ALMA MATER STUDORIUM -
UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA

Dipartimento di Ingegneria dell'Energia Elettrica e
dell'Informazione "Guglielmo Marconi"

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA
PER L'ENERGIA E L'INFORMAZIONE

SVILUPPO DI UN SISTEMA EMBEDDED PER APPLICAZIONI DI TELEMETRIA E ACQUISIZIONE DATI WIRELESS

Relatore:
Romani Aldo

Presentata da:
Guidi Andrea

Anno Accademico 2022/2023

Sommario

Introduzione

In un mondo in cui la connettività e la raccolta di dati sono diventate fondamentali per il successo di molte industrie, lo sviluppo di sistemi in grado di monitorare e registrare informazioni da remoto in maniera wireless, riveste un ruolo sempre più cruciale.

La tecnologia embedded risulta una soluzione ideale per implementare sistemi semplici da gestire e in grado di fornire accesso da remoto tramite smartphone o pc, usando la tecnologia Wi-Fi o Bluetooth.

Questi sistemi avanzati consentono di monitorare e raccogliere informazioni critiche da dispositivi remoti, come sensori, macchinari industriali e veicoli autonomi, facilitando una vasta gamma di applicazioni nelle industrie automobilistica, manifatturiera, energetica, sanitaria e molte altre.

Il cuore di un sistema embedded di questo tipo è rappresentato da un microprocessore o un microcontrollore, che agisce come cervello intelligente del sistema. Questo dispositivo integrato è responsabile di elaborare i segnali provenienti dai sensori e di trasmettere i dati attraverso reti wireless.

Lo sviluppo di questi sistemi è però contraddistinto da varie sfide e limitazioni. La gestione dell'energia è un aspetto fondamentale, poiché spesso questi sistemi operano su batteria o fonti di alimentazione limitate, in quanto essendo collocati in punti non sempre gestibili manualmente, necessitano comunque di un'autonomia energetica particolarmente lunga. Allo stesso tempo, è essenziale ottimizzare l'utilizzo delle risorse, come la memoria di archiviazione, che risulta estremamente limitata in molti sistemi di questo tipo.

L'elaborato si basa sull'utilizzo di una scheda di sviluppo a microcontrollore e all'implementazione di varie soluzioni per rendere il sistema efficiente per applicazioni di telemetria con acquisizioni dati tramite tecnologia Wi-Fi.

Dopo aver descritto la struttura della scheda di sviluppo e alcune caratteristiche che mette a disposizione per lo sviluppo di un sistema di questo tipo, verranno introdotte più nel dettaglio varie soluzioni software implementate. Infine, verranno riportati alcuni test effettuati in laboratorio riguardanti la qualità del convertitore analogico digitale (la periferica alla base del sistema di acquisizione dati) e le opzioni di risparmio energetico, fornendo una misurazione quantitativa di quelli che sono i consumi della scheda rispetto a una modalità di normale funzionamento.

1 Descrizione del sistema

1.1 Presentazione scheda di sviluppo

La ESP32-C3-DevKitM-1 (*figura 1*) è una scheda di sviluppo a microcontrollore su cui è installato un modulo ESP32-C3, sviluppato dalla Espressif Systems e basato su un processore RISC-V a 32 bit e 160Mhz, ottimizzato per applicazioni a basso consumo energetico.

Una delle caratteristiche del modulo è la sua capacità di connessione wireless, tramite Wi-Fi 802.11 b/g/n, consentendo alla scheda di essere impostata facilmente sia come station, sia come access point.

Dispone, inoltre, di varie porte GPIO (general purpose input-output), che consentono il collegamento di attuatori, sensori e altre periferiche esterne.

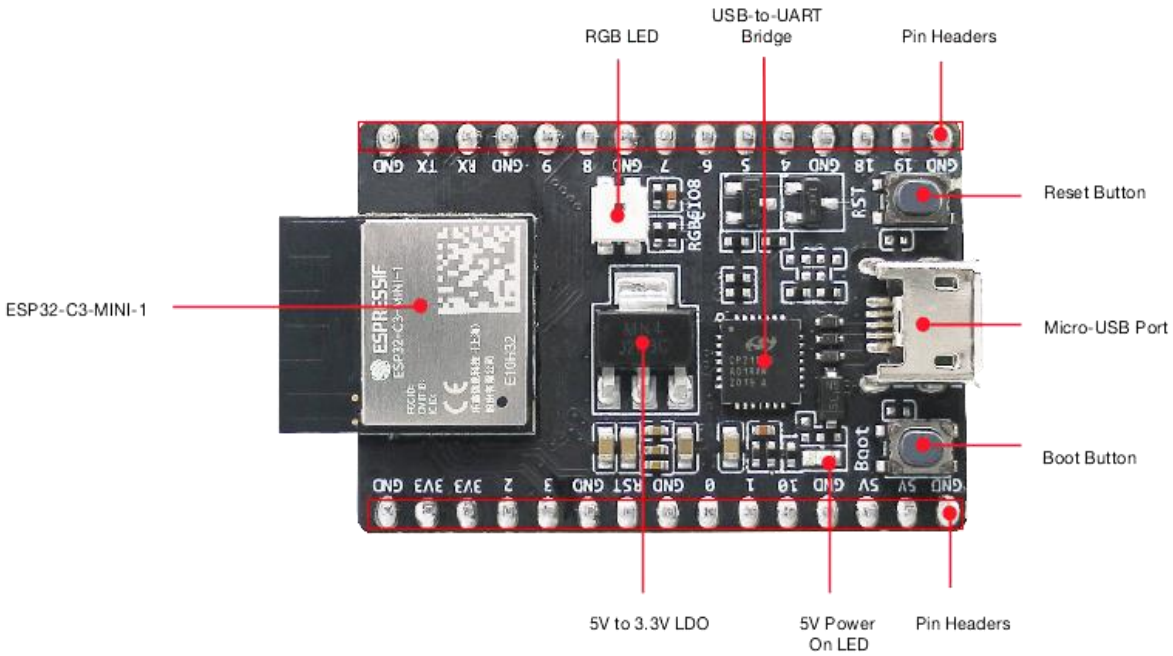

Figura 1 : scheda ESP-C3-DevKit-M1(fonte: documentazione Espressif, [ESP32-C3-DevKitM-1 - ESP32-C3 - — ESP-IDF Programming Guide latest documentation (espressif.com)]).

La scheda può essere alimentata con semplicità tramite una tensione di ingresso fino a circa 12 volt, grazie a un regolatore LDO integrato. Dispone di una porta UART che permette comunicazioni seriali con bit rate fino a 3Mbs e una porta USB tipo B tramite convertitore USB-UART. Dispone poi di due diodi LED, tra cui un LED di controllo, la cui accensione indica la corretta alimentazione della scheda, e un LED RGB, che non è stato utilizzato nel progetto.

La scheda rende inoltre disponibili due pulsanti, uno di reset, e uno per il caricamento del bootloader.

Un altro punto di forza della scheda è la sua semplicità di sviluppo, essendo supportata sia dall'ambiente di sviluppo integrato (IDE) dell'Espressif, ESP-IDF, sia dall'IDE di Arduino (utilizzato nel progetto).

La scheda è progettata per essere una soluzione all'avanguardia per le applicazioni IOT, garantendo ottima connettività wireless, flessibilità di sviluppo e ottimizzazione per basso consumo energetico.

1.2 Convertitore analogico digitale (ADC)

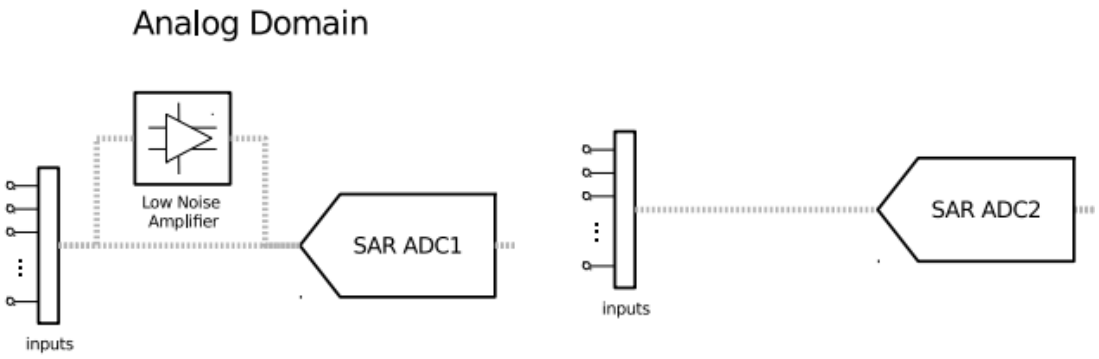

Figura 2 : schema dei due ADC SAR. Gli ingressi passano per due multiplexer (fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

L’ESP32-C3 presenta al suo interno 2 convertitori analogici digitali con risoluzione fino a 12 bit di tipo SAR (*Figura 2*). Questo li rende adatti per applicazioni a consumo energetico ridotto pur mantenendo una risoluzione più che sufficiente per la maggior parte delle applicazioni.

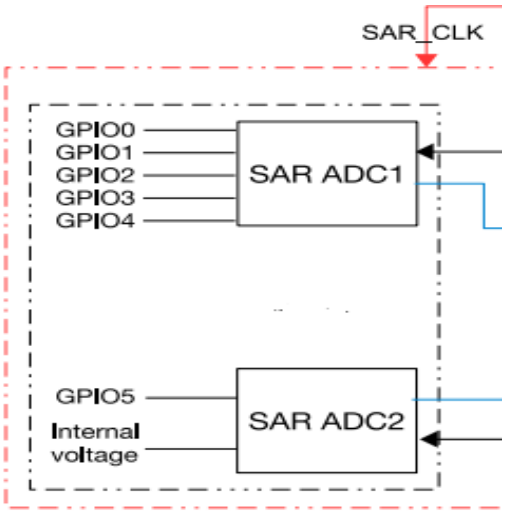

Figura 3: Cinque pin GPIO per l'ADC1 e un GPIO, più un riferimento in volt per l'ADC2(fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

Il primo dei due ADC dispone di più canali, il che permette di gestire, con ritardi relativamente brevi, campioni analogici provenienti da differenti sensori al tempo stesso. Come viene però specificato nella documentazione ufficiale, l’ADC1 presenta una maggior robustezza a problematiche di rumore, per cui si è preferito utilizzare quest’ultimo.

Come mostrato in figura (*Figura 3),* l’ADC1 presenta cinque canali di cui però, ne sono stati resi disponibili sulla scheda solo quattro. Per quanto riguarda l’ADC2, invece, viene reso disponibile solo un canale, oltre che una tensione interna al microcontrollore per effettuare dei test di conversione. Inoltre, oltre alle problematiche di rumore riportate nella documentazione della scheda sull’ADC2, viene inoltre sconsigliato l’utilizzo di questo ADC, in quanto può in parte compromettere il pieno utilizzo di alcune modalità Wi-Fi e Bluetooth, essendo utilizzato alla base del funzionamento di questi ultimi.

La tensione di riferimento interna per entrambi i convertitori è di 1.1 V, è però possibile reimpostare questa tensione di riferimento attenuando direttamente i valori analogici in ingresso. Nella tabella (*Figura 4*), sono mostrati vari range ottenibili dal processo di attenuazione.

Attenuation	Measurable input voltage range
ADC_ATTEN_DB_0	0 mV ~ 750 mV
ADC_ATTEN_DB_2_5	0 mV ~ 1050 mV
ADC_ATTEN_DB_6	0 mV ~ 1300 mV
ADC_ATTEN_DB_11	0 mV ~ 2500 mV

Figura 4: Differenti opzioni di attenuazione. Maggiore è l'attenuazione, maggiore sarà il range ammissibile in ingresso (fonte: Documentazione Espressif per il framework di Arduino, ADC — Arduino-ESP32 2.0.6 documentation (espressif.com)).

In particolare, è stata impostata l'attenuazione massima che permette di misurare tensioni fino a 2.5 V, in modo da non dover amplificare il segnale esternamente tramite amplificatore operazionale (in quanto molti sensori lavorano a tensioni superiori a 1.1 V).

Tramite sperimentazioni sull'ADC, si è notato che questo valore di riferimento risulta in parte indicativo e si può discostare di qualche centinaio di millivolt rispetto ai valori riportati.

Essendo questo valore di riferimento variabile e differente per ogni scheda, l'Espressif ha messo a disposizione funzioni per la calibrazione dell'ADC al fine di contenere queste problematiche.

1.3 Modulo WiFi

Il modulo WiFi è un sottosistema dell'ESP32-C3, ed è assistito da un'antenna in microstriscia depositata sulla parte superiore del modulo.

Può essere sia impostato per lavorare sia come station (STA), sia come access point (AP), sia entrambe le due modalità al tempo stesso.

Nel caso venga utilizzato come station, la scheda si comporta come un client, cioè come un qualsiasi dispositivo che si connette a una rete locale, mentre nel caso venga utilizzato come access point, si comporta come un router virtuale, con una sua rete associata a cui possono accedere gli altri dispositivi client.

1.4 Serial peripheral interface file system (SPIFFS)

Lo SPIFFS (serial peripheral interface file system) è un file system presente in alcune serie di schede di sviluppo basate sui moduli ESP32.

Lo SPIFFS può essere visto anche come una memoria flash integrata, separata dalla flash principale del microcontrollore, che può essere utilizzata per memorizzare file come pagine web o insiemi di dati acquisiti, fungendo da memoria non volatile che non va a occupare lo spazio preposto a contenere la memoria programma.

Lo SPIFFS organizza i file in memoria in maniera simile a un file system tradizionale, sfruttando un sistema di directory e sottodirectory, dove la radice della struttura contiene le directory principale e da queste si ramificano ulteriori sottodirectory.

