Università degli Studi di Bergamo

SCUOLA DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto di Sensori

Activity tracker con Arduino Nano 33 BLE Sense

Prof. Gianluca Traversi

Dott. Paolo Lazzaroni

Candidati

Davide Salvetti
Matricola 1057596

Martina Fanton
Matricola 1059640

Matteo Verzeroli
Matricola 1057926

Anno Accademico 2022–2023

1 Introduzione

Lo scopo di questo progetto consiste nella classificazione dell'attività fisica svolta da un individuo grazie a delle misure ottenute con la board Arduino Nano 33 BLE Sense posizionata sopra la caviglia del soggetto. Dopo essersi connessi alla board tramite BLE e a seconda del firmware caricato sull'Arduino, è possibile utilizzare l'applicazione in due modalità:

- modalità acquisizione dati: l'applicazione riceve e salva i dati inviati dalla piattaforma;
- modalità activity tracker: l'applicazione visualizza la predizione dell'attività che il soggetto sta svolgendo (camminata, cyclette, salto con la corda oppure l'individuo è fermo).

Per questo progetto sono stati utilizzati i seguenti sensori presenti sulla board:

- accelerometro triassiale;
- giroscopio triassiale;
- sensore di temperatura e di umidità.

In realtà, i dati relativi alla temperatura e all'umidità non sono stati utilizzati poiché non si sono mostrati significativi per l'obiettivo di questo progetto. Inoltre, è stato necessario modificare la libreria che gestisce la comunicazione con la IMU (file `LSM9DS1.cpp`) per cambiare il fondo scala dell'accelerometro (che di default è impostato a 4g):

- accelerometro: fondo scala 8g, frequenza di campionamento 119 Hz;
- giroscopio: fondo scala 2000 dps, frequenza di campionamento 119 Hz.

Per l'attività di classificazione dell'attività si è scelto di seguire un approccio *black-box*, utilizzando una rete neurale opportunamente allenata con i dati raccolti dalla board. La rete neurale, sviluppata tramite il framework Edge Impulse, è stata poi installata sull'Arduino. Utilizzando quindi un modello di *edge computing* è stato poi possibile realizzare un'applicazione, sviluppata grazie al framework Qt, che si occupa solamente della visualizzazione dei dati e del risultato della predizione dell'attività ricevuta dall'Arduino. In questo modo, si è potuto mantenere una frequenza di campionamento dei dati relativamente alta (circa 66 Hz) senza sovraccaricare la comunicazione BLE.

2 Acquisizione dei dati

Figura 1: Arduino posizionato sulla gamba del soggetto con relativo sistema di riferimento.

Per poter acquisire i dati con Arduino, è stato necessario sviluppare un firmware apposito che consentisse di inviare le misure utilizzando la tecnologia Bluetooth Low Energy (BLE), in cui l'Arduino viene utilizzato nel ruolo di *peripheral* mentre l'applicazione assume il ruolo di *central*. La board di Arduino espone un servizio con una caratteristica su cui vengono scritti i dati delle misure secondo il seguente formato:

`"accx,accy,accz,gyrox,gyroy,gyroz,temp,hr,timestamp"`.

Il central viene notificato ogni volta che il *peripheral* aggiorna la caratteristica e, dopo averne letto il contenuto, salva le misure in un file .csv che contiene le seguenti colonne: *accx*, *accy*, *accz*, *gyrox*, *gyroy*, *gyroz*, *temp*, *hr* e *timestamp*. Sull'applicazione è possibile visualizzare i dati ricevuti in tempo reale su un grafico (Fig.2). Inizialmente, si è cercato di utilizzare una frequenza di campionamento dei dati pari a 100 Hz. Tuttavia, si è verificato che il flusso di dati non era sostenibile dal BLE. Tramite delle prove si è poi stabilito che una frequenza di campionamento dei dati sostenibile era di circa 66 Hz (un campione ogni 15 ms). Per assicurare una buona precisione nella frequenza di campionamento,

