4.1.5 Accelerometro

Come precedentemente descritto, l'accelerometro serve per segnalare inclinazioni e vibrazioni. Il modo più efficace per collaudare tale sensore è mediante l'uso di un motore vibrazionale, che attraverso dei pesi eccentrici solidali all'albero motore genera vibrazioni meccaniche quando è in movimento.

In Figura 17 è presente lo schema del circuito. La parte di controllo rimane invariata, così come l'utilizzo del diodo in controfase e il morsetto per poter collegare esternamente il motore.

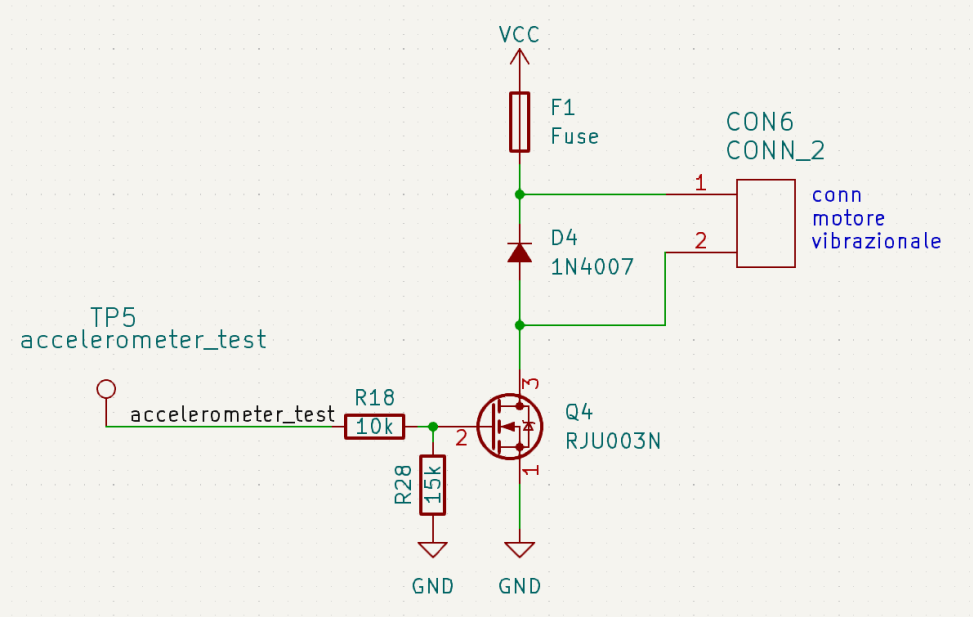

Figura 17: Schema circuitale per la verifica dell'accelerometro.

Bisogna porre l'attenzione sul fusibile e sul suo dimensionamento. Alimentando il motore a 13,2V, il consumo a regime è intorno ai 70-80mA (consumo verificato tramite amperometro). Per valutare lo spunto, invece, è stata posta una resistenza di shunt in serie al motore di valore pari a 0.883Ω; tramite l'oscilloscopio è stata analizzata la caduta di tensione, dal quale è possibile valutare il picco di corrente. Come si nota dalla Figura 18, l'assorbimento istantaneo iniziale è:

$$I_{spunto} = \frac{V_{picco}}{R_{shunt}} = \frac{1,08V}{0,883\Omega} = 1,22\text{A}$$

Siccome il fusibile (un PTC) ha un tempo di intervento piuttosto lungo dovuto alle costanti di tempo delle temperature, durante lo spunto non riesce ad entrare in funzione. Esso deve garantire una protezione in caso di guasto e non durante il normale funzionamento.

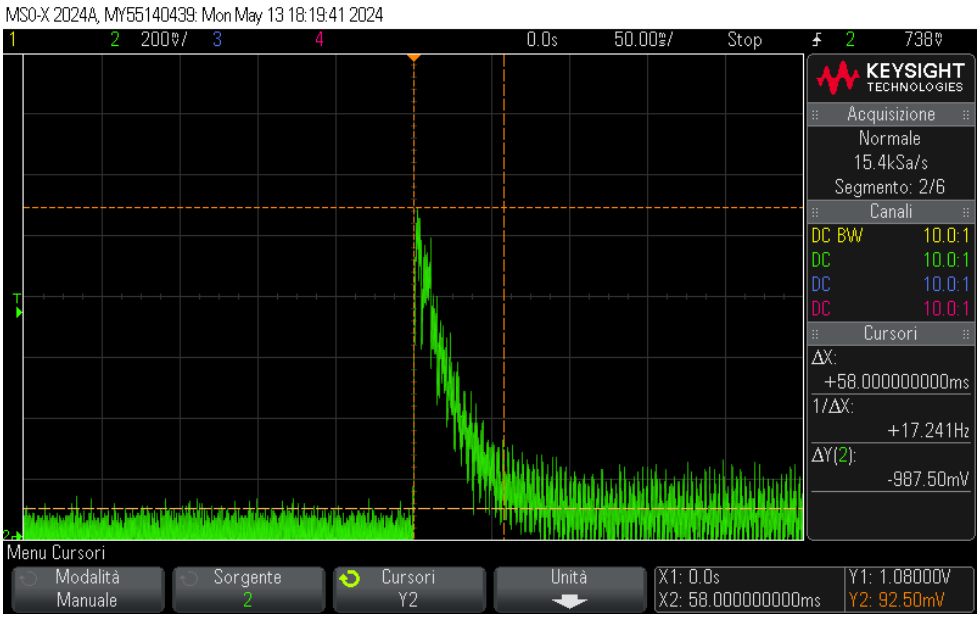

Figura 18: Assorbimento del motore durante lo spunto.

4.1.6 Alimentazioni e μC

Oltre alle parti necessarie per svolgere il collaudo, di seguito si riportano gli schematici relativi all'alimentazione, al microcontrollore e ai circuiti discussi precedentemente, mostrati nel loro insieme.

In Figura 19 è mostrata la parte di potenza. L'alimentazione principale viene fornita tramite il morsetto J1, filtrata per ridurre rumore e interferenze e abbassata con un LDO a 3.3V. Il diodo in serie ai 13.8V ha lo scopo di evitare danni alla scheda in caso di inversione dei cavi sul morsetto. Inoltre, in presenza di problemi a valle, si potrebbero avere potenziali reverse sul morsetto di ingresso che potrebbero arrivare anche ad altre schede connesse. Il diodo serve per evitare anche questa problematica. Il 3.3V si distribuisce e alimenta il microcontrollore, fornendo anche i riferimenti di tensione.
Infine, si può notare la linea "current loop", una tecnologia di comunicazione di cui Saet I.S. è proprietaria, non utilizzata in questa applicazione. Si tratta di una linea per la comunicazione seriale, half duplex. In un impianto, tale linea serve per lo scambio dei dati tra il concentratore SC8R e la centrale di controllo.

Inoltre, si può notare il corredo di condensatori, consigliati dal costruttore del microcontrollore.

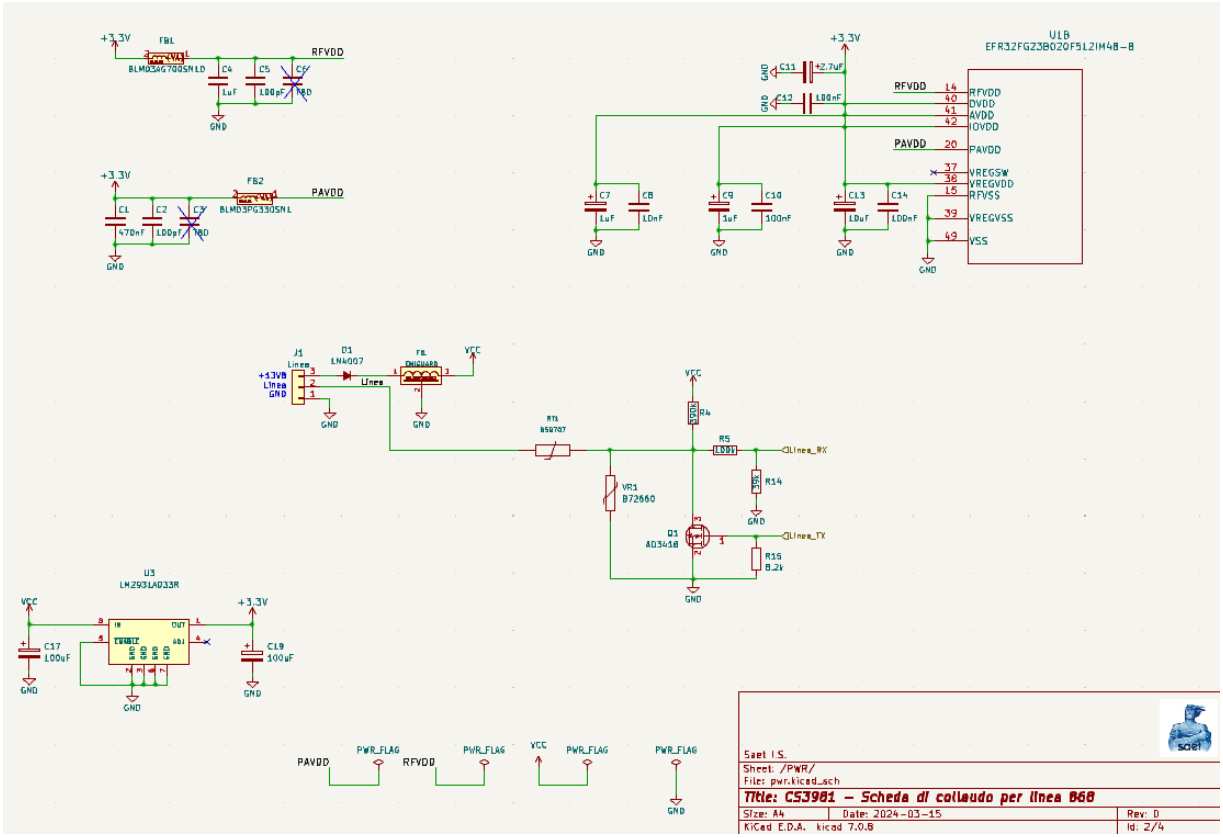

Figura 19: Schema circuitale per alimentazioni.

Nella Figura 20 viene mostrato lo schematico del micro, insieme ad alcune parti precedentemente descritte. In particolare, si possono notare: due quarzi, uno a bassa frequenza (32kHz) e uno ad alta frequenza (40MHz); l'antenna, per le comunicazioni radio, con il relativo circuito di condizionamento; alcuni LED utilizzati per segnalare l'arrivo di pacchetti in ricezione e altri GPIO connessi allo strip di programmazione per poter aggiornare il firmware (attraverso SWCLK e SWDIO) o comunicare in seriale con un altro dispositivo.

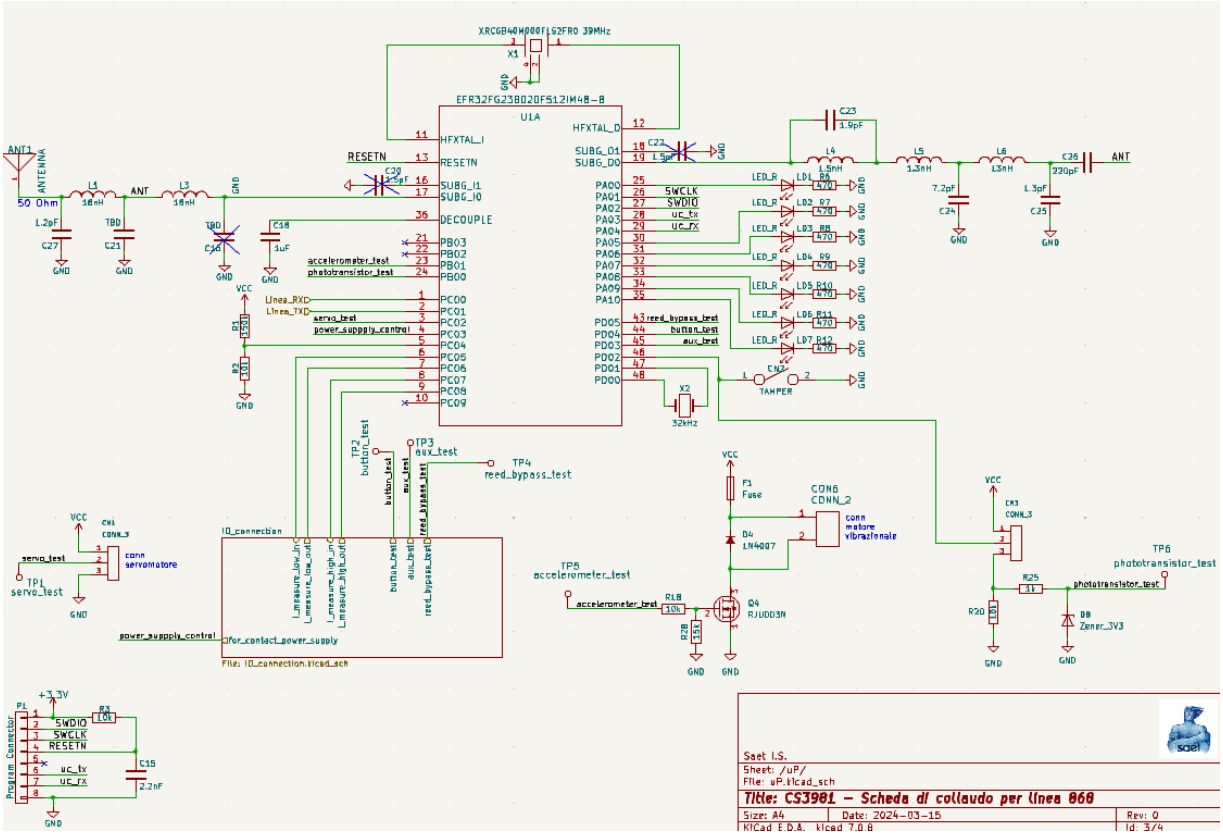

Figura 20: Schema circuitale del microcontrollore.

Infine, in Figura 21 è mostrato il foglio completo dei circuiti che verranno utilizzati per il collaudo effettivo, come descritto ed evidenziato in precedenza.

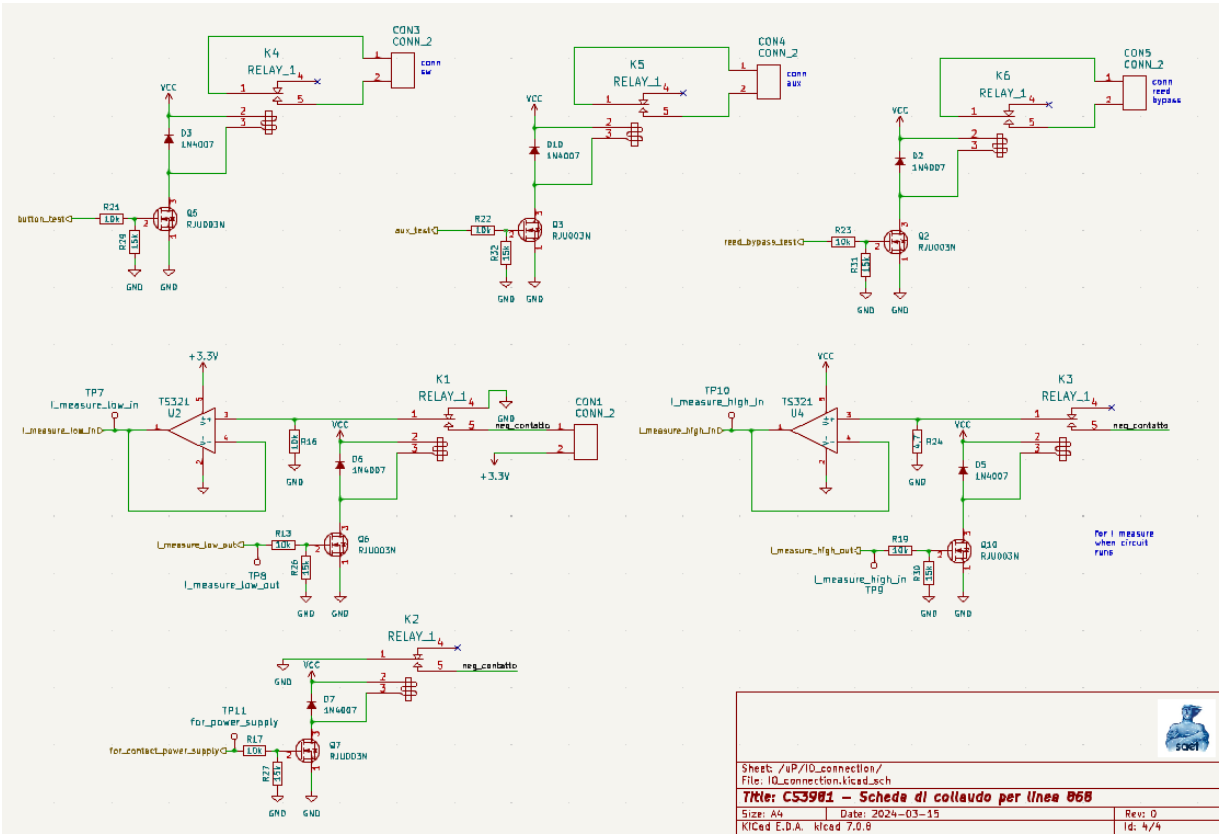

Figura 21: Schema circuitale d'insieme per gli step funzionali.

4.2 Architettura completa

Salendo ad un livello di astrazione superiore, si è passati allo sviluppo dell'intero sistema, nel quale la scheda CS3981 precedentemente descritta verrà inserita. In Figura 22 viene mostrato lo schema elettrico completo. Successivamente verranno approfonditamente trattate le varie parti che lo compongono e le scelte progettuali adottate.

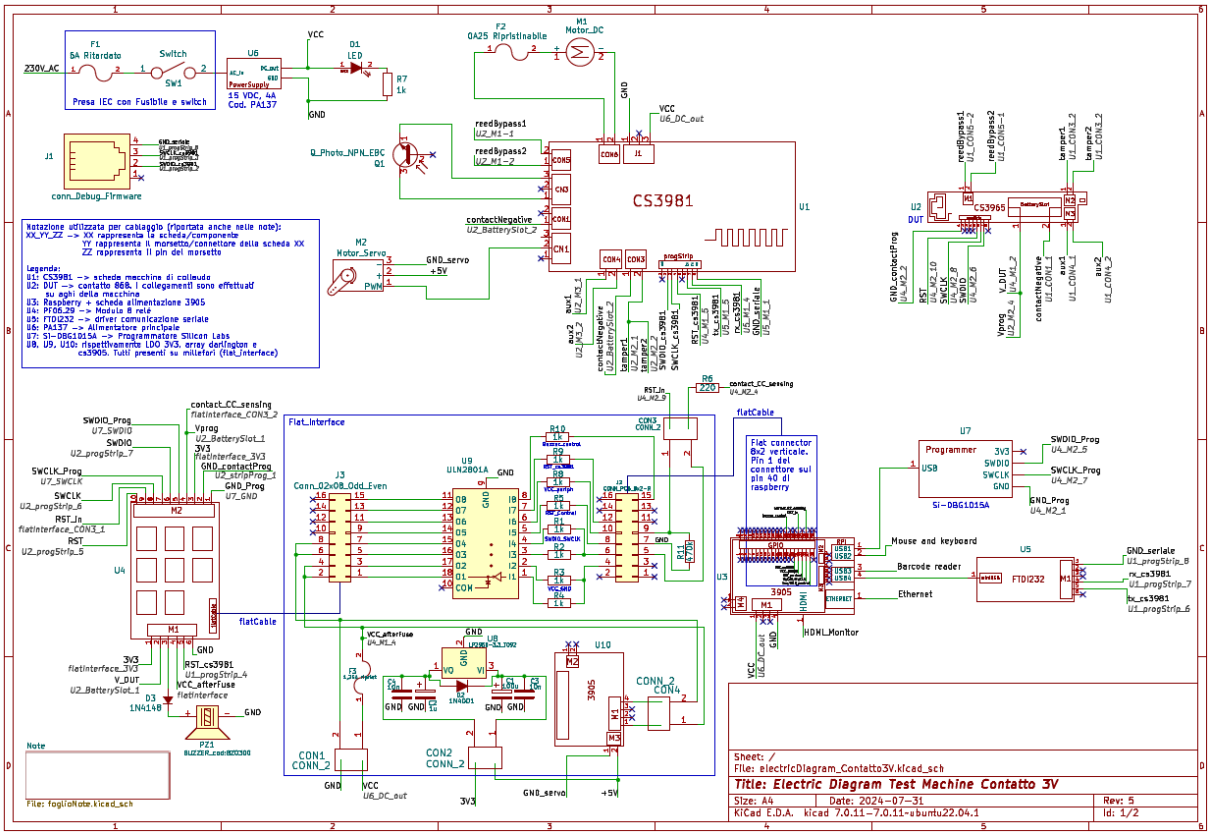

Figura 22: Schema elettrico del sistema completo.

L'intero sistema sarà inserito all'interno di un contenitore plastico che sarà dotato di connettori per interfacciarsi con l'esterno. L'utente potrà interagire con la macchina di collaudo tramite un monitor, mouse, tastiera e uno scanner barcode. È inoltre presente un connettore che consente ad un utilizzatore specializzato di aggiornare il firmware della scheda CS3981 e una presa di alimentazione, tramite la quale viene fornita l'alimentazione primaria. Al fine di soddisfare le caratteristiche e le specifiche dei componenti e delle schede utilizzate, la rete di alimentazione è stata strutturata in modo gerarchico: sono presenti tre diversi livelli di tensione (14V, 5V e 3V), progettati in modo tale che se si verificasse un problema su un livello inferiore, questo non possa propagarsi, evitando che la sorgente di energia superiore possa subire interruzioni o malfunzionamenti. Questo è stato reso possibile adottando protezioni come l'inserimento di fusibili (con curve di intervento

selettive, che garantiscono una protezione progressiva e sequenziale) o LDO con limitazione di corrente da cortocircuito. L'argomento verrà approfondito in seguito.

L'alimentazione principale è costituita da un alimentatore switching 14V, 4A. Lo schema è mostrato in Figura 23. Il valore di corrente è stato opportunamente dimensionato considerando i contributi di tutta la componentistica presente e moltiplicando per un fattore di contemporaneità stimato.

Il consumo misurato della scheda CS3981 a riposo è di circa 30mA. L'assorbimento misurato durante una trasmissione è di circa 100mA. Sulla scheda sono presenti sette relè alimentati a 12V. Da datasheet [Rel24], la corrente necessaria per mantenere eccitata la bobina è di 30mA. L'ultimo contributo significativo è dato dal motore vibrazionale, che a regime assorbe 80mA. Si ottiene quindi:

$$\begin{aligned} I_{\text{CS3981}} &= I_{quiescentCurrent} + I_{TX} + I_{rele'} + I_{motore} \\ &= 30 + 100 + 7 \cdot 30 + 80 \\ &= 420\text{mA} \end{aligned}$$

In condizioni di lavoro, ossia durante un test, due relè al massimo sono controllati contemporaneamente. Pertanto, l'assorbimento in worst case stimato è intorno ai 270mA.

Il consumo medio di una Raspberry a riposo è intorno ai 300-500mA e può salire fino a 1-1,2A con un intenso utilizzo della CPU. Inoltre, il collegamento con dispositivi USB aumenta l'assorbimento di corrente: mouse e tastiera assorbono intorno ai 100mA; discorso analogo per lo scanner barcode; programmatore firmware [Lab23b] e FTDI232 per la comunicazione seriale circa 100mA. Sommando i vari contributi si ottiene:

$$\begin{aligned} I_{\text{Raspberry}} &= I_{operating} + I_{mouseTasiera} + I_{scannerBarcode} + (I_{programmatore} + I_{FTDI232}) \\ &= 1200 + 100 + 100 + 100 \\ &= 1500\text{mA} \end{aligned}$$

Gli ultimi contributi significativi sono dati dal servomotore, dalla periferica DUT e dal modulo ad otto relè. Il servomotore appartiene alla famiglia dei modelli MG996R, viene alimentato a 5V e l'assorbimento massimo dichiarato da datasheet in condizioni operative è di circa 500mA (confermato dalla verifica sperimentale e dai calcoli svolti nella Sezione 4.1.4). Il contatto 3V da testare ha un consumo decisamente ridotto a riposo (si tratta di un dispositivo low power); durante una trasmissione radio, però, si assorbono 80-100mA per un breve intervallo di tempo. Per il modulo otto relè, il discorso è analogo al caso del CS3981. Il massimo consumo è intorno a $I_{ottoRele'} = 8 \cdot I_{bobina} = 240\text{mA}$. Considerando un fattore di contemporaneità del 60-65%, il consumo massimo è di 160mA.

I consumi finali vengono riassunti nella tabella 1.

Componente	Assorbimento a riposo [mA]	Massimo assorbimento [mA]
CS3981	30	270
Raspebrry Pi 3B+	300-500	1500
Servomotore MG996R	<10	500
Modulo otto relè	0	160

Tabella 1: Tabella riassuntiva degli assorbimenti di corrente

Sommando tutti i contributi in worst case delle correnti assorbite, si ottiene:

$$\begin{aligned} I_{\text{Tot}} &= I_{CS3981} + I_{Raspberry} + I_{Servomotore} + I_{Modulo8rele'} \\ &= 270 + 1500 + 500 + 160 \\ &= 2430\text{mA} = 2,43\text{A} \end{aligned}$$

Si evidenzia che tutti i valori riportati sono riferiti a temperatura ambiente (20°C) e in aria libera. Siccome la componentistica viene inserita in un contenitore plastico chiuso che non è dotato di sistemi di dissipazione del calore particolari, l'influenza della temperatura non è trascurabile. Considerando un margine del 20% dovuto alla temperatura e un margine del 20% sul valore ottenuto, si ottiene un consumo massimo finale di circa 3,5A. Tra gli alimentatori presenti in magazzino, la scelta è ricaduta su uno switching con corrente pari a 4A.

Nello schema è possibile notare un fusibile ritardato con valore di intervento di 6A. Infatti, durante lo startup della macchina, i transitori portano ad avere picchi di corrente che potrebbero essere elevati. L'alimentatore è un grado di soddisfare questa richiesta per un breve intervallo di tempo, ma il fusibile non deve interrompere il circuito in tale situazione. Infine è stato previsto un indicatore luminoso posto esternamente al contenitore plastico che segnala la presenza di alimentazione.

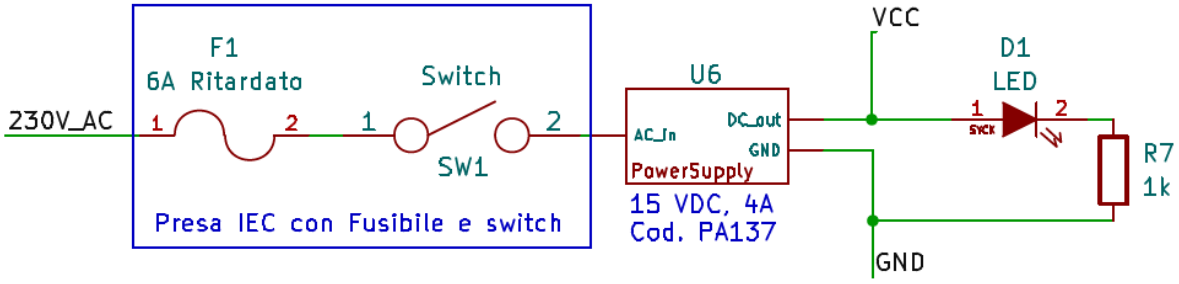

Figura 23: Alimentazione primaria.

Il successivo livello di alimentazione è costituito da una tensione di 5V. Sono presenti due schede proprietarie che abbassano la tensione in ingresso per ottenere 5V in output. La prima scheda fornisce la potenza per la Raspberry Pi Model 3B+ (che funziona come master del circuito), mentre la seconda serve per alimentare il servomotore, il modulo ad otto relè e ricavare i 3,3V del terzo livello di alimentazione. Una sola scheda di alimentazione non era sufficiente per via della massima corrente che è in grado di fornire, inferiore alla richiesta. Inoltre, avendo due differenti sorgenti a 5V, si evita il problema di interruzione sul master in caso di corti dovuti a malfunzionamenti o gusti, ad esempio sul servomotore. Il fusibile interviene interrompendo il circuito relativo al servomotore, ma il master rimarrebbe regolarmente alimentato, consentendo di rilevare il potenziale problema e arrestare la macchina. Il valore di intervento è stato dimensionato tenendo in considerazione le correnti elencate precedentemente.