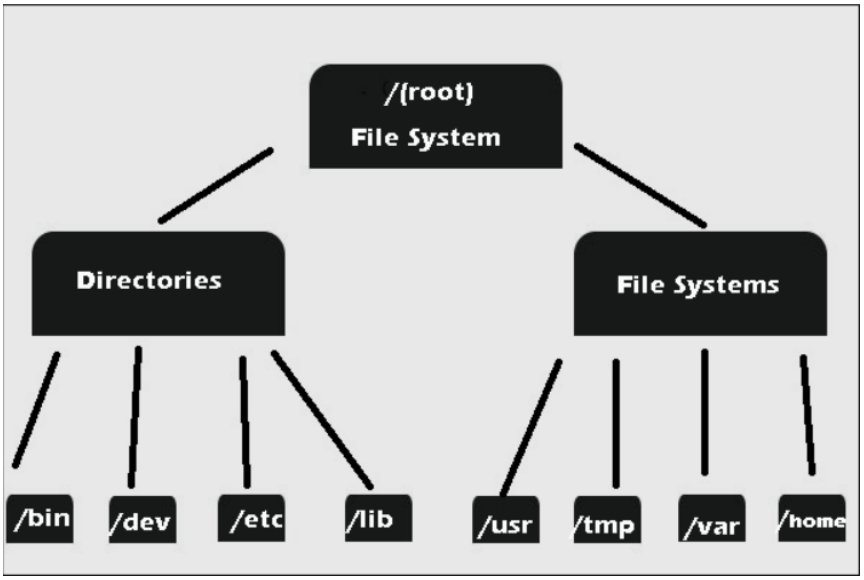

Figura 5: Struttura semplificata di un file system

Ogni file è caratterizzato da un percorso che ne indica la posizione all'interno del file system rispetto a una certa radice (root).

Nel caso dello SPIFFS, l'architettura è simile a quella di un file system tradizionale, ma ottimizzata per un dispositivo con risorse limitate come l'ESP32-C3.

Infatti, mentre un classico file system può essere utilizzato su diversi dispositivi di archiviazione come SSD o HDD, lo SPIFFS è un file system appositamente progettato per le schede dell'ESPRESSIF.

Inoltre, entrambi i sistemi differiscono tra loro per quanto riguarda la gestione dello spazio di memoria, in quanto in un file system generico, l'allocazione in memoria viene effettuata tramite blocchi di dimensione variabile per i vari dati, sfruttando una tabella di allocazione per definire quali blocchi siano occupati e quali siano liberi.

Questo porta a problematiche quali la frammentazione dovuta alla disposizione discontinua dei blocchi, che porta a creare spazi vuoti in memoria non abbastanza capienti a ospitare un nuovo file.

Lo SPIFFS, invece, si basa sul concetto delle pagine come sistema di allocazione, le quali hanno dimensione fissa e sono e sono scritte in maniera sequenziale, inoltre possono essere allocate e liberate individualmente, riducendo la problematica della frammentazione.

In definitiva lo SPIFFS è un sistema molto utile per contenere dati risultanti da misurazioni del sistema embedded e, in generale, per contenere informazioni senza andare a occupare spazio sulla flash principale, motivo per cui è stato utilizzato per contenere i dati acquisiti nel progetto.

1.5 Opzioni risparmio energetico

Una delle caratteristiche dell'ESP32-C3 è quella di disporre di diverse modalità di risparmio energetico (SLEEP MODES), al fine di ridurre il consumo di energia

quando il dispositivo non è attivamente impegnato nell'esecuzione di operazioni.

Table 9-3. Predefined Power Modes

Power Mode	Power Domain								
	PMU	PD Peripherals	Digital System	Wireless Digital Circuits	CPU	FOSC_ CLK	XTAL_ CLK	PLL	RF Circuits
Active	ON	ON	ON	ON	ON	ON	ON	ON	ON
Modem-sleep	ON	ON	ON	ON*	ON	ON	ON	ON	OFF
Light-sleep	ON	ON	ON	OFF*	OFF*	OFF*	OFF	OFF	OFF
Deep-sleep	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Figura 6: Descrizioni delle differenti modalità di sleep mode(fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

Lo sleep mode si basa sul concetto dei "power domains" (*figura 6*), differenti porzioni del chip che possono essere alimentate o sottoalimentate in maniera discreta da parte di un'unità principale detta power management unit (PMU). Tra questi power domains possiamo avere, ad esempio, il PLL, la CPU, i circuiti a radiofrequenza per la trasmissione e ricezione di dati e i circuiti digitali per il WiFi che si occupano della connessione a un access point tramite protocollo WiFi.

Il sistema dispone di tre differenti modalità di sleep, in base alle esigenze di progetto:

- **MODEM SLEEP MODE:** Questa modalità è ottimizzata per ridurre il consumo energetico quando il dispositivo non risulta attivamente coinvolto nella trasmissione o ricezione di dati WiFi. Vengono, quindi, spenti la maggior parte dei componenti del modem, riducendo così il consumo.

  Tuttavia, è importante notare che l'efficacia della modalità modem sleep dipende dall'utilizzo specifico della scheda, in particolare se viene configurato come access point (AP) o come stazione(station) in una rete WiFi.

  Quando il dispositivo è impostato come access point, significa che sta agendo come punto di accesso per gli altri dispositivi, consentendo loro di connettersi ad esso. In questa modalità, l'ESP32-C3 deve rimanere sempre attivo per gestire le richieste di connessione dai dispositivi connessi. D'altra parte, quando la scheda è configurata come stazione(station) e si connette ad un altro punto di accesso WiFi, la modalità modem sleep può risultare utile per risparmiare energia.

In questo modo, quando il sistema è inattivo, ad esempio quando non sta trasmettendo o ricevendo dati, la scheda può entrare nella modalità modem sleep per ridurre il consumo energetico senza interrompere la connessione WiFi.

In sintesi, la modalità modem sleep offre vantaggi solo nel caso dell'utilizzo del dispositivo come station, mentre non offre vantaggi in termini di risparmio energetico nel caso di un suo utilizzo come access point.

- **LIGHT SLEEP MODE**: In questa modalità vengono disabilitati i circuiti wireless digitali e i circuiti RF, disabilitando totalmente le funzionalità del modulo WiFi.

  Oltre a questi viene tolta l'alimentazione anche alla CPU, impedendo l'esecuzione di un programma precedentemente caricato.

  Vengono inoltre disabilitati il PLL e i vari oscillatori interni.

  Rimane invece operativo il power management unit (PMU), quella componente che si occupa della gestione delle alimentazioni e dell'impostazione delle varie modalità di risparmio energetico, essendo l'unità principale che regola le alimentazioni verso i vari power domains, in particolare, in questo caso rimangono alimentate la maggior parte delle periferiche analogiche e digitali come ad esempio il convertitore analogico-digitale.
- **DEEP SLEEP MODE:** Questa è la modalità che fornisce il maggior risparmio energetico, lasciando alimentato soltanto il power management unit e una periferica real time clock (RTC), la quale consiste in un contatore che potrà essere utilizzato per svegliare il chip, ed è strettamente correlato al PMU.

Per quanto riguarda il sistema per il risveglio della scheda dallo stato di sleep, il PMU presenta una macchina a stati realizzata alla base del real time clock.

In base allo stato attuale, l'RTC comanderà ogni power controller per alimentare o sottoalimentare ogni power domain. Quando il wakeup controller (che può essere rappresentato da un pin RTC della scheda) invia un segnale alla macchina a stati, significa che si è verificato un evento in grado di svegliare il chip, di conseguenza, il cambio di stato porta a ripristinare le alimentazioni a tutti i power domains tramite i

power controller, che non sono altro che i controller che portano l'alimentazione a ogni power domain.

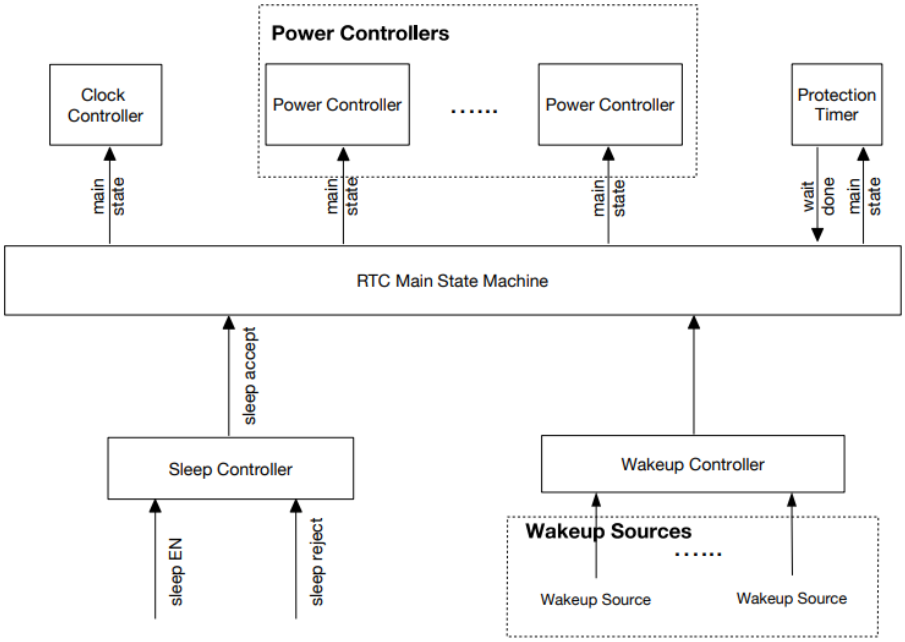

Figure 9-2. Power Management Unit Workflow

Figura 7: Struttura del power management unit alla base dell'RTC(fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

Gli eventi che possono svegliare il chip sono essenzialmente due, un evento di interrupt su un pin esterno, il quale deve essere definito come pin RTC, quindi un pin comunicante con il sistema RTC, oppure un certo valore di conteggio raggiunto dal timer RTC.

2 Soluzioni software implementate

Si è scelto come ambiente di sviluppo integrato (IDE) per la programmazione della scheda l'IDE Arduino, il quale a differenza di ESP-IDF (il framework di sviluppo software open source dell'Espressif), risulta estremamente più semplice da utilizzare e si basa su un linguaggio C/C++, quindi molto più adatto per la scrittura di codice a basso livello.

È bene inoltre specificare che varie funzioni rese disponibili nella documentazione associate l'ESP-IDF sono totalmente utilizzabili anche all'interno dell'IDE di Arduino. Per quanto riguarda la struttura del codice, quest'ultimo può essere suddiviso in cinque blocchi principali:

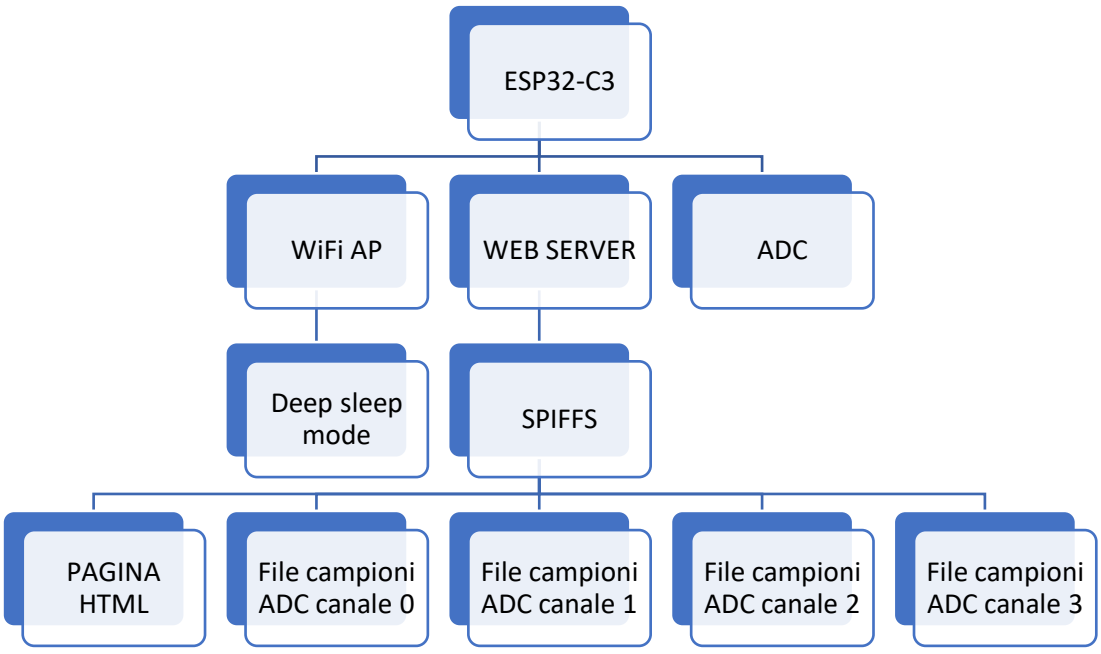

Figura 8: Il diagramma mostra l'idea di base del progetto, evidenziando la gerarchia dei vari blocchi.

1) **Convertitore analogico digitale (ADC)**: L'unità fondamentale per la lettura di dati provenienti da sensori.

Si è utilizzato il canale 0 dell'ADC1, ma si è reso disponibile l'utilizzo degli altri 3 canali, mentre si è evitato l'utilizzo dell'ADC2, essendo più soggetto a problematiche di rumore come riportato nella documentazione.

Si è inoltre utilizzata una funzione di calibrazione sull'ADC1 per correggere in piccola parte variazioni della tensione di riferimento del convertitore.

2) **WiFi AP**: Per rendere il sistema embedded facilmente accessibile da remoto si è impostato quest'ultimo come access point con determinate credenziali di rete. A differenza dell'impostazione station, in questo caso si rende il dispositivo indipendente da eventuali spostamenti o variazioni di una rete locale.

3) **Web Server Asincrono**: È stato inizializzato un web server http, accessibile da ogni client connesso all'access point. Tramite un sistema a richieste http, viene restituita una pagina web contenente l'interfaccia per l'utilizzo dell'ADC e la scrittura dei campioni sullo SPIFFS. Il web server è stato inizializzato in maniera asincrona, in questo modo le singole richieste http non risultano bloccanti e si possono gestire più richieste simultaneamente, rendendo il tutto più robusto.

4) **SPIFFS**: Si è utilizzato il sistema SPIFFS sia per contenere la pagina web, sia per contenere i file di testo contenenti i valori campionati da ogni canale ADC.

5) **Risparmio Energetico (DEEP SLEEP)**: Vista l'esigenza di utilizzare la scheda come access point, si è optato per la modalità a maggior risparmio energetico, cioè la modalità deep sleep.

2.2 Calibrazione software ADC

Come introdotto precedentemente, il convertitore analogico digitale dispone di una tensione di riferimento interna di 1100 mV. Nonostante tutto, questa tensione di riferimento non risulta totalmente fissa ma può variare tra differenti schede, portando a lievi errori per quanto riguarda il corrispondente valore digitale misurato.