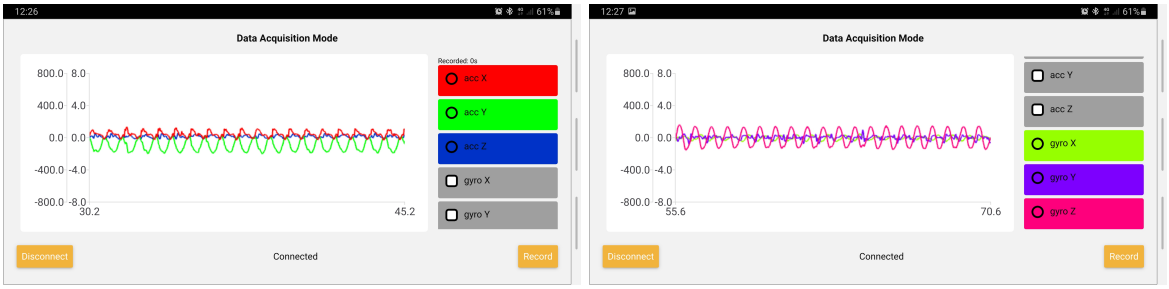

Figura 2: Schermata di acquisizione dati dell'applicazione: l'immagine a sinistra mostra i segnali dell'accelerometro mentre quella a destra mostra i segnali del giroscopio durante l'attività cyclette.

si è scelto di utilizzare il meccanismo degli interrupt (Interrupt Service Routine), in cui una funzione di callback viene chiamata ogni 15 ms. Infatti, non è possibile utilizzare la funzione *delay* poiché non garantisce intervalli regolari di acquisizione. Inoltre, ciò permette di realizzare un'applicazione *pseudo* concorrente, in cui la funzione *loop* principale si occupa dell'invio dei dati tramite BLE, mentre un'altra funzione, attivata da un interrupt, si occupa di acquisire i dati. Tuttavia, l'utilizzo degli interrupt non si è mostrato essere sufficiente a garantire una frequenza di campionamento accurata in quanto si è visto che il tempo necessario all'invio dei dati tramite BLE non è costante e ciò portava alla perdita di alcuni dati. Per questo motivo, è stato introdotto un buffer circolare tramite la libreria `https://github.com/rlogiacco/CircularBuffer`. In questo modo, se la comunicazione subisce un ritardo non si ha perdita di dati. Di seguito è stato riportato un esempio dei dati dell'accelerometro e giroscopio ottenuti durante una camminata di un soggetto (Fig.3).

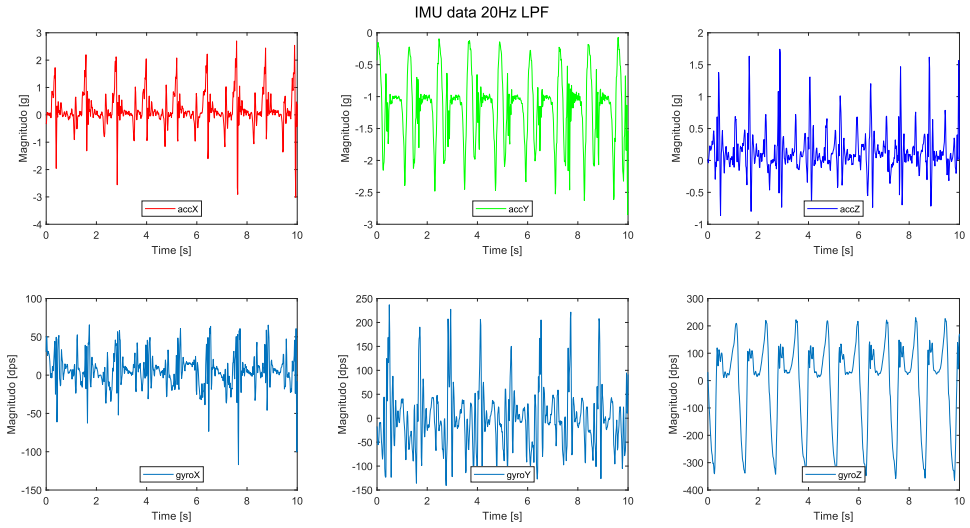

Figura 3: Dati acquisiti dall'accelerometro e giroscopio durante una camminata. I dati sono campionati ogni 15 ms e sono stati filtrati con un filtro passa basso con frequenza di taglio di 20 Hz.