Infine, per quanto riguarda l'ultimo livello di alimentazione a 3,3V, è stato previsto un LDO con limitazione da corto circuito [Ins24]. Questo dispositivo è in grado di fornire fino a 120mA di corrente in output in condizioni standard. Inoltre, grazie ad un feedback interno, è in grado di rilevare un cortocircuito e limitare la corrente fornita a 200-230mA senza danneggiarsi e senza causare problemi alla tensione in ingresso (i 5V). Siccome il regolatore è l'ultimo livello di alimentazione e si deve interfacciare con il mondo esterno (con il DUT), questa caratteristica risulta ottimale: non è possibile sapere a priori se la periferica in esame presenta un cortocircuito, ma anche se questo problema fosse presente non causerebbe buchi di alimentazione, la macchina di collaudo risulterebbe comunque operativa e in grado di individuare la problematica, comportandosi opportunamente. Il corredo e le scelte progettuali dell'LDO sono basate sulle considerazioni del datasheet e dell'application note [Les10]. I condensatori hanno lo scopo di filtrare il segnale, limitando le fluttuazioni e rendendo stabile la tensione. Un condensatore più grande (quello 100uF) agisce come serbatoio di carica riducendo le variazioni di tensione dovute a cambiamenti nel carico o nella sorgente di alimentazione, mentre il condensatore con capacità inferiore (10nF) risponde più veloce al rumore ad alta frequenza. La combinazione dei due condensatori migliora la regolazione, riducendo il ripple e migliorando la risposta transitoria; I condensatori in uscita svolgono una funzione simile. Il diodo funge da protezione contro le correnti inverse, alleggerendo il diodo del body del transistor interno all'LDO, ad esempio durante lo spegnimento della macchina.

In Figura 24 è possibile osservare la porzione di schema appena descritta con gli ultimi due stadi di alimentazione.

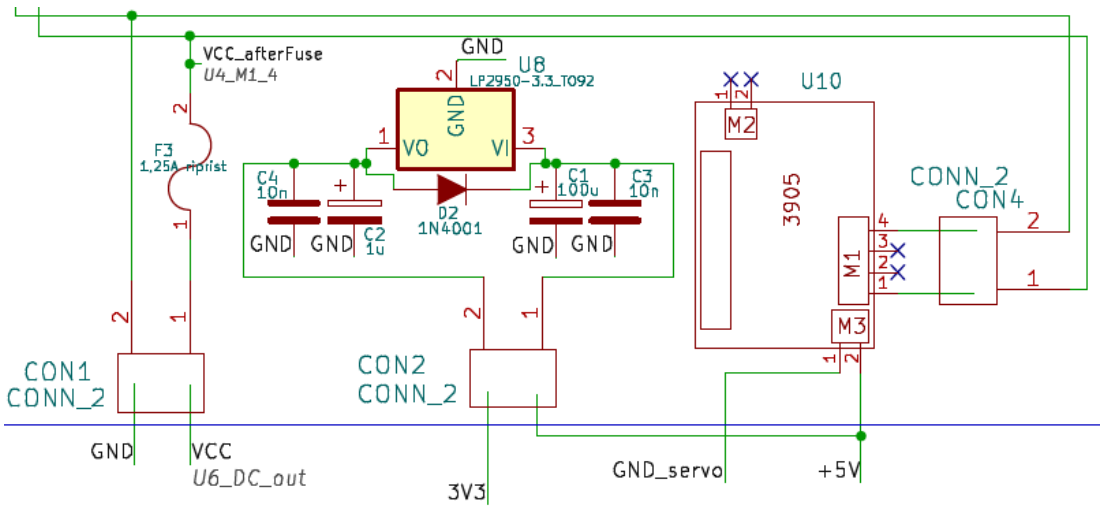

Figura 24: Secondo e terzo livello di alimentazione.

Il ruolo di master è svolto da una scheda a microprocessore, una Raspberry Pi Model 3B+ [Ras23]. Essa gestisce la comunicazione tra l'utente (attraverso un'interfaccia web, descritta nella Sezione 4.5) e il circuito CS3981, tramite un protocollo seriale. Tra i due moduli, è presente un'interfaccia FTDI232 che permette lo scambio dati all'avvio di ogni procedura di collaudo. La raspeberry permette di gestire anche la programmazione delle periferiche da testare, pilotando un programmatore della Silicon Laboratories; inoltre, tramite le porte USB, il connettore HDMI e la LAN è possibile interfacciarsi esternamente. In Figura 25 viene mostrata questa porzione dello schema.

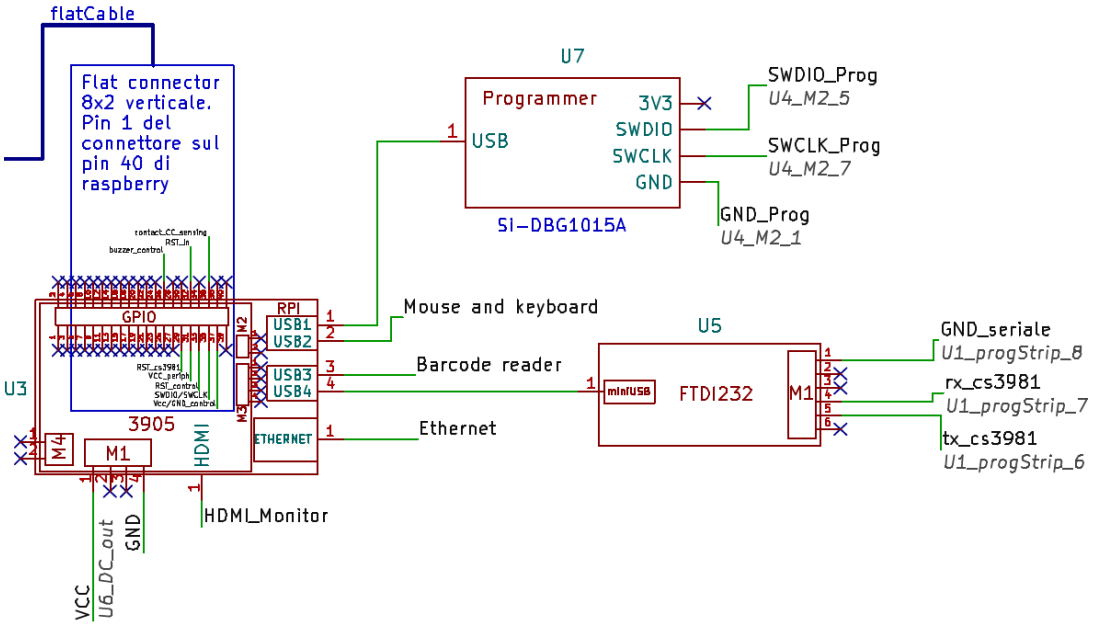

Figura 25: Raspberry, Programmatore SiLab e FTDI232.

All'inizio di una procedura di test, l'operatore deve inserire il DUT e armare la macchina. Fino a quando non viene premuto il pulsante di avvio, non deve essere fornita potenza alla periferica. Tramite dei relè, si fornisce alimentazione al momento opportuno, così come viene controllata la presenza di un cortocircuito, della corretta armatura della macchina e la programmazione. I relè aprono e chiudono i collegamenti sotto il controllo dei GPIO della raspberry. Siccome la massima corrente che un pin è in grado di fornire è di circa 15mA, insieme alle considerazioni svolte sulla corrente di eccitazione della bobina di un relè, non è possibile pilotare quest'ultimo direttamente. Pertanto, è stato previsto un livello intermedio costituito da un array di Darlington [STM18], visibile in Figura 26. Si tratta di un integrato disponibile in varie versioni a seconda dell'applicazione, con una hfe molto alta, che consente di pilotare carichi grandi con una corrente di base inferiore ad 1mA. Il modello scelto è ULN2801 poiché è l'unico pilotabile con una tensione più bassa di 5V, al quale è stata aggiunta una resistenza sulla base da 1kΩ per condizionarlo. Nel caso il darlington si deteriorasse e danneggiasse, la resistenza serve per limitare la corrente fornita dal GPIO nel caso base ed emettitore siano in corto. Il valore di 1kΩ limita la corrente a 3,3mA.

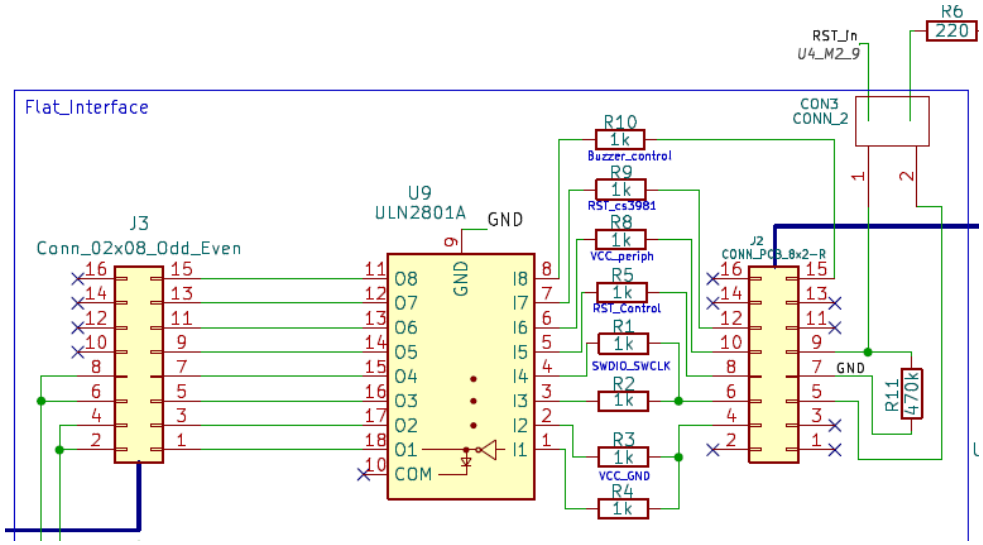

Figura 26: Array Darlington per pilotaggio relè.

La macchina di collaudo, inoltre, è in grado di rilevare un possibile corto sull'alimentazione della periferica in collaudo e di verificare la corretta armatura. In particolare, la verifica del cortocircuito viene effettuata fornendo alimentazione e acquisendo in ingresso a Raspberry il livello di tensione, preso direttamente sul polo positivo del DUT. In condizioni normali, si ha un valore logico alto, ma nel caso di cortocircuito, la tensione crolla a 0V, la scheda lo rileva e rimuove immediatamente alimentazione, segnalando il problema. Per quanto riguarda la verifica della corretta armatura, il procedimento è similare e sfrutta il pin di reset della periferica. Se il microcontrollore è alimentato, viene

inserito un pullup interno e si ha un livello logico alto su di esso. Pertanto, fino a quando la macchina non risulta chiusa correttamente, gli aghi non contattano la periferica e il valore acquisito risulta ad un livello logico basso. Dal datasheet del microcontrollore, la resistenza di pullup è intorno a 56kΩ, pertanto è stato previsto un resistore da 470kΩ in serie (R11) e la tensione viene acquisita tra le due resistenze.

In Figura 27 è mostrata la porzione di circuito a valle dell'array di Darlington. Si tratta di un modulo costituito da otto relè. Essendo dei contatti puliti, ossia elettricamente isolati dalla parte di controllo e con una resistenza di contatto molto bassa, permettono di aprire e chiudere il circuito senza perturbarlo. In questa applicazione, hanno lo scopo di fornire alimentazione all'avvio di un test (e di rimuoverla al termine), di collegare il programmatore per il caricamento del firmware e di controllare un emettitore acustico che segnala la fine del ciclo di collaudo.

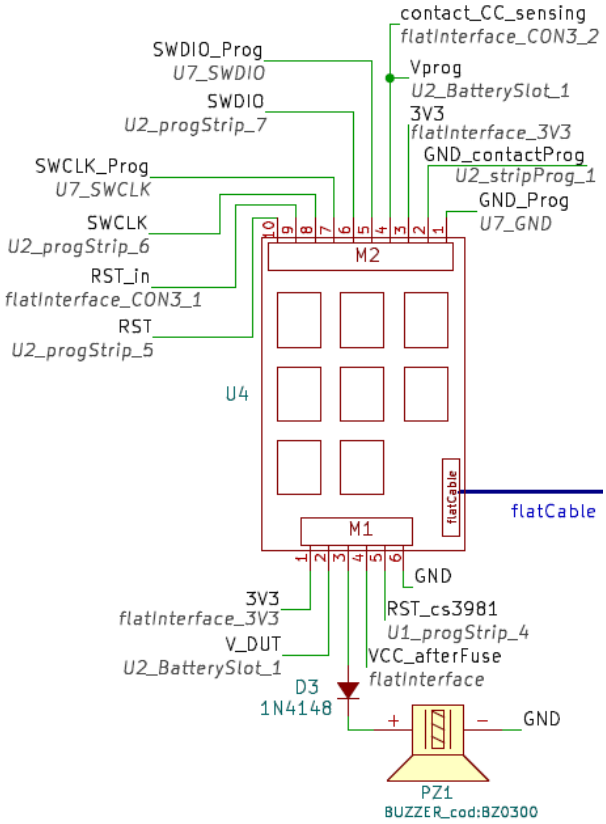

Figura 27: Modulo ad 8 relè.

In Figura 28 è presente la scheda (già descritta precedentemente) responsabile del vero collaudo funzionale. Essa si interfaccia al DUT tramite collegamento diretto sul letto di aghi o utilizzando dispositivi quali motore, fototransistor e servomotore. Siccome è inserita all'interno del contenitore plastico, è stato previsto un connettore che permette la riprogrammazione della scheda senza la necessità di aprire il tutto. Il connettore si collega allo strip di programmazione e tramite GND, SWCLK e SWDIO si può aggiornare il firmware.

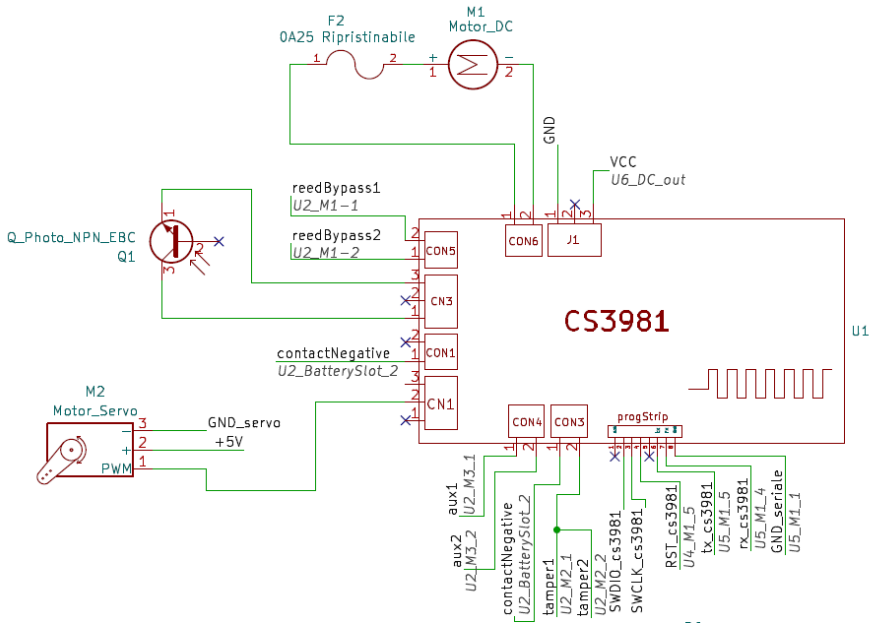

Figura 28: scheda CS3981.

Infine, in Figura 29 è presente il contatto magnetico, la periferica da testare.

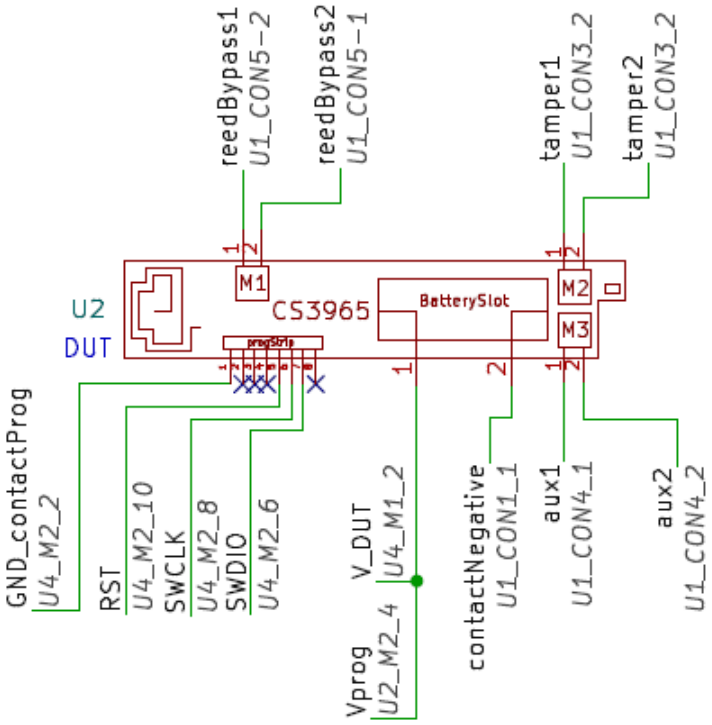

Figura 29: Contatto magnetico.

4.3 Parti meccaniche e struttura esterna

La struttura meccanica esterna riveste un ruolo fondamentale della macchina di test. Oltre a proteggere i componenti elettronici e hardware da fattori esterni, essa garantisce il corretto posizionamento delle periferiche da testare, permettendo un funzionamento preciso e sicuro del sistema. Nelle Figure 30 e 31 si mostra la struttura esterna della macchina di test per i contatti magnetici.

Nella parte superiore della macchina è presente il meccanismo dedicato all'inserimento e al collaudo della periferica. Attraverso una slitta è possibile inserire il dispositivo da testare, garantendo un allineamento corretto con i punti di contatto. Una volta posizionata la scheda, entra in funzione il meccanismo di armatura. Tirando la leva di attivazione, il letto d'aghi si sposta verticalmente, muovendosi dal basso verso l'alto fino a entrare in contatto con i punti designati del DUT.

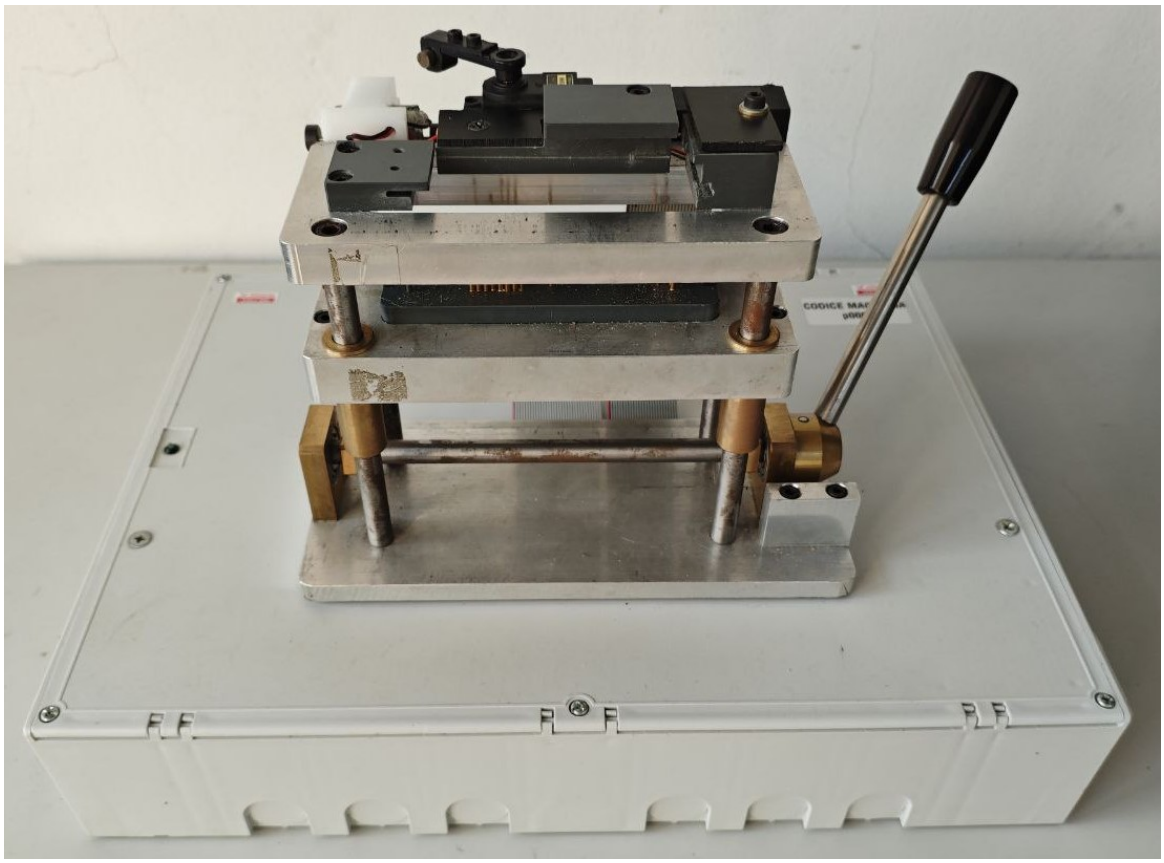

Figura 30: Vista frontale della meccanica.

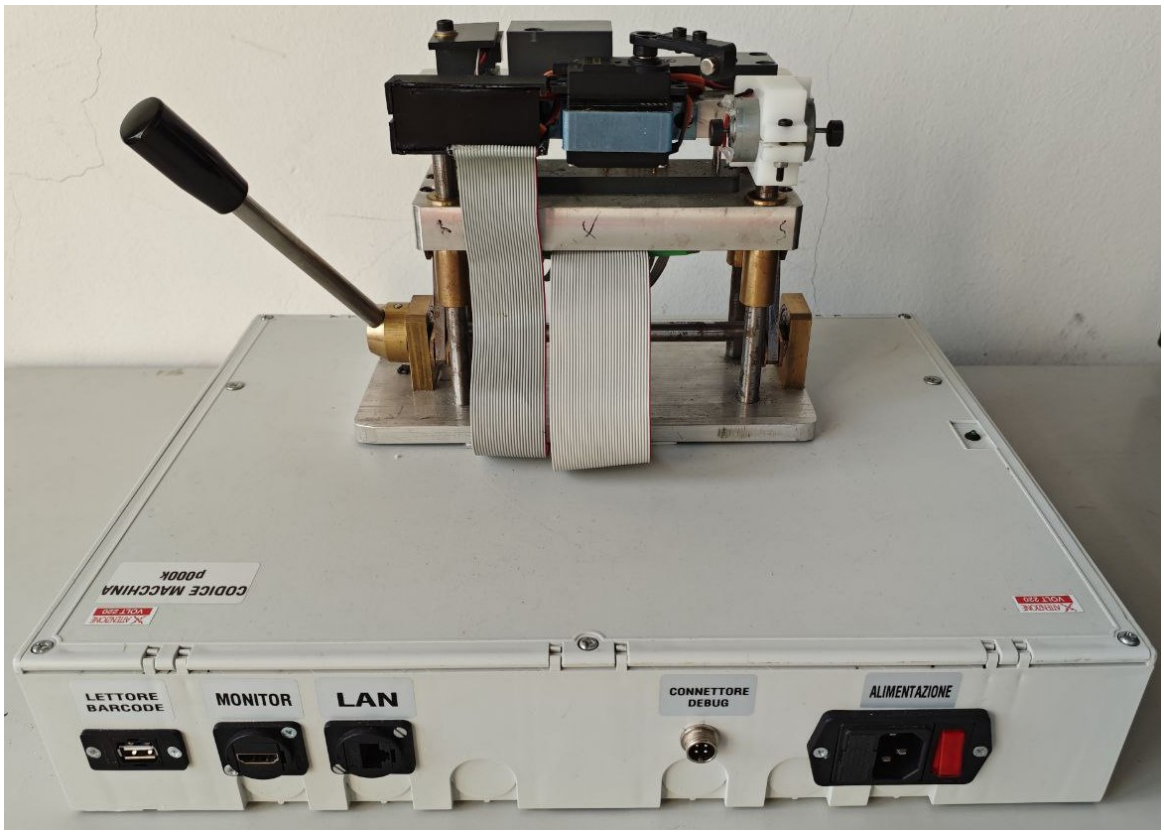

Figura 31: Vista posteriore della meccanica.

Oltre al meccanismo di armatura e al letto d'aghi, la parte superiore della macchina ospita il motore, il servomotore e il fototransistor. Come descritto precedentemente, il motore, solidale alla struttura, è progettato per generare vibrazioni che si trasmettono direttamente al DUT. Queste vibrazioni permettono di testare il corretto funzionamento dell'accelerometro integrato nella periferica. Il servomotore, invece, è dotato di un magnete fissato all'estremità del suo albero. Questo sistema consente di stimolare il sensore reed presente nel DUT, verificandone il corretto funzionamento in risposta ai cambiamenti del campo magnetico. Infine, sempre nella parte superiore della macchina, dal lato della leva di armatura, è stata ricavata una camera oscurata progettata per ospitare il fototransistor. Questo componente è posizionato in modo da allinearsi perfettamente con il LED presente sul dispositivo durante il collaudo. Grazie all'isolamento dalla luce esterna garantito dalla camera, il sistema è in grado di discriminare con precisione lo stato del LED della periferica, rilevandone l'accensione o lo spegnimento.

Sul lato posteriore sono stati montati i connettori per interfacciarsi con il mondo esterno e alimentare la macchina. Inserendo il cavo di alimentazione e premendo il pulsante rosso, la macchina viene avviata. Siccome l'interruttore luminoso non risulta facilmente visibile durante l'operatività, è stato aggiunto un LED verde sul contenitore plastico, che offre

un'indicazione chiara e immediatamente visibile dello stato di accensione della macchina all'utilizzatore finale.

Alla base della struttura si ha il contenitore plastico con all'interno tutta l'elettronica e la circuiteria descritta nella Sezione 4.2. Nell'angolo in basso a destra è presente l'alimentatore e il rispettivo connettore; alla sua sinistra, è montata la scheda CS3981, responsabile del collaudo vero e proprio, che comunica con il master tramite lo schedino seriale FTDI (quello rosso). La Raspberry si trova nell'angolo in alto a sinistra, insieme al suo alimentatore da 5V, inserito immediatamente sopra ad essa. Di fianco è presente una millefori con l'array di darlington, il programmatore firmware e LDO per la generazione del 3.3V. Infine, nell'angolo in alto a destra, si ha il modulo 8 relè. I flat cable che coprono la parte centrale dell'immagine escono dal contenitore plastico e si collegano al letto d'aghi, al motore, al servomotore e al fototransistor. In Figura 32 si mostra quanto appena descritto.

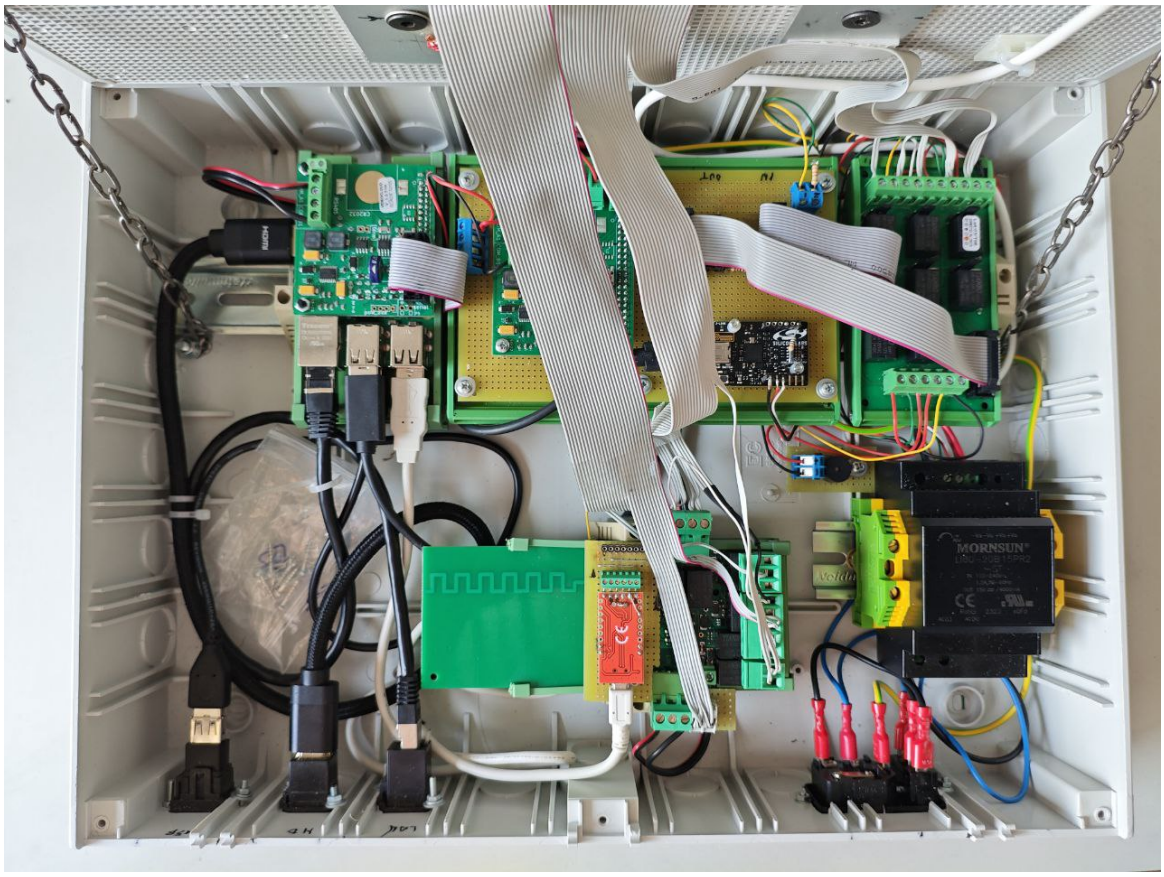

Figura 32: Elettronica interna al contenitore plastico.