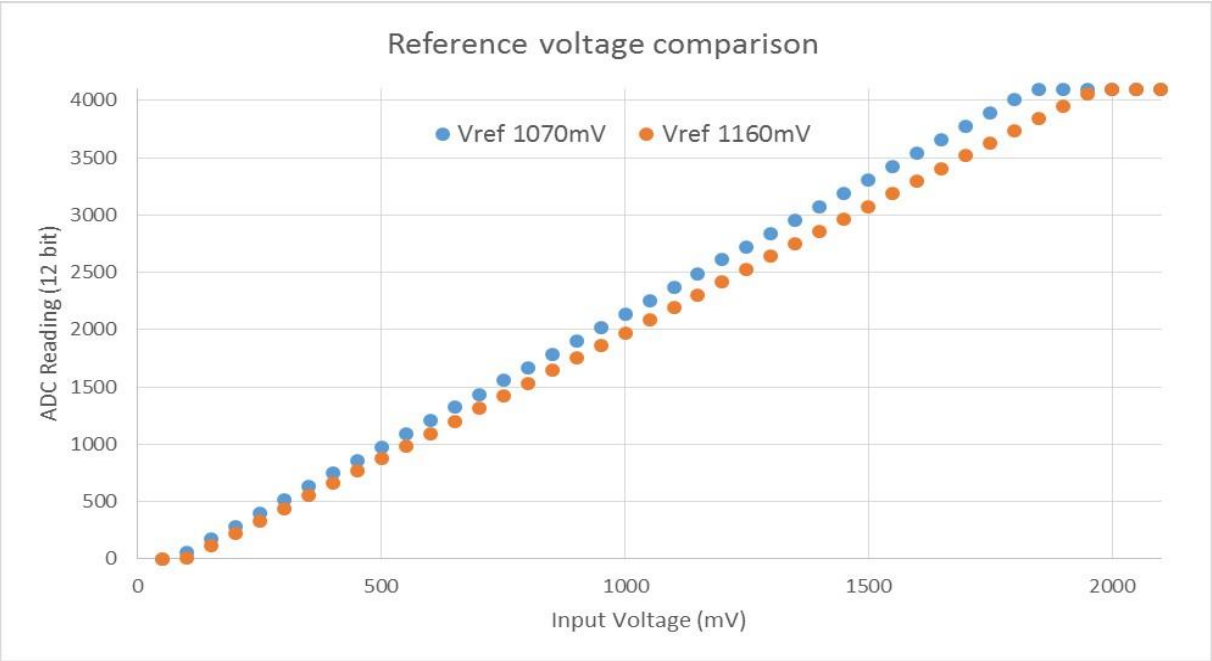

Figura 9: caratteristica dell'ADC dell'ESP32-C3. Si nota come variazioni della tensione di riferimento possano influenzare principalmente la parte più alta della caratteristica. In questo esempio riportato nella documentazione si è effettuata l'attenuazione sugli ingressi dell'ADC per avere una tensione di riferimento massima di 1750 mV (fonte: documentazione Espressif, [Analog to Digital Converter (ADC) - ESP32 - — ESP-IDF Programming Guide release-v4.4 documentation (espressif.com)]).

È stata perciò utilizzata una libreria messa a disposizione dall'Espressif, "esp_adc_cal.h", la quale corregge la corrispondenza tra un valore di tensione misurato e il suo corrispettivo valore digitale.

Questa calibrazione avviene tramite il metodo "Two Points", con cui si effettua la misurazione di due punti di riferimento noti da parte dell'ADC che possano all'incirca comprenderne la dinamica di tensione. Essendo la dinamica dell'ADC compresa tra 0 a 1.1 V circa, le misurazioni avvengono internamente a 150 mV e a 850 mV.

Tramite l'eFuse (Electrically Fuse), una porzione di memoria permanente a 4096 bit utilizzata per dati critici della scheda, presente all'interno del real time clock, vengono confrontatati i valori delle due misurazioni con un'altra tensione di riferimento sempre a 1.1 V, ma di valore più preciso contenuta nell'eFuse. Questo fornisce due punti della

caratteristica totalmente corrette, e a partire da questi dati vengono calcolati dei fattori di correzione che vengono poi salvati in una porzione dell'eFuse.

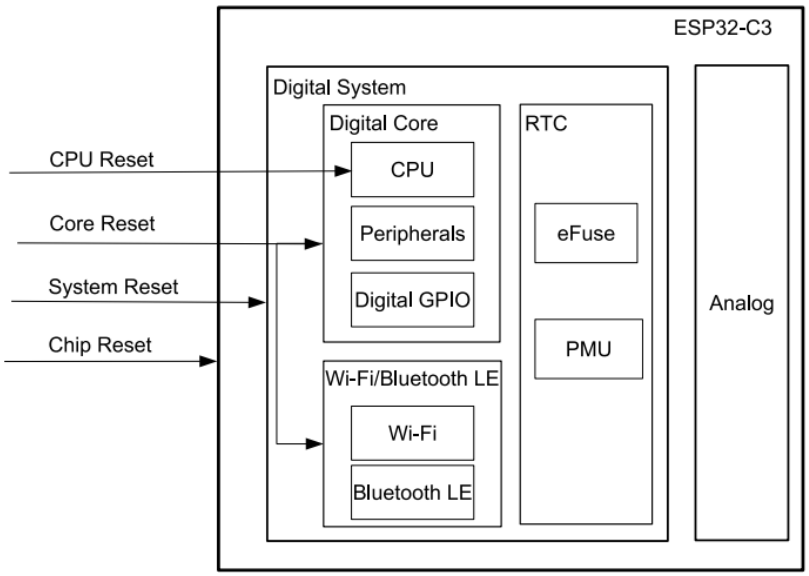

Figura 10 : Struttura semplificata dell'ESP32-C3. Si osservano l'eFuse e il PMU presenti alla base dell'RTC, oltre che la suddivisione tra periferiche wi-fi/bluetooth e le periferiche analogiche e digitali(fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

Dopodiché, in seguito a ogni misurazione, vengono prelevati dall'eFuse questi fattori di correzione, i quali forniscono una migliore corrispondenza tra ogni valore analogico e digitale misurato.

Normalmente questa procedura dovrebbe essere necessaria soltanto una volta, in quanto i fattori di correzioni rimangono salvati in maniera permanente nell'eFuse, ma, come riportato nella documentazione ufficiale, effettuare nuovamente l'operazione di calibrazione più volte può correggere eventuali variazioni nel tempo della tensione di riferimento dell'ADC, ad esempio dovute a variazioni di temperatura.

Nonostante per la calibrazione ci si è basati sulla misurazione interna, tramite ADC, dei due punti a 150 mV e a 850 mV, vi è inoltre la possibilità di effettuare manualmente, tramite due tensioni esterne, la misurazione di due differenti punti, nel caso in cui si voglia ad esempio migliorare la qualità di una determinata regione della caratteristica del convertitore.

2.1 Wi-Fi software enabled access point (softAp)

Per quanto riguarda la connettività, si è deciso di optare per la comunicazione WiFi invece di quella Bluetooth LE. Sia per il fatto che la comunicazione tramite Wi-Fi risulta notevolmente più veloce, sia per la maggior complessità di un firmware personalizzato per il Bluetooth, portando alla necessità di implementare anche un'applicazione mobile dedicata.

Per la realizzazione, si è utilizzata la libreria integrata "WiFi.h", compatibile per varie schede ESP-32, inoltre un vantaggio della libreria, è la sua gestione ad eventi, completamente compatibile con le altre funzioni messe a disposizione dalla ESPRESSIF.

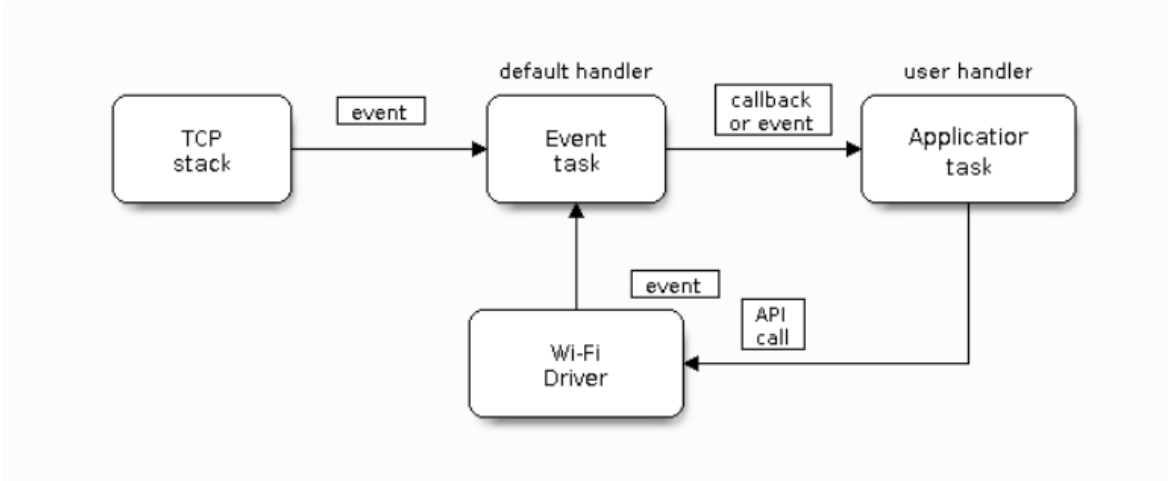

Figura 11 : Modello gestione eventi WiFi per le varie schede ESP-32 (fonte: documentazione Espressif, Wi-Fi Driver - ESP32-S2 - — ESP-IDF Programming Guide latest documentation (espressif.com)).

In particolare (*figura 11*), vi è l'application task, la parte di firmware scritta per inizializzare il microcontrollore come access point e inizializzare vari eventi tramite il driver WiFi, tra cui vi possono essere ad esempio la disconnessione o la connessione di un utente all'access point, oppure la connessione o disconnessione del dispositivo da una rete locale, come station.

Tramite il driver WiFi, il default event loop controlla ciclicamente il verificarsi di un determinato evento tramite delle funzioni di callback.

Nel caso del codice sviluppato, sono stati inizializzati due eventi, uno al verificarsi della connessione di un client all'access point e uno al verificarsi della disconnessione del client dall'access point.

È stata utilizzata questa gestione ad eventi per non saturare il loop principale del codice, potendolo così sfruttare per altre operazioni come letture continue dell'ADC.

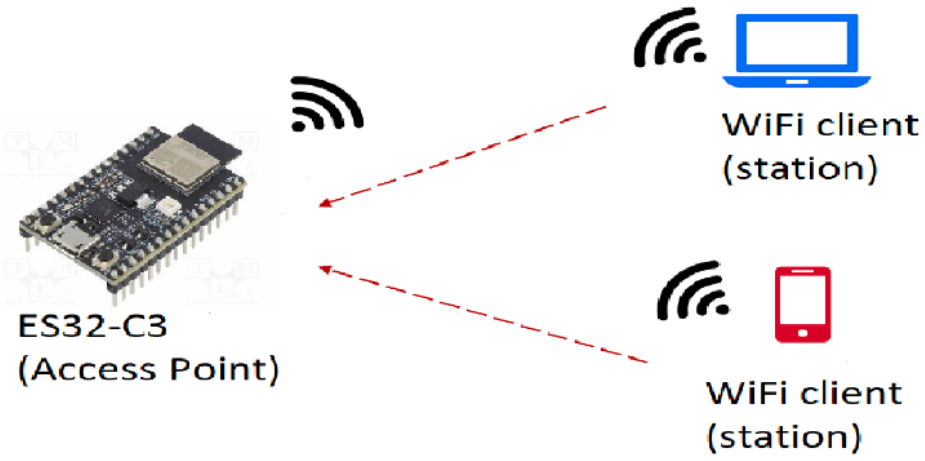

Figura 12 : L'ESP32-C3 impostato come access point, mentre gli altri dispositivi sono identificati come client a lui associati.

Per quanto riguarda la connessione da parte di un client, sono state impostate delle credenziali per l'accesso all'ESP32-C3 e, avvenuta la connessione, al client viene assegnato staticamente un indirizzo IP che sarà l'IP associato al web server che restituirà la pagina web.

2.2 Web server asincrono

Un web server è un applicazione software che supervisiona la comunicazione tra un client browser e il server hardware vero e proprio. In questo caso l'accesso dell'utente tramite browser, inserendo l'IP nella barra di ricerca, definisce la parte client, la parte web server è rappresentata dal software sviluppato sulla scheda ESP32-C3, mentre il server vero è proprio è la scheda ESP32-C3.

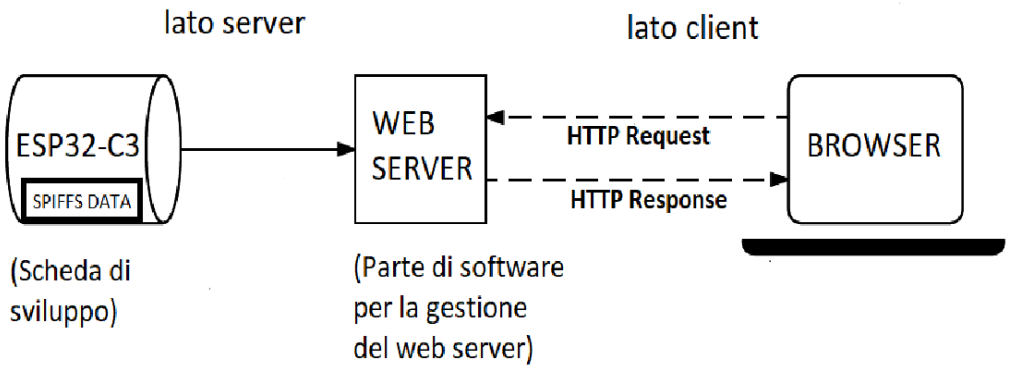

Figura 13: Struttura architettura client- server impiegata.

In particolare, a ogni client connesso alla scheda ESP32-C3, viene assegnato un indirizzo IP, il quale, inserendo nella barra di ricerca di un qualsiasi browser l'indirizzo IP del server ESP32-C3, effettua una richiesta http di tipo GET, con cui si richiede una risorsa al server, che in questo caso è la pagina web. Il web server in questo caso agisce da supervisore tra client e server, prelevando la risorsa richiesta dal server (più precisamente dalla memoria SPIFFS della scheda), restituendola al browser, il quale tradurrà i vari blocchi HTML, rendendo visibile all'utente la pagina web. Per quanto riguarda le operazioni di lettura e scrittura tramite SPIFFS dei campioni ADC, si sono implementate delle richieste HTTP POST, le quali vengono inoltrate al web server alla pressione di alcuni pulsanti presenti sulla pagina HTML.