Nella figura 4 è stato riportato un confronto tra gli intervalli di tempo tra due acquisizioni successive su una misura di 76 s eseguita su un soggetto durante una camminata con e senza buffer circolare. Nella figura 4a) è stato utilizzato il buffer circolare e il tempo medio risultante è di 15.0057 ms con una deviazione standard di 0.4405 ms. Senza buffer circolare (Fig.4b)) è stata ottenuta una media di circa 16.1630 ms con una deviazione standard di 6 ms: infatti, è possibile vedere nell'immagine che si hanno molti più outlier. Quindi l'utilizzo del buffer rende più robusto il sistema in caso di eventuali rallentamenti nell'invio dei dati. Si tenga in considerazione che l'istante di tempo in cui si eseguiva l'acquisizione è stato misurato con la funzione *millis()*, che ha errore di ± 1 ms.

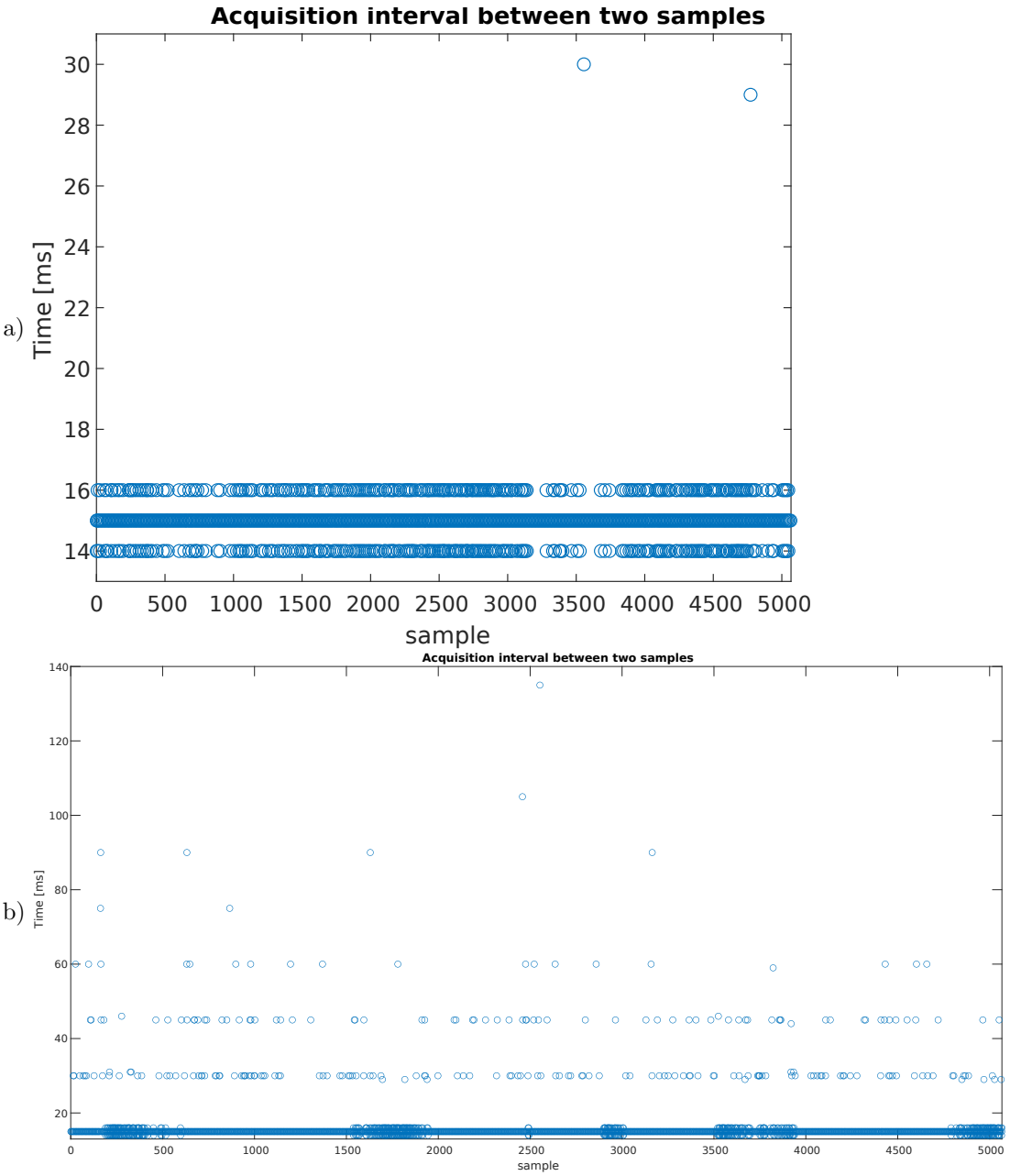

Figura 4: Tempo tra due campioni successivi su un'acquisizione di 76 s: a) con buffer circolare, b) senza buffer circolare.

3 Classificazione delle attività

Nella figura 3 è rappresentato un esempio di dati raccolti durante la camminata di un soggetto. Osservando i dati dell'accelerometro è possibile notare dei picchi che corrispondono ai movimenti dei passi del soggetto. In particolare, per come è stato posizionato l'Arduino (Fig.3), è possibile notare il contributo della forza di gravità sull'asse y che causa un offset di circa 1g nelle misure e i picchi corrispondono a quando il piede entra in contatto con il terreno. Invece, per quanto riguarda le misure del giroscopio, è interessante notare lungo l'asse z il movimento di rotazione del piede attorno alla caviglia durante la camminata.

Per realizzare la classificazione dei dati acquisiti dai sensori di Arduino e poi per visualizzarne la previsione sull'applicazione è stato sviluppato un ulteriore firmware al cui interno sono stati utilizzati tre thread differenti, analizzati più in dettaglio nella sezione 3.2, che si occupano di ricavare i dati dalla IMU, effettuare la predizione ed inviare il risultato tramite BLE all'applicazione. In generale, il funzionamento del firmware prevede che inizialmente venga riempito un buffer circolare con i dati provenienti dai sensori. Successivamente, il buffer viene trasformato in un elenco di features tramite un DSP. Le features raccolte vengono poi passate al classificatore, realizzato grazie al framework Edge Impulse, che si occupa della classificazione. Infine, il risultato della classificazione viene trasmesso tramite BLE all'applicazione che mostra a video l'immagine relativa all'attività fisica identificata. Le schermate che possono essere visualizzate sull'applicazione si distinguono in base all'attività fisica rilevata come mostrato nella figura 5. Nei momenti subito dopo la connessione del dispositivo al BLE oppure nei momenti in cui non si riesce a classificare l'attività in corso sull'applicazione viene visualizzata la scritta *"Unknown"*, che indica che il sistema non è in grado di riconosce l'attività.