4.3.1 Valutazione dei rischi elettrici e meccanici

Nell'ambito della progettazione e realizzazione della macchina di collaudo, è stata svolta una valutazione preliminare dei rischi elettrici e meccanici. Sebbene non sia stato possibile approfondire ogni aspetto nel dettaglio a causa delle competenze specifiche richieste in questo ambito, l'analisi è stata orientata a garantire la sicurezza operativa e l'affidabilità del sistema. In particolare, sono stati considerati i principali rischi associati all'alimentazione elettrica, ai movimenti meccanici e all'interazione dell'utente con la macchina, con l'obiettivo di ridurre al minimo i pericoli per l'operatore e per il dispositivo stesso.

Uno dei possibili rischi elettrici è il contatto accidentale con parti sotto tensione. Per ridurre al minimo questo possibile rischio, tutta l'elettronica principale è completamente racchiusa da un contenitore plastico, che impedisce l'accesso senza l'ausilio di strumenti appositi, come il cacciavite. Inoltre, sul coperchio sono state apposte delle etichette indicanti chiaramente la presenza di alta tensione, accompagnate da un indicatore luminoso che indica quando la macchina è in funzione.

Il contatto con il letto d'aghi e i motori non rappresenta un pericolo per l'operatore, in quanto i cavi sono opportunamente isolati, scorrono basse correnti e in bassa tensione. Per quanto riguarda i connettori, sono stati utilizzati connettori differenti o polarizzati per evitare il rischio di errato inserimento.

Tra i rischi meccanici si ha la possibilità di schiacciamento o pizzicamento durante l'armatura della macchina e il contatto con le parti vibranti del motore o con il braccio del servo. Sebbene esistano potenziali rischi, al momento non è stata installata una protezione meccanica, in quanto il rischio è stato valutato come non sufficientemente alto da giustificare tale misura.

È stato steso un manuale d'uso per garantire il corretto funzionamento della macchina. Il manuale include tutte le avvertenze necessarie e le istruzioni su come comportarsi durante il funzionamento e l'operatività dell'attrezzo di collaudo, al fine di garantire la sicurezza dell'operatore e l'efficienza del sistema.

4.4 Implementazione firmware

Come per la parte hardware, anche il firmware iniziale ha come base quello di un SC8R. Le parti radio e di comunicazione seriale sono state notevolmente arricchite per poter soddisfare le specifiche della macchina di collaudo. È stato ideato un protocollo di comunicazione utile per lo scambio dei dati e sono state implementate tutte le funzioni necessarie per poter svolgere ciascun test, sia all'interno di una procedura completa, utile per il reparto produzione poiché permette di collaudare interamente il DUT, sia come procedura singola, ad esempio per scopi di debug o sviluppo.

In sintesi, l'utente invia un comando specifico in funzione del test che vuole svolgere tramite un menù di selezione (descritto successivamente nella Sezione 4.5). Il firmware riceve tale informazione in seriale e avvia la procedura di test corretta. In fase di collaudo, dopo aver azionato gli attuatori, attende una risposta da parte della periferica tramite la gestione della comunicazione radio e confronta i dati ricevuti con quelli attesi. L'esito viene poi memorizzato e, una volta rispedito indietro (sempre per mezzo della seriale), verrà visualizzato dall'operatore. All'interno del main è quindi presente un ciclo infinito in cui viene periodicamente controllata la porta seriale, si avvia la procedura di test (se richiesto) e si imposta l'antenna in ricezione per attendere la risposta, o in trasmissione nel caso di avvio o termine del test.

In Figura 33 è presente un flow chart generale per quanto riguarda il singolo step. Essenzialmente, ogni step del collaudo funzionale è costituito dalla struttura rappresentata nel diagramma di flusso. In particolare, dopo che il firmware ha ricevuto un comando in seriale, tramite una struttura switch case viene selezionato lo step funzionale corrispondente. Nel caso di test completo, il primo controllo riguarda l'assorbimento di corrente a riposo, mentre i successivi verranno svolti in cascata.

All'interno dello step viene innanzitutto fornita alimentazione al dispositivo da collaudare (in caso di test completo, per gli step successivi al primo, non si rimuove l'alimentazione fino al termine del collaudo), vengono effettuate le configurazioni iniziali (ad esempio l'inizializzazione dell'ADC ove necessario) e si setta un opportuno delay. L'inserimento del ritardo ha essenzialmente tre scopi: il primo consiste nel permettere alla periferica di accendersi correttamente e finire la sua routine di inizializzazione, fornedole il tempo necessario per entrare a regime; il secondo scopo è quello di permettere ai relè di eccitarsi opportunamente poiché l'esecuzione del codice risulta più veloce del tempo di apertura e chiusura di un relé; infine, permette di slegare step adiacenti, al fine di limitare l'influenza di uno step precedente su quello successivo, nel caso di test completo.

Un timer è stato opportunamente configurato (consultando il reference manual [Lab24], sezione Timer) al fine di decrementare la variabile ad ogni millisecondo. la formula utilizzata è:

$$f_{timer} = \frac{f_{clk}}{(\mathrm{PRESC} + 1) \times (\mathrm{TOP} + 1)}$$

Dove f_{timer} risulta essere 1kHz e f_{clk} è la frequenza del clock pari a 40MHz; Impostando $PRESC$ a 7 e invertendo la formula, si ottiene $TOP = 4999$.

Al termine del timeout, si effettuata una misura o una stimolazione in base allo step del collaudo e viene settato un ulteriore timeout. Nel caso di una misura, rappresenta l'intervallo di tempo nel quale vengono campionati i valori di tensione; nel caso di stimolazione, rappresenta il tempo a disposizione entro il quale la periferica deve segnalare alla macchina l'effettivo cambio di stato di uno dei sensori presenti. Questa parte verrà chiarita durante la descrizione dei singoli collaudi.

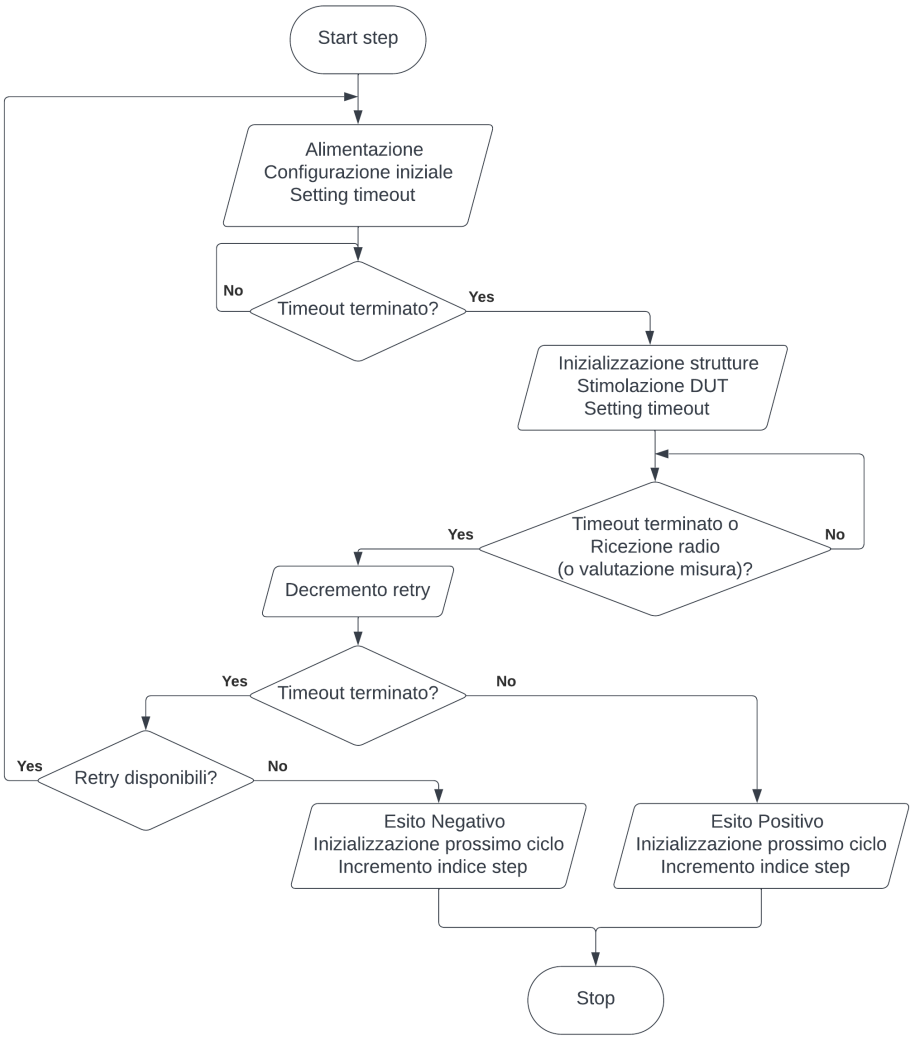

Figura 33: Flow chart generale del singolo step.

In seguito, avviene la valutazione dell'esito. Nel caso della stimolazione, se il timeout viene raggiunto, la macchina non ha ricevuto una risposta attraverso un pacchetto radio. Possono esserci due motivazioni:

1. La periferica ha rilevato la stimolazione, ma la macchina non ha ricevuto la risposta via radio.

2. La periferica non ha rilevato lo stimolo.

Nel primo caso, la mancata ricezione di un pacchetto radio può essere attribuita ad interferenze dovute ad altri dispositivi vicini, che disturbano la trasmissione. Per evitare che vengano forniti esiti falsi negativi, è stato implementato un meccanismo di retry in cui il ciclo viene ripetuto dall'inizio. Con un numero sufficiente di retry, risulta altamente improbabile (se non impossibile) che questa situazione si verifichi nuovamente. Se, nonostante le varie ripetizioni dello step, la macchina non avesse ancora ricevuto risposte dalla periferica, allora il problema rientra nel secondo caso.

Il secondo punto raccoglie tutti quei dispositivi la cui risposta non è stata rilevata dalla macchina in tutti i retry a disposizione, portando ogni tentativo all'azzeramento del timeout. Questo significa che la porzione di circuito in esame risulta essere difettosa, pertanto l'esito dello step sarà negativo e il DUT verrà etichettato come scarto.

Ovviamente, se nessuna delle due situazioni si verifica, significa che il timeout non è stato raggiunto e la macchina ha ricevuto la risposta che si aspettava dalla periferica, pertanto l'esito dello step risulta essere positivo.

Dopo la valutazione dell'esito, l'indice dello step viene incrementato in base alla tipologia iniziale di test. Nel caso di test completo, l'indice aumenta di un'unità, mentre nel caso di test singolo l'indice raggiunge il valore limite che porta alla termine dell'intera procedura di collaudo.

4.4.1 Protocollo seriale

Come detto, tramite l'interfaccia utente, l'operatore può selezionare il tipo di test da effettuare e avviare il ciclo di collaudo. Ogni tipologia di collaudo corrisponde ad un diverso codice che l'interfaccia invia ad un programma in C che gira in background. Questa informazione viene poi ribaltata alla scheda CS3981 tramite seriale, mediante un protocollo specifico. La comunicazione è di tipo full-duplex, punto-punto tra il master (la Raspberry) e la scheda della macchina di collaudo. Viene utilizzata una FTDI 232 (quindi sono presenti tre cavi, ossia TX, RX, GND). La velocità di trasmissione è di 115200 baud, con tensione 0-3.3V. Per quanto riguarda il formato dei dati, il protocollo utilizza una lunghezza variabile, una codifica esadecimale per rappresentare i byte trasmessi con un byte di header; non è stato previsto un `checksum` per il controllo degli errori, né un byte di stop.

La struttura del frame è la seguente:

header	len	periph	cmd	data[0]	data[1]	...	data[len-2]

Dove i campi rappresentano:

- `header`: è il preambolo del messaggio. Nell'applicazione corrisponde a `0x3C`. Se il messaggio non inizia con questo byte, esso viene considerato non valido e pertanto scartato.
- `len`: Specifica la lunghezza del frame da `periph` in avanti. Nel conteggio non fanno parte `header` e `len`.
- `periph`: Indica il dispositivo da testare. Per il contatto magnetico il byte è `0x00`.
- `cmd`: Si tratta del comando da eseguire. In particolare, questo byte rappresenta la tipologia del test. `0x16` corrisponde al test completo, mentre i codici da `0x18` in poi rappresentano il tipo di test singolo da eseguire.
- `data[0]` ... `data[len-2]`: Sono dati aggiuntivi necessari per alcuni `cmd`.

4.4.2 Assorbimenti di corrente

La misura degli assorbimenti di corrente per questo tipo di periferiche risulta molto importante poiché esse sono alimentate a batteria. Un consumo a riposo superiore del $15 - 20\%$ rispetto al massimo ritenuto accettabile può ridurre la durata del dispositivo di diversi mesi. La macchina di collaudo non è progettata per essere uno strumento di misura, ma deve comunque fornire misure attendibili, che rientrano in determinate soglie considerate valide.

Il primo step del test completo riguarda la corrente a riposo, ovvero in assenza di trasmissioni radio. Si riporta l'andamento della tensione acquisito ad oscilloscopio, con una resistenza di shunt da 153,04Ω.

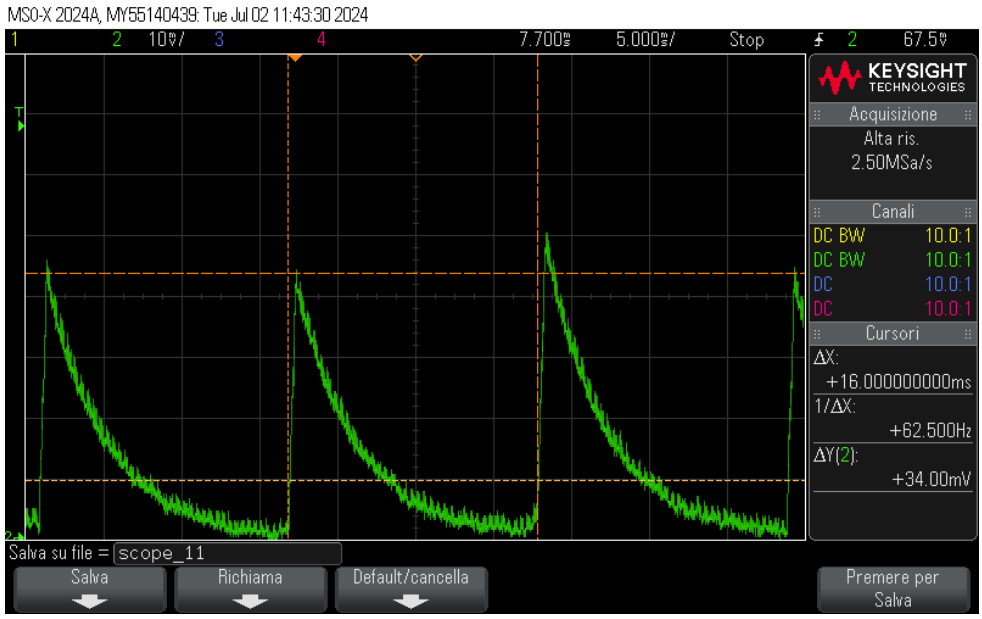

Figura 34: Routine del microcontrollore a riposo.

Al fine di ottenere una misura accurata, sono state prese in considerazioni alcune accortezze. In primo luogo, la singola misura è in realtà la media di 10 campioni, al fine di limitare l'impatto del rumore. In secondo luogo, collegando l'ingresso dell'operazionale direttamente a massa, viene valutato l'offset; questo valore andrà a correggere le successive misure. Infine, siccome la corrente media rappresenta il valore effettivo della corrente che fluisce attraverso il circuito in un determinato intervallo di tempo, essa viene misurata effettuando una sorta di media integrale. La corrente istantanea $I(t)$, con R nota, risulta essere:

$$I(t) = \frac{V(t)}{R}$$

Siccome l'intervallo di tempo per la misura è 500ms, la corrente media $\langle I(t) \rangle$ è data dalla media integrale della funzione $I(t)$, ovvero:

$$\langle I(t) \rangle = \frac{1}{500\,\text{ms} - 0} \int_0^{500\,\text{ms}} \frac{V(t)}{R}\, dt$$

Semplificando:

$$\langle I(t) \rangle = \frac{1}{500\,\text{ms}} \cdot \frac{1}{R} \int_0^{500\,\text{ms}} V(t)\, dt$$

Ai fini pratici, l'integrale viene calcolato mediante il metodo dei rettangoli: ogni misura effettuata corrisponde all'altezza del rettangolo, mentre la larghezza dipende dal tempo necessario per acquisire 10 campioni e mediarli tra di loro. Inoltre, per migliorare ulteriormente l'accuratezza di misura, si tiene in conto la correzione dell'area tra due rettangoli consecutivi, rappresentata da un triangolo. In formule, si ha quindi:

$$\langle I(t) \rangle \approx \frac{\sum_{i=0}^{n-1} V(t_i)\Delta t + \sum_{i=1}^{n-1} \frac{1}{2} \left(V(t_i) - V(t_{i-1})\right) \Delta t}{500ms \cdot R}$$

Dove:

- $V(t_i)$ è il valore della funzione al punto t_i,
- Δt è la larghezza dei rettangoli (con n sottointervalli),
- $\frac{1}{2} \left(V(t_i) - V(t_{i-1})\right) \Delta t$ rappresenta l'area del triangolo formato tra due rettangoli successivi.

Oltre al calcolo della corrente media, si ha anche la valutazione della corrente massima e minima acquisite durante l'intervallo di misura. Terminato il processo, il valore medio deve rientrare all'interno dei limiti previsti per considerare la periferica accettabile. Le soglie sono state determinate attraverso l'analisi di un discreto numero di campioni e sono il risultato di una valutazione statistica basata sulla distribuzione dei valori misurati. Si riportano i valori acquisiti nel grafico in Figura 35.

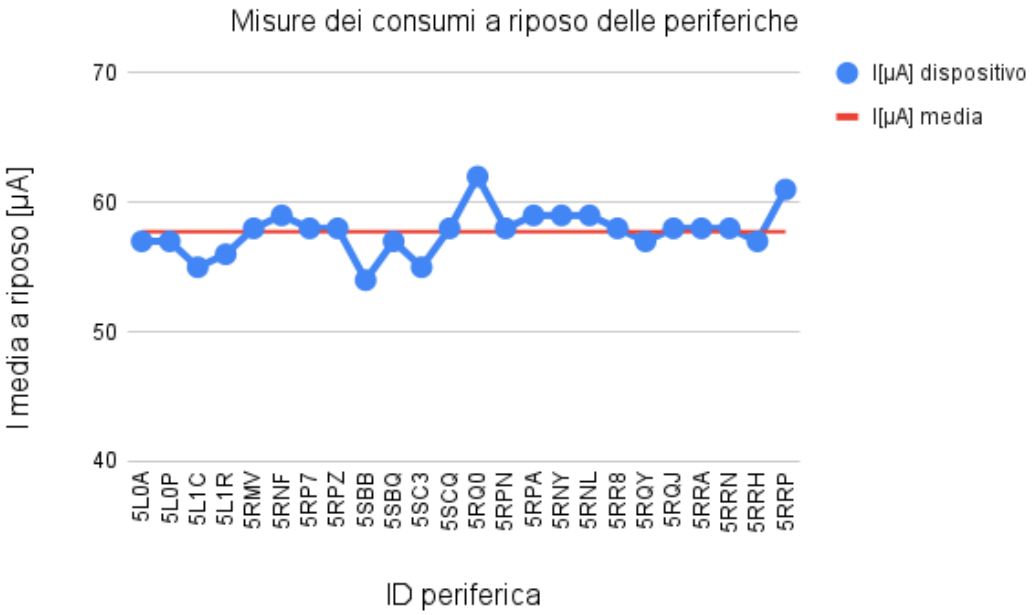

Figura 35: Misure assorbimenti a riposo delle periferiche.

Queste misure sono state effettuate mediante l'utilizzo di un amperometro da banco. La media dei consumi effettuata su questi 25 campioni corrisponde a 57,7μA. La macchina di collaudo, come già detto, non si tratta di uno strumento di misura, ma deve effettuare una sorta di caratterizzazione e verificare se questa rientra nei limiti. Bisogna tenere in considerazione incertezze di misura dovute alle tolleranze dei componenti in gioco come operazionale e resistenze, l'incertezza dovuta all'ADC e al processo di valutazione della misura ed un eventuale margine dovuto all'influenza di fattori esterni come la temperatura, che possono variare i consumi della periferica. Pertanto, per la determinazione delle soglie, si è preso un margine del 30% rispetto al valore medio e le soglie finali sono state fissate $40\mu A$ e $80\mu A$. La batteria con cui viene alimentato il DUT è una CR123, che presenta una capacità di 1400mAh. Il consumo energetico a riposo è il principale fattore responsabile dello scaricamento della batteria, pertanto è possibile valutare la durata della batteria con queste soglie. Trascurando le trasmissioni, si ha:

$$t_{\text{scarica}} = \frac{\text{Capacità}}{\text{Consumo}} = \frac{1400 \text{ mAh}}{80 \ \mu\text{A}} = \frac{1400 \times 10^3 \ \mu\text{Ah}}{80 \ \mu\text{A}} = 17500 \text{ ore}$$

Per convertire il tempo di scarica in anni, utilizziamo:

$$\text{Anni} = \frac{t_{\text{scarica}}}{24 \times 365} = \frac{17500 \text{ ore}}{24 \times 365} = \frac{17500}{8760} \approx 1,99 \text{ anni}$$

Per politica aziendale, si raccomanda la sostituzione della batteria ogni anno. Pertanto, per considerazioni sulle tolleranze e sul margine effettuate prima, la scelta della soglia risulta ragionevole anche dal punto di vista energetico e di durata della batteria.

La porzione di codice responsabile della misura della corrente a riposo è presentata qui di seguito.

```
1    uint8_t I_measureLowContatto3V(uint8_t *testStep, uint8_t functionalTestMode, int16_t
         testResultArrayMeasurements[]){
2   static uint16_t I_low_value_min;
3   static uint16_t I_low_value_max;
4   static uint32_t I_low_value_mean, triangleSum;
5   static uint16_t num_measure, previousReadValue;
6   uint16_t readValue;
7
8   static uint8_t initFlag = 1;                    // flag useful for first run
9   static uint16_t initDelay = 1;
10  static uint16_t offset = 0;
11  static uint8_t firstMeasure = 1;
12  float R_shuntLow = 0.153;
13
14  if(initDelay){
15      GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
16      setInputADCContatto3V(I_LOW_ADC_IN);
17      offset = ReadAdcTestContatto3V();
18      GPIO_PinOutSet(I_LOW_PORT, I_LOW_OUT_PIN);
19      timeoutTestMachine = 2000;
20      initDelay = 0;
21  }
22  else{
23      if(initFlag){                                   // initialization if first run
24          if(!timeoutTestMachine){
25              GPIO_PinOutClear(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
26              timeoutTestMachine = MAX_TIMEOUT;       // timeout set to 1s to close
                    correctly the relay and extinguish transients
27              I_low_value_min = INT16_MAX;
28              I_low_value_max = 0;
29              I_low_value_mean = 0;
30              triangleSum = 0;
31              num_measure = 0;
32              initFlag = 0;
33              firstMeasure = 1;
34          }
35      }
36      else{
37          if(firstMeasure){
38              if(!timeoutTestMachine){
39                  firstMeasure = 0;
40                  timeoutTestMachine = 500;  // timeout set to 500 ms for the I_low
                        measure
41                  readValue = ReadAdcTestContatto3V() - offset;  // measure of current
42                  previousReadValue = readValue;
43                  if((int16_t)readValue < 0)
44                      readValue = 0;
45                  num_measure++;
46                  I_low_value_mean += readValue;
47
48                  if (readValue < I_low_value_min){           // find min and max current
                         values for determine the range
49                      I_low_value_min = readValue;
50                  }
51                  if (readValue > I_low_value_max){
52                      I_low_value_max = readValue;
```

```
53                  }
54              }
55          }
56          else{
57              readValue = ReadAdcTestContatto3V() - offset;  // measure of current
58              if((int16_t)readValue < 0)
59                  readValue = 0;
60              triangleSum += abs((int16_t)(readValue-previousReadValue));
61              num_measure++;
62              I_low_value_mean += readValue;
63              previousReadValue = readValue;
64
65              if (readValue < I_low_value_min){           // find min and max current
                    values for determine the range
66                  I_low_value_min = readValue;
67              }
68              if (readValue > I_low_value_max){
69                  I_low_value_max = readValue;
70              }
71
72              if(!timeoutTestMachine){                              // if timeout arrived
73                  GPIO_PinOutClear(I_LOW_PORT, I_LOW_OUT_PIN);      // de-initialization
74                  I_low_value_min = I_low_value_min/R_shuntLow;     // conversion from
                        millvolt to uA. R = 153ohm
75                  I_low_value_max = I_low_value_max/R_shuntLow;
76                  I_low_value_mean = (REQUIRED_ADC_TIME*(I_low_value_mean+triangleSum/2))
                        /(500000*R_shuntLow);  // store the mean measure obtained, integral
                         mean
77                  testResultArrayMeasurements[I_MEASURE_LOW_MIN] = I_low_value_min;  //
                        store the min measure obtained
78                  testResultArrayMeasurements[I_MEASURE_LOW_MAX] = I_low_value_max;  //
                        store the max measure obtained
79                  testResultArrayMeasurements[I_MEASURE_LOW_MEAN] = I_low_value_mean;
80
81                  if(functionalTestMode > COMPLETE_TEST_CMD)                    // if in
                        single test, increment to the max to finish the test routine
82                      (*testStep) = NUM_TEST_CONTATTO3V;
83                  else
84                      (*testStep)++;                                // else increment of
                            test step for the next step
85
86                  initFlag = 1;                                     // reset value when I
                        low test is done for next cycle
87                  initDelay = 1;
88
89                  if(I_low_value_mean < 40 || I_low_value_mean > 80){        // if I out
                        of range 40uA < I < 80uA => ERROR
90                      return TEST_ERROR;
91                  }
92                  else{
93                      return TEST_PASSED;                       // no error, return 0
94                  }
95              }
96          }
97      }
98  }
99  return 0;
100 }
```

Lo step successivo all'interno del collaudo completo è la valutazione dell'assorbimento durante una trasmissione radio. Si ricorda che l'andamento tipico è mostrato nella seguente figura.