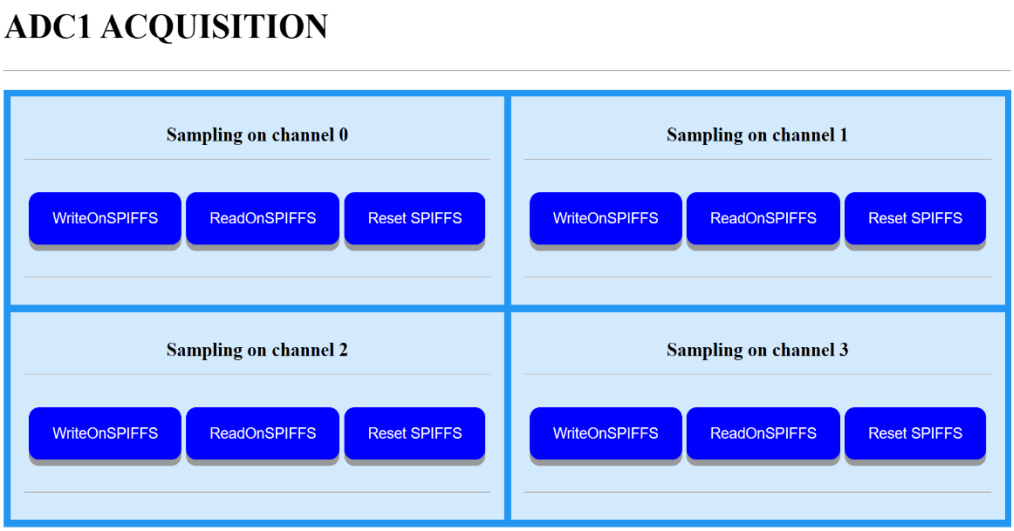

Figura 14: Pagina html restituita dal web server. Permette l'acquisizione tramite ognuno dei 4 canali dell'ADC1

È stata resa possibile l'acquisizione di una certa quantità di campioni (nei test effettuati vengono salvati circa 13 secondi di campioni con una frequenza di campionamento di 1Khz, quindi 13000 campioni), per ognuno dei quattro canali dell'ADC1. In particolare (*Figura 14*), tramite i pulsanti "WriteOnSpiffs" vengono misurati un certo numero di campioni da un determinato canale dell'ADC, che vengono poi salvati in modo non volatile nello SPIFFS. Il pulsante "ReadOnSPIFFS", invece, restituisce in download un file di testo contenente i campioni precedentemente salvati. Infine, il pulsante "Reset SPIFFS", cancella tutti i campioni salvati associati a un determinato canale. Per quanto riguarda la struttura del codice, si è utilizzata la libreria "ESPasyncWebServer", la quale fornisce già un'implementazione di base di un web server asincrono per scheda della Espressif. Il vantaggio principale di un web server asincrono consiste nell'avere richieste http non bloccanti, cioè, durante la gestione di una richiesta, il web server può già cominciare a servire quelle successive in maniera non bloccante. Questo riduce il tempo di risposta e rende il sistema di gestione client-server più robusto.

Dopodiché, oltre alla parte di codice HTML-CSS per la parte contenutistica ed estetica della pagina, si è scritto anche del codice javascript per la gestione delle richieste post in maniera asincrona. Ognuno di questi file di testo (*figura 15*) è stato impostato per riportare l'istante temporale di campionamento in microsecondi, il valore digitale del campione e il suo corrispondente valore in millivolt.

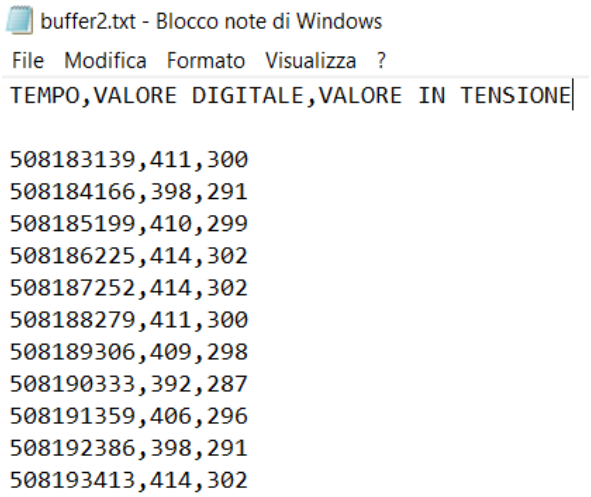

Figura 15 : In figura viene mostrata una piccola parte di un file di testo restituito dallo SPIFFS contenente dei campioni letti dal canale 2 dell'ADC, per questo il nome buffer2. Da sinistra a destra viene riportato l'istante di campionamento in microsecondi, il valore digitale del campione e il corrispondente valore in tensione.

2.3 SPIFFS come flash secondaria e datalogger

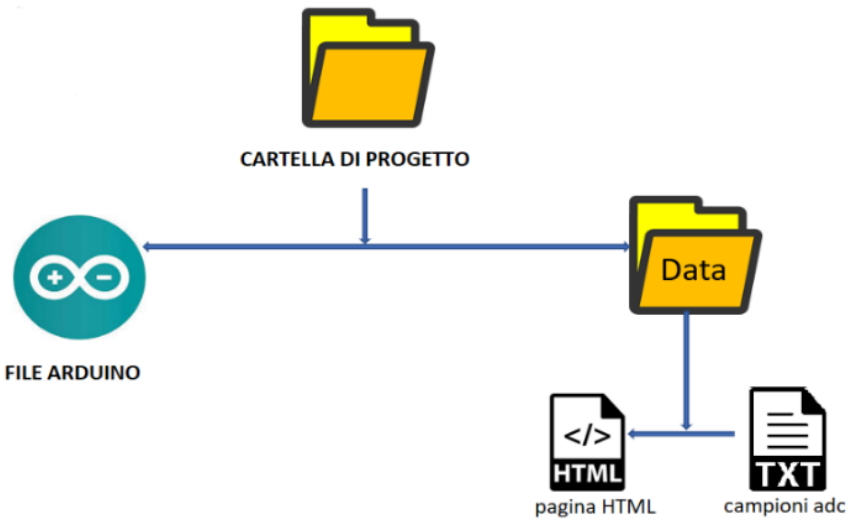

Figura 16: Directory del file system SPIFFS dentro la cartella del progetto.

Per quanto riguarda il file system SPIFFS, si è utilizzata la libreria "SPIFFS.h", la quale semplifica particolarmente la gestione del file system. La comodità principale consiste nel fatto che lo SPIFFS può essere gestito nel classico modo con cui si effettuano letture e scritture da file in c/c++.

L'IDE di Arduino rende la gestione dello SPIFFS particolarmente comoda e veloce, infatti, risulta solamente necessario creare una cartella "data" nella directory del progetto di Arduino (*figura 16*), la quale conterrà i vari file di base che si vogliono caricare nello SPIFFS. Dopodiché basta impostare la cartella data come la radice del file system SPIFFS, utilizzando un'opzione disponibile sull'IDE di Arduino (*Figura 17*).

Configurato il file system, si è utilizzato lo SPIFFS per la lettura e la scrittura su quest'ultimo in differenti file, in base al canale di acquisizione.

In particolare, oltre al file main.html contenente la pagina web, si sono creati nello SPIFFS altri quattro file, con il nominativo "buffer", ognuno associato a un canale ADC differente.

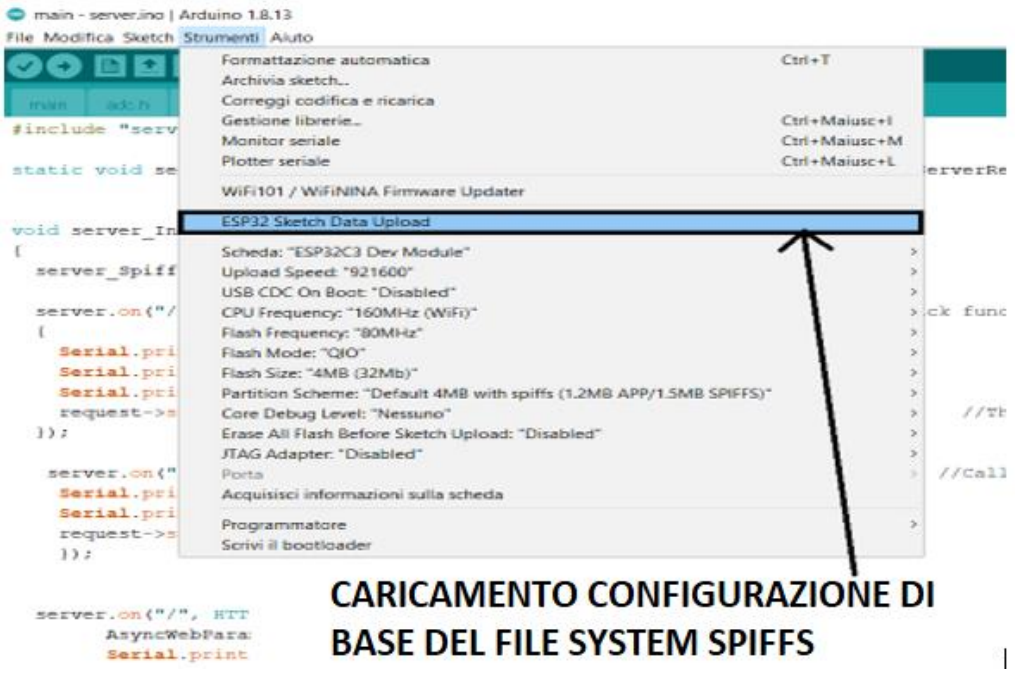

Figura 17: Impostazione struttura file system di base sull'IDE di Arduino.

Per quanto riguarda la capienza dello SPIFFS, la documentazione ufficiale non riporta valori precisi, fornendo soltanto valori che oscillano tra 1Mb e i 2Mb.

Tramite le funzioni rese disponibili dalla libreria SPIFFS, si è potuta definire indicativamente la capienza massima, che si attesta intorno agli 1.2Mb (*figura 18*). Per verificarlo, si è forzatamente provato a saturare la memoria, confermando all'incirca uno spazio massimo di 1.2Mb, concorde con quello mostrato dalle funzioni della libreria.

È bene però specificare che lo spazio massimo occupato dallo SPIFFS può risultare differente per schede ESPRESSIF di tipologia anche lievemente differente.

Ad esempio, testando una scheda molto simile, l'ESP32-C3-13, si sono avuti problemi di scrittura anche per valori massimi di circa 1Mb.

In ogni caso, per l'applicazione richiesta, questo spazio risulta più che sufficiente, in quanto la pagina web occupa poco più di una decina di Kilobyte, mentre i quattro file con i campioni dell'ADC sono stati utilizzati come buffer circolari; quindi, vengono scritti e riscritti in maniera ciclica, occupando tutti insieme nel caso di scritture particolarmente lunghe, meno di 300Kb.

Inoltre, un vantaggio dello SPIFFS, mostrato precedentemente nella struttura della pagina web, è quello di permettere non solo la cancellazione della memoria nella sua interezza, ma anche dei singoli file in esso contenuti.

```
//////SPIFFS File system///////
Total Size: 1287 Kbytes
Used Size: 31 Kbytes
Free Size: 1255 Kbytes

File: buffer0.txt.txt - Size: 7 Kbytes
File: buffer1.txt.txt - Size: 0 Kbytes
File: buffer2.txt.txt - Size: 7 Kbytes
File: buffer3.txt.txt - Size: 0 Kbytes
File: main.html - Size: 14 Kbytes
```

Figura 18 : Informazione sullo SPIFFS, stampata nel monitor seriale di Arduino. Si osserva una capienza massima dello SPIFFS di circa 1.2Mb, oltre che la presenza della pagina web e dei 4 file che contengono i valori campionati da ognuno dei 4 canali.

2.6 Deep sleep mode

Si è optato per la modalità a risparmio energetico "deep sleep mode", in cui rimangono attive solo alcune periferiche del real time clock, tra cui il PMU, mentre il sistema Wi-Fi rimane totalmente inattivo.

Le modalità con cui è stata sfruttata questa tipologia di risparmio è fortemente legata alla parte di firmware riguardante la configurazione access point della scheda, in particolare la gestione a eventi fornita dalla libreria "WiFi".

Al verificarsi dell'evento per cui un client effettua un collegamento all'access point per accedere alla pagina web, non vi è alcun sistema di risparmio energetico, essendo fondamentale mantenere la connessione, mentre al verificarsi dell'evento della disconnessione dell'utente, è stato impostato un timer interno del microcontrollore, precisamente il general purpose TIMER0 a 54 bit, per effettuare un conteggio fino a 30

secondi, al termine del quale, se non avvengono connessioni, il dispositivo entra in modalità deep sleep mode.

Più precisamente, per evitare blocchi del codice nel loop principale, si è impostato un interrupt, la cui routine, che porta il microcontrollore in deep sleep, viene chiamata ogni volta che il timer raggiunge il conteggio di trenta secondi, in pratica un semplice sistema di stand by.

Definita l'idea di base per portare il chip in risparmio energetico, è poi arrivata la necessità di trovare un sistema per svegliare il microcontrollore, infatti come definito precedentemente, i metodi di risveglio consistono in un segnale esterno su un pin (come, ad esempio, un pulsante) oppure un timer contenuto nell'RTC, il quale raggiunto il fondo scala causa il risveglio.

Per definizione, essendo il dispositivo accessibile solo da remoto, si è bocciata la prima metodologia, utilizzando invece quella del timer RTC. Si è impostato anche in questo caso un tempo di sleep di 30 secondi, al fine di permettere a un utente di non dover aspettare tempi troppo lunghi, in quanto, il modulo wi-fi non è abilitato in sleep mode.

Raggiunti i trenta secondi di sleep mode, il microcontrollore va in reset, ricominciando da capo l'esecuzione del programma, garantendo trenta secondi di normale funzionamento per permettere connessioni da parte di client e trenta secondi di risparmio energetico per il risparmio della batteria.