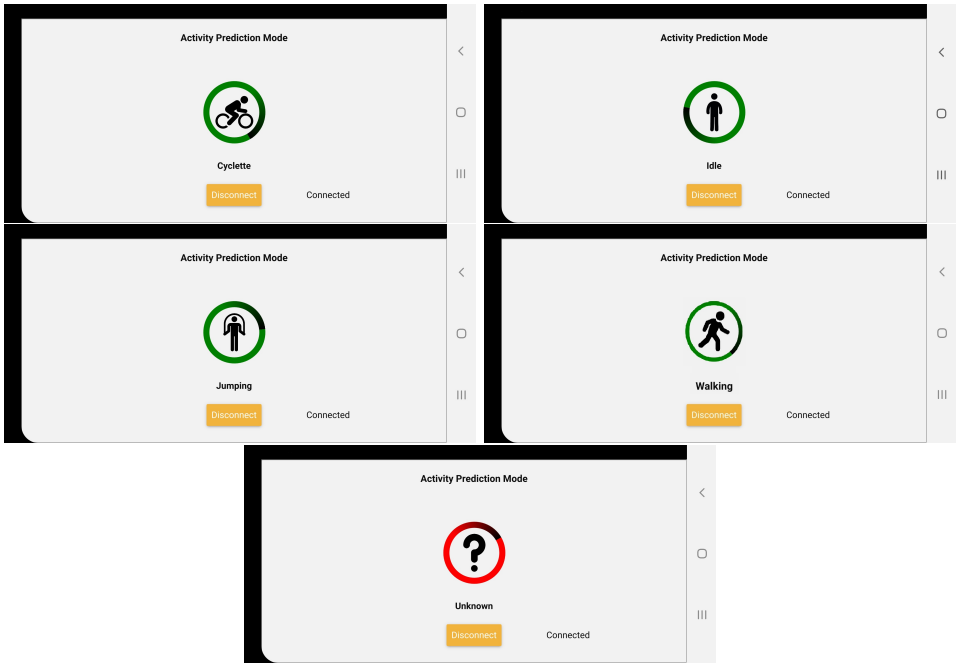

Figura 5: Schermate per le possibili attività: cyclette, fermo, salto della corda, camminata e attività non riconosciuta.

3.1 Edge Impulse

Edge Impulse è una piattaforma online per lo sviluppo di algoritmi di machine learning per *edge devices*. Tramite Edge Impulse è possibile creare un modello da poter caricare sul dispositivo finale

seguendo quattro semplici passi: acquisizione dei dati, design del modello, test del modello e deploy del modello.

Per prima cosa sono stati acquisiti i dati delle quattro categorie di attività fisica da classificare (idle, walking, jumping e cycling). Per raccogliere i dati, è stato utilizzato il firmware precedentemente descritto, misurando attività per circa 70 s. Essendo un modello di *machine learning*, maggiore è il numero dei dati, maggiore è la qualità del modello stimato. I dati acquisiti sono quindi stati caricati sulla piattaforma Edge Impulse dove sono stati divisi in *training set* e *test set* (Fig.6). Edge Impulse suggerisce di dividere i dati di train e di test con una percentuale rispettivamente di 80% e 20%.

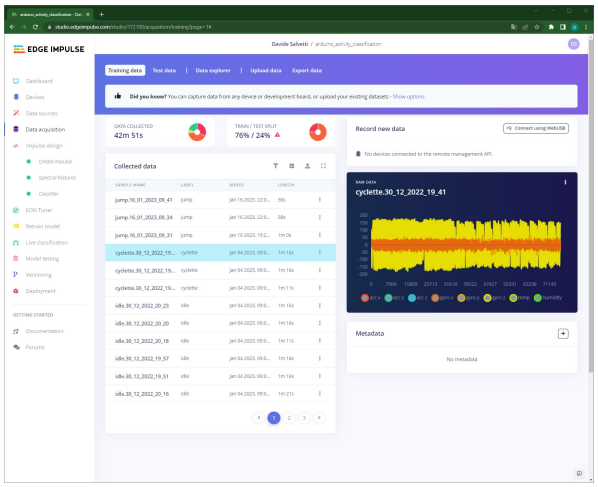

Figura 6: Pagina di Edge Impulse per l'inserimento di nuovi campioni.

Successivamente, è stato creato un *impulse*: questa fase prevede la selezione della larghezza della finestra dei dati che vengono poi processati, di un blocco di processamento dei dati e di un blocco di learning (figura 7). I campioni vengono analizzati su finestre di 15 secondi con una *slinding window* di 10 secondi. La frequenza di campionamento viene ricavata a partire dai dati grazie al campo `timestamp` ed è pari circa a 66 Hz. Per il processamento dei dati è stato scelto un DSP che analizza le features spettrali dei segnali provenienti da accelerometro e giroscopio. Come anticipato in precedenza, non

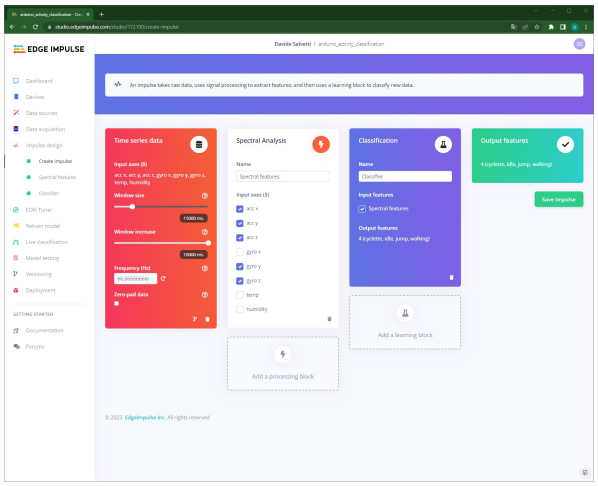

Figura 7: Schermata di Edge Impulse per la creazione di un nuovo *impulse* tramite la scelta di ciascun blocco.