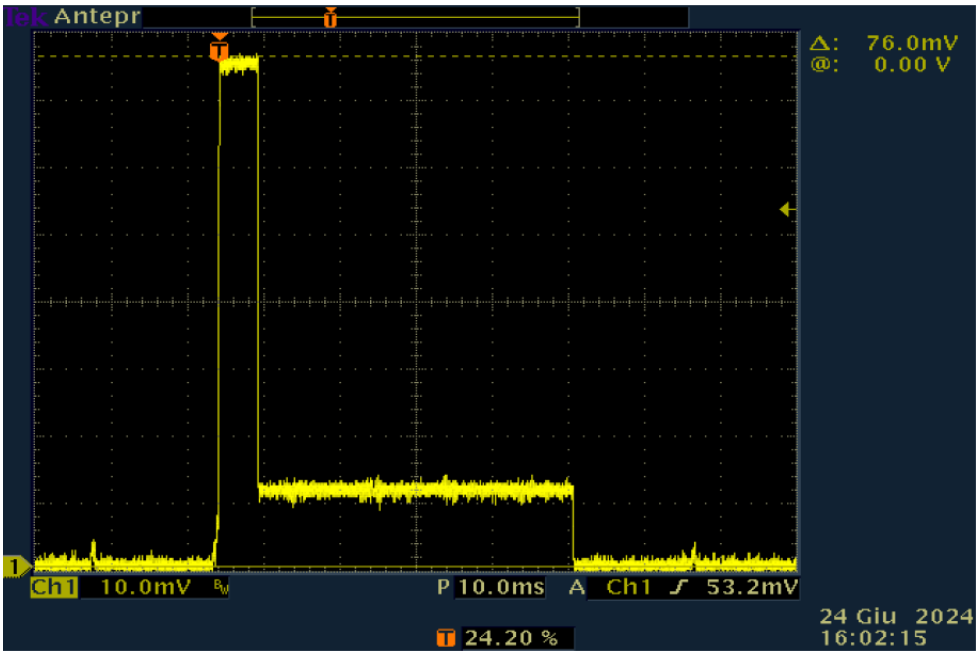

Figura 36: Fase di trasmissione del contatto magnetico.

Il primo gradino corrisponde all'intervallo di tempo in cui viene effettuata una trasmissione radio, seguito da una fase di ricezione. Valutando gli assorbimenti sugli stessi campioni citati prima, si è ottenuto un valore medio di 79,4mA in TX e 11,2mA in RX. Per via della diversa forma d'onda, il metodo di misura è variato rispetto alla valutazione delle correnti a riposo; infatti, non si effettua un'approssimazione dell'integrale, ma vengono ricercate delle soglie e acquisiti i campioni all'interno di ciascuna soglia. Inoltre, si valutano il valore massimo, quello minimo e la media dei campioni all'interno di ogni intervallo. In particolare, dopo aver effettuato una stimolazione, la periferica trasmette un pacchetto radio. Conseguentemente, i consumi del DUT inizieranno ad aumentare e la macchina scarterà le misure fino a quando la soglia 1, mostrata in Figura 37, non verrà raggiunta. A questo punto, inizia la fase di acquisizione, dove ogni misura (che risulta essere la media di 10 campioni, come prima) viene memorizzata all'interno di un vettore dimensionato adeguatamente per memorizzare l'intera forma d'onda. Nel caso in cui la soglia 1 non venga identificata, significa che la stimolazione non ha avuto successo per via di un difetto nella periferica in collaudo oppure potrebbe indicare un potenziale problema sulla radio.

Al termine dell'acquisizione, avviene la fase di processamento ed elaborazione dei dati. Vengono innanzitutto ricercate le due soglie restanti (la soglia 1 corrisponde all'inizio della

memorizzazione delle misure) e per ognuno dei tre intervalli viene valutato l'assorbimento massimo, quello minimo e il valore medio dei campioni presenti. Siccome la transizione non è immediata per via dei fronti che non sono esattamente verticali, alcuni campioni sono rimossi sia in salita che in discesa. Questo porta ad avere una valutazione più accurata poiché valori acquisiti a ridosso delle soglie, essendo dovuti a transitori, andrebbero a ridurre la corrente media all'interno dell'intervallo, e questo porterebbe ad una stima sempre in difetto. Oltre agli assorbimenti, si ha anche una valutazione dei tempi in TX ed RX. Se le soglie non venissero identificate o le correnti medie risultassero al di fuori dei limiti previsti, lo step viene concluso con esito negativo.

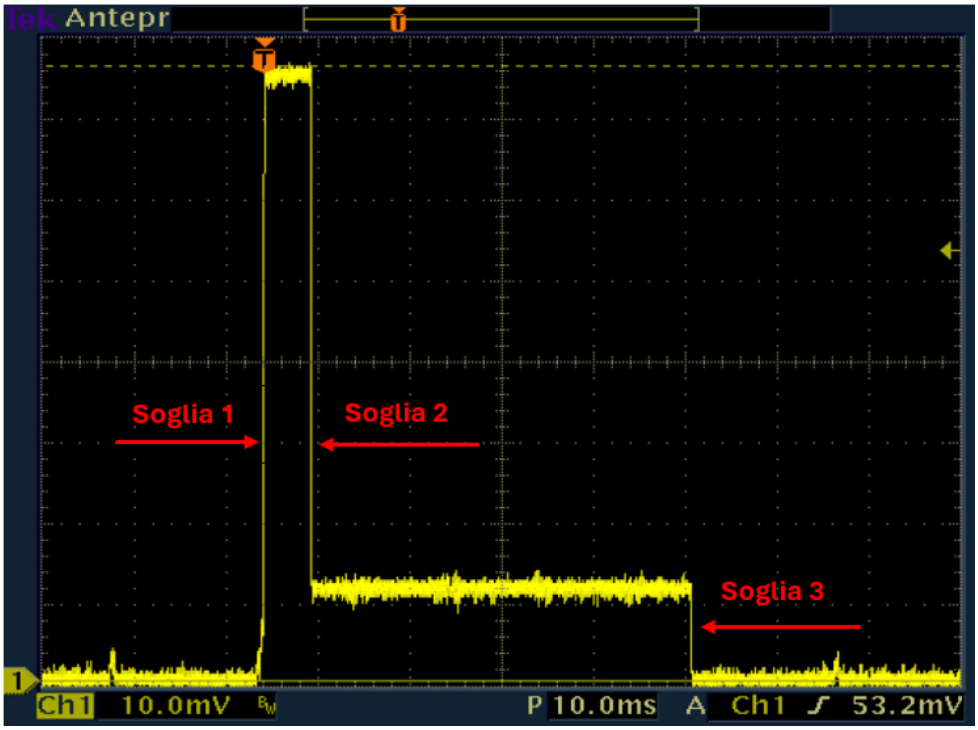

Figura 37: Fase di trasmissione con soglie per valutazione della corrente in TX.

Per motivi di brevità, si riportano solo alcune parti del codice inerenti alla caratterizzazione delle correnti in trasmissione, ritenute rilevanti.

A seguito delle inizializzazioni e della stimolazione, inizia la fase di acquisizione con la ricerca della prima soglia. Se viene trovata, l'intero vettore viene riempito e si ricercano le soglie 2 e 3.

```
1 readValue = (ReadAdcTestContatto3V() - offset)/R_shuntHigh; // measure of current in mA.
      Conversion from millvolt to mA. R = 4.7ohm
2 if (foundThreshold1 || (readValue > currentThreshold1)){    // threshold1 matched
```

```
foundThreshold1 = 1;                                    // re-initialization for next
    cycle test
 I_measureArray[idx] = readValue;
 idx++;
 if (idx == DIM_I_HIGH_ARRAY){                          // if array is full start check
      values
     idxThreshold2 = 1;
     foundThreshold2 = 0;
     while((idxThreshold2 < DIM_I_HIGH_ARRAY) && !foundThreshold2){    // find
         threshold2 (when TX stops)
         if (I_measureArray[idxThreshold2] < currentThreshold2){
             foundThreshold2 = 1;
         }
         idxThreshold2++;
     }
     idxThreshold2--;                                                  // to avoid last
         increment

     idxThreshold3 = idxThreshold2;
     foundThreshold3 = 0;
     while((idxThreshold3 < DIM_I_HIGH_ARRAY) && !foundThreshold3){    // find
         threshold3 (when RX stops)
         if (I_measureArray[idxThreshold3] < currentThreshold3){
             foundThreshold3 = 1;
         }
         idxThreshold3++;
     }
     idxThreshold3--;                                                  // to avoid last
         increment
```

Se le soglie sono state identificate, avviene la valutazione delle correnti minime, massime e medie e dei tempi di TX ed RX. In base a tali valutazioni, si fornisce l'esito della misura.

```
 if(foundThreshold2 == 1 && foundThreshold3 == 1){                  // if threshold2
     and threshold3 present, check I values
     testResultArrayMeasurements[TX_INTERVAL] = ((idxThreshold2)-1) * REQUIRED_ADC_TIME;
                        // evaluation of TX interval
     testResultArrayMeasurements[RX_INTERVAL] = ((idxThreshold3)-idxThreshold2) *
         REQUIRED_ADC_TIME;    // evaluation of RX interval

     for(idx = 1+removedSamples; idx < (idxThreshold2-1-removedSamples); idx++){  //
         remove first 3 samples respect to first threshold and last 3 samples
         if (I_measureArray[idx] < testResultArrayMeasurements[I_MEASURE_HIGH_TX_MIN]){
              // find min and max current values for determine the range
             testResultArrayMeasurements[I_MEASURE_HIGH_TX_MIN] = I_measureArray[idx];
         }
         if (I_measureArray[idx] > testResultArrayMeasurements[I_MEASURE_HIGH_TX_MAX]){
             testResultArrayMeasurements[I_MEASURE_HIGH_TX_MAX] = I_measureArray[idx];
         }
         testResultArrayMeasurements[I_MEASURE_HIGH_TX_MEAN] += I_measureArray[idx];
     }
     testResultArrayMeasurements[I_MEASURE_HIGH_TX_MEAN] = testResultArrayMeasurements[
         I_MEASURE_HIGH_TX_MEAN]/(idxThreshold2-1-removedSamples-(1+removedSamples));
     if(testResultArrayMeasurements[I_MEASURE_HIGH_TX_MEAN] < 30 ||
         testResultArrayMeasurements[I_MEASURE_HIGH_TX_MEAN] > 50){   // check if 30mA <
          I_TX_MEAN < 50mA, otherwise ERROR
```

```
16          if(functionalTestMode > COMPLETE_TEST_CMD)                  // if in single
                test, increment to the max to finish the test routine
17              (*testStep) = NUM_TEST_CONTATTO3V;
18          else
19              (*testStep)++;                              // else increment of test step
                    for the next step
20          initFlag = 1;
21          initDelay = 1;
22          GPIO_PinOutClear(I_HIGH_PORT, I_HIGH_OUT_PIN);
23          GPIO_PinOutClear(SW_PORT, SW_PIN);
24          return TEST_ERROR;
25      }
26
27      for(idx = idxThreshold2+removedSamples; idx < (idxThreshold3-1); idx++){   //remove
             first 3 samples respect to first threshold and last 3 samples
28          if (I_measureArray[idx] < testResultArrayMeasurements[I_MEASURE_HIGH_RX_MIN]){
                 // find min and max current values for determine the range
29              testResultArrayMeasurements[I_MEASURE_HIGH_RX_MIN] = I_measureArray[idx];
30          }
31          if (I_measureArray[idx] > testResultArrayMeasurements[I_MEASURE_HIGH_RX_MAX]){
32              testResultArrayMeasurements[I_MEASURE_HIGH_RX_MAX] = I_measureArray[idx];
33          }
34          testResultArrayMeasurements[I_MEASURE_HIGH_RX_MEAN] += I_measureArray[idx];
35      }
36      testResultArrayMeasurements[I_MEASURE_HIGH_RX_MEAN] = testResultArrayMeasurements[
            I_MEASURE_HIGH_RX_MEAN]/(idxThreshold3-1-(idxThreshold2+removedSamples));
37
38      if(testResultArrayMeasurements[I_MEASURE_HIGH_RX_MEAN] < 8 ||
            testResultArrayMeasurements[I_MEASURE_HIGH_RX_MEAN] > 15){     // check if 8mA
            < I_RX_MEAN < 15mA, otherwise ERROR
39          if(functionalTestMode > COMPLETE_TEST_CMD)                  // if in single
                test, increment to the max to finish the test routine
40              (*testStep) = NUM_TEST_CONTATTO3V;
41          else
42              (*testStep)++;                              // else increment of test step
                    for the next step
43          initFlag = 1;
44          initDelay = 1;
45          GPIO_PinOutClear(I_HIGH_PORT, I_HIGH_OUT_PIN);
46          GPIO_PinOutClear(SW_PORT, SW_PIN);
47          return TEST_ERROR;
48      }
49
50      //remove first 3 samples respect to first threshold
51      for(idx = idxThreshold3+removedSamples; idx < DIM_I_HIGH_ARRAY; idx++){         //
            check if I_after_RX < 5, otherwise ERROR
52          if (I_measureArray[idx] < testResultArrayMeasurements[
                I_MEASURE_HIGH_AFTER_RX_MIN]){  // find min and max current values for
                determine the range
53              testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MIN] = I_measureArray[
                    idx];
54          }
55          if (I_measureArray[idx] > testResultArrayMeasurements[
                I_MEASURE_HIGH_AFTER_RX_MAX]){
56              testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MAX] = I_measureArray[
                    idx];
57          }
58          testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MEAN] += I_measureArray[idx
                ];
59      }
```

```
60      testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MEAN] =
            testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MEAN]/(DIM_I_HIGH_ARRAY-(
            idxThreshold3+removedSamples));
61      if(testResultArrayMeasurements[I_MEASURE_HIGH_AFTER_RX_MEAN] > 5){
62          if(functionalTestMode > COMPLETE_TEST_CMD)                  // if in single
                test, increment to the max to finish the test routine
63              (*testStep) = NUM_TEST_CONTATTO3V;
64          else
65              (*testStep)++;                                  // else increment of test step
                    for the next step
66          initFlag = 1;
67          initDelay = 1;
68          GPIO_PinOutClear(I_HIGH_PORT, I_HIGH_OUT_PIN);
69          GPIO_PinOutClear(SW_PORT, SW_PIN);
70          return TEST_ERROR;
71      }
72
73      if(functionalTestMode > COMPLETE_TEST_CMD)           // if in single test,
            increment to the max to finish the test routine
74          (*testStep) = NUM_TEST_CONTATTO3V;
75      else
76          (*testStep)++;                                // else increment of test step for
                the next step
77      initFlag = 1;
78      initDelay = 1;
79      GPIO_PinOutClear(I_HIGH_PORT, I_HIGH_OUT_PIN);
80      GPIO_PinOutClear(SW_PORT, SW_PIN);
81      return TEST_PASSED;                               // if here, all tests passed
82  }
83  else{
84      if(functionalTestMode > COMPLETE_TEST_CMD)           // if in single test,
            increment to the max to finish the test routine
85          (*testStep) = NUM_TEST_CONTATTO3V;
86      else
87          (*testStep)++;                                // else increment of test step for
                the next step
88      initFlag = 1;
89      initDelay = 1;
90      GPIO_PinOutClear(I_HIGH_PORT, I_HIGH_OUT_PIN);
91      GPIO_PinOutClear(SW_PORT, SW_PIN);
92      return TEST_ERROR;                                // no threshold detected for
            correct mask => ERROR
93  }
```

4.4.3 Calibrazione

La calibrazione consiste nell'aggiustamento della frequenza portante del dispositivo in collaudo. A causa delle tolleranze dei componenti che costituiscono la radio, è possibile che la frequenza di riferimento possa essere fuori specifica e quindi debba essere aggiustata. In caso contrario, potrebbero sorgere problemi di comunicazione dovuti ad una errata interpretazione dei bit.

Siccome la macchina di collaudo rappresenta il riferimento, inizialmente è necessario calibrarla. Durante la manutenzione, tramite una funzione che sarà descritta nella Sezione 4.5, è possibile mandare la macchina in trasmissione di portante fissa e confrontare il picco del segnale tramite un analizzatore di spettro. Infatti, la portante fissa è un segnale a frequenza costante utilizzato come riferimento per una modulazione; tramite l'analizzatore di spettro è possibile misurare lo sfasamento rispetto alla frequenza di riferimento e aggiustarlo andando ad agire sull'offset, inserendo nell'interfaccia il valore necessario per regolare lo shift. Calibrata la macchina, è quindi possibile tarare la periferica in collaudo.

Per prima cosa, è necessario configurare la macchina di collaudo in modo che comunichi al DUT l'entrata nella fase di calibrazione (impostando la modalità di test in `TEST_CALIBRATE`). Inizia così uno scambio di messaggi radio con il device in collaudo, in cui vengono invocate determinate funzioni, fornite dal costruttore del microcontrollore; essendo proprietarie, tali routine non sono accessibili e consultabili, ma il loro compito è quello di ricavare lo sfasamento rispetto al riferimento e andarlo ad aggiustare, andando a scriverlo in flash.

Ad ogni ricezione radio, la macchina riceve dalla periferica il numero di step per la calibrazione e il valore di RSSI, che si tratta di un parametro per la misura della potenza di un segnale. Questi valori devo rientrare in determinati limiti: un RSSI troppo basso è sintomo di un problema sull'antenna, poiché non esce potenza dalla radio. In Figura 38 si mostra l'andamento dei valori misurati attraverso la macchina di collaudo su una cinquantina di campioni. Tutte le misure sono state svolte nelle stesse condizioni e con pezzi appartenenti a diversi lotti di produzione. Il valore medio, indicato dalla linea rossa, risulta essere -25dBm. Le soglie sono state fissate a -15 e -35 dBm. Siccome l'indicazione del valore di RSSI non è molto accurata, a seguito del test tutti i campioni sono stati testati sul campo per verificare l'effettiva portata e la distanza a cui erano ancora in grado di comunicare e scambiare messaggi via radio. Inoltre, sono stati effettuati collaudi anche su periferiche volontariamente manomesse, a cui sono stati rimossi componenti lungo il percorso dell'antenna oppure sono stati sostituiti con valori diversi. Questo porta ad avere un mismatch dell'antenna e quindi un degrado delle performance, che si dovrebbero tradurre in un significativo calo dell'RSSI. Effettivamente, per questi campioni, la potenza rilevata dalla macchina di collaudo risultava essere inferiore a -50 dBm. Maggiori dettagli saranno forniti nella sezione delle validazioni (Sezione 5).

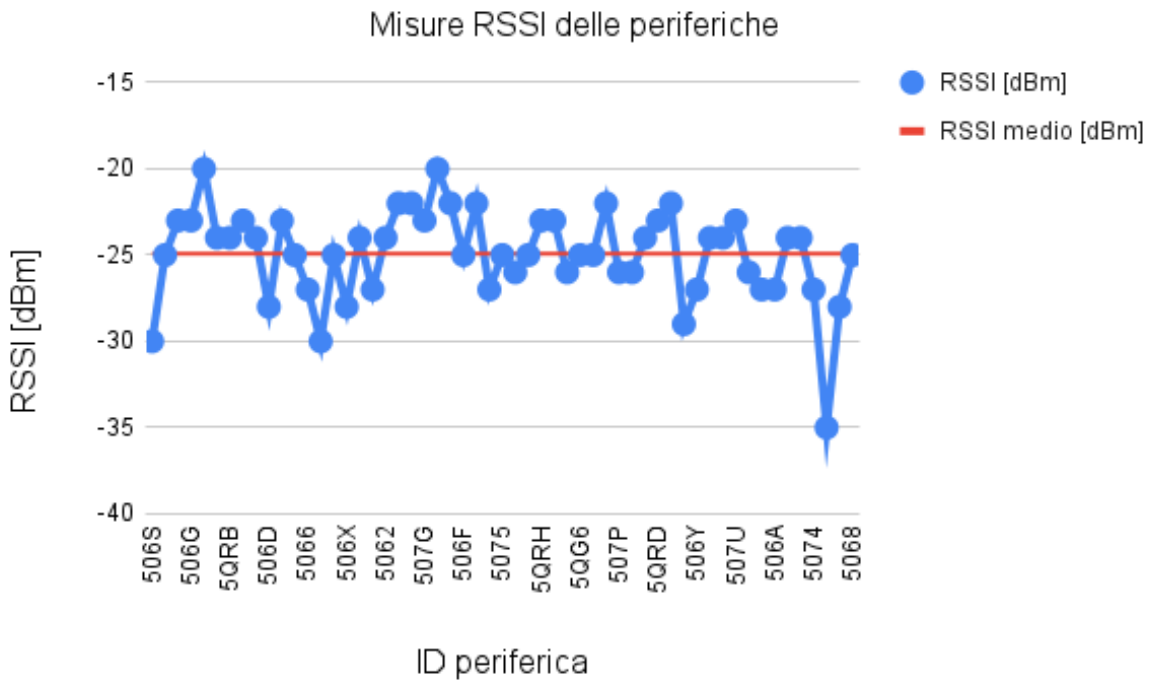

Figura 38: Valori di RSSI su campioni NON manomessi.

Di seguito si riporta la porzione di codice relativa alla calibrazione e alla stima della potenza dell'antenna.

```
1    uint8_t calibrationContatto3V(uint8_t *testStep, uint8_t functionalTestMode, int16_t
         testResultArrayMeasurements[]){
2   static uint8_t initFlag = 1;                        // flag useful for first run
3   static uint8_t initDelay = 1;
4   int8_t minRSSI = -35;
5   int8_t maxRSSI = -15;
6
7   if(initDelay){
8       inhibitTX = 0;
9       rssiPeriphTest = 127;
10      periphTestCalibration = 0xFFFF;
11
12      GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
13      if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
             correct routine of periph
14          timeoutTestMachine = 2000;
15      }
16      else{
17          timeoutTestMachine = MAX_TIMEOUT/4; // else initial delay to ensure non-
                influence of previous step
18      }
19      initDelay = 0;
20  }
21  else{
22      if(initFlag){                                   // initialization if first run
```

```
23            if(!timeoutTestMachine){
24                // Calibration
25                inp_out.timer_led[LED_RADIO_RX_S] = 255;
26                inp_out.timer_led[LED_MAN_SCR8]   = 255;
27                rssiPeriphTest = 127;
28                testMode = TEST_CALIBRATE;
29                workingMode = MODE_TEST;
30                ram_gest.cnt_sec_collaudo = 60;   //sec
31                ram_gest.stato_concentratore = COLLAUDO;
32                GPIO_PinOutSet(REED_BYPASS_PORT, REED_BYPASS_PIN);
33                timeoutTestMachine = 10000; // set timeout to 10 seconds. In this time
                      interval thera are 100 Tx and Rx
34                initFlag = 0;
35            }
36        }
37        else{
38            if(!timeoutTestMachine){       // if timeout arrived
39                GPIO_PinOutClear(REED_BYPASS_PORT, REED_BYPASS_PIN); // de-initialization
40                testMode = TEST_ON;
41                exitFromTest = 0;
42                workingMode = MODE_TEST;
43                ram_gest.cnt_sec_collaudo = 60;   //sec
44                ram_gest.stato_concentratore = COLLAUDO;
45
46                initFlag = 1; // re-initialization for next cycle test
47                initDelay = 1;
48
49                testResultArrayMeasurements[CALIB_VALUE] = periphTestCalibration;
50                if(rssiPeriphTest < 0){
51                    testResultArrayMeasurements[RSSI] = - rssiPeriphTest; // change of sign
52                    testResultArrayMeasurements[RSSI_SIGN] = 1;
53                }
54                else{
55                    testResultArrayMeasurements[RSSI] = rssiPeriphTest;
56                    testResultArrayMeasurements[RSSI_SIGN] = 0;
57                }
58
59                if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                      increment to the max to finish the test routine
60                    (*testStep) = NUM_TEST_CONTATTO3V;
61                else
62                    (*testStep)++; // else increment of test step for the next step;
63
64                if(periphTestCalibration == 0xFFFF || periphTestCalibration == 0 ||
                      rssiPeriphTest < minRSSI || rssiPeriphTest > maxRSSI )
65                    return TEST_ERROR;
66                else
67                    return TEST_PASSED;
68            }
69        }
70    }
71    return 0;
72 }
```

4.4.4 Verifica ADC tramite controllo batteria

Sul contatto 3V è presente un partitore resistivo: esso è collegato ad un pin del microcontrollore e il valore di tensione viene acquisito dall'ADC. In condizioni di normale funzionamento, la lettura e l'analisi di questo dato serve per stabilire la durata residua della batteria. Se la tensione scende sotto un determinato livello, la batteria è scarica e deve essere sostituita al più presto per garantire il corretto funzionamento della periferica. Il device legge periodicamente e comunica all'SC8R la tensione rilevata ad ogni trasmissione, poiché fa parte del frame radio.

Pertanto, per verificare se le resistenze risultano correttamente montate e l'ADC funzionante, è sufficiente effettuare una stimolazione, acquisire il pacchetto radio e verificare se la tensione rientra nei limiti consentiti. Siccome la periferica è alimentata con una tensione costante, il valore che la macchina si aspetta è pressoché il medesimo. La tensione rilevata dalla periferica viene convertita in step, ognuno da 50mV. Se il valore ricevuto non rientra nei limiti, si potrebbe avere un problema sul partitore (almeno una delle due resistenze non è saldata bene o ha un valore effettivo diverso da quello di progetto) oppure un malfunzionamento dell'ADC.

In seguito si mostra la porzione di codice relativa allo step appena descritto.

```
1   uint8_t batteryVoltageContatto3V(uint8_t *testStep, uint8_t functionalTestMode){
2   static uint8_t initFlag = 1;                    // flag useful for first run
3   static uint8_t initDelay = 1;
4   static uint8_t retry = 3;
5
6   if(initDelay){
7       inhibitTX = 0;
8       GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
9       if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
            correct routine of periph
10          if(retry == 3)
11              timeoutTestMachine = 2000;
12          else
13              timeoutTestMachine = 250;
14      }
15      else
16          timeoutTestMachine = 250;               // else initial delay to ensure non-
                influence of previous step
17      initDelay = 0;
18  }
19  else{
20      if(initFlag){                                // initialization if first run
21          if(!timeoutTestMachine){
22              batteryVoltage = 0;
23              GPIO_PinOutSet(SW_PORT, SW_PIN);
24              timeoutTestMachine = MAX_TIMEOUT;
25              initFlag = 0;
26          }
27      }
28      else{
29          if((batteryVoltage > 57 && batteryVoltage < 69) || !timeoutTestMachine){     //
                check if battery voltage 2900mV<V_batt<3400mV -> every step 50mV
30              GPIO_PinOutClear(SW_PORT, SW_PIN);          // de-initialization
```

```
31              initFlag = 1;                                   // re-initialization for next
                    cycle test
32              initDelay = 1;
33              retry--;
34
35              if(!timeoutTestMachine){                        // check if timeout is reached
36                  if(retry == 0){
37                      retry = 3;
38                      if(functionalTestMode > COMPLETE_TEST_CMD){  // if in single test,
                            increment to the max to finish the test routine
39                          (*testStep) = NUM_TEST_CONTATTO3V;
40                      }
41                      else
42                          (*testStep)++;                          // else increment of test
                                step for the next step
43                      return TEST_ERROR;                          // no periph response
                            within the timeout, error = 1
44                  }
45              }
46              else{
47                  if(functionalTestMode > COMPLETE_TEST_CMD)       // if in single test,
                        increment to the max to finish the test routine
48                      (*testStep) = NUM_TEST_CONTATTO3V;
49                  else
50                      (*testStep)++;                          // else increment of test step
                             for the next step
51                  retry = 3;
52                  return TEST_PASSED;                         // no error, return 0
53              }
54          }
55      }
56  }
57  return 0;
58 }
```

4.4.5 Tamper, aux e reed bypass

In questa sezione vengono raccolti tre step funzionali: il controllo del tamper, dell'ingresso ausiliario e del reed bypass. La struttura risulta essere praticamente la stessa per tutti e tre i controlli, pertanto verrà esposto solo l'esempio relativo allo step del tamper. Dopo una fase di inizializzazione, avviene la stimolazione del pulsante anti-manomissione. A livello firmware, il GPIO corrispondente viene settato ad un livello logico alto, si porta il mosfet in conduzione e il relè viene eccitato. Gli aghi che sono posti sulle piazzole dello switch vengono cortocircuitati, emulando la pressione del tamper; contemporaneamente si resetta anche un timeout. Se la periferica funziona correttamente, invia un pacchetto radio contenente le informazioni dei suoi sensori, tra cui lo stato del pulsante. Se la ricezione avviene entro il tempo a disposizione e lo stato del sensore in esame risulta compatibile con il dato atteso dalla macchina, l'esito dello step risulta positivo.

L'ingresso ausiliario e il reed bypass sono testati in modo analogo. Occorre precisare che nel caso del pulsante anti-manomissione viene verificata la correttezza delle piste e

dei collegamenti sul circuito, ma non il funzionamento meccanico. Se per qualche ragione non facesse contatto o fosse difettoso dal punto di vista meccanico, la macchina non è in grado di scartare la periferica. Lo stimolo effettuato è esclusivamente elettrico, poiché non è presente un attuatore che preme meccanicamente il pulsante. Al termine dell'intera procedura di collaudo, questa operazione sarà effettuata dall'operatore durante il reset di fabbrica del dispositivo.