I tempi impiegati costituiscono solo un esempio, infatti, si potrebbero anche impostare trenta secondi di risparmio in deep sleep e solo dieci secondi in modalità normale, ma si è ipotizzato che trenta secondi fossero un tempo ottimale per permettere a agli utenti di effettuare connessioni.

In questo caso, quindi, il risparmio energetico riguarda soltanto gli istanti in cui nessun client risulta collegato, permettendo un parziale risparmio energetico quando non risultano esserci collegamenti all'access point.

Oltre a questo, dato che il Real time clock è l'unica periferica che non viene mandata in sleep mode, vi è la possibilità di utilizzare l'indicatore di variabile

"RTC_DATA_ATTR" nell'IDE Arduino, per salvare nella memoria interna RTC le variabili con valori importanti che devono essere mantenute.

Un esempio potrebbe essere quello di campioni ADC che non sono stati salvati correttamente sullo SPIFFS, rendendo in questo caso la memoria RTC un buffer momentaneo prima del salvataggio sul file system SPIFFS, essendo anche la memoria RTC non volatile.

La capienza della memoria RTC per il salvataggio delle variabili risulta però molto ridotta, all'incirca 20Kb, di cui una parte viene utilizzata per i conteggi dei timer dello sleep mode; quindi, non può certamente essere sostituita al sistema SPIFFS.

Risulta però un sistema più veloce per la scrittura di piccole quantità di dati in memoria e la sua scrittura risulta molto più veloce rispetto allo SPIFFS, essendo naturalmente presente tra le periferiche del microprocessore, mentre lo SPIFFS risulta essere una porzione di memoria flash esterna che sfrutta il protocollo SPI.

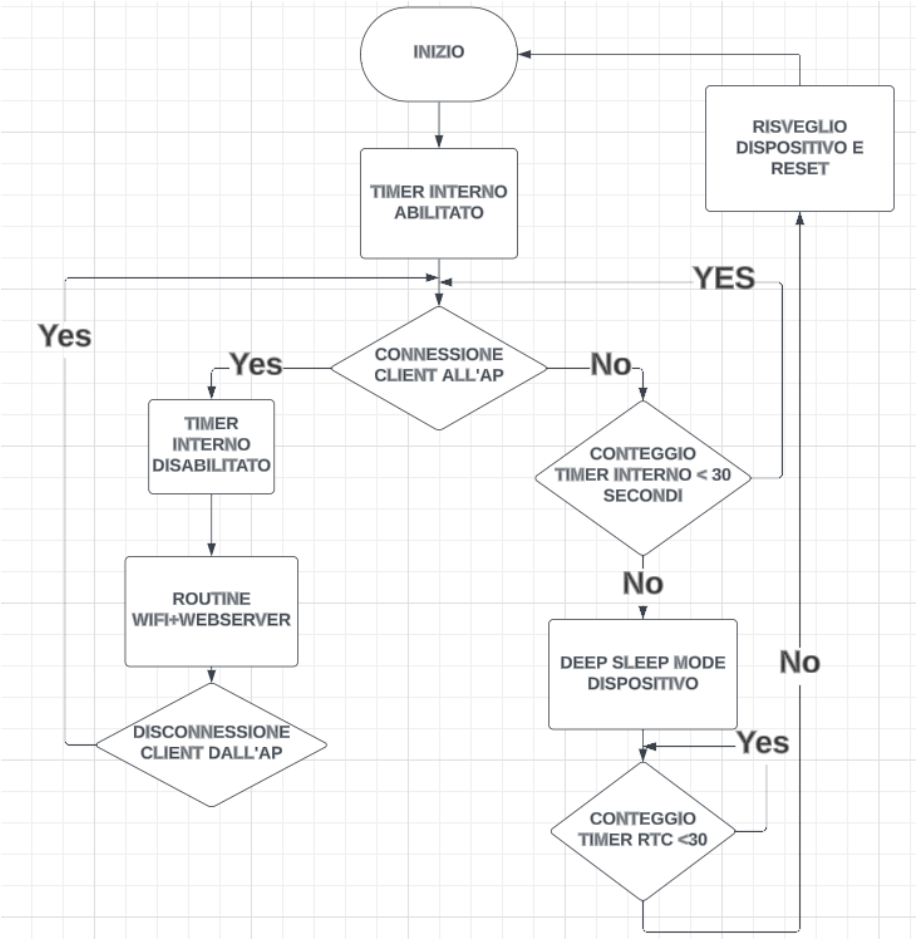

Figura 19 : Struttura dell'algoritmo che impiega la modalità di risparmio energetico

3 Misure sperimentali

In questa parte dell'elaborato, vengono trattate varie misure effettuate in laboratorio per testare alcune soluzioni implementate. In particolare, si sono effettuati dei test sul convertitore analogico digitale al fine di ricostruirne la caratteristica, mettendo anche in evidenza per quali valori si raggiunge la saturazione. Si è inoltre messa in evidenza la caratteristica dell'ADC sia con la calibrazione, sia senza calibrazione. Sono stati, inoltre, effettuati dei test su delle sinusoidi di differente ampiezza e con una frequenza di 50Hz tramite generatore di funzioni, sempre al fine di parametrare la qualità della misurazione. Si è poi misurato il segnale associato a un sensore piezoelettrico, sufficientemente condizionato, tramite un semplice circuito raddrizzatore, confrontando il tutto anche con le misurazioni effettuate direttamente da un oscilloscopio digitale.

Si è infine testato il sistema di sleep mode, confrontando il consumo energetico rispetto a condizioni normali di lavoro.

3.1 Caratteristica ADC

Tramite l'ausilio di un alimentatore da banco, sono stati dati in ingresso al canale 0 dell'ADC1 differenti valori di tensione. Si è testato solo il canale 0, in quanto si è constatato come le prestazioni sugli altri 3 canali risultassero piuttosto simili.

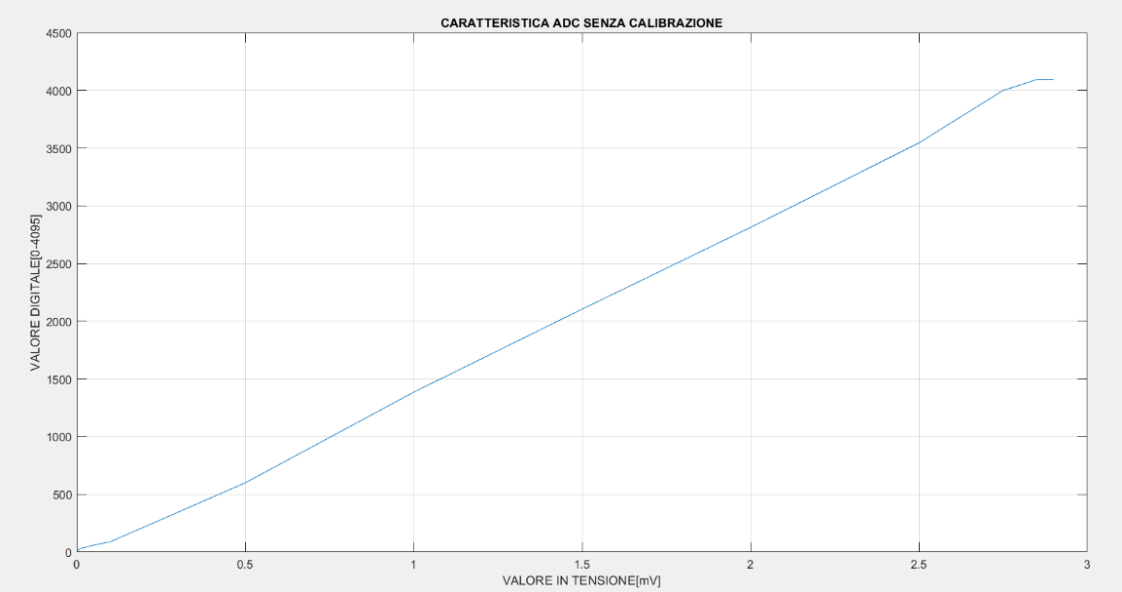

Figura 20: Caratteristica adc senza calibrazione.

In particolare, sono stati dati in ingresso al canale ADC valori di tensione, con intervalli di circa 200 mV, con intervalli più fitti per quanto riguarda le parti più critiche, quindi i valori presenti agli estremi della caratteristica. Dopo aver misurato i vari punti, si è graficata la caratteristica ottenuta tramite il software Matlab. Si sono notate lievi non linearità nelle zone poste alle estremità della caratteristica, ma nonostante tutto la caratteristica totale risulta accettabile (*figura 20*).

Si è notato, però, che la tensione di riferimento, con l'attenuazione da noi impostata, avrebbe dovuto avere un valore di 2.5 V (*Figura 4*), mentre in questo caso raggiunge un valore di 2.85 V circa, quindi ben superiore. Si è inoltre osservato che, su un'altra scheda di sviluppo è stata misurata una tensione ancora differente, quindi si ipotizza che la tabella mostrata nella documentazione (Figura 4), fornisce un valore non preciso per quella che può essere la nuova tensione di riferimento.

In parallelo, si è tentato di calcolare la caratteristica dell'ADC, anche in presenza di calibrazione, la quale tramite la correzione della tensione di riferimento dovrebbe risolvere problematiche di non linearità.

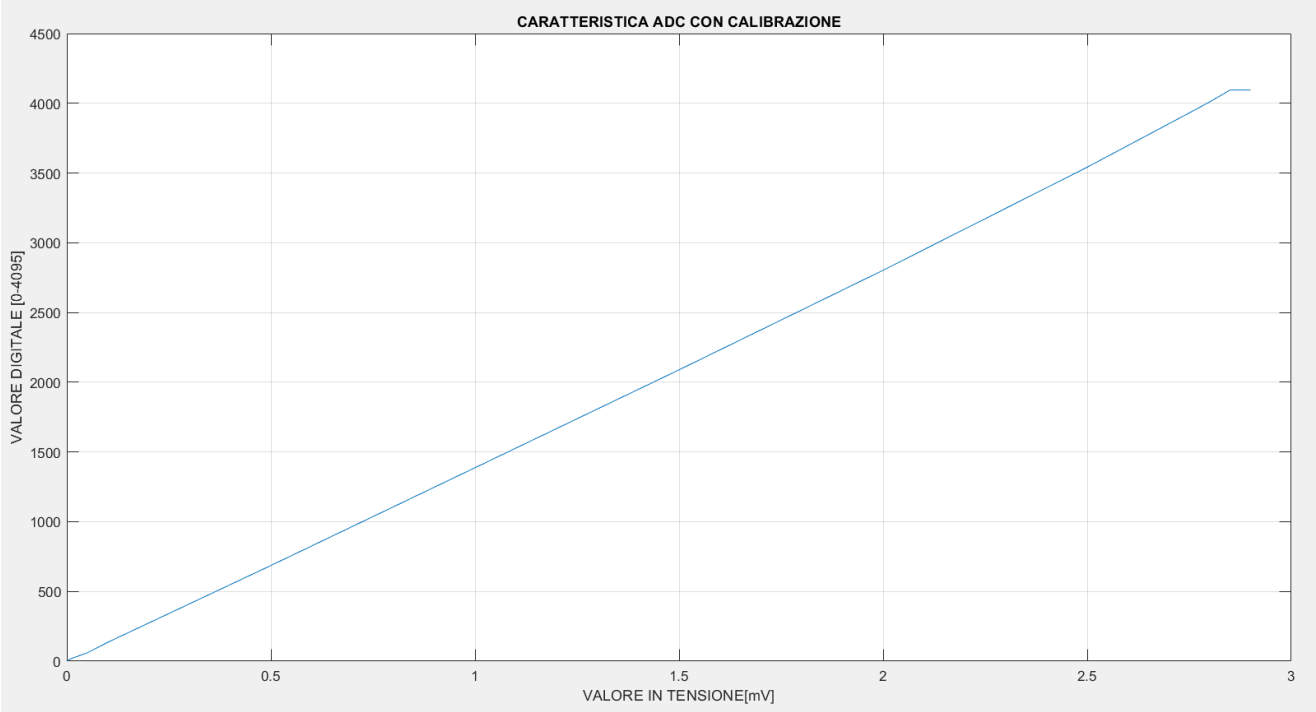

Figura 21: Caratteristica adc con calibrazione

Si osserva (*figura 21*) che, in questo caso, non sono più presenti quelle lievi linearità mostrate precedentemente nell'altra caratteristica. Nonostante tutto, è bene specificare che misurando le differenze tra i vari punti delle due caratteristiche risultano esserci variazioni abbastanza irrisorie, ma le misurazioni sono state effettuate sia una per volta, sia in un ambiente controllato come un laboratorio. In situazioni con maggior presenza di rumore si suppone che l'ausilio della calibrazione risulti maggiormente evidente.

Per quanto riguarda i risultati numerici, la calibrazione fornisce una maggiore precisione dei valori digitali ottenuti, con un errore massimo corrispondente di circa +- 35 mV per quello che riguarda l'opzione di attenuazione che è stata scelta nel progetto

Parameter	Description	Min	Max	Unit
Total error	ATTEN0, effective measurement range of 0 ~ 750	–10	10	mV
	ATTEN1, effective measurement range of 0 ~ 1050	–10	10	mV
	ATTEN2, effective measurement range of 0 ~ 1300	–10	10	mV
	ATTEN3, effective measurement range of 0 ~ 2500	–35	35	mV

Figura 22: Tabella che mostra i risultati della calibrazione, evidenziando il range di errore possibile per ogni opzione attenuazione (fonte: ESP32-C3 datasheet, esp32-c3_datasheet_en.pdf (espressif.com)).

3.2 Test ADC tramite sinusoide

Tramite l'ausilio di generatore di funzioni, si è poi testato l'ADC, fornendogli in ingresso varie sinusoidi a 50 Hz. In questo caso l'accortezza principale riguarda il range di tensione da fornire all'ingresso all'ADC dell'ESP32-C3; infatti, fornire tensioni negative può portare al rischio di danneggiamento della scheda, per questo è stato fornito un offset a queste sinusoidi di test, al fine di fornire solo valori di tensione positivi.