In seguito viene riportata la porzione di firmware relativa al test del tamper.

```
1   uint8_t swTestContatto3V(uint8_t *testStep, uint8_t functionalTestMode){
2   static uint8_t initFlag = 1;                    // flag useful for first run
3   static uint8_t initDelay = 1;
4   static uint8_t retry = 3;
5
6   if(initDelay){
7       inhibitTX = 0;
8       GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
9       if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
            correct routine of periph
10          if(retry == 3)
11              timeoutTestMachine = 2000;
12          else
13              timeoutTestMachine = 250;
14      }
15      else
16          timeoutTestMachine = 250;                   // else initial delay to ensure non-
                influence of previous step
17      initDelay = 0;
18  }
19  else{
20      if(initFlag){                                   // initialization if first run
21          if(!timeoutTestMachine){
22              radioRxInputState.tamper = 1;
23              GPIO_PinOutSet(SW_PORT, SW_PIN);
24              timeoutTestMachine = MAX_TIMEOUT;
25              initFlag = 0;
26          }
27      }
28      else{
29          if((radioRxInputState.tamper == 0) || !timeoutTestMachine){       // if periph
                response or timeout arrived
30              GPIO_PinOutClear(SW_PORT, SW_PIN);          // de-initialization
31              initFlag = 1;                               // re-initialization for next
                    cycle test
32              initDelay = 1;
33              retry--;
34
35              if(!timeoutTestMachine){                    // check if timeout is reached
36                  if(retry == 0){
37                      retry = 3;
38                      if(functionalTestMode > COMPLETE_TEST_CMD){  // if in single test,
                            increment to the max to finish the test routine
39                          (*testStep) = NUM_TEST_CONTATTO3V;
40                      }
41                      else
42                          (*testStep)++;                      // else increment of test
                                step for the next step
```

```
43                      return TEST_ERROR;                          // no periph response
                            within the timeout, error = 1
44                  }
45              }
46              else{
47                  if(functionalTestMode > COMPLETE_TEST_CMD)       // if in single test,
                        increment to the max to finish the test routine
48                  (*testStep) = NUM_TEST_CONTATTO3V;
49                  else
50                      (*testStep)++;                          // else increment of test step
                            for the next step
51                  retry = 3;
52                  return TEST_PASSED;                         // no error, return 0
53              }
54          }
55      }
56  }
57  return 0;
58 }
```

4.4.6 LED

Il controllo del LED avviene tramite un fototransistor posto fisicamente in corrispondenza della fonte luminosa. Generalmente, la normale operatività del contatto 3V prevede l'accensione del LED per un breve intervallo di tempo ad ogni trasmissione radio; durante la procedura di collaudo, però, il dispositivo mantiene la luce fissa, come indicazione della modalità di funzionamento attiva.

La valutazione del corretto funzionamento del diodo avviene acquisendo la tensione presente in input al GPIO tramite il condizionamento descritto nella sezione hardware. Sebbene possa essere sufficiente un'indicazione ON/OFF, l'acquisizione viene effettuata utilizzando un ADC: è stata identificata la tensione acquisita in condizioni di buio e la soglia è stata opportunamente marginata.

La scelta dell'ADC è giustificata dal fatto che, in caso di manutenzione, il rilevatore luminoso potrebbe essere sostituito e, siccome il beta del fototransistor è variabile da componente a componente, la tensione a valle del sensore potrebbe non essere sufficiente per essere riconosciuta come un livello logico alto dal microcontrollore. Un discorso analogo può essere fatto nel caso la periferica monti un LED differente, con caratteristiche di luminosità che potrebbero variare rispetto alla situazione attuale.

Si riporta il codice relativo allo step di verifica del LED.

```
1     uint8_t ledTestContatto3V(uint8_t *testStep, uint8_t functionalTestMode){
2   uint16_t phototr_voltage;
3   static uint8_t initFlag = 1;                    // flag useful for first run
4   static uint8_t initDelay = 1;
5
6   if(initDelay){
7       inhibitTX = 0;
```

```
8       GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
9       if(functionalTestMode > COMPLETE_TEST_CMD)       // if in single test, wait for the
            correct routine of periph
10          timeoutTestMachine = 2000;
11      else
12          timeoutTestMachine = 50;
13      initDelay = 0;
14    }
15    else{
16      if(initFlag){                                    // initialization if first run
17          setInputADCContatto3V(LED_TEST_ADC_IN);
18          initFlag = 0;
19      }
20
21      if(!timeoutTestMachine){
22          phototr_voltage = ReadAdcTestContatto3V();      // in testing mode led always
                on
23          if(functionalTestMode > COMPLETE_TEST_CMD)       // if in single test,
                increment to the max to finish the test routine
24              (*testStep) = NUM_TEST_CONTATTO3V;
25          else
26              (*testStep)++;                           // else increment of test step for
                    the next step
27          initFlag = 1;                                // re-initialization for next
                cycle test
28          initDelay = 1;
29
30          if(phototr_voltage < 500){                   // if phototr_voltage < 500mV =>
                ERROR
31              return TEST_ERROR;
32          }
33          else{
34              return TEST_PASSED;                      // no error, return 0
35          }
36      }
37    }
38    return 0;
39  }
```

4.4.7 Reed

La verifica del reed consiste nell'emulare l'apertura e la chiusura dell'infisso e verificare che la stimolazione venga rilevata correttamente dalla periferica in collaudo. Il servomotore viene pilotato tramite un segnale PWM, pertanto è necessario utilizzare un timer del microcontrollore.

Di seguito si riporta la configurazione del timer in modalità PWM.

```
1     void timerPWMInit(void)
2   {
3     CMU_ClockEnable(cmuClock_TIMER3, false);    // disable clock for timer
4     CMU_CLOCK_SELECT_SET(TIMER3, HFXO);        // selection of clock source
5     CMU_ClockEnable(cmuClock_TIMER3, true);    // enable clock for timer
6
```

```
  GPIO->TIMERROUTE[3].CC0ROUTE = (uint32_t) ((SERVO_PIN << 16) | (SERVO_PORT));
                        // CC0 of TIMER3 connected to reed
  GPIO->TIMERROUTE[3].ROUTEEN = 0b011;        // route enable for CC0 e CC1

  TIMER3->EN = 0;                  // timer disabled
  while(TIMER3->EN & 0X02)
      ;
  TIMER3->CFG = (99 << 18);        // PRESCALER = 99+1 = 100
  TIMER3->CC[0].CFG = 0x03;        // PWM mode on CC0
  TIMER3->EN = 1;

  TIMER3->TOP = 999;               // interrupt @ 400Hz = 40MHz / (100 * (TOP + 1))
  PWM_ReedSetContatto3V(0);                // 0 degree

  TIMER3->CMD = 0x01;              // start timer
}
```

Da datasheet, l'angolo di rotazione del servomotore è definito dalla durata dell'impulso alto del segnale PWM: con un valore di 500μs si ha un angolo di 0°, mentre con 2500μs si a un angolo di circa 180° (che corrisponde alla massima escursione).

Pertanto, per settare il timer è stata usata la seguente espressione:

$$\mathrm{PWM_{up}} = \frac{f_{\mathrm{CLK}}}{(\mathrm{PRESC}+1)\times(\mathrm{TOP}+1)}$$

Dove f_{CLK} corrisponde a 40MHz, `PRESC` è il prescaler del timer ed è stato definito a 99; $\mathtt{PWM_{up}}$ deve corrispondere al periodo massimo (e quindi la frequenza minima, di 400Hz) e può essere ottenuto con:

$$f_{min} = \frac{1}{2500\mu s} = 400\mathrm{Hz}$$

Invertendo la prima formula, è quindi possibile ricavare `TOP`:

$$\mathrm{TOP} = \frac{f_{\mathrm{CLK}}}{(\mathrm{PRESC}+1)\times \mathrm{PWM_{up}}} - 1 = 999$$

Infine, è possibile variare l'angolo cambiando il valore memorizzato nel registro `OC` (output compare). Con la seguente funzione, inserendo un valore di duty cycle, si ottiene una variazione di angolo corrispondente.

```
    void PWM_ReedSetContatto3V(uint8_t PWM_DC){
  TIMER3->CC[0].OC = (8*PWM_DC) + 200;
}
```

Il servomotore ha un magnete montato sull'estremità del braccio. Inizialmente, si trova in posizione di riposo con il magnete lontano dal sensore reed; successivamente, il braccio viene chiuso avvicinando il magnete al sensore. Se la periferica funziona correttamente, si ha una prima trasmissione radio dal DUT in cui il sensore deve risultare chiuso; verificato

tale passaggio, il braccio del servo si allontana nuovamente, ritornando nella situazione di riposo. Questa variazione deve scaturire un'altra trasmissione radio, il cui il bit del sensore reed dovrebbe assumere il valore opposto. Se questa sequenza viene verificata, l'esito dello step sarà positivo.

La porzione di codice descritta è riportata in seguito.

```
  int8_t reedTestContatto3V(uint8_t *testStep, uint8_t functionalTestMode){
static uint8_t initFlag = 1;                    // flag useful for first run
static uint8_t initDelay = 1;
static uint8_t retry = 3;
static uint8_t alreadyClosed = 0;

if(initDelay){
    inhibitTX = 0;
    GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
    PWM_ReedSetContatto3V(0);
    if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
         correct routine of periph
        if(retry == 3)
            timeoutTestMachine = 2000;
        else
            timeoutTestMachine = 500;
    }
    else
        timeoutTestMachine = 500;                // else initial delay to ensure non-
            influence of previous step
    initDelay = 0;
    alreadyClosed = 0;
}
else{
    if(initFlag){                                // initialization if first run
        if(!timeoutTestMachine){
            radioRxInputState.reed = 1;
            PWM_ReedSetContatto3V(65);                    // PWM set to 65% => reed
                opened
            timeoutTestMachine = MAX_TIMEOUT;
            initFlag = 0;
        }
    }
    else{
        if((radioRxInputState.reed == 0 || !timeoutTestMachine) && !alreadyClosed){
                  // if periph response or timeout arrived
            PWM_ReedSetContatto3V(0);

            if(!timeoutTestMachine){                    // check if timeout is reached
                initFlag = 1;                               // re-initialization for
                    next cycle test
                initDelay = 1;
                retry--;
                if(retry == 0){
                    retry = 3;
                    if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                        increment to the max to finish the test routine
                        (*testStep) = NUM_TEST_CONTATTO3V;
                    else
                        (*testStep)++;                      // else increment of test
                            step for the next step
```

```
45                      return TEST_ERROR;                          // no periph response
                            within the timeout, error = 1
46                  }
47              }
48              else{
49                  alreadyClosed = 1;
50                  timeoutTestMachine = MAX_TIMEOUT;
51              }
52          }
53
54          if((radioRxInputState.reed == 1 || !timeoutTestMachine) && alreadyClosed){
55              initFlag = 1;                                   // re-initialization for next
                    cycle test
56              initDelay = 1;
57              retry--;
58              if(!timeoutTestMachine){                        // check if timeout is reached
59                  if(retry == 0){
60                      retry = 3;
61                      if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                            increment to the max to finish the test routine
62                          (*testStep) = NUM_TEST_CONTATTO3V;
63                      else
64                          (*testStep)++;                      // else increment of test
                                step for the next step
65                      return TEST_ERROR;                      // no periph response
                            within the timeout, error = 1
66                  }
67              }
68              else{
69                  if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                        increment to the max to finish the test routine
70                      (*testStep) = NUM_TEST_CONTATTO3V;
71                  else
72                      (*testStep)++;                          // else increment of test step
                             for the next step
73                  retry = 3;
74                  return TEST_PASSED;                         // no error, return 0
75              }
76          }
77      }
78  }
79  return 0;
80 }
```

4.4.8 Accelerometro

Per la verifica dell'accelerometro, viene utilizzato un motore vibrazionale, con due pesi eccentrici montati sull'albero. Il motore è fissato alla struttura metallica ed essendo solidale ad essa, consente il trasferimento delle vibrazioni al DUT quando è in movimento.

Inizialmente è stato pensato di accendere e spegnere ripetutamente il motore per avere vibrazioni ravvicinate tra di loro, ma ad intermittenza. Sperimentalmente, invece, tale soluzione non è risultata robusta poiché alcune periferiche funzionanti fornivano un esito negativo su questo step (falsi negativi). Questo era dovuto al fatto che il motore è piuttosto

compatto e le vibrazioni risultavano troppo deboli per sollecitare l'accelerometro del DUT. Tuttavia, mantenendo il motore costantemente in rotazione durante lo step e aumentando il duty cycle (e quindi l'alimentazione media fornita al motore), il problema degli esiti falsi negativi è stato risolto.

Di seguito viene riportata la porzione di codice appena descritta. Risulta ancora presente qualche riga di codice relativa alla gestione ad intermittenza poiché nel caso il motore venga sostituito con uno più grande è possibile adottare questa soluzione.

```
1   uint8_t accelerometerTestContatto3V(uint8_t *testStep, uint8_t functionalTestMode){
2   static uint8_t initFlag = 1;                    // flag useful for first run
3   static uint8_t initDelay = 1;
4   static uint8_t retry = 30, motorState = 0;
5
6   if(initDelay){
7       inhibitTX = 0;
8       GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
9       if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
            correct routine of periph
10          if(retry == 3)
11              timeoutTestMachine = 2000;
12          else
13              timeoutTestMachine = 1000;
14      }
15      else
16          timeoutTestMachine = 1000;              // else initial delay to ensure non-
                influence of previous step
17      initDelay = 0;
18  }
19  else{
20      if(initFlag){                               // initialization if first run
21          if(!timeoutTestMachine){
22              radioRxInputState.inertial = 0;
23              PWM_accelerometerSetContatto3V(100);          // PWM set => motor ON
24              timeoutTestMachine = MAX_TIMEOUT/4;
25              initFlag = 0;
26              retry = 30;                         // 20 means 10 activation
27              motorState = 1;
28          }
29      }
30      else{
31          if(retry){
32              if(radioRxInputState.inertial == 1){
33                  retry = 0;
34              }
35              else{
36                  if(timeoutTestMachine == 0){
37                      timeoutTestMachine = MAX_TIMEOUT/4;
38                      retry--;
39                      //motorState ^= 1; // uncomment if you want to perform ON/OFF of
                            the motor
40                      if(motorState)
41                          PWM_accelerometerSetContatto3V(100);
42                      else
43                          PWM_accelerometerSetContatto3V(0);
44                  }
```

```
45              }
46          }
47          else{
48              PWM_accelerometerSetContatto3V(0);                    // PWM set to 0 =>
                    motor OFF
49              if(functionalTestMode > COMPLETE_TEST_CMD)       // if in single test,
                    increment to the max to finish the test routine
50                  (*testStep) = NUM_TEST_CONTATTO3V;
51              else
52                  (*testStep)++;                              // else increment of test step
                        for the next step
53              initFlag = 1;                                   // re-initialization for next
                    cycle test
54              initDelay = 1;
55              retry = 3;
56
57              if(radioRxInputState.inertial == 1)             // if periph response or
                    timeout arrived
58                  return TEST_PASSED;                         // no error, return 0
59              else
60                  return TEST_ERROR;                          // no periph response within
                        the timeout, error = 1
61
62          }
63      }
64  }
65  return 0;
66 }
```

4.4.9 Update ID periferica

L'ultimo step di collaudo della macchina risulta essere l'update del codice univoco delle periferiche.

Quando un dispositivo viene programmato, ha un ID corrispondente a `FFFFFF`; alla prima accensione, in base al firmware presente in flash, si riconosce la tipologia della periferica e le prime due cifre vengono modificate permanentemente. Per il contatto magnetico, ad esempio, le prime due cifre sono 00 e il codice, a seguito della modifica iniziale, sarà `00FFFF`.

Tutti i contatti, se non viene effettuato un update dell'ID, saranno pertanto indistinguibili poiché presenterebbero tutti lo stesso codice.

Per ovviare a questo problema, al termine del collaudo deve essere effettuato un aggiornamento del serial number. Tale operazione è nuovamente effettuata via radio, mediante una procedura apposita. Inizialmente vengono controllati gli esiti di tutti gli step precedenti: nel caso anche solo uno di questi abbia fallito, l'update viene saltato poiché non ha senso aggiornare l'ID di uno scarto e consumare inutilmente un codice. Nel caso che tutti gli step siano stati positivi, invece, la macchina di collaudo popola opportunamente le variabili della radio, configurandosi in modo tale che venga mandato un messaggio di tipo `TEST_WR_SERIAL_NUM` alla periferica. A questo punto, con una prima stimolazione,

si comunica alla periferica il serial number che è stato inserito dall'utente tramite l'interfaccia; in seguito si effettua una seconda stimolazione con lo scopo di controllare se la scrittura del nuovo identificativo è stata completata con successo, analizzando il pacchetto radio ricevuto. Nel caso il codice target e quello ricevuto coincidano, l'esito risulterà essere positivo.

Di seguito si riporta la porzione di codice relativa all'aggiornamento del serial number.

```
1   uint8_t updatePeriphIDContatto3V(uint8_t *testStep, uint8_t functionalTestMode,
        uint8_t *testResultArray){
2   uint8_t i=0;
3   static uint8_t completeTestFailed = 0;
4
5   static uint8_t initFlag = 1;                        // flag useful for first run
6   static uint8_t initDelay = 1;
7   static uint8_t verifyPhase = 0;
8
9   uint8_t tot_retry = 3;
10  static uint8_t retry = 3;
11
12  if(initDelay){
13      inhibitTX = 0;
14      verifyPhase = 0;
15      GPIO_PinOutSet(POWER_SUPPLY_CONTR_PORT, POWER_SUPPLY_CONTR_PIN);
16      if(functionalTestMode > COMPLETE_TEST_CMD){      // if in single test, wait for the
            correct routine of periph
17          completeTestFailed = 0;
18          if(retry == tot_retry)
19              timeoutTestMachine = 2000;
20          else
21              timeoutTestMachine = 250;
22      }
23      else{
24          timeoutTestMachine = 250;               // else initial delay to ensure non-
                influence of previous step
25
26          completeTestFailed = 0;
27          while((i<(UPDATE_PERIPH_ID_CONTATTO3V)) && !completeTestFailed){
28              if(testResultArray[i] == 1){
29                  completeTestFailed = 1;
30              }
31              i++;
32          }
33      }
34
35      initDelay = 0;
36      if(!completeTestFailed){
37          inp_out.timer_led[LED_MAN_SENS] = 255;
38          testMode = TEST_WR_SERIAL_NUM;
39          workingMode = MODE_TEST;
40          ram_gest.cnt_sec_collaudo = 60;   //sec
41          ram_gest.stato_concentratore = COLLAUDO;
42      }
43      else{
44          initFlag = 1;                                   // re-initialization for next
                cycle test
45          initDelay = 1;
```

```
46          retry = tot_retry;
47          if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test, increment to
                the max to finish the test routine
48              (*testStep) = NUM_TEST_CONTATTO3V;
49          else
50              (*testStep)++;                          // else increment of test step for the
                    next step
51          return TEST_ERROR;
52      }
53  }
54  else{
55      if(initFlag){                                   // initialization if first run
56          if(!timeoutTestMachine){
57              GPIO_PinOutSet(REED_BYPASS_PORT, REED_BYPASS_PIN);  //  stimolation to
                    write new ID
58              timeoutTestMachine = 1250;
59              initFlag = 0;
60          }
61      }
62      else{
63          if(!timeoutTestMachine && !verifyPhase){       // if periph response or timeout
                arrived
64              GPIO_PinOutClear(REED_BYPASS_PORT, REED_BYPASS_PIN); // stimolation to read
                    ID
65              verifyPhase = 1;
66              timeoutTestMachine = MAX_TIMEOUT/4;
67
68              //Exit from wb_serial ID
69              periphToTest = newPeriphId;
70              testMode = TEST_ON;
71              exitFromTest = 0;
72              workingMode = MODE_TEST;
73              ram_gest.cnt_sec_collaudo = 60;   //sec
74              ram_gest.stato_concentratore = COLLAUDO;
75          }
76          else if(!timeoutTestMachine && verifyPhase){
77              initFlag = 1;                               // re-initialization for next
                    cycle test
78              initDelay = 1;
79              retry--;
80
81              if(retry == 0){
82                  retry = tot_retry;
83                  if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                        increment to the max to finish the test routine
84                      (*testStep) = NUM_TEST_CONTATTO3V;
85                  else
86                      (*testStep)++;                      // else increment of test step
                            for the next step
87
88                  if(actualPeriphID == newPeriphId)
89                      return TEST_PASSED;                     // no error, return 0
90                  else
91                      return TEST_ERROR;
92              }
93              else{
94                  if(actualPeriphID == newPeriphId){
95                      if(functionalTestMode > COMPLETE_TEST_CMD)   // if in single test,
                            increment to the max to finish the test routine
96                          (*testStep) = NUM_TEST_CONTATTO3V;
```

```
97                          else
98                              (*testStep)++;                          // else increment of test
                                    step for the next step
99                          retry = tot_retry;
100                         return TEST_PASSED;                         // no error, return 0
101                     }
102                     else
103                         return TEST_ERROR;
104                 }
105             }
106         }
107     }
108     return 0;
109 }
```

4.5 Implementazione software e pagina web

Terminata la descrizione hardware e firmware della macchina di collaudo, si procede alla progettazione dell'interfaccia utente, dell'interazione tra frontend e backend e della comunicazione bidirezionale con il firmware. Tutte queste operazioni vengono svolte dalla Raspberry Pi 3B+, il master del sistema.

4.6 User interface e backend

Quando la macchina viene avviata, la prima finestra che compare è l'interfaccia utente. L'operatore si troverà davanti una schermata con la quale è possibile interagire per svolgere una qualsiasi procedura di collaudo. La macchina di collaudo dei contatti 868MHz è stata la prima ad essere realizzata, ma l'interfaccia è stata pensata e predisposta al fine di poter integrare nuove funzioni e periferiche, in ottica di futuri sviluppi. L'interfaccia utente è stata sviluppata in HTML, Javascript e CSS. Essa si presenta come in Figura 39.

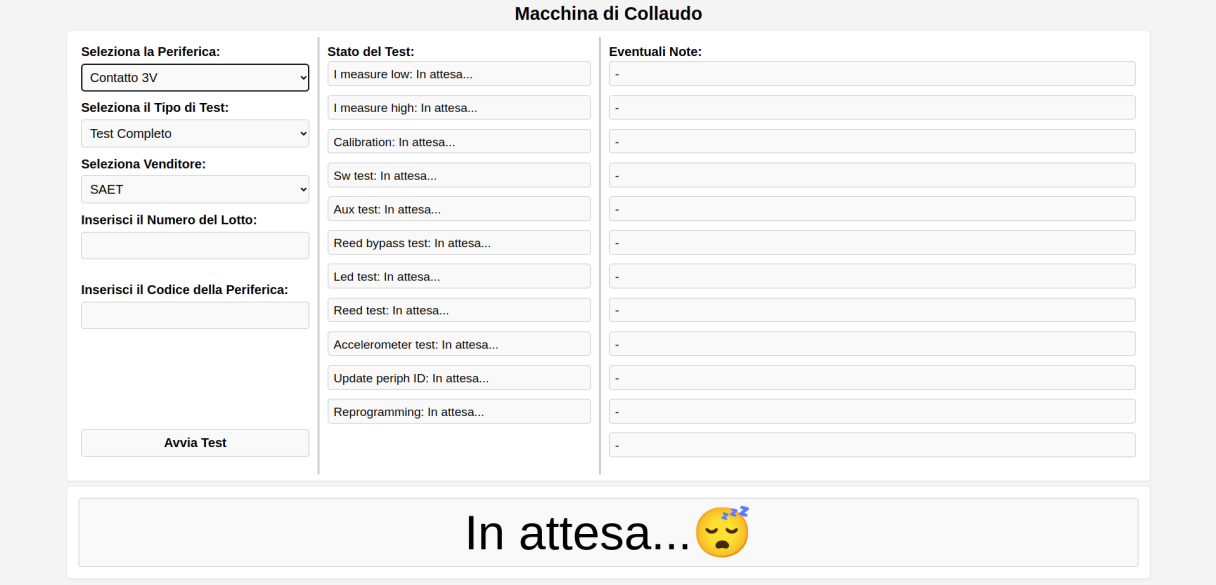

Figura 39: Interfaccia utente.

La finestra è suddivisa in 3 pannelli verticali che occupano gran parte dello schermo, mentre un pannello orizzontale occupa la parte inferiore. Nel primo pannello a sinistra sono presenti tre menù di selezione a tendina, due menù per l'inserimento di codici e un pulsante di avvio test. In questa sezione ogni menù è modificabile dall'utente, ed è possibile selezionare il tipo di periferica da collaudare, la tipologia di test da effettuare e infine l'azienda al quale è destinato quel prodotto. Quest'ultima selezione permette di caricare in flash al DUT la variante firmware per il cliente selezionato in modo che i prodotti destinati ad un installatore non siano compatibili e non possano comunicare con

quelli di un altro venditore. Gli altri campi servono per assegnare un serial number alla periferica e per la stesura del report. Infine, il pulsante di start avvia la procedura di test in base alle selezioni precedenti.

Nel pannello centrale sono indicati gli step che verranno eseguiti in base alle selezioni effettuate. Ad ogni riga è associato uno step funzionale differente, nel quale si indica lo stato attuale. Cambiando la tipologia di test, l'elenco viene aggiornato immediatamente.

Nel pannello destro sono presenti le eventuali note. Esse riportano un messaggio al termine della procedura di test: nel caso uno degli step all'interno del pannello centrale fosse negativo, si ha una breve descrizione della motivazione del fallimento dello step; la casella rimane vuota nel caso lo step fornisca un esito positivo.

Infine, il pannello orizzontale mostra la fase in cui il collaudo si trova. Prima del test, come si evince dall'immagine, è presente la scritta "In attesa..."; durante il collaudo, compare il messaggio "In esecuzione..." e una barra di caricamento avanza, indicando lo svolgimento del test. Al termine della procedura, in questo pannello compare il resoconto dell'esito: viene effettuata una OR logica tra tutti gli step svolti e se ce ne fosse anche solo uno con esito negativo, il pezzo è etichettato come scarto.

Si evidenzia l'inserimento delle emoticon nell'interfaccia utente. Sono presenti una decina di smile differenti, ciascuno associato ad un diverso messaggio. L'introduzione delle emoji risponde a una richiesta del capo produzione, e pertanto rappresenta una specifica di progetto. Nonostante possa sembrare poco professionale, questa scelta può avere un impatto positivo sulla motivazione e sul benessere degli operatori. Da un punto di vista psicologico, l'utilizzo di elementi grafici informali può alleggerire l'atmosfera e ridurre la percezione di formalità nell'ambiente di lavoro. Pertanto, si rende l'interazione con la macchina user friendly e meno stressante, migliorando la produttività in modo indiretto.

Di seguito si inseriscono alcune immagini per mostrare come si presenta una schermata con esito positivo ed una che mostra un esito negativo. Nel primo esempio, i test `I_measure_low` e `Sw_test` sono falliti. Questo ha portato ad avere un esito finale negativo e gli step `Update_periph_ID` e `Reprogramming` non sono stati svolti, pertanto risultano negativi. Nel pannello a destra si possono notare le note associate ai test falliti: il numero indica il tipo di errore e la descrizione indica in modo puntuale una delle possibili cause. Questo dispositivo risulta essere difettoso e non può essere venduto nelle attuali condizioni. Tuttavia, tramite il codice indicato, è possibile riprendere la periferica e correggere l'errore o il malfunzionamento, laddove possibile, andando a sostituire i componenti opportuni.

In Figura 41 si mostra l'esempio di una periferica che ha superato con successo tutte le fasi del collaudo. Dopo il montaggio finale, sarà possibile venderla.

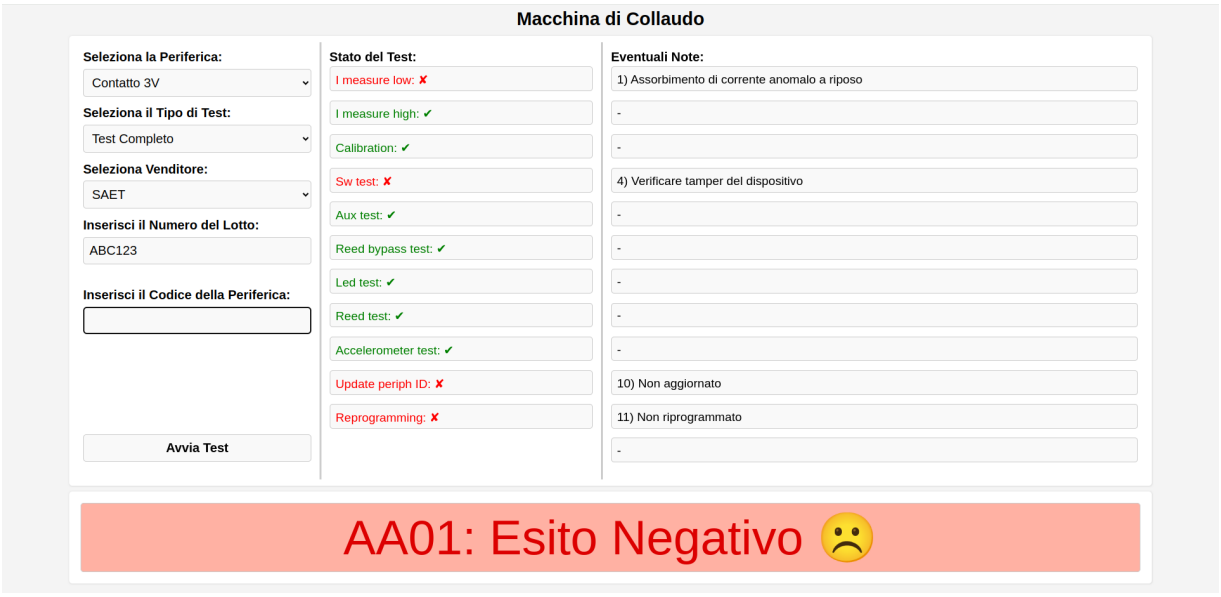

Figura 40: Interfaccia utente con esito negativo.

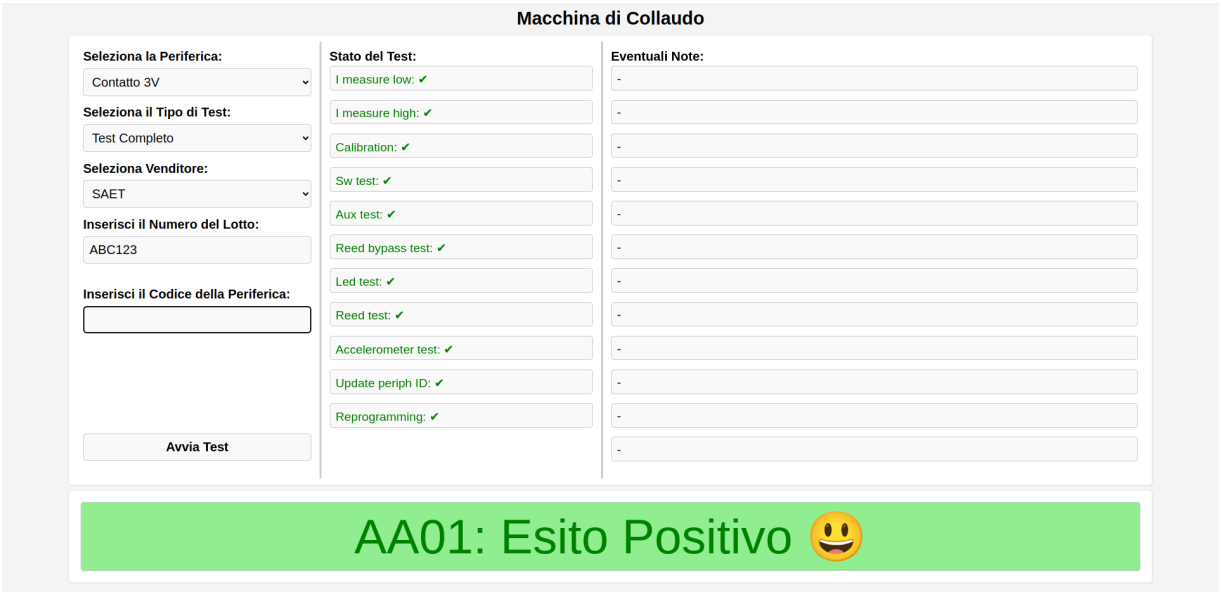

Figura 41: Interfaccia utente con esito positivo.

4.6.1 Struttura del codice della pagina web

A basso livello, il codice è strutturato in diversi file e moduli. Il file HTML contiene il codice strutturale di una pagina web. Pertanto, vengono introdotti i menù e i campi di selezione, pulsanti, caselle di testo, riquadri e immagini e molti altri elementi grafici. Lo stile, la forma, il font, i colori, i margini e le varie dimensioni sono specificate all'interno del file CSS; in sostanza, si tratta di una sorta di file di stile, che descrive come gli elementi grafici devono essere visualizzati. Ogni elemento ha un tag con il quale viene richiamato, e la pagina web lo mostra in base alle caratteristiche dell'oggetto. Infine, l'HTML ha collegamenti a file e risorse esterne, scritte in JavaScript. Il loro scopo è quello di rendere la pagina interattiva e dinamica, permettendo la gestione di funzionalità avanzate come il controllo eventi. In questa sezione saranno presi in considerazione solo alcuni punti chiave, evitando di approfondire dettagli meno rilevanti.

Tra i file JavaScript, *config.js* è utilizzato per settare i parametri di default della pagina, come la periferica e il tipo di test da eseguire. Quando la pagina viene caricata, in base alle configurazioni presenti in questo file, vengono mostrate automaticamente le selezioni nei relativi menù. Di seguito si riporta una breve porzione di codice. In questo caso, l'impostazione di default risulta essere `DEVICE_CONTATTO3V` poiché la macchina è quella dei contatti magnetici. Quando la pagina web verrà caricata, nel primo menù comparirà automaticamente questa selezione.

La successiva struttura fa riferimento al codice hardware della macchina. Siccome verranno realizzate più macchine di collaudo con una struttura in comune, queste saranno componibili. Il codice serve come sicurezza aggiuntiva, per assicurarsi che non vengano montate parti che fanno riferimento a macchine differenti. Questo verrà approfondito in seguito, nella Sezione 6.

```
    const configData = {
    selectedOptionPeriph: DEVICE_CONTATTO3V,

    // Hardware codes present on the machine.
    // K = kit, P = produzione
    hardwareCodes: {
        [DEVICE_CONTATTO3V]:        "p000k",
        [DEVICE_PLATINO]:           "p001k",
        [DEVICE_TELECOMANDO]:       "p002k",
        [DEVICE_SIRENA_EXT]:        "p003k",
        [DEVICE_SIRENA_INT]:        "p006k",
        [DEVICE_VXS_RAM]:           "p008k",
        [DEVICE_KAPTURE]:           "p013k",
        [DEVICE_SC8R]:              "p127k"
    }
};
```

Nel file *functions.js* vengono gestite le principali funzionalità dell'interfaccia utente. Tra queste vi è la gestione dei menù di selezione e il riempimento dei campi in base alle scelte effettuate, l'aggiornamento dei test, delle note e alcuni controlli come l'accesso a menù non destinati alla produzione (protetti da password).

La seguente porzione di codice gestisce proprio le impostazioni di default. Viene creato l'elenco delle possibili periferiche che si possono testare e si setta in automatico questo menù con il contenuto della variabile presente nel file *config.js*.

```
// Function to initialize the options
function initializeDeviceOptions() {
    // Map of options
    const options = [
        { value: CALIBRAZIONE_MACCHINA_TEST, text: "Calibrazione Machine Test" },
        { value: DEVICE_CONTATTO3V,          text: "Contatto 3V" },
        { value: DEVICE_PLATINO,             text: "Platino" },
        { value: DEVICE_TELECOMANDO,         text: "Telecomando" },
        { value: DEVICE_SIRENA_EXT,          text: "Sirena Esterno" },
        { value: DEVICE_SIRENA_INT,          text: "Sirena Interno" },
        { value: DEVICE_VXS_RAM,             text: "Optex" },
        { value: DEVICE_KAPTURE,             text: "Kapture" },
        { value: DEVICE_SC8R,                text: "SC8R" },

        { value: DEVICE_DEBUG_CONTATTO3V,    text: "Debug Contatto 3V" },
        { value: DEVICE_DEBUG_PLATINO,       text: "Debug Platino" },
        { value: DEVICE_DEBUG_TELECOMANDO,   text: "Debug Telecomando" },
        { value: DEVICE_DEBUG_SIRENA_EXT,    text: "Debug Sirena Esterno" },
        { value: DEVICE_DEBUG_SIRENA_INT,    text: "Debug Sirena Interno" },
        { value: DEVICE_DEBUG_VXS_RAM,       text: "Debug Optex" },
        { value: DEVICE_DEBUG_KAPTURE,       text: "Debug Kapture" },
        { value: DEVICE_DEBUG_SC8R,          text: "Debug SC8R" },

    ];

    // Function to populate options
    function populateOptions() {
        const selectElement = document.getElementById('device-select');
        options.forEach(option => {
            const optionElement = document.createElement('option');
            optionElement.value = option.value;
            optionElement.textContent = option.text;
            selectElement.appendChild(optionElement);
        });
    }

    // Function to set the selected option based on configuration
    function setSelectedOption() {
        const selectElement = document.getElementById('device-select');
        selectElement.value = configData.selectedOptionPeriph;
    }

    populateOptions();
    setSelectedOption();
}
initializeDeviceOptions();
```

Un'altra funzionalità importante risulta essere il controllo sull'effettivo mounting dell'archivio. Tale operazione permette di capire se la macchina risulta connessa alla rete e se è in grado di raggiungere l'archivio aziendale, per l'aggiornamento dei firmware e per il backup dei report. Il mount vero e proprio è effettuato dal backend, al termine del quale viene inviato l'esito; a questo punto, la pagina interpreterà lo stato e farà comparire un popup che indica l'irraggiungibilità dell'archivio in caso di mancanza di connessione.

```
function archiveMountedCheck() {
    var payload = {
        "archive": "checking",
    };

    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'http://localhost:8080', true);
    xhr.onreadystatechange = function () {
        if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
            var responseFromC = JSON.parse(xhr.responseText);
            console.log(responseFromC);
            if (responseFromC.archive === "KO") {
                showPopupArchiveMountedCheck("ARCHIVIO NON RAGGIUNGIBILE");
            }
        }
    }
    xhr.send(JSON.stringify(payload));
}

function showPopupArchiveMountedCheck(message) {
    var popup = document.createElement('div');
    popup.className = 'popup show';
    popup.innerHTML = '<p>' + message + '<br><br>Contattare ufficio tecnico<br>oppure<br>
        Cliccare sul riquadro<br>per proseguire comunque</p>';
    document.body.appendChild(popup);

    popup.addEventListener('click', function () {
        popup.classList.remove('show');
        document.body.removeChild(popup);
    });
}
window.onload = archiveMountedCheck;
```

Infine, per ogni periferica sono presenti ulteriori due file; nel caso del CS3965, questi sono *testContatto3V.js* e *debugContatto3V.js* e gestiscono il test vero e proprio. La macchina di collaudo ha la possibilità di svolgere un test completo, un test singolo o entrare nella modalità debug. I primi due casi sono destinati al reparto produzione e permettono di svolgere operazioni e controlli aggiuntivi, come la verifica di un eventuale cortocircuito e la programmazione del DUT, mentre la modalità di debug è destinata ad operatori qualificati che vogliono effettuare test particolari o attività di sviluppo. Non è possibile variare la selezione liberamente, ma un meccanismo di sicurezza richiede una password per proseguire, al fine di evitare che per sbaglio un operatore cambi involontariamente un menù e svolga test che sono parziali o diversi rispetto alla sua mansione.

La struttura interna dei due file è molto simile: si ha una funzione sensibile al pulsante di start e quando questo viene premuto, si avvia il test. Inizialmente si controlla che i campi relativi al lotto e al codice della periferica siano compilati correttamente; se tutte le verifiche vengono superate, si apre la comunicazione tra il frontend, che rappresenta il client, e il backend, che svolge il ruolo di server. Il client invia una richiesta al server locale `localhost:8080` in cui specifica in formato JSON tutti i dati inseriti nel pannello sinistro dell'interfaccia utente e li invia al programma in C che gira in background. Il server scomporrà e analizzerà tutti i parametri e li manderà al firmware, dove verranno gestiti i vari step del collaudo. Al termine, la risposta contenente gli esiti del test sarà spedita indietro al client. A quel punto, la pagina si aggiornerà mostrando all'operatore i risultati, come descritto nella Sezione 4.6.

In questo segmento di codice si mostra come avviene il popolamento del `payload` e com'è effettuata la richiesta del client. Essendoci la parola chiave `true`, la richiesta non è bloccante e la pagina web continua ad eseguire normalmente. Quando il test è finito, lo stato della richiesta diventa `ready` e siccome la funzione è sensibile al cambiamento, prosegue oltre. La differenza tra i file di test e debug risiede nella compilazione dei risultati. Nel primo caso, ogni step è associato al relativo esito e la nota corrispondente viene popolata in caso di risultato negativo, mentre nel secondo caso si mostrano anche eventuali misure.

```
    var payload = {
                "deviceSelect": deviceSelect,
                "testSelect": testSelect,
                "brand": brand,
                "periphID": periphID,
                "lottoID": lottoID,
                "futureID": "FFFF",
                "volumeStatus": volumeStatus,
    };

    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'http://localhost:8080', true);
    xhr.onreadystatechange = function () {
        if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
            ...
            // Lettura esiti e aggiornamento interfaccia con risultati
            ...
        }
        ...
    }
```

4.6.2 Codice del backend

Il backend è un programma scritto in linguaggio C che mette in comunicazione l'interfaccia utente con il firmware della macchina; si tratta di un programma che viene eseguito in background e funge da ponte. Anche in questo caso, i file sono organizzati in una struttura modulare, comprendendo funzioni generali e routine specifiche per ogni periferica.

In Figura 42 si mostra il flowchart del backend della macchina di test. Il main è diviso in una prima parte in cui viene configurato il server per la comunicazione con l'interfaccia utente e la porta seriale per lo scambio di dati con il firmware. Inoltre, si verifica la raggiungibilità dell'archivio e vengono estrapolate le versioni firmware presenti in locale. Queste informazioni verranno ribaltate sull'interfaccia, e nel caso la rete non fosse presente, un popup comparirà per avvisare l'operatore dell'assenza di connessione.

Successivamente, si ha il loop infinito responsabile dei test. Ogni volta che viene avviato un test dall'interfaccia, si verifica la presenza di un cortocircuito sull'alimentazione della periferica e la corretta armatura della macchina tramite il pin di reset del DUT. Se entrambi i controlli risultano positivi, si effettuata la programmazione con una variante firmware necessaria per svolgere i test. A questo punto, tramite un messaggio in seriale, il controllo viene lasciato alla scheda di collaudo CS3981, che svolgerà uno step alla volta (nel caso di test completo), come descritto nella Sezione 4.4. Al termine, il firmware restituirà il controllo al backend, insieme agli esiti di ciascun test e alle relative misure e caratterizzazioni. Se tutti i risultati sono positivi, la periferica non presenta anomalie e può essere riprogrammata con il codice definitivo, con la quale sarà commercializzata. Verrà anche generato un report con tutte le informazioni principali, che sarà reso disponibile per future consultazioni.

Gli esiti vengono comunicati all'interfaccia utente, attraverso la risposta e la chiusura della comunicazione con il client. Ora sarà possibile ripetere nuovamente la procedura al fine di collaudare un altro sensore.

Si mostra di seguito il diagramma di flusso principale del backend e saranno riportati segmenti di codice ritenuti importanti.

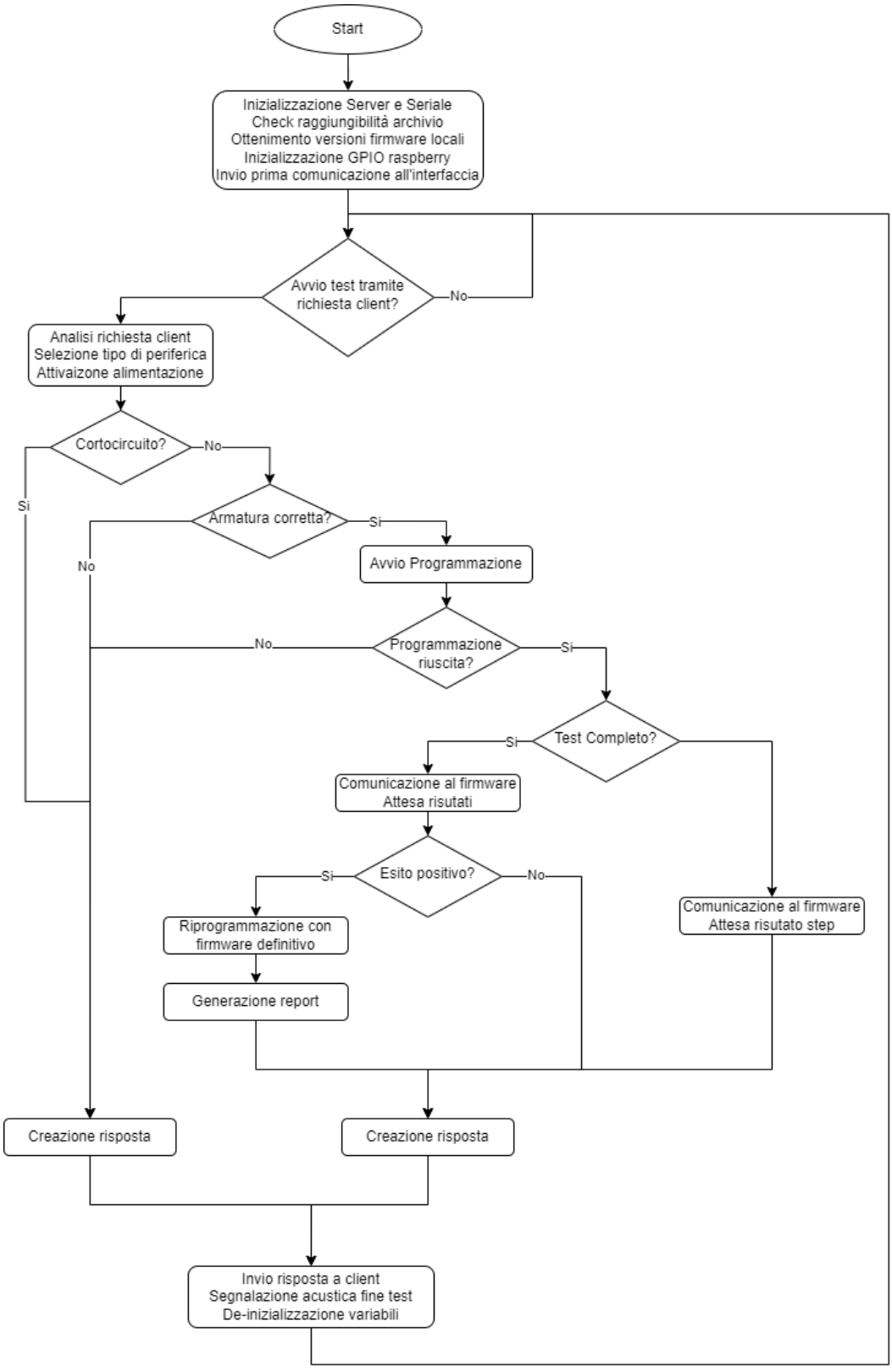

Figura 42: Diagramma di flusso Backend.

Il primo controllo che viene eseguito risulta essere la verifica di un cortocircuito che potrebbe essere potenzialmente distruttivo. Dopo aver fornito alimentazione al DUT tramite l'attivazione dei relè, si controlla il livello logico del polo positivo tramite il sensing in input ad un GPIO della raspeberry. Nel caso fosse effettivamente presente un corto, l'alimentazione viene rimossa immediatamente e il test è concluso con un esito negativo. In caso contrario, si procede oltre.

```
int shortCircuitVerification(){
    printf("\nPower supply ON\n");
    digitalWrite(GPIO_RPI_VCC_GND, HIGH);
    usleep(25000); // 25000 us => 25 ms
    printf("Short circuit verification...\n");
    if(digitalRead(GPIO_RPI_CC_SENSING) == 1){
        //digitalWrite(GPIO_RPI_VCC_GND, LOW);
        return OK;
    }
    else{
        digitalWrite(GPIO_RPI_VCC_GND, LOW);
        return ERROR;
    }
}
```

Il secondo controllo è la corretta armatura della macchina. Il pin di reset del sensore è sempre alto tramite un pullup, pertanto è possibile identificare la presenza del DUT o la non completa chiusura della meccanica acquisendo il livello logico di quel pin.

```
int armorVerification(){
    printf("\nArmor verification...\n");
    digitalWrite(GPIO_RPI_RST_CONTROL, HIGH);
    usleep(500000); // 500000 us => 500 ms
    if(digitalRead(GPIO_RPI_RST_IN) == 1){
      digitalWrite(GPIO_RPI_RST_CONTROL, LOW);
      usleep(500000); // 500000 us => 500 ms
      return OK;
    }
    else{
        digitalWrite(GPIO_RPI_RST_CONTROL, LOW);
        digitalWrite(GPIO_RPI_VCC_GND, LOW);
        return ERROR;
    }
}
```

Queste due funzioni appena descritte sono generali e sono svolte su tutte le periferiche.

La programmazione firmware del DUT risulta invece specifica e si adatta alla periferica da testare. La funzione che svolge questo compito, nel caso del contatto magnetico, è la `programContatto3V()`.

Oltre all'alimentazione, vengono collegati i segnali di `SWCLK` e `SWDIO` e si seleziona il codice da caricare in flash tramite la funzione `setFileProgramContatto3V()`, riportata poco sotto. La funzione `popen()` serve a eseguire un comando o un programma esterno, permettendo la comunicazione con esso tramite una pipe. Nello specifico, si imposta il file da caricare e viene lanciata la procedura di flash tramite un eseguibile fornito direttamente dal costruttore del programmatore. Sempre tramite la `popen()`, si può anche leggere l'output del comando, andando a ricercare la stringa *"Flashing completed successfully!"* che indica il corretto completamento dell'operazione.

```
int programContatto3V(int brand) {
    FILE *fp;
    char lineBuffer[MAX_LINE_BUFFER_SIZE];
    int resultProg = 1;

    // setup GPIO to program the peripheral
    wiringPiSetup();
    wiringPiSetupGpio();
    pinMode(GPIO_RPI_SWDIO_SWCLK, OUTPUT);
    pinMode(GPIO_RPI_VCC_GND, OUTPUT);
    pinMode(GPIO_RPI_VCC, OUTPUT);

    printf("\nErase and programming...\n\n");
    digitalWrite(GPIO_RPI_SWDIO_SWCLK, HIGH);
    digitalWrite(GPIO_RPI_VCC_GND, HIGH);
    usleep(100000); // 100000 us => 100 ms

    setFileProgramContatto3V(brand);
    fp = popen(flashCommandStr, "r");

    if (fp == NULL) {
        fprintf(stderr, "Error opening pipe\n");
        digitalWrite(GPIO_RPI_SWDIO_SWCLK, LOW);
        digitalWrite(GPIO_RPI_VCC_GND, LOW);
        return ERROR;
    }

    // search if the command is successfully completed
    while (fgets(lineBuffer, MAX_LINE_BUFFER_SIZE, fp) != NULL) {
        printf("%s", lineBuffer);
        if(strcmp(lineBuffer, "Flashing completed successfully!\n") == 0){
            resultProg = 0;
        }
    }

    // close the pipe pipe
    pclose(fp);

    // if the previous string is not found, there is an ERROR
    if(resultProg == 1){
        printf("\nProgrammingError\n");
        digitalWrite(GPIO_RPI_SWDIO_SWCLK, LOW);
        digitalWrite(GPIO_RPI_VCC_GND, LOW);
```

```
        usleep(100000); // 100000 us => 100 ms
        return ERROR;
    }
    else{
        digitalWrite(GPIO_RPI_SWDIO_SWCLK, LOW);
        digitalWrite(GPIO_RPI_VCC_GND, LOW);
        digitalWrite(GPIO_RPI_VCC, HIGH);
        //to ensure correct initialization of the peripheral and the first contact TX
        delay(2000);
        return OK;
    }
}
```

```
void setFileProgramContatto3V(int brand){
    FILE *fp;
    char findFirmwareNameStr[256];

    if(brand){ // brand=1 means SECURGROSS
        strcpy(firmwareName, "~/Desktop/test_868_longrange/08_scripts/
            firmwareForTestMachine_bin/Contatto868MHzLongRange*_MachineTest_SG.bin");
    }
    else{
        strcpy(firmwareName, "~/Desktop/test_868_longrange/08_scripts/
            firmwareForTestMachine_bin/Contatto868MHzLongRange*_MachineTest_Saet.bin");
    }

    sprintf(findFirmwareNameStr, "ls %s", firmwareName);

    fp = popen(findFirmwareNameStr, "r");
    if(fp == NULL) {
        printf("File program not found\n");
        exit(1);
    }

    if(fgets(firmwareName, sizeof(firmwareName)-1, fp) != NULL) {
        size_t len = strlen(firmwareName);
        if (len > 0 && firmwareName[len-1] == '\n') {
            firmwareName[len-1] = '\0';
        }
    }
    // close the pipe pipe
    pclose(fp);
    printf("Firmware name: %s\n\n", firmwareName);
    sprintf(flashCommandStr, "~/Desktop/test_868_longrange/08_scripts/commander-cli/
        commander-cli flash --address 0x08000000 %s -d %s --masserase --speed %d",
        firmwareName, programmingDevice, programmingSpeed);
    }
```

A seguire, si hanno le funzioni per comunicare al firmware gli step funzionali ed elettrici da svolgere. Come si evince dal flow chart (Figura 42), si ha una distinzione nel caso il test sia completo o singolo, e di conseguenza si ha un diverso messaggio da inviare in seriale. Si riporta la porzione di codice responsabile dello svolgimento del test completo. In particolare, tale informazione viene comunicata al firmware, che inizierà la procedura

di test. Il backend attenderà un ACK e successivamente aspetterà gli esiti e le misure. Nel caso ci fosse un problema di comunicazione o un'anomalia sull'interfaccia seriale FTDI232, un meccanismo di timeout è stato implementato per garantire comunque l'avanzamento del flusso logico. Pertanto, la macchina non si blocca in caso di errore, ma procede oltre e segnalerà il problema tramite l'interfaccia. Prima che la funzione si chiuda, si aggiornata la stringa da comunicare al frontend, che viene gestita in formato JSON con tutti i dati relativi al collaudo.

```
int completeTestContatto3V(int fd, int testResultArray[], \
                           int testResultArrayMeasurements[], \
                           char risposta[], int *flagHtmlCommunication, \
                           int *completeTestFailed){
    int i;
    int timeoutExpired = 0;

    printf("Test Command:\n");
    write(fd, cmdTest[NUM_TOT_GENERAL_COMMANDS], LEN_COMPLETE_TEST_CMD);
    printf("TX : ");
    for(i=0;i<LEN_COMPLETE_TEST_CMD;i++)
        printf("%02X ", cmdTest[NUM_TOT_GENERAL_COMMANDS][i]);
    printf("\n");

    timeoutExpired |= ser_ackRead(fd);

    printf("\nRX Outcomes = ");
    timeoutExpired |= ser_responseReadNormal(fd, testResultArray);

    //printing outcomes of steps
    printf( "......" );

    i=0;
    while(i<(NUM_TEST_CONTATTO3V) && !(*completeTestFailed)){
        if(testResultArray[i] == 1){
            (*completeTestFailed) = 1;
        }
        i++;
    }

    printf("\nRX Measurements = ");
    timeoutExpired |= ser_responseReadandReconstruct(fd, testResultArrayMeasurements);

    //printing measurements
    printf( "......." );

    if((*completeTestFailed)){
        printf("\n----------TEST FAILED----------\n\n");
    }
    else{
        printf("\n----------TEST PASSED----------\n\n");
    }

    printf("Test Exit...\n");
    write(fd, cmdTest[TEST_EXIT], LEN_TEST_EXIT); // <sensor_code> enter in test mode
    printf("TX : ");
    for(i=0;i<LEN_TEST_EXIT;i++)
        printf("%02X ",cmdTest[TEST_EXIT][i]);
    printf("\n");
```

```
50
51      (*flagHtmlCommunication) = 0;
52      timeoutExpired |= ser_ackRead(fd);
53      digitalWrite(GPIO_RPI_VCC_GND, LOW);
54
55      char str_tmp[20];
56
57      strcat(risposta, "\"testResultArray\":[\"");
58      for(int i=0;i<NUM_TEST_CONTATTO3V;i++){
59          //printf("%d,",testResultArray[i]);
60          sprintf(str_tmp, "%d", testResultArray[i]);
61          strcat(risposta, str_tmp);
62          if(i<NUM_TEST_CONTATTO3V-1){
63              strcat(risposta, "\", \"");
64          }
65      }
66      strcat(risposta,"\"], ");
67
68      strcat(risposta, "\"testResultArrayMeasurements\":[\"");
69      for(int i=0;i<NUM_MEASUREMENTS_TEST;i++){
70          //printf("%d,",testResultArray[i]);
71          sprintf(str_tmp, "%d", testResultArrayMeasurements[i]);
72          strcat(risposta, str_tmp);
73          if(i<NUM_MEASUREMENTS_TEST-1){
74              strcat(risposta, "\", \"");
75          }
76      }
77      strcat(risposta,"\"], ");
78
79      return timeoutExpired;
80  }
```

Infine, `writeReportContatto3V(...)` è responsabile della generazione dei report. In caso di test completo ed esito positivo, si genera un documento in formato .txt che riporta data e ora del test, brand, lotto di produzione, ID della periferica, effettivo firmware caricato e relativa versione, una lista dei test effettuati e l'elenco delle misure raccolte. Nel caso di test singolo non viene generato nessun report, mentre in caso di esito negativo si aggiorna un file contenente le statistiche di fallimento dei test. Questi documenti vengono memorizzati in opportuni percorsi, attraverso la distinzione tra collaudi positivi e negativi, lotto di produzione e brand del sensore; lo scopo è quello di facilitare la ricerca di una periferica specifica e velocizzare l'accesso e il caricamento dei dati.