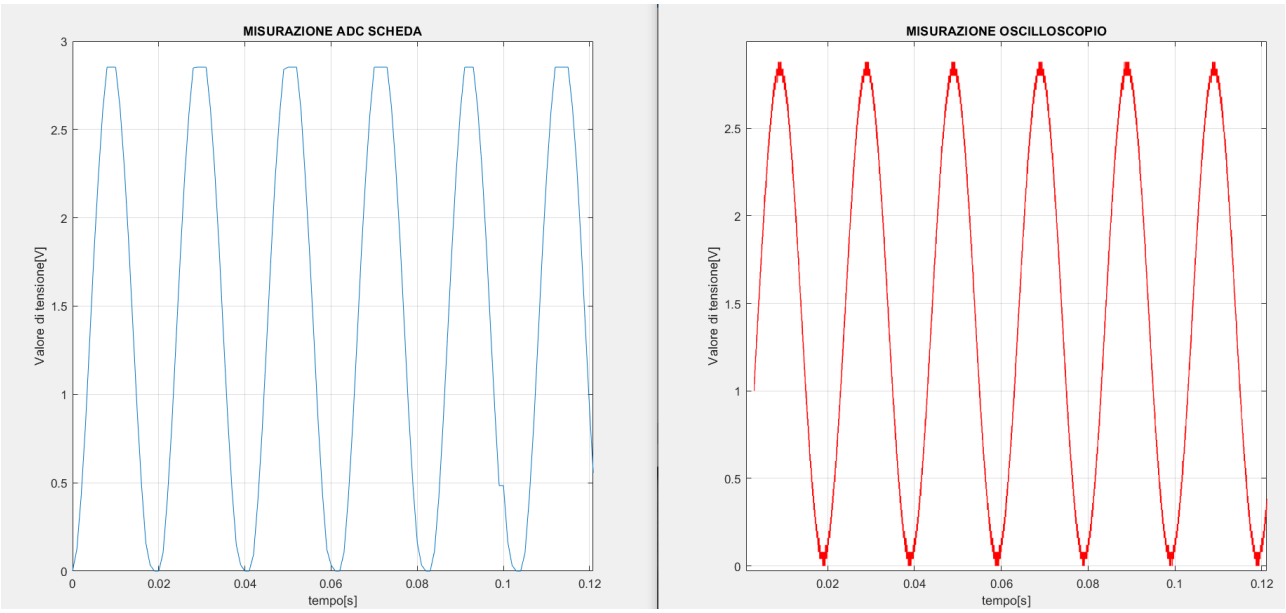

Figura 23 : Confronto di una sinusoide. A destra si può osservare la lettura effettuata dall'oscilloscopio, mentre a sinistra quella effettuata dall'ADC della scheda. In quest'ultimo caso si nota un lieve clipping della sinusoide intorno i 2.85 V.

Si mette in evidenza (*Figura 23*) un test effettuato con una sinusoide a 50 Hz, 2.9 V picco-picco e un offset di 1.45 V. Questa misurazione conferma le limitazioni discusse precedentemente sul convertitore analogico digitale, mostrando un lieve clipping della sinusoide intorno ai 2.85 V, mentre la sinusoide misurata direttamente dall'oscilloscopio non mostra queste problematiche. Si può inoltre osservare che nel caso dell'ADC del microcontrollore, i campioni più critici sono quei valori di tensione vicini al limite inferiore e al limite superiore della dinamica, come discusso precedentemente.

3.3 Test ADC tramite sensore piezoelettrico

Continuando i test sul convertitore analogico-digitale, si è utilizzato un sensore piezoelettrico flessibile, il quale sfrutta il principio dell'effetto piezoelettrico per convertire l'energia meccanica in segnali elettrici.

L'effetto piezoelettrico si verifica nei materiali che hanno la capacità di generare una carica elettrica quando vengono sottoposti a una deformazione meccanica. Quando un materiale di questa tipologia viene soggetto a una tensione o compressione, le sue

strutture cristalline subiscono una deformazione, causando una certa distribuzione delle cariche elettriche. Di conseguenza, la distribuzione di carica aumenta all'aumentare della sollecitazione, invertendo la distribuzione di carica con una deformazione nel senso opposto.

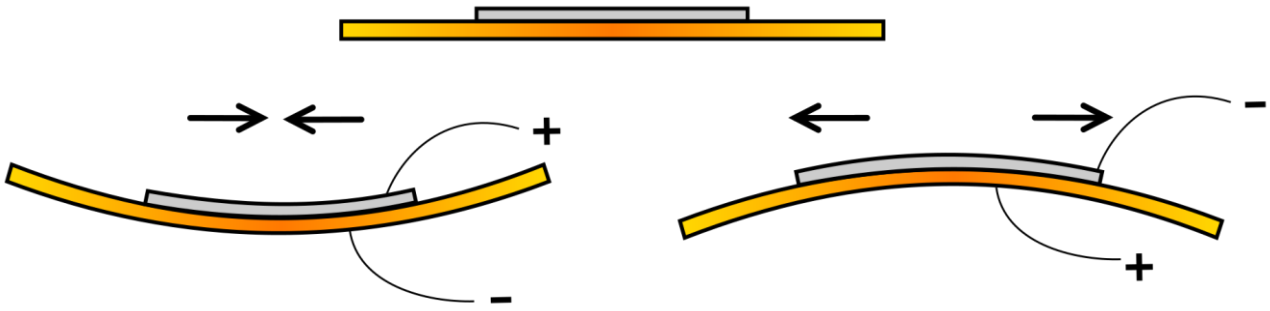

Figura 24 : Schema di base di un sensore piezoelettrico flessibile. Una deformazione in un senso o nell'altro produce una distribuzione di cariche invertita

In questo caso, un sensore di questo tipo risulta valido per rilevare grandezze fisiche come forza, pressione, flessione e vibrazioni, inoltre, la sua flessibilità dovuta a materiali polimerici, lo rende facilmente aderente a diverse superfici, come ad esempio l'interno di uno pneumatico, dove può essere utilizzato in un sistema embedded per monitorare vibrazioni e urti a cui questo pneumatico può risultare sottoposto.

Non avendo a disposizione uno pneumatico per il test del sensore, si è utilizzato un vibration test system, detto anche "shaker", un dispositivo in grado di effettuare test di vibrazione su varie tipologie di oggetti, al fine di valutare resistenza, durata e prestazioni di questi ultimi in condizioni di vibrazioni reali.

L'intero sistema può essere schematizzato in questo modo:

- **ATTUATORE:** è l'elemento principale del sistema, ed è responsabile della generazione delle vibrazioni. Quando una certa corrente elettrica viene fatta passare all'interno di una bobina, viene generato un campo magnetico, il quale interagendo con il campo magnetico di un magnete permanente genera così una forza proporzionale alla corrente.

- **AMPLIFICATORE:** è quel dispositivo che si occupa di amplificare il segnale in ingresso, fornito in questo caso da un generatore di funzioni, in modo da fornire all' attuatore(shaker) la potenza necessaria per ottenere le vibrazioni desiderate.
- **SISTEMA DI FISSAGGIO:** è quella parte del sistema in cui viene inserito il dispositivo da testare, in questo caso il sensore piezoelettrico, il quale comincerà a oscillare sotto l'effetto delle vibrazioni date dallo shaker.

Figura 25: Vibration test system con il sensore piezoelettrico posizionato.

In particolare, tramite generatore di funzioni, si sono generati all'ingresso dell'amplificatore dello shaker delle sinusoidi a 10Hz e 1.5 V picco-picco a intervalli di circa 500 ms.

Misurando tramite oscilloscopio l'uscita del sensore piezoelettrico, si è osservata un'oscillazione del sensore con una frequenza intorno ai 26Hz, la quale è dipesa dal materiale del sensore e dalla sua frequenza di risonanza.

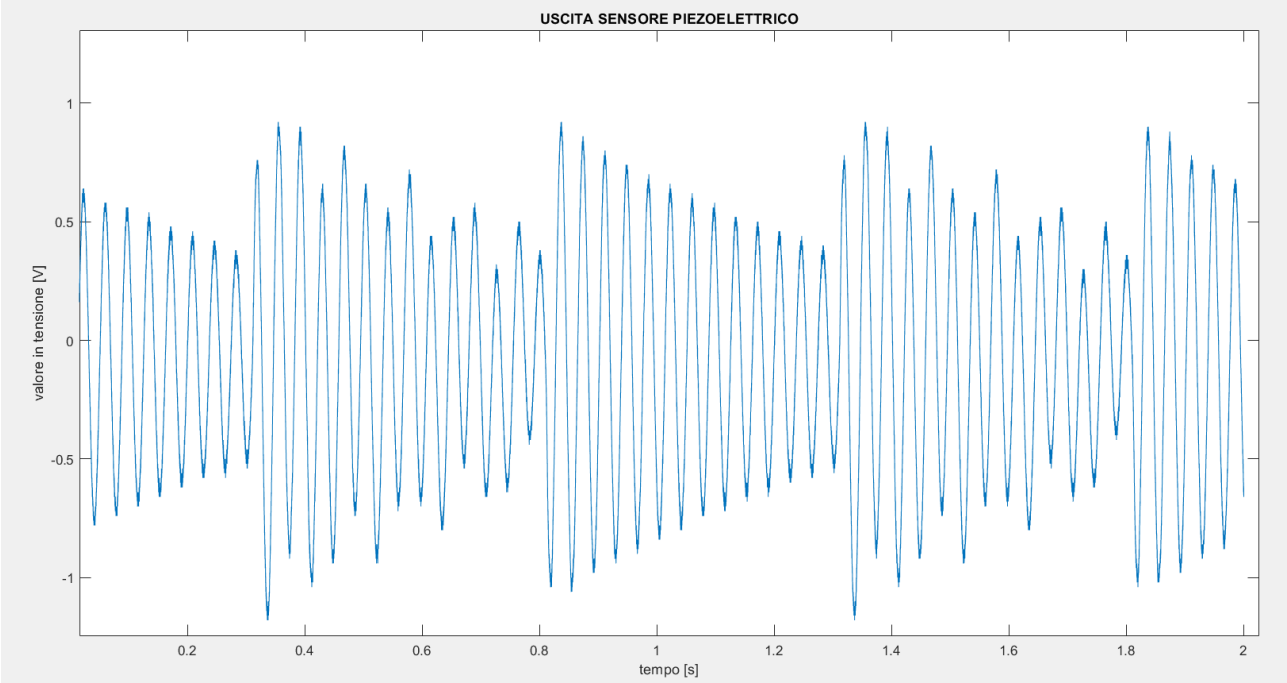

Figura 26: Oscillazioni a 26Hz misurate in uscita dal sensore piezoelettrico da parte dell''oscilloscopio.

Il segnale ottenuto, però, non risulta adatto per essere campionato dall'ADC della scheda, in quanto quest'ultima permette il campionamento solo di valori di tensione positivi. È risultato quindi necessario il raddrizzamento del segnale tramite diodo, al fine di poter rilevare solo campioni positivi.

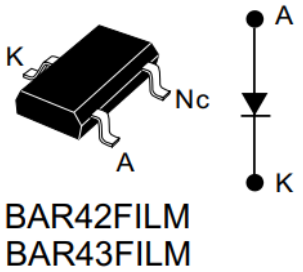

Table 1. Device summary

Symbol	Value
$I_{F(AV)}$	0.1 A
V_{RRM}	30 V
T_j	150 °C
V_F (max)	0.33 and 0.40 V

Figura 27: Diodo BAR42 con le sue caratteristiche principali (fonte: datasheet diodo schottky BAR42, Small signal Schottky diode (mouser.it)).

Si è impiegato un diodo SMD BAR42 per il raddrizzamento del segnale, il quale essendo un diodo Schottky, presenta una caduta di tensione diretta inferiore rispetto ai diodi convenzionali (figura 27).

La caduta di tensione diretta, infatti, è bene che non sia troppo grande, avendo a che fare con un segnale in tensione in uscita dal sensore abbastanza contenuto.

Il segnale in uscita dal diodo è stato posto in ingresso al canale 0 dell'ADC della scheda e al contempo, è stato misurato da un ulteriore canale dell'oscilloscopio, al fine di poter confrontare il campionamento di quest'ultimo con quello del microcontrollore.

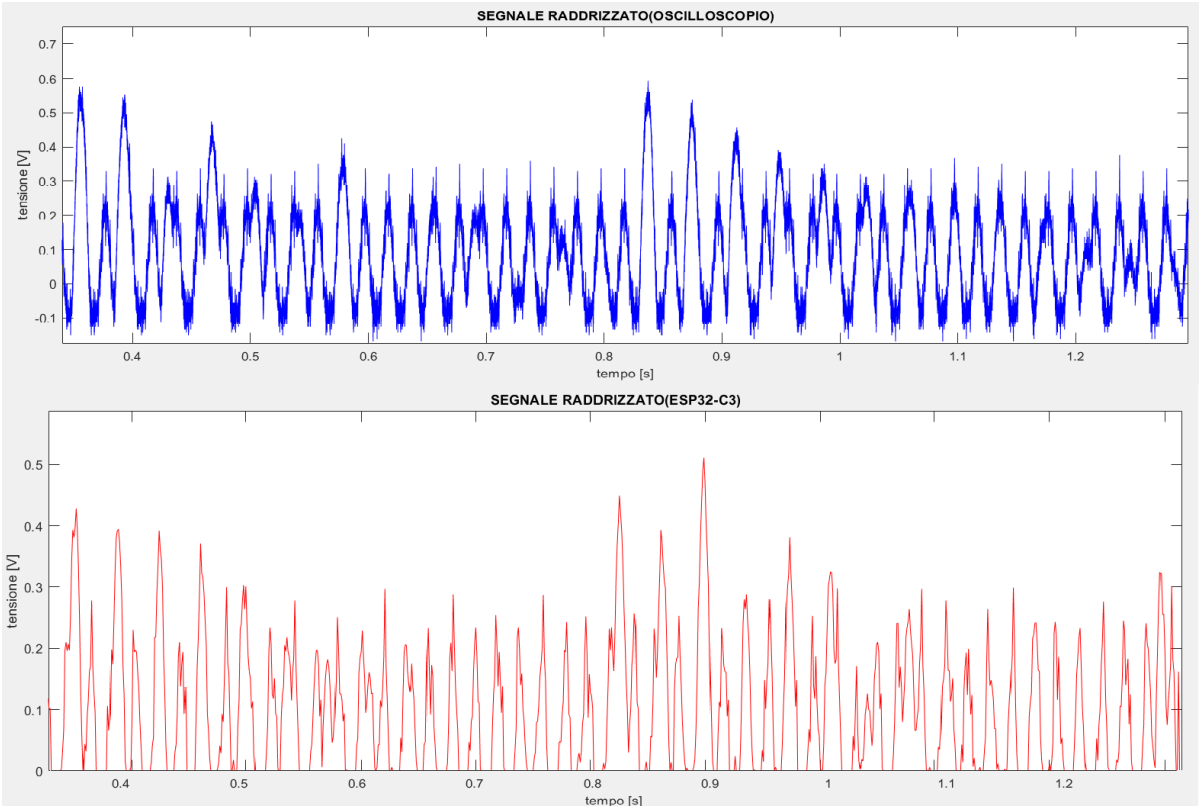

Figura 28 : Confronto andamento dello stesso segnale raddrizzato, misurato dall'oscilloscopio (grafico superiore) e misurato dall'ADC del microcontrollore (grafico inferiore).