Considerati gli elevati volumi di vendita, le dimensioni dei file .txt non sono trascurabili. Siccome il sistema operativo su Raspberry occupa già diversi gigabyte di memoria, è fondamentale che i report contengano solo gli elementi essenziali, evitando dati superflui. In questo modo, si limita l'occupazione di memoria, consentendo lo storage in locale dei test di molti dispositivi.

4.6.3 Ulteriori funzionalità di contorno

In aggiunta a tutto l'ecosistema necessario per svolgere i collaudi, sono presenti altri script per funzionalità utili. Uno degli step fondamentali è la programmazione firmware del DUT. Periodicamente, è possibile che siano rilasciate nuove versioni che risolvono bug o implementano nuove logiche. È importante che le macchine di test carichino le ultime versioni firmware rilasciate, al fine di avere le periferiche sempre allineate.

I file devono essere scaricati dall'archivio (dove avvengono i rilasci) e quelli in locale devono essere aggiornati. Pertanto, ad ogni avvio della macchina, viene eseguito in automatico il seguente script in Bash tramite `Crontab`. Crontab è uno strumento presente nei sistemi Unix/Linux usato per automatizzare l'esecuzione di comandi a intervalli regolari, permettendo di programmare operazioni ricorrenti, come backup, aggiornamenti o esecuzione di script. In questo caso, una volta montato l'archivio (se il cavo di rete è collegato), i file sono copiati nella cartella di destinazione in locale, si esegue un umount e si crea un file di log.

```
1  #!/bin/bash
2
3  #This script is used to update firmware files by copying from the archive
4
5  localArchiveFolder="/home/kevin/ArchivioSaet_ProduzioneLongRange/"
6  destinationFolder="firmwareForTestMachine_bin"
7  SUFFIX="bin"
8
9  echo -e "$(date)"
10 echo -e "\nStart updating firmware files..."
11
12
13 echo -e "\nMounting server path:"
14 echo -e "//192.168.XX.YYY/archive/Produzione/Firmware 868 DSSS/\n"
15 #Mount archive/Produzione/Programmazione e Collaudo 868 Long Range/ in local folder /home
       /kevin/ArchivioSaet_ProduzioneLongRange/
16 sudo mount -t cifs //192.168.XX.YYY/archive/Produzione/Firmware\ 868\ DSSS -o gid=kevin,
       uid=kevin,username=**********,password=**********,! $localArchiveFolder
17
18 #check if mount is OK
19 if [ $? -ne 0 ]; then
20   echo -e "\nMount Failed!"
21   exit 1
22 fi
23
24 # Delete entire folder to avoid double firmware files
25 if [ -d "$destinationFolder" ]; then
26     rm -rf "$destinationFolder"
27 fi
28
29 if [ ! -d "$destinationFolder" ]; then
30   mkdir "$destinationFolder"
31 fi
32
33 #Copy files from archive to local folder
34 for file in $(ls $localArchiveFolder); do
35   if [ -f "$localArchiveFolder/$file" ]; then
```

```
    filename=$(basename "$file")
    if [[ $filename == *.$SUFFIX ]]; then
      cp "$localArchiveFolder/$file" $destinationFolder
      echo -e "Coping of:\t$file"
    fi
  fi
done

echo -e "\nUmounting server path..."

MAX_ATTEMPTS=100
attempt=0
success=0

while [ $attempt -lt $MAX_ATTEMPTS ] && [ $success -ne 1 ]; do
  #Umount /home/kevin/ArchivioSaet_ProduzioneLongRange/
  sudo umount $localArchiveFolder
  if [ $? -eq 0 ]; then
    success=1
    echo "Umount Successful at attempt $attempt!"
  else
    attempt=$(($attempt+1))
    sleep 1
  fi
done

echo -e "\nProcess Finished!"
```

Per quanto riguarda i backup dei report, si lancia il seguente script ad intervalli regolari. Esso ha lo scopo di aggiornare le differenze tra la cartella locale in cui sono generati i report e la corrispondente in archivio. Pertanto, i backup automatici che vengono svolti consentono di poter archiviare i file, evitandone la perdita in caso di guasto alla macchina o al dispositivo di storage locale. Con questo meccanismo, è possibile visualizzare i log da qualsiasi workstation che abbia accesso all'archivio aziendale, migliorando così la consultazione.

```
#!/bin/bash

localArchiveFolder="/home/kevin/ArchivioSaet_ProduzioneLongRange/"

echo -e "$(date)"

echo -e "Starting Backup of reports..."

echo -e "\nMounting server path:"
echo -e "//192.168.XX.YYY/archive/Produzione/Programmazione e Collaudo 868 DSSS/"
#Mount archive/Produzione/Programmazione e Collaudo 868 DSSS/ in local folder /home/kevin
    /ArchivioSaet_ProduzioneLongRange/
sudo mount -t cifs //192.168.XX.YYY/archive/Produzione/Programmazione\ e\ Collaudo\ 868\
    DSSS -o gid=kevin,uid=kevin,username=**********,password=**********,
    $localArchiveFolder

#check if mount is OK
if [ $? -ne 0 ]; then
```

```
16    echo -e "\nMount Failed!"
17    exit 1
18  fi
19
20  echo -e "\nUpdating differences...\n"
21  #Backup only difference
22  rsync -av --update /home/kevin/Desktop/test_868_longrange/01_software/
        collaudoForTestMachineWeb/Report_Contatto3V/ $localArchiveFolder/Contatto3V/
23
24  echo -e "\nUmounting server path..."
25
26  MAX_ATTEMPTS=100
27  attempt=0
28  success=0
29
30  while [ $attempt -lt $MAX_ATTEMPTS ] && [ $success -ne 1 ]; do
31    #Umount /home/kevin/ArchivioSaet_ProduzioneLongRange/
32    sudo umount $localArchiveFolder
33    if [ $? -eq 0 ]; then
34      success=1
35      echo "Umount Successful at attempt $attempt!"
36    else
37      attempt=$(($attempt+1))
38      sleep 1
39    fi
40  done
41
42  echo -e "\nProcess Finished!\n\n"
```

Tramite l'utilizzo di `Crontab`, `Autostart` ed `rc.local` vengono svolte ulteriori funzioni che consistono nel lanciare automaticamente il programma di backend in un terminale e aprire l'interfaccia web direttamente all'avvio del sistema. Inoltre, si gestisce un task che permette di allineare data e ora della raspberry anche in assenza di connessione internet, attraverso l'utilizzo di un superconsendatore che mantiene attivo un RTC ed uno script che manipola opportunamente un registro nel quale si mantengono tali informazioni.

5 Collaudo e validazione

La fase di validazione della macchina rappresenta l'ultima parte fondamentale del processo di sviluppo, in quanto permette di verificare che i risultati forniti siano affidabili e accurati. Il sistema deve garantire la minimizzazione di falsi positivi e negativi.

Verranno descritte le metodologie e le prove sperimentali effettuate, traendo conclusioni sulle prestazioni e sulla capacità di individuazione di scarti e malfunzionamenti reali. Ogni singolo step del processo è stato validato, ponendo particolare attenzione sulle misure e caratterizzazioni elettriche, sulla calibrazione della portante, sulla potenza di trasmissione e sulla verifica dei vari sensori e componenti che costituiscono il DUT.

L'approccio utilizzato è prettamente sperimentale e comprende prove sul campo, confronti tramite l'ausilio di strumenti di misura, manomissioni fisiche dei circuiti e simulazioni di guasti comuni.

5.1 Assorbimenti di corrente

Gli assorbimenti di corrente sono importanti per garantire la durata della batteria. Per la corrente a riposo, come già evidenziato nella Sezione 4, il microcontrollore si risveglia ogni 15,625ms. L'acquisizione dura 500ms e viene effettuata la media integrale per ottenere il valore medio.

La prima verifica è rappresentata dalla ripetibilità delle misure sulla stessa periferica e nelle stesse condizioni. Il valore medio, sui campioni testati, risulta essere sempre lo stesso, con una variazione massima di un solo microampere.

La seconda verifica riguarda l'accuratezza della misura: utilizzando un multimetro, la tensione misurata per un segnale periodico è approssimativamente pari al valore medio. Pertanto, il risultato fornito dalla macchina dovrà essere pari alla lettura del multimetro, o comunque nel suo intorno. È stata anche acquisita la forma d'onda con un oscilloscopio e valutato il valore medio del segnale. Anche in questo caso, i risultati devono essere compatibili.

Prendendo in esame una trentina di periferiche, i valori ottenuti dalla macchina di test (mostrati nel grafico in Figura 43) sono risultati consistenti con le misure effettuate da strumenti di misura tarati e certificati, discostandosi nel peggiore dei casi di 4μA. Considerando che le soglie software sono opportunamente marginate e garantiscono una durata della batteria superiore a 12-18 mesi, la validazione sulla corrente a riposo ha avuto esito positivo.

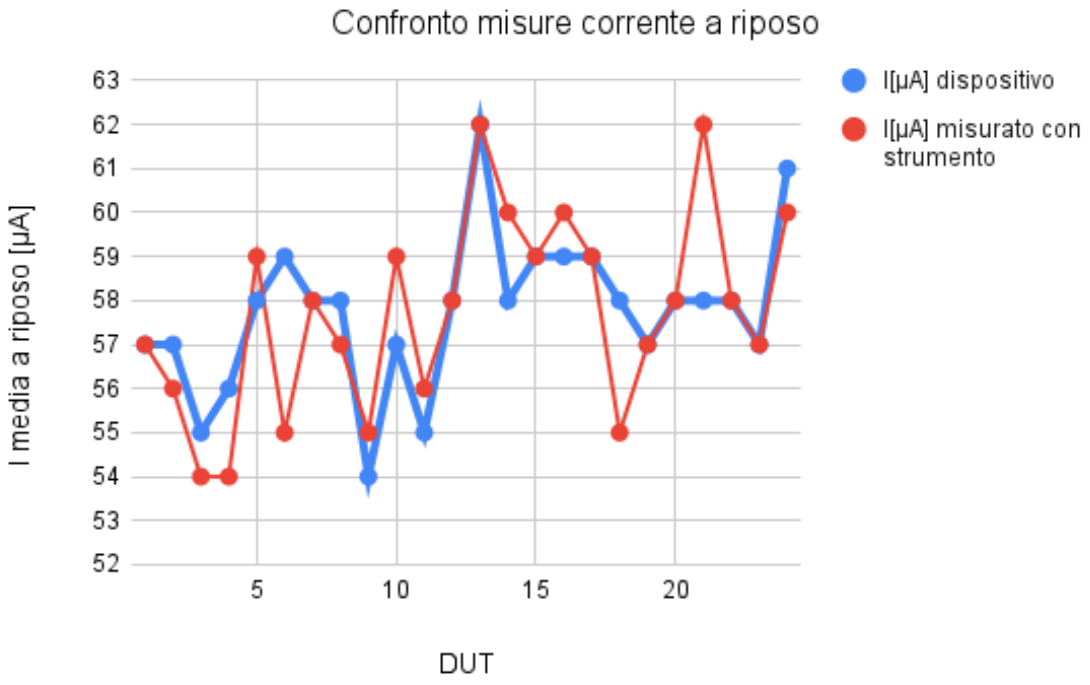

Figura 43: Misure di corrente a riposo e confronto con misure tramite strumento di misura.

Successivamente, alcune periferiche sono state volontariamente sabotate e manomesse, al fine di forzare un consumo anomalo. In questi casi, le correnti misurate risultavano decisamente superiori alle soglie impostate. La macchina ha etichetto correttamente questi dispositivi come scarto. Sono stati collegati anche dei carichi noti tra alimentazione e massa (ad esempio una resistenza) e si è verificato che gli assorbimenti fossero quelli del normale funzionamento più il contributo dovuto al resistore. Anche queste prove hanno fornito esiti positivi.

Per quanto riguarda la corrente durante una trasmissione radio, la verifica è stata svolta mediante un oscilloscopio ed una resistenza di valore pari a quella inserita nel progetto della macchina. Durante questa valutazione, si è notato un fenomeno particolare. Riproducendo il test a banco, i valori di corrente in fase di TX misurati ad oscilloscopio sono risultati quasi doppi rispetto a quanto misurato dalla macchina. Eppure, inserendo la periferica tramite l'apposita slitta all'interno dell'attrezzo di collaudo e svolgendo un test con la sonda dell'oscilloscopio posta sul test point in ingresso al microcontrollore, essa misurava correttamente, ma la tensione era nuovamente dimezzata. Indagando le cause di questa anomalia, si è compreso che la slitta sulla quale è inserita la periferica, andando a sovrapporsi meccanicamente alla porzione di circuito stampato ove è disegnata l'antenna, si accoppiava con essa, determinando una variazione di impedenza vista dal power amplifier, alterando i consumi.

Effettivamente, questa tesi è stata confermata svolgendo nuovamente la misura a banco con l'oscilloscopio, riproducendo le stesse condizioni, ponendo due blocchi dello stesso materiale della slitta sopra e sotto l'antenna. La forma d'onda acquisita ad oscilloscopio è risultata più che dimezzata rispetto al caso senza la presenza dei due blocchi.

Una volta identificato questo fenomeno, sono state variate opportunamente le soglie a livello firmware, al fine di correggere questa problematica. Inoltre, è stata condotta una piccola statistica al fine di identificare una funzione di correzione, per avere una stima reale dei consumi senza l'influenza sull'antenna. È stato identificato un coefficiente moltiplicativo pari a 2,3 rispetto al valore acquisito tramite l'attrezzo di collaudo.

Per quanto riguarda la fase di RX e la successiva di comunicazione radio, i consumi sono risultati compatibili e consistenti.

In definitiva, gli step riguardanti gli assorbimenti di corrente sono stati soddisfacenti, per cui questi due controlli hanno superato la fase di validazione.

5.2 Calibrazione

La validazione della calibrazione serve a garantire una corretta comunicazione radio tra periferiche e ricevitore una volta che verranno installati sul campo. La calibrazione sfrutta funzioni fornite dal costruttore, ma che non sono documentate. Non sono presenti informazioni sugli algoritmi e sulle procedure utilizzate, pertanto non è possibile fornire dettagli circa le incertezze associate all'ottenimento dei valori. L'unica valutazione disponibile è stata effettuata sperimentalmente, utilizzando i risultati restituiti da tali funzioni. Infatti, al termine dello step, viene fornita la differenza in hertz rispetto alla macchina di collaudo. La frequenza della portante è 868,5MHz.

La prima verifica effettuata è rappresentata dalla ripetibilità dei valori forniti se viene eseguito più volte il test di calibrazione sulla stessa periferica. In particolare, sono state prese in esame un determinato numero di periferiche e sono stati raccolti i risultati dopo aver svolto 10 test completi. Nel grafico in Figura 44, per motivi di leggibilità, si riportano i valori di calibrazione per due sole periferiche.

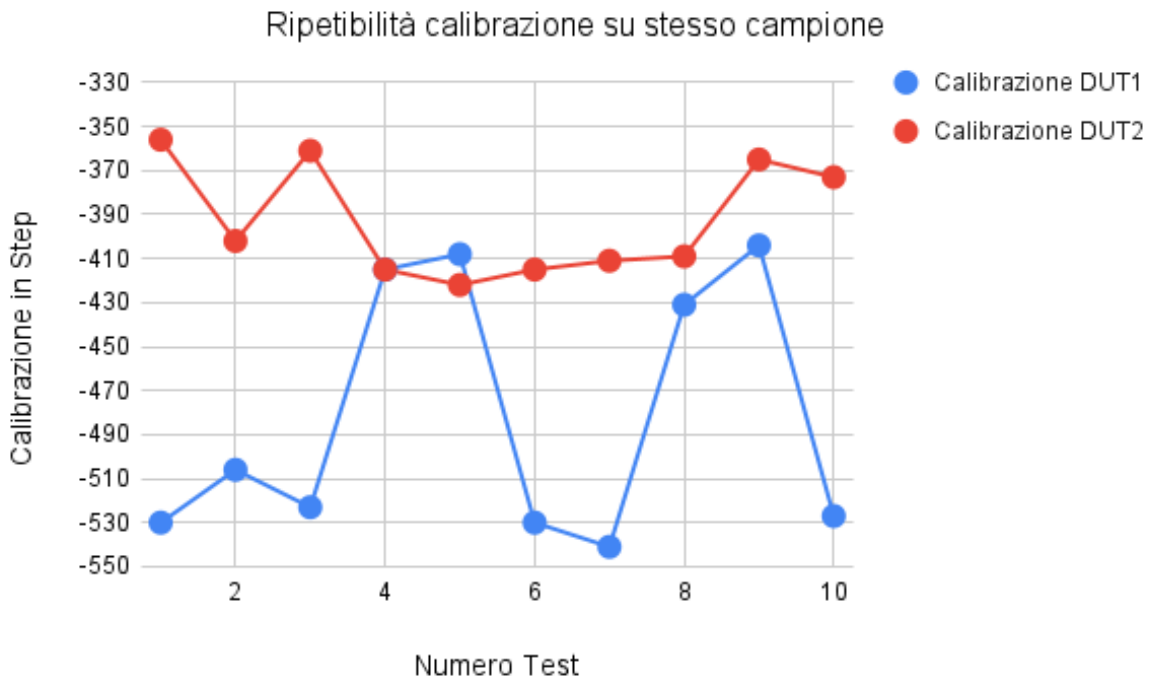

Figura 44: Ripetibilità calibrazione su stesso DUT.

Il valore riportato sulle ordinate è in step, ognuno dei quali vale 24,8Hz. Dal grafico emergono due osservazioni principali: la prima risulta essere il valore medio, differente nei due DUT; questo è un risultato atteso e perfettamente normale. Siccome i due circuiti stampati contengono componenti con tolleranze diverse, la calibrazione serve proprio a

compensare queste variazioni. Infatti, lo scopo di questo step consiste nell'adattare ciascun dispositivo alla frequenza di riferimento, correggendo le differenze introdotte dai componenti hardware. La seconda osservazione riguarda la dispersione dei valori, che sulle prove condotte risulta essere al massimo 120 step, indipendentemente dalla periferica. Pertanto, la calibrazione può fornire risultati che possono discostarsi di massimo 60 step rispetto al valore medio del DUT, che corrispondono a circa 1,5kHz.

Il secondo test condotto riguarda lo scostamento rispetto alla portante teorica, fissata a 868,5MHz. Una cinquantina di campioni calibrati attraverso la macchina di test sono stati messi in portante fissa e il segnale è stato acquisito con un analizzatore di spettro. La differenza rispetto alla portante è stata riportata nel grafico in Figura 45.

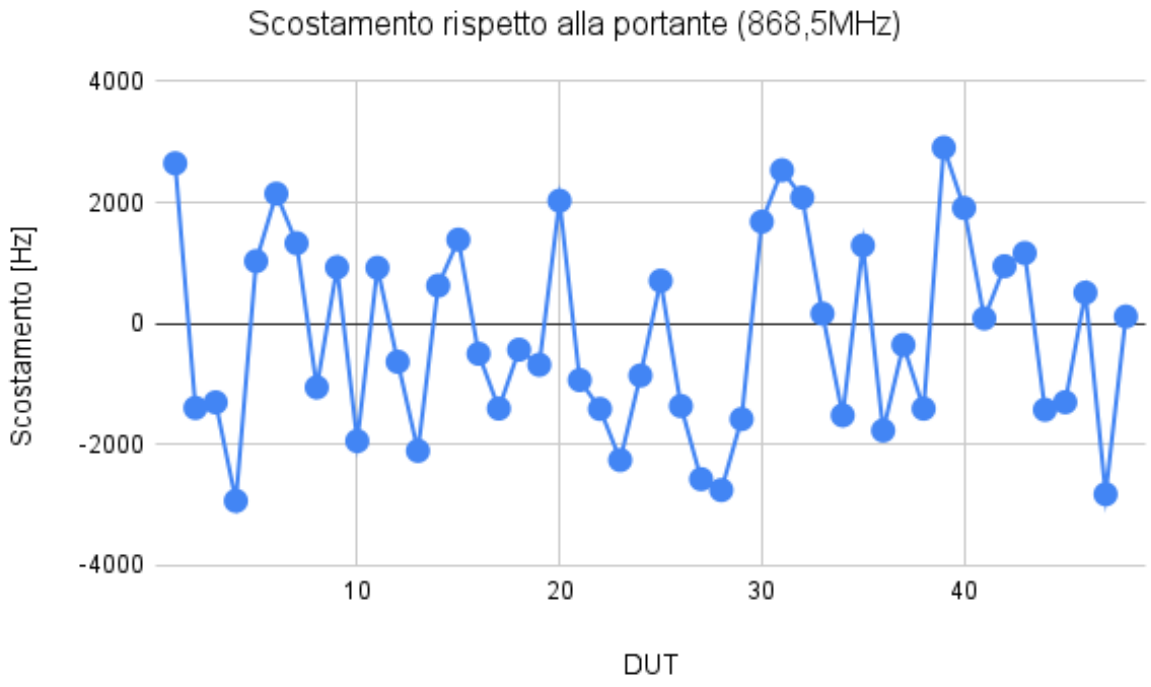

Figura 45: Scostamento rispetto alla portante di 868,5MHz.

Si evince che ogni periferica risulta essere scostata di un massimo di 3kHz rispetto alla portante teorica. Se si considera anche il contributo precedentemente descritto, si raggiunge uno scostamento massimo di 4,5kHz. Sperimentalmente si è visto che la comunicazione tra SC8R e contatto magnetico non ha significativi errori o cali di performance fino ad uno scostamento di 20-25kHz rispetto alla portante, poiché la modulazione DSSS è a banda larga ed è molto robusta in questo senso. Si può concludere che lo step riguardante la calibrazione risulta efficace e accurato.

5.3 Potenza in trasmissione ed RSSI

Oltre alla calibrazione, un altro parametro importante per la caratterizzazione della comunicazione radio è il Receive Signal Strenght Indicator. Tramite questo indicatore, la macchina è in grado di valutare la potenza di trasmissione della periferica da testare e quindi discriminare problematiche sul circuito dell'antenna. Come in precedenza, si è valutata la ripetibilità svolgendo più test sullo stesso campione.

In Figura 46 sono mostrati i risultati condotti su due diversi DUT, svolgendo 10 cicli completi. Si nota che la differenza tra il valore massimo e il valore minimo è di 3dBm, che risulta accettabile. Questo scostamento si può attribuire anche all'ambiente rumoroso (dal punto di vista della radio) in cui sono stati condotti i test e all'incertezza degli algoritmi utilizzati. Siccome le funzioni che gestiscono la stime del valore di RSSI sono nuovamente fornite dal costruttore del microcontrollore e non sono documentate, non è possibile verificare il procedimento utilizzato per queste stime. Pertanto, come per la calibrazione, sono stati osservati i risultati finali.

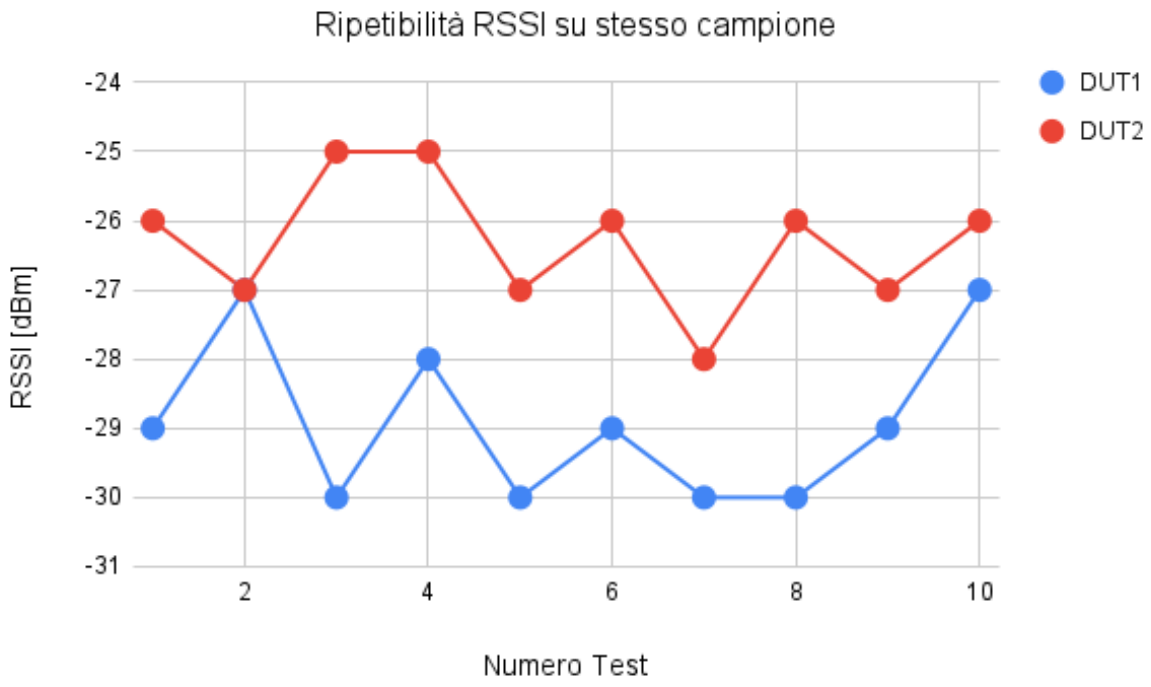

Figura 46: Ripetibilità misura di RSSI su stesso DUT.

A differenza della calibrazione, avere un'indicazione sull'accuratezza del valore di RSSI richiede prove più complesse. Per validare la potenza di trasmissione, è stato condotto un test sperimentale: recandosi in un ambiente senza ostacoli (come edifici, vegetazione, mezzi di trasporto, ecc), si è in una condizione vicina al campo libero (ossia un ambiente ideale

per in cui le onde elettromagnetiche, come quelle radio, possono propagarsi liberamente senza riflessioni o interferenze). SAET garantisce il funzionamento dei dispositivi radio in campo libero per diverse centinaia di metri, pertanto ogni periferica ritenuta funzionante dalla macchina è stata posta ad una distanza maggiore rispetto al minimo garantito, mantenendo il ricevitore in una posizione fissa. Effettuando delle stimolazioni, l'SC8R avrebbe dovuto ricevere il pacchetto radio.

Tutte le periferiche hanno superato con successo questo test, anche quelle con RSSI vicini alla soglia che discrimina il prodotto funzionante rispetto allo scarto. In Figura 47 sono mostrati i valori di RSSI delle periferiche che sono state testate in campo.