Si può osservare (Figura 28), che nella misurazione data dall'oscilloscopio, il raddrizzamento del segnale non è totale, avendo circa 100 mV negativi, mentre nella misurazione della scheda di sviluppo, non essendo realizzata per campionare valori di tensione negativi, questi campioni risultano tagliati.

Table 14: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter	Min	Typ	Max	Unit
C_{IN}	Pin capacitance	—	2	—	pF
V_{IH}	High-level input voltage	0.75 × VDD[1]	—	VDD[1]+ 0.3	V
V_{IL}	Low-level input voltage	–0.3	—	0.25 × VDD[1]	V

Figura 29: Tabella che mostra le caratteristiche delle tensioni in ingresso all'ADC(fonte: ESP32-C3 datasheet, esp32-c3_datasheet_en.pdf (espressif.com)).

Come descritto precedentemente, il convertitore analogico-digitale del microcontrollore non prevede ai suoi ingressi tensioni negative, che possono causare danneggiamento della periferica, ma questo valore non risulta problematico, in quanto l'ESP32-C3 può supportare senza danni tensioni negative in ingresso fino a circa 300 mV(*figura 29*).

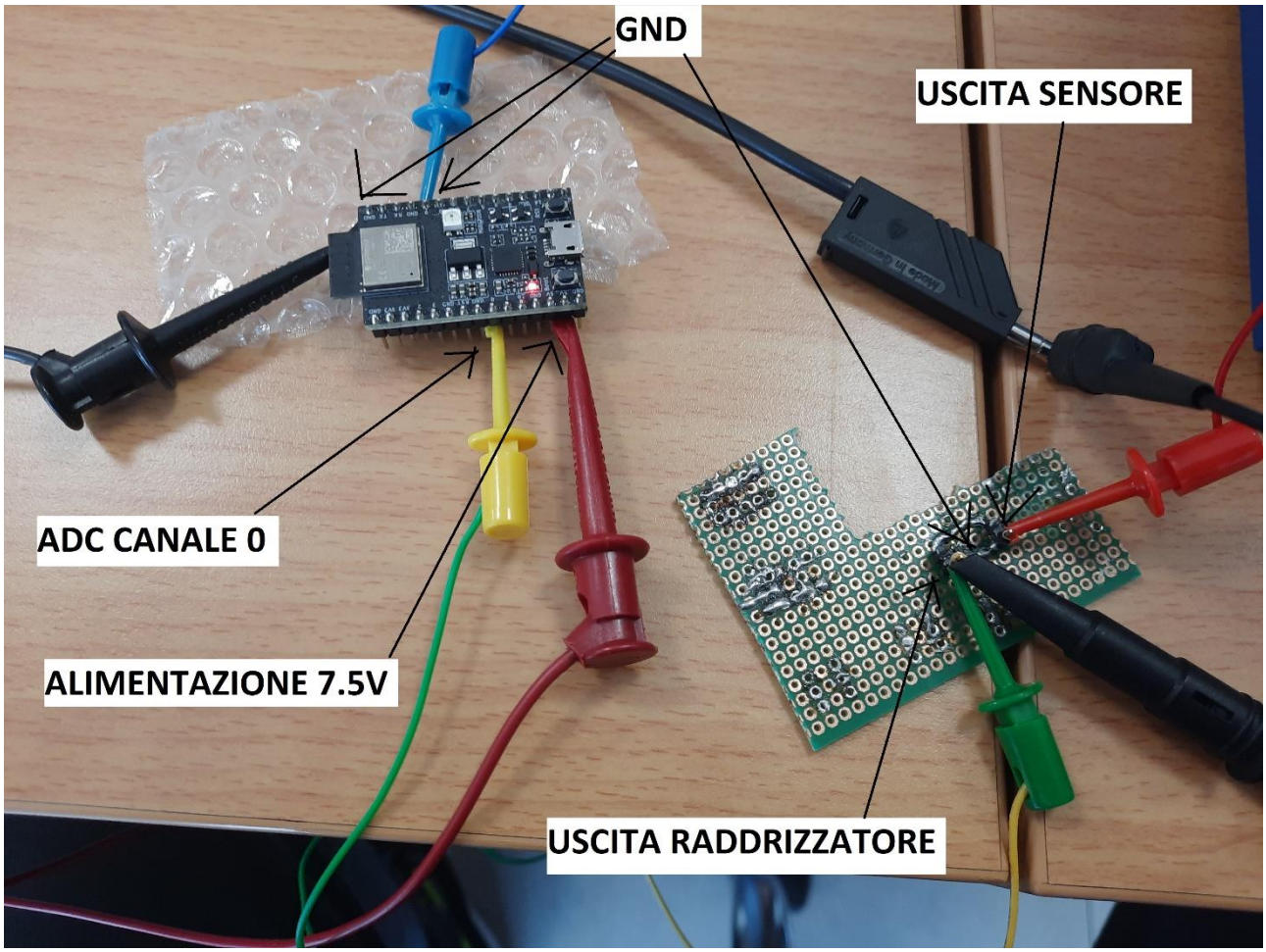

Figura 30 : Struttura collegamenti tra la scheda di sviluppo e la scheda con il diodo raddrizzatore.

Il diodo raddrizzatore SMD è stato saldato su una scheda millefori, fornendo due connettori per il collegamento di ingresso del segnale proveniente dal sensore piezoelettrico e due connettori per il collegamento della massa e del canale 0 del convertitore analogico-digitale. I vari test, compresi quelli effettuati precedentemente, sono avvenuti fornendo un'alimentazione di 7.5 V alla scheda. Si è provato a effettuare dei test anche con diversi valori di alimentazione come 5.5 V o 9 V, ma non si sono riscontrate variazioni significative sulle prestazioni del convertitore analogico-digitale.

Dopo aver effettuato questi test, si è deciso di verificare anche la qualità delle misurazioni dell'ADC2 in maniera sperimentale. Come menzionato precedentemente, l'ADC2 non dovrebbe essere utilizzato per campionamenti a una certa frequenza e per un certo numero di campioni, per via del suo utilizzo a basso livello per le funzionalità Wi-Fi e Bluetooth.

Si è comunque provato a modificare la pagina web, al fine di permettere momentaneamente la misurazione anche da parte dell'ADC2.

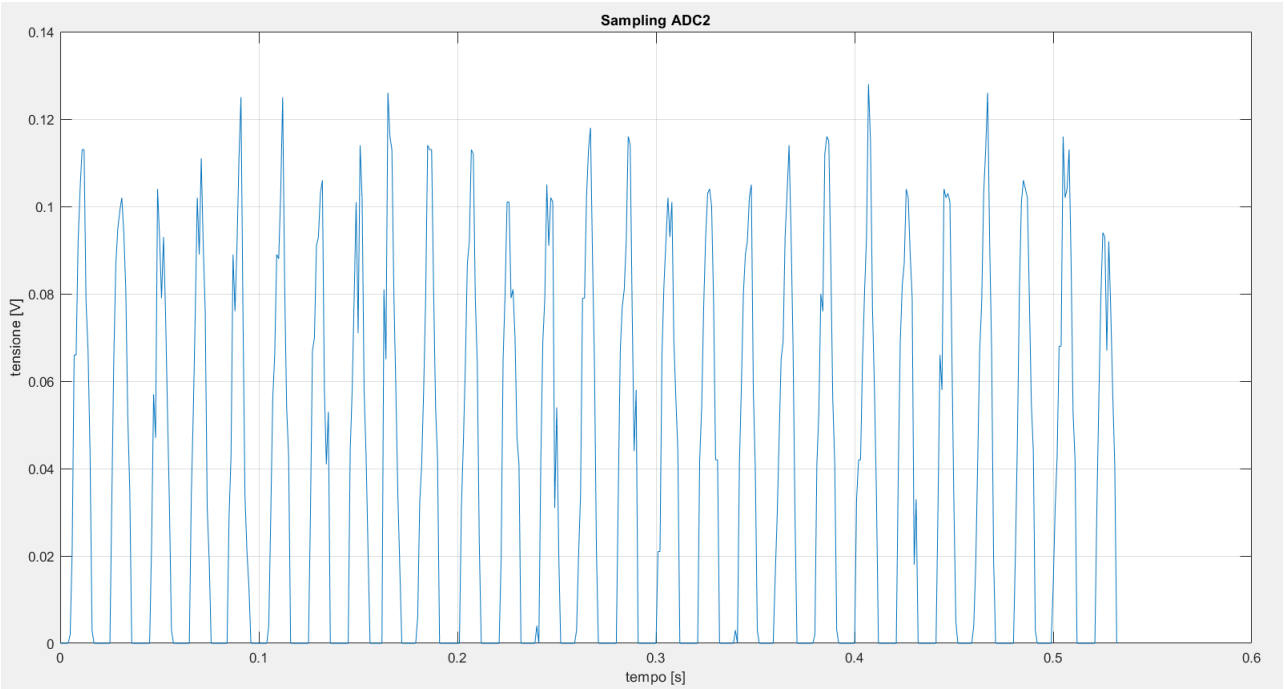

Figura 31: Misurazione campioni da parte dell'ADC2. Intorno ai 500 ms si è avuta una disconnessione dall'access point della scheda,

Il risultato non è stato particolarmente soddisfacente, infatti, come osservato anche nell'elaborazione dati su matlab (figura 31), si è avuta una disconnessione dell'utente

client dalla scheda access point intorno ai 500 millisecondi, proprio durante la scrittura dei campioni sullo SPIFFS. Infatti, andando a leggere successivamente i dati campionati sullo SPIFFS, si è osservata la presenza di soltanto 530 campioni circa, contro i 13000 previsti.

Questo conferma il fatto che, nonostante l'ADC2 possa essere utilizzato per effettuare campionamenti, il suo uso non risulta compatibile con l'utilizzo del modulo Wi-Fi al tempo stesso.

Per quanto riguarda i pochi campioni ottenuti, si è inoltre notato che i valori campionati presentano valori in tensione estremamente più bassi di quelli previsti, rendendo per niente validi neanche i pochi valori ottenuti nel breve intervallo di 500 ms.

3.4 Consumo energetico scheda sleep mode

Sono stati effettuati dei test sulla funzionalità di risparmio energetico "DEEP SLEEP", in particolare effettuando un confronto con le condizioni di funzionamento normale.

Per effettuare questi test, si è utilizzato Joulescope, un sistema di misurazione di consumo energetico ad alta precisione, progettato per l'analisi e il monitoraggio dei consumi di energia dei dispositivi elettronici. Il sistema consiste in un dispositivo hardware, più un software dedicato, che permette di visualizzare con alta precisione i valori di corrente, tensione e potenza, tramite grafici in tempo reale.

Si sono effettuati questi test fornendo un'alimentazione lievemente superiore rispetto ai test precedenti (9 V tramite alimentatore), al fine di testare il tutto in condizioni più critiche.

Figura 32: Dispositivo Joulescope.

Si è testato inizialmente il sistema in condizioni di normale utilizzo, assicurandosi che la scheda non finisse nello stato di sleep. Si è osservato (*figura 33*) un valore medio di circa 100 mA di assorbimento, con dei picchi anomali di corrente di circa 300 mA ogni 100 ms. Questi picchi sono probabilmente dovuti ai "WiFi beacon", pacchetti di gestione trasmessi periodicamente dagli access point che contengono informazioni come l'identificatore SSID, la tipologia di rete, i parametri di sicurezza e altro ancora. Risultano fondamentali per fornire informazioni ai dispositivi client che desiderano connettersi alla rete.

Consultando la documentazione, in particolare quella riferita al WiFi driver dell'ESP32-C3, si è infatti confermato che l'intervallo di beacon (il tempo tra l'invio di due pacchetti beacon consecutivi) è impostato di default a 100 ms, ma può essere impostato fino a un massimo di 60 s.

Figura 33: Analisi del consumo della scheda tramite il software Joulescope. Si osservano i picchi di circa 300 mA che si verificano periodicamente ogni 100 ms dovuti all'invio dei pacchetti beacon.

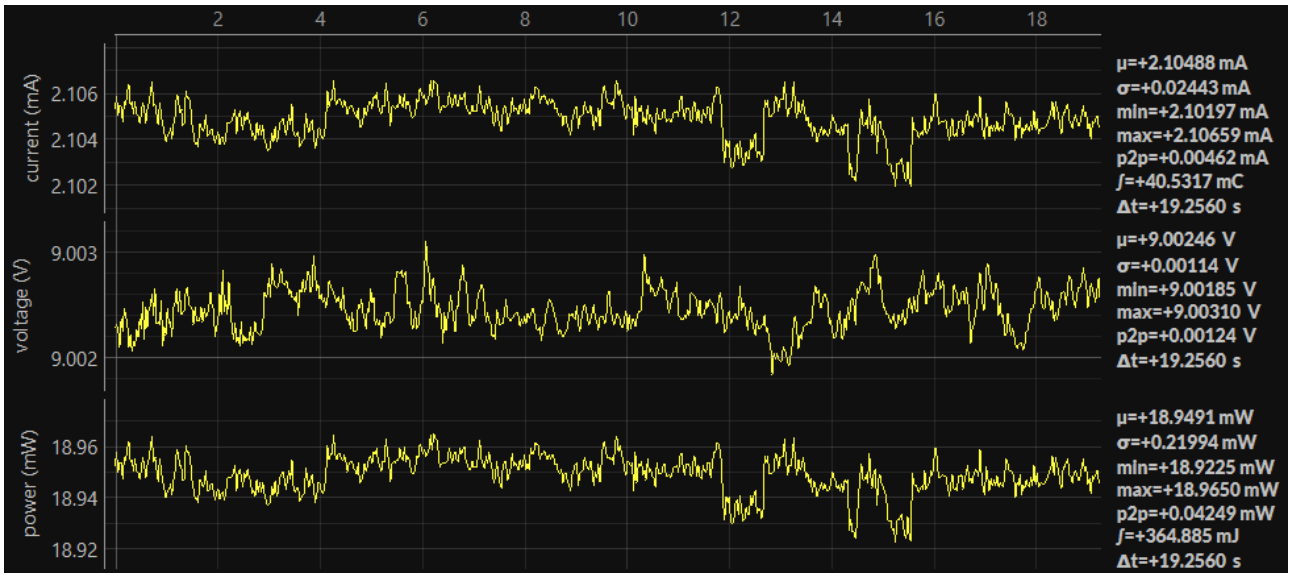

Figura 34 : Analisi del consumo della scheda nello stato di deep sleep. Si osserva un assorbimento di corrente di circa 2 mA.