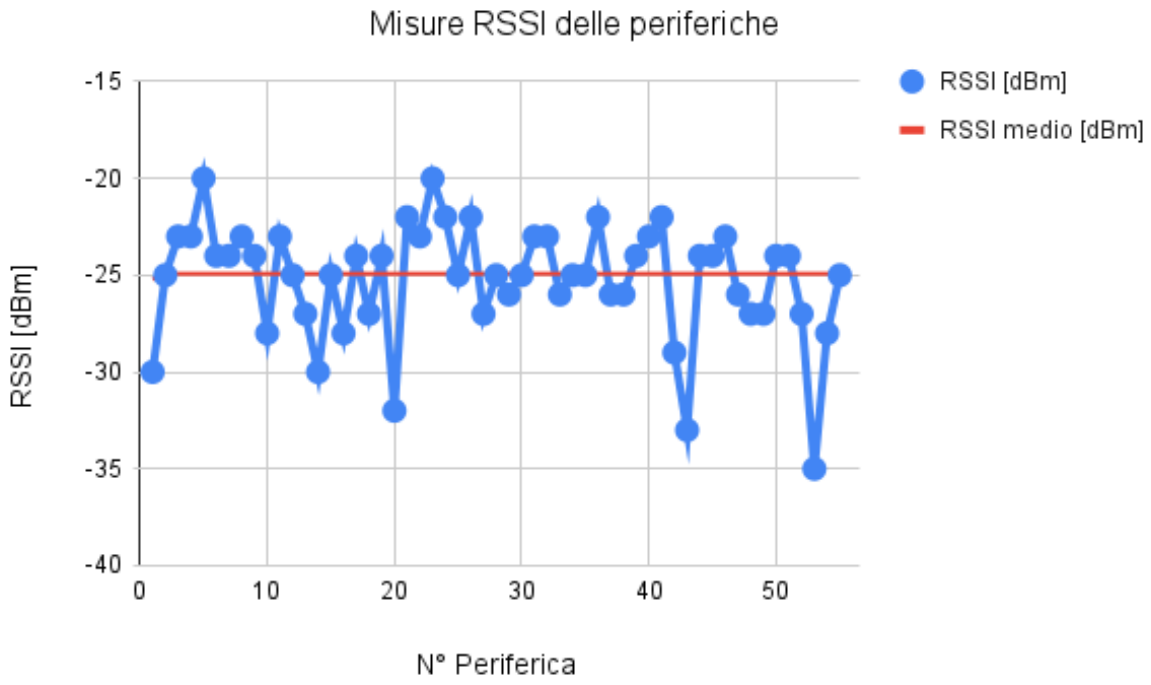

Figura 47: Misure di RSSI su diversi campioni.

Inoltre, si è cercato di verificare empiricamente che dispositivi con potenza di trasmissione più alta avessero effettivamente una portata maggiore. Questa prova può considerarsi positiva, anche se ci sono stati alcuni risultati discrepanti. Infatti, alcune periferiche hanno mostrato la stessa portata anche con un valore di RSSI inferiore di due o tre unità, mentre alcuni contatti magnetici con pari RSSI avevano una portata che si discostava anche di 25-30 metri.

Come ultima prova, sono stati passati in macchina dei dispositivi volutamente sabotati, andando a disconnettere o bypassare componenti lungo l'antenna, oppure sostituendoli con valori differenti rispetto al progetto. In tutti questi casi, la potenza radio risultava inferiore alla soglia impostata. I valori vengono riportati nel grafico in Figura 48.

In campo, i campioni con RSSI più alti riuscivano a garantire la portata dichiarata, ma con un margine ridotto. Pertanto, nonostante si possa abbassare la soglia inferiore, si è preferito avere una maggiore selettività per garantire con certezza la portata, scartando eventualmente campioni con RSSI prossimi alla soglia.

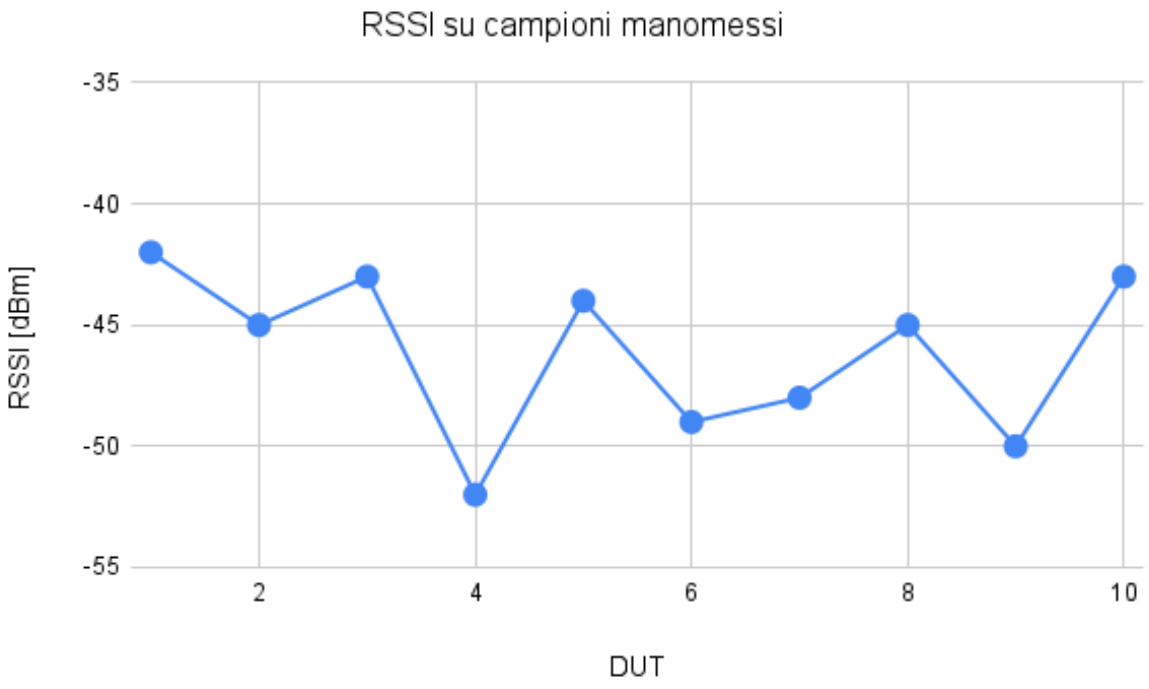

Figura 48: Misure di RSSI su campioni manomessi fisicamente.

5.4 Tamper, aux, reed bypass e LED di segnalazione

La procedura di validazione di tamper, aux e reed bypass è risultata pressoché la stessa. Un insieme di campioni è stato testato dalla macchina e successivamente i dispositivi sono stati alimentati mediante l'inserimento dell'apposita batteria. Il controllo effettuato consisteva nel premere il tamper, verificare che il LED di segnalazione effettuasse un lampeggio indicante la trasmissione radio del cambiamento di stato e assicurarsi tramite un ricevitore collegato al sensore la ricezione del pacchetto. Inoltre, rilasciando il pulsante, un'ulteriore trasmissione doveva essere effettuata. In modo analogo sono stati verificati l'ingresso AUX e il reed bypass, mentre il corretto funzionamento del LED, come illustrato nell'esempio del tamper, è stato confermato attraverso un riscontro visivo.

Siccome alcuni di questi componenti sono gli unici presenti lungo il percorso dal microcontrollore all'alimentazione (al massimo in serie è presente un resistore), non sono state effettuate manomissioni delle schede. Tuttavia, durante i test con la macchina di collaudo, sono stati svolti cicli con tamper, ingresso ausiliario e reed bypass chiusi/cortocircuitati e LED oscurato (che emula un errore nell'orientamento del componente durante il montaggio).

Tutte le verifiche effettuate, sia per gli esiti positivi, sia per i test volutamente alterati, hanno confermato i risultati forniti dalla macchina con un tasso del 100%.

5.5 Reed e accelerometro

Il contatto magnetico e l'accelerometro rappresentano due dei sensori più stimolati durante la vita della periferica, ma anche i più critici e delicati.

Il sensore reed è costituito da due sottili lamelle metalliche racchiuse in un tubo di vetro riempito di gas, pertanto urti o vibrazioni possono facilmente rompere o danneggiare il componente. Siccome l'incidenza di questi problemi non è bassissima, durante i test non è stato difficile trovare scarti relativi al reed. In questi casi, stimolando a mano la periferica mediante un magnete, la segnalazione del LED e la trasmissione radio non è avvenuta, confermando l'esito dell'attrezzo di collaudo.

Anche il test dell'accelerometro è stato validato con successo. Testando a banco le periferiche, in caso di spostamento, inclinazione o urto sul contenitore plastico, esse dovevano trasmettere il relativo cambio di stato e confermare i risultati ottenuti dalla macchina. Come verifica finale, su alcuni campioni si è rimosso completamente l'accelerometro o si sono interrotte alcune piste importanti, come i percorsi di alimentazione o quelli che comunicano i parametri in seriale con il microcontrollore. In tutti questi casi, il sistema ha correttamente etichettato le periferiche come guaste.

6 Future implementazioni

Oltre al contatto magnetico già testato con la macchina attuale, l'attenzione si è estesa allo sviluppo di macchine di test dedicate ad altre tipologie di periferiche di sicurezza. Ogni dispositivo presenta caratteristiche e requisiti specifici, che hanno richiesto la progettazione di sistemi personalizzati per garantire test affidabili ed efficienti. In questo capitolo verranno illustrate brevemente le nuove soluzioni sviluppate per adattarsi a questi dispositivi.

6.1 Periferiche da testare

In aggiunta al contatto magnetico, si riporta un elenco con le altre periferiche da testare, insieme ad una rapida descrizione delle loro funzionalità. Tutte le periferiche devono superare i controlli sugli assorbimenti a riposo, sulle correnti durante una trasmissione radio, devono essere calibrati in frequenza e l'antenna deve fornire una stima di RSSI adeguata. Oltre a questi step, comuni a tutte i dispositivi, vengono indicati ulteriori test specifici per la periferica trattata.

- `SC8R`: Si tratta del dispositivo descritto nella Sezione 3.2. È il ricevitore al quale tutte le periferiche si interfacciano e funge da ponte tra i sensori e la centralina dell'impianto.

  I test specifici da svolgere riguardano l'accensione/spegnimento di tutti i LED di segnalazione presenti sulla scheda, il funzionamento del tamper e la comunicazione attraverso il current loop, che si tratta del protocollo di comunicazione proprietario di SAET con il quale l'SC8R scambia messaggi con la centrale.

- `Telecomando`: Consente di attivare e disattivare l'impianto di allarme o solo alcune zone di esso, individuare lo stato attuale del sistema, comandare un attuatore specifico o segnalare un'emergenza.

  L'esito dell'intero test dipende dal corretto funzionamento di tutti i pulsanti e dei LED associati ad essi.

- `Sirena da Esterno`: Il principale dispositivo di dissuasione acustica, attivato quando l'impianto rileva un'intrusione non autorizzata. Viene installata all'esterno e grazie alla tromba di cui è dotata segnala al vicinato la possibile presenza di un malintenzionato.

  Si devono svolgere controlli sul pulsante antischiuma e sul tamper della manomissione, sul funzionamento della tromba durante una suonata e sul flash della sirena.

- `Sirena da Interno`: Simile alla sorella da esterno, viene installata all'interno dell'edificio.

  Siccome non dispone di sensore antischiuma e flash di segnalazione, tra i controlli specifici si ha la verifica della porzione di circuito responsabile della segnalazione acustica.

- `Platino`: È un'interfaccia radio che si collega ad un sensore volumetrico da interno ad infrarossi, rendendolo wireless.

  Il sensore vero è proprio non è prodotto da SAET, pertanto la macchina di test dovrà emulare il suo funzionamento, testando l'interfaccia platino attraverso dei GPIO in ingresso e uscita, leggendo i livelli logici a fronte di differenti stimoli.

- `Kapture`: È un'interfaccia radio che si collega ad un "sensore a barriera" a doppia tecnologia (PIR e microonda) da esterno. Quando la barriera viene attraversata o interrotta, il sensore rileva l'intrusione e comunica la situaizone di allarme. L'interfaccia da testare consente di rendere wireless questo dispositivo. Si deve verificare il corretto funzionamento della comunicazione con il sensore, che in questo caso avviene tramite seriale.

- `Optex`: È un'interfaccia per una famiglia di sensori principalmente volumetrici che vengono installati all'esterno. Analogamente al Kapture, questi sensori sono dotati di doppia tecnologia. Anche in questo caso, l'interfaccia rende wireless i dispositivi appartenenti a questa famiglia.

  I test riguardano la prova della linea seriale, come nel precedente caso.

6.2 Sviluppo hardware e firmware

Per la realizzazione delle successive macchine di test, al fine migliorare la riusabilità e sfruttare la struttura modulare del progetto, è stata utilizzata nuovamente la scheda CS3981 come punto di partenza. Il flusso progettuale adottato è stato lo stesso della prima macchina, così come l'architettura complessiva che risulta molto simile. La prima differenza riguarda la circuiteria su millefori, che è stata modificata (ove necessario) per consentire lo svolgimento dei test specifici sulle nuove periferiche. Inoltre, per ottimizzare i costi, è stato riutilizzato un meccanismo di armatura proveniente da vecchie postazioni di collaudo. In questo modo, le nuove macchine sono state rese componibili (l'argomento verrà approfondito nella Sezione 6.3).

Per garantire un'interfaccia comune tra tutte le macchine, è stato progettato un sistema di connessione che sfrutta esclusivamente due cavi D-sub a 25 pin, semplificando l'interconnessione e riducendo la complessità del cablaggio.

Il firmware è stato integrato con gli algoritmi necessari per svolgere i diversi test specifici. Sono presenti file .c e .h per ogni dispositivo da testare, ma il corpo di molte funzioni è stato riutilizzato, eventualmente con cambiamenti riguardanti le soglie di correnti, calibrazioni ed RSSI.

6.3 Struttura meccanica componibile

Come accennato in precedenza, per la realizzazione della struttura meccanica sono stati riutilizzati i meccanismi di armatura di postazioni non più operative. Ovviamente, per ogni dispositivo è stato creato l'apposito letto d'aghi e il coperchio superiore dotato di pressori.

Ogni macchina è composta da quattro parti, ciascuna identificata da un'etichetta che indica il tipo di dispositivo a cui è destinata, al fine di facilitare il riconoscimento e la localizzazione. Negli esempi riportati di seguito si fa riferimento alla periferica SC8R.
Le quattro parti sono:

- `Il meccanismo di armatura`, comune a tutte le macchina di collaudo. Agendo sulla leva, a differenza della macchina dei contatti magnetici, è la parte superiore a scendere e a esercitare pressione sul circuito per permettere agli aghi di stabilire un contatto elettrico. Sono inoltre presenti dei meccanismi di sicurezza, come l'impossibilità di aprire il coperchio senza il disarmo della macchina grazie ad un blocco meccanico.

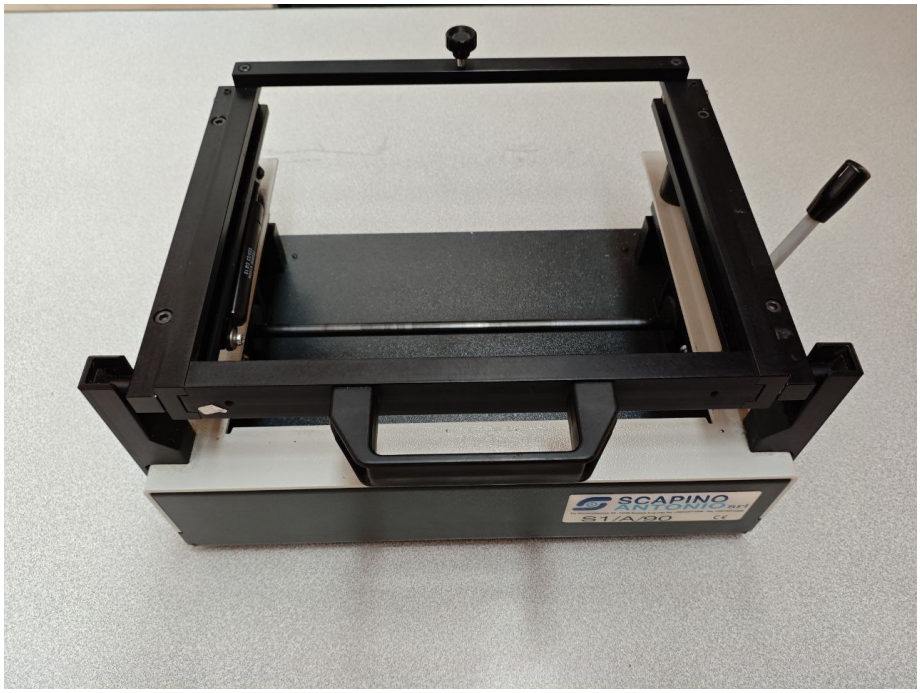

Figura 49: Meccanismo di armatura delle macchine componibili.

- `La meccanica inferiore` comprende il letto di aghi su cui si appoggia la periferica. Grazie ai testimoni che si inseriscono nei fori del circuito da testare, si garantisce il corretto inserimento del DUT.

Figura 50: Meccanica inferiore con letto d'aghi.

- `La meccanica superiore` è costituita dal pannello trasparente con i pressori. Durante l'armatura della macchina, il pannello viene spostato verticalmente verso il basso. I pressori in gomma esercitano una pressione uniforme sulla periferica, garantendo il contatto elettrico con il letto d'aghi. Nel caso dell'SC8R, è presente un vano con all'interno dei fototransistor per la verifica dei LED.

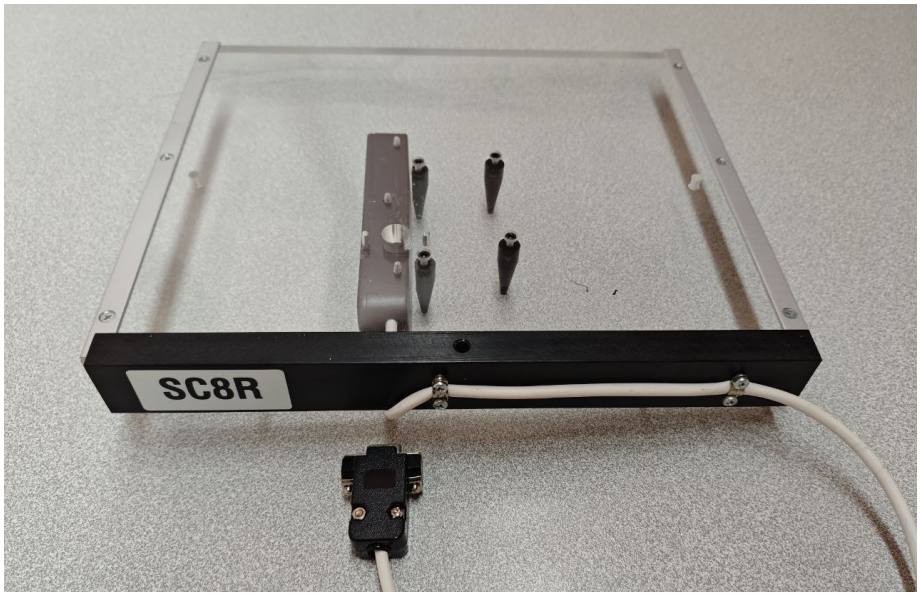

Figura 51: Meccanica superiore con pressori.

- `L'elettronica` è situata all'interno del contenitore plastico. Le due staffe metalliche consentono al meccanismo di armatura di appoggiarsi nella stessa identica posizione, per calibrare e svolgere i test radio sempre nelle medesime condizioni.

Figura 52: Contenitore plastico con elettronica.

Tutte e quattro le parti vengono scelte e montate in base alla periferica da collaudare. In Figura 53 si mostra il momento in cui vengono inseriti i blocchi tramite le apposite slitte. Mediante scorrimento fino a battuta, fissandoli con delle viti, la macchina risulta assemblata e montata. Una volta collegati i cavi, sarà immediatamente operativa.

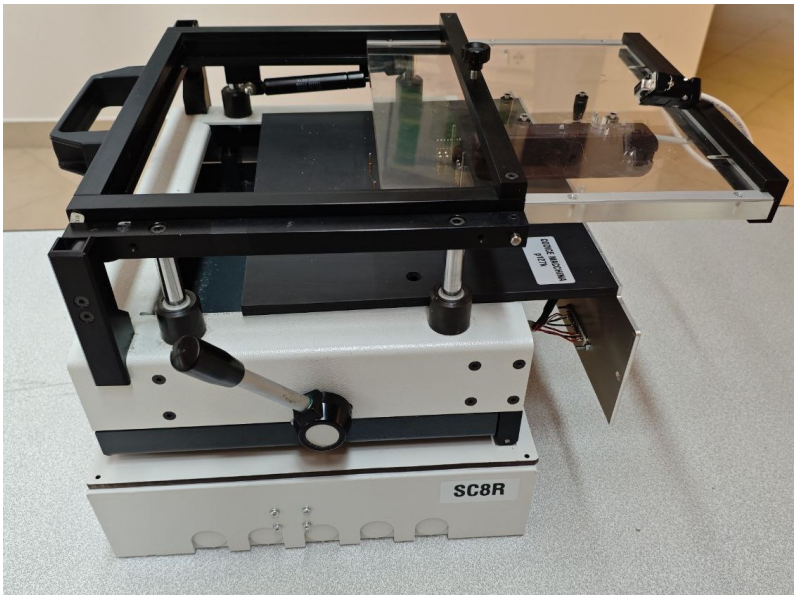

Figura 53: Inserimento delle parti durante il montaggio della macchina.

6.4 Gestione interfacciamento e prevenzione errori di montaggio

Le macchine per i dispositivi sopra elencati sono progettate per essere componibili, pertanto è stata prevista un'interfaccia comune in grado di connettere l'elettronica con il letto d'aghi. Tuttavia, questa configurazione flessibile può comportare potenziali problemi dovuti ad errori di montaggio, come l'inserimento della meccanica di una determinata periferica abbinata erroneamente a un contenitore plastico contenente l'elettronica di un altro dispositivo.

Per prevenire tali errori, come prima misura, si è scelto di utilizzare un connettore D-sub a 25 pin. Per garantire l'uniformità, alcuni cavi sono stati dedicati esclusivamente all'alimentazione, mentre gli altri sono stati raggruppati, per quanto possibile, in base alla tipologia di segnale e alla funzione. In Figura 54 è mostrata la scelta adottata per distinguere i segnali delle diverse periferiche; si riporta solo il connettore 2, poiché il primo è destinato esclusivamente alla programmazione dei DUT ed è quindi analogo per tutti. Le caselle bianche indicano che la connessione non è effettuata e per quella macchina risulta disponibile (ad esempio per future funzionalità). Il simbolo "X" segnala la presenza del collegamento e l'utilizzo di quel cavo; infine, l'etichetta presente specialmente per le connessioni di tipo *Other_* identifica la funzione di quel cavo per quella periferica.

	Pin Type	Pin	Pin Name	Sirena Ext	Sirena Int	SC8R	Telecomando	Platino	Kapture	Optex
CONNECTOR 2	Power Supply	1	3,3V_1	X	X			X	X	X
		2	3,3V_2	X	X			X	X	X
		3	12V_1	X	X	X				
		4	12V_2	X	X	X				
		5	12V_3	X	X	X				
		6	12V_4	X	X	X				
		7	GND_1	X	X	X		X	X	X
		8	GND_2	X	X	X		X	X	X
		9	GND_3	X	X	X		X	X	X
		10	GND_4	X	X	X		X	X	X
	Phototransistor	11	Photo VCC			X				
		12	Photo GND			X				
	Other	13	Current Loop			X				
		14	Tamper1	X	X	X		Tamper		
		15	Tamper2	X	X	X				
		16	Other_1	Foam1				Alarm	TX_serial	TX_serial
		17	Other_2	Foam2				Walktest	RX_serial	RX_serial
		18	Other_3	Siren1_1	BuzSens_1			Inhibit		
		19	Other_4	Siren1_2	BuzSens_2			Sens		
		20	Other_5	Siren1_3				LedON		
		21	Other_6	Siren1_4				Mask		
		22	Other_7	Siren2_1				MaskEN		
		23	Other_8	Siren2_2						
		24	Other_9	Siren2_3						
		25	Other_10	Siren2_4						

Figura 54: Tabella di interfacciamento dei segnali.

Questa misura primaria limita i potenziali danni causati da un errore di assemblaggio, ma non li elimina completamente. Per questo motivo è stata implementata una seconda misura di prevenzione. Quando la macchina viene accesa, l'interfaccia utente si avvia mostrando un popup iniziale che richiede l'inserimento di un codice identificativo. La Raspberry Pi, responsabile della gestione dell'interfaccia, è stata preconfigurata con impostazioni di default specifiche per la periferica da testare. Una volta caricata la pagina, la Raspberry si aspetta il codice relativo alla propria configurazione.

Sulla meccanica inferiore, nell'angolo in alto a destra, è presente un codice identificativo associato al letto d'aghi. L'operatore legge questo codice e lo inserisce nell'apposito campo. Se il codice corrisponde a quello atteso dalla macchina, è possibile proseguire ed effettuare i test; in caso contrario, l'operazione viene bloccata, impedendo ulteriori azioni e invitando l'operatore a ricontrollare i collegamenti effettuati. I codici da inserire per ciascun dispositivo sono visibili nella Sezione 4.6.1, quando si esamina il file *config.js*.

Alcune di queste macchine sono già state realizzate e sono pienamente operative, mentre per le ultime periferiche in elenco sono ancora in corso le fasi di sviluppo, che consentiranno il completamento e la messa in funzione entro poche settimane.

Riferimenti bibliografici

[Les10] Scot Lester. *Packaging Limits Range of Linear Regulators*. 2010. URL: https://www.ti.com/lit/an/sbva027/sbva027.pdf?ts=1726318169193&ref_url=https%253A%252F%252Fwww.google.com%252F.

[STM18] STMicroelectronics. *ULN2801A, ULN2802A, ULN2803A, ULN2804A*. 2018. URL: https://www.st.com/resource/en/datasheet/uln2801a.pdf.

[Alf19] Alfasicurezza. *I Sistemi di Sicurezza nel corso del tempo: una breve storia della moderna Sicurezza nelle Abitazioni*. 2019. URL: https://www.alfasicurezza.com/i-sistemi-di-sicurezza-nel-corso-del-tempo-una-breve-storia-della-moderna-sicurezza-nelle-abitazioni/.

[Wik22] Wikipedia. *Reed (dispositivo)*. 2022. URL: https://it.wikipedia.org/wiki/Reed_(dispositivo).

[Lab23a] Silicon Lab. *EFR32FG23 Wireless SoC Family Data Sheet*. 2023. URL: https://www.silabs.com/documents/public/data-sheets/efr32fg23-datasheet.pdf.

[Lab23b] Silicon Lab. *UG548: Simplicity Link Debugger User's Guide*. 2023. URL: https://www.silabs.com/documents/public/user-guides/ug548-brd1015a-user-guide.pdf.

[Ras23] Raspberrypi. *Raspberry Pi 3 Model B+ product brief*. 2023. URL: https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf.

[Lab24] Silicon Lab. *EFR32xG23 Wireless SoC Reference Manual*. 2024. URL: https://www.silabs.com/documents/public/reference-manuals/efr32xg23-rm.pdf.

[Pro24] Proto-electronics. *Panoramica dei metodi di test e di ispezione dei PCB*. 2024. URL: https://www.proto-electronics.com/it/blog/panoramica-dei-metodi-di-test-e-di-ispezione-dei-pcb.

[Rel24] Taimbo Relay. *scheda-tecnica-1366519-tianbo-electronics-hjr-3ff-s-z-12vdc-rele-per-pcb-12-vdc-15-a-1-scambio-1-pz*. 2024. URL: https://asset.conrad.com/media10/add/160267/c1/-/en/001366519DS01/scheda-tecnica-1366519-tianbo-electronics-hjr-3ff-s-z-12vdc-rele-per-pcb-12-vdc-15-a-1-scambio-1-pz.pdf.

[Sic24] Sicura24. *Dove e quando nasce l'antifurto che oggi vende Sicura 24?* 2024. URL: https://www.sicura24.it/la-storia-dell-antifurto/#:~:text=Il%20primo%20antifurto%20fu%20brevettato,il%20suono%20di%20una%20campana.

[Umb24] SAET Umbria. *Chi ha inventato l'antifurto? Breve storia di questo utile strumento*. 2024. URL: https://saetumbria.com/chi-ha-inventato-lantifurto-breve-storia-di-questo-utile-strumento/.

[Ins24] Texas Instruments. *LP295x Adjustable Micropower Voltage Regulators With Shutdown*. 2006, REVISED 2024. URL: https://www.ti.com/lit/ds/symlink/lp2950.pdf?ts=1726324498159&ref_url=https%253A%252F%252Fwww.mouser.ch%252F.