In queste sperimentazioni non si è variato l'intervallo di beacon, ma questa modifica potrebbe notevolmente ridurre il consumo, in quanto diminuendo la frequenza con cui si presentano questi picchi si diminuisce il consumo totale.

Dopodiché (*figura 34*), si è effettuato il test della scheda nello stato di Deep Sleep, dove si è riscontrato un assorbimento di corrente di circa 2 mA piuttosto stabili. Questo valore misurato di circa 2 mA è, però, influenzato da quello che è il tempo in cui la scheda

viene portata in modalità di sleep. In questo caso si è testato un tempo di sleep di 30 s (come menzionato precedentemente per l'implementazione della modalità di risparmio), ma provando a portare la scheda in modalità deep sleep per periodi più lunghi come cinque o dieci minuti, si è notato dopo qualche minuto un assorbimento ancora inferiore, di circa 900 uA.

È necessario specificare che nella documentazione ufficiale del microcontrollore, la funzionalità di deep sleep descrive assorbimenti in corrente estremamente più ridotti, intorno ai 5 uA. Questa differenza può essere dovuta da vari fattori, come, ad esempio, dall' aver testato il sistema con tempi di sleep di soli trenta secondi, quindi particolarmente ridotti, oppure dal fatto che la documentazione ufficiale è riferita al solo modulo ESP32-C3 e non alla scheda di sviluppo ESP32-C3 DevKit-M1 che lo contiene, la quale risulta provvista di più periferiche e quindi si ipotizza che i suoi consumi siano maggiori.

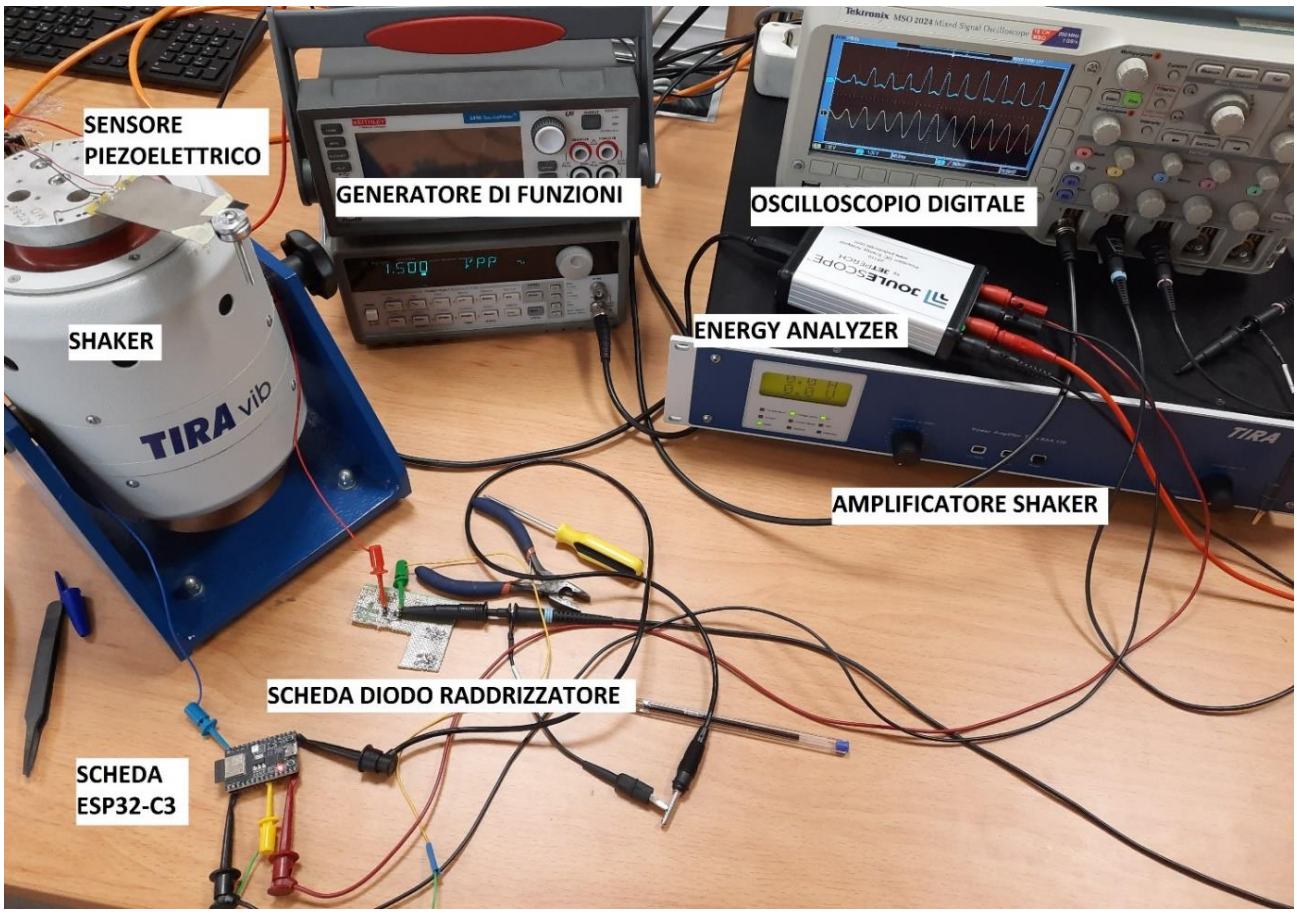

Figura 35: Strumentazione utilizzata per i vari test. Nell'inquadratura non è presente l'alimentatore DC.

4 Possibili miglioramenti

Nel progetto di questo sistema embedded per applicazioni di telemetria e acquisizione dati wireless, si è cercato di implementare varie soluzioni offerte dalla scheda a microcontrollore al fine di ottimizzare il sistema per questa tipologia di applicazioni. Tramite lo studio del sistema si sono trovate ulteriori migliorie al dispositivo finale che però non sono state implementate per motivi di tempo.

Tra queste ci potrebbero essere:

- **UTILIZZO PERIFERICA DMA PER LETTURE CONTINUE DELL'ADC:** il sistema DMA (Direct Memory Access) è una periferica integrata nel microcontrollore ESP32-C3, che consente al processore di trasferire dati direttamente tra una periferica come l'ADC e la memoria DMA. Questo approccio risulta particolarmente interessante perché permette operazioni di lettura e scrittura in background senza coinvolgere direttamente il core della CPU.

  Inoltre, tramite alcune funzioni rese disponibili dalla Espressif, utilizzando la libreria "adc_continuous.h", si rende possibile impostare via software il numero di campioni ADC da misurare prima di essere salvati nella memoria DMA.

  Inoltre, il DMA permette non soltanto la comunicazione periferica-memoria, ma anche la comunicazione memoria-periferica in maniera bidirezionale.

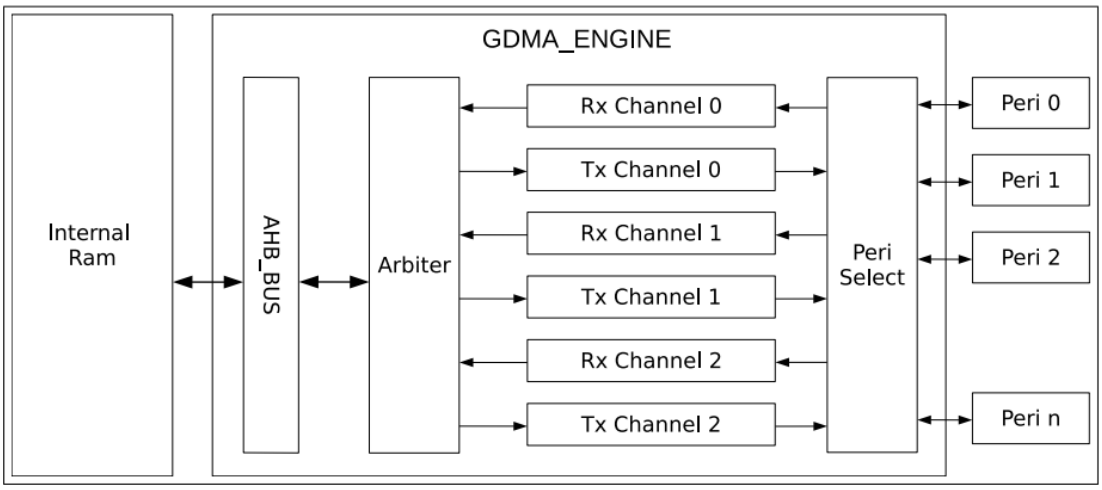

Figure 2-2. GDMA Engine Architecture

Figura 36: Struttura base del General Direct Memory Access (GDMA) presente nel microcontrollore(fonte: Documentazione tecnica ESP32-C3, esp32-c3_technical_reference_manual_en.pdf (espressif.com)).

- **AUMENTO INTERVALLO DI BEACON:**

  Come menzionato precedentemente l'invio dei pacchetti beacon è uno delle principali cause del consumo energetico in quella modalità in qui il dispositivo non si trova ancora in sleep mode. Aumentando questo tempo si rende il consumo del sistema ancora più ridotto. Purtroppo, non si è riuscito ad aumentare questo intervallo né tramite Api fornite dall'Espressif, ne tramite la programmazione via registri del microcontrollore.

- **ANALISI DETTAGLIATA DEL TRAFFICO DI RETE (PACKET SNIFFER):**

  Le varie schede ESP32 dispongono della "Wi-Fi promiscuous mode", una funzionalità che permette di monitorare e analizzare il traffico di rete in modo dettagliato.

  In questa modalità, la scheda può captare tutti i pacchetti Wi-Fi trasmessi nell'area circostante e ottenerne informazioni specifiche.

  Questa funzionalità può essere utilizzata per monitorare la comunicazione tra la scheda e una station a essa collegata, risultando particolarmente utile per scopi di

debugging, analisi delle prestazioni di rete e per tutte quelle applicazioni in cui risulta necessaria una comprensione dettagliata del traffico di rete Wi-Fi.

- **CARICAMENTO FIRMWARE DA REMOTO(OTA):**

  Una funzionalità particolarmente interessante per questo tipo di applicazioni è la funzionalità OTA(Over-the-Air), che consente di caricare e installare firmware su una qualsiasi scheda della serie ESP-32, senza dover collegare fisicamente la scheda al computer di sviluppo.

  Il funzionamento si basa sulla creazione di un server OTA, il quale consiste in un semplice web server, in cui l'utente, provvisto di PC, si collegherà e inserirà il firmware che desidera caricare sulla scheda.

  La scheda Espressif, dopodiché, invierà anch'essa una richiesta al server OTA, il quale si occuperà di caricarci sopra il nuovo firmware.

  Il sistema OTA risulta quindi una soluzione particolarmente rapida ed efficiente che elimina la necessità di un cablaggio diretto, rendendo il sistema embedded completamente realizzabile da remoto anche in fase di sviluppo.

Conclusioni

In conclusione, il progetto di sviluppo di un sistema embedded per telemetria e acquisizione dati wireless utilizzando la scheda ESP32-C3 DevKit-M1 si è dimostrata una scelta valida, essendo il chip estremamente versatile e disponendo di una buona potenza di calcolo, connettività wireless e funzionalità di acquisizione di dati analogici.

In particolare, si è apprezzata la sua capacità di gestire il risparmio energetico del sistema, per cui è stato possibile minimizzare il consumo quando il sistema non era attivamente impegnato nell'acquisizione dati o nella trasmissione wireless. Questo ha reso il sistema adatto per applicazioni a batteria o in situazioni in cui il consumo energetico è una considerazione critica.

Un vantaggio a favore del microcontrollore scelto, è stato senz'altro la forte presenza di un ambiente open source, il quale ha permesso di usufruire in maniera completamente trasparente di librerie realizzate e rese disponibili, oltre che alla grande quantità di informazioni dettagliate sull'hardware presenti nella documentazione, che ha permesso di studiare a fondo il dispositivo fino alle sue più piccole funzionalità.

Figure

Bibliografia

1: ESP32-C3 technical reference manual

https://www.espressif.com/sites/default/files/documentation/esp32-c3_technical_reference_manual_en.pdf

2: ESP32-C3 datasheet

https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf

3: ESP3-C3 Espressif documentation

https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/index.html

4. Asynchronous web server library “EspAsyncWebServer”

https://github.com/me-no-dev/ESPAsyncWebServer

5. Espressif library for ADC calibration “esp_adc_cal”

https://github.com/espressif/esp-idf/blob/8b94183c9c/components/esp_adc_cal/include/esp_adc_cal.h

6. WiFi driver library “WiFi”

https://github.com/arduino-libraries/WiFi

7. SPIFFS management library “SPIFFS”

https://github.com/espressif/esp-idf/tree/cf7e743a9b/examples/storage/spiffs

8. ESP32-C3 Devkit-M1 datasheet

https://www.espressif.com/sites/default/files/documentation/esp32-c3-mini-1_datasheet_en.pdf

9. Schottky diode datasheet

https://www.mouser.it/datasheet/2/389/bar42-1849103.pdf

10. Espressif documentation for arduino framework

https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html

11. Development board circuit schematic

https://dl.espressif.com/dl/schematics/SCH_ESP32-C3-DEVKITM-1_V1_20200915A.pdf

12. ESP32-C3 hardware design guidelines

https://www.espressif.com/sites/default/files/documentation/esp32-c3_hardware_design_guidelines_en.pdf

13. ESP32-C3 ADC DMA library

https://github.com/espressif/esp-idf/tree/cf7e743a9b/examples/peripherals/adc/continuous_read