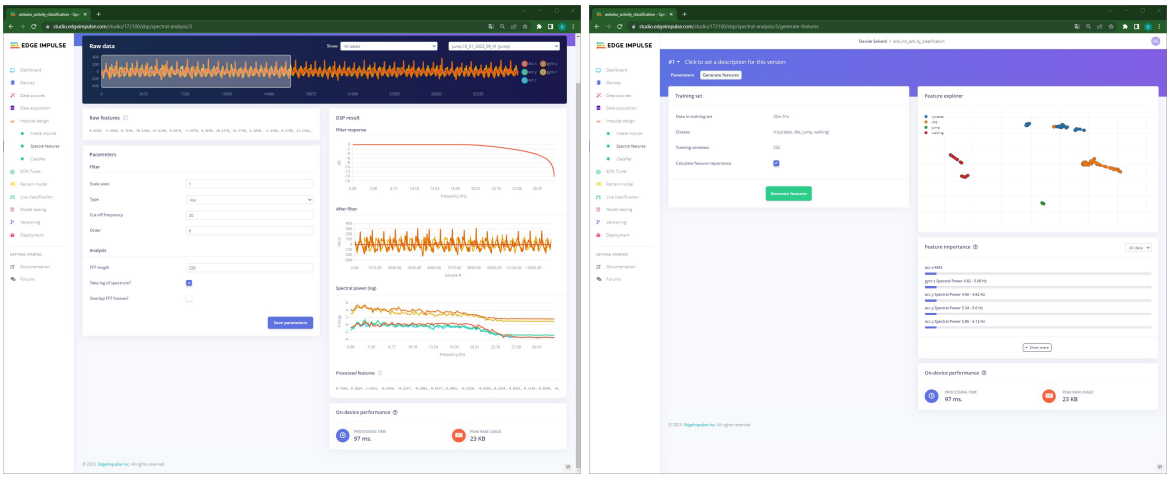

Figura 8: A sinistra, schermata di Edge Impulse per il settaggio dei parametri per l'estrazione delle features spettrali. A destra, schermata di Edge Impulse per la visualizzazione dell'importanza di ciascuna feature estratta per la distinzione delle varie classi. Inoltre, è possibile osservare le performance stimate sull'Arduino.

sono stati inclusi i campi temperatura e umidità e nemmeno il giroscopio sull'asse x dal momento che non fornisce dati significativi. Per l'analisi spettrale è stato scelto di introdurre un filtro passa-basso con frequenza di taglio a 20 Hz del sesto ordine ed è stato analizzato il logaritmo della densità spettrale di potenza (figura 8). Una volta definiti i parametri dell'analisi spettrale, il modello estrae le features dai dati di test e viene visualizzato un grafico con i dataset divisi per "somiglianza" delle features; inoltre mostra l'importanza delle features estratte per comparare le diverse classi: alcune delle features più importanti sono il valore quadratico medio dell'accelerometro sull'asse x, la densità spettrale di potenza del giroscopio sull'asse z e la densità spettrale di potenza dell'accelerometro sull'asse y (figura 8). Come blocco di apprendimento è stato scelto un classificatore. La rete neurale riceve in ingresso 400 features ricavate dal blocco DSP precedente e le analizza tramite due Dense Layer, composti rispettivamente da 32 e 16 neuroni, ed un Output Layer che fornisce in uscita la probabilità di appartenenza delle features in ingresso alle 4 classi. La predizione finale è la classe con probabilità maggiore. Una volta sviluppato il modello è possibile testarlo sui dati di test (Fig.9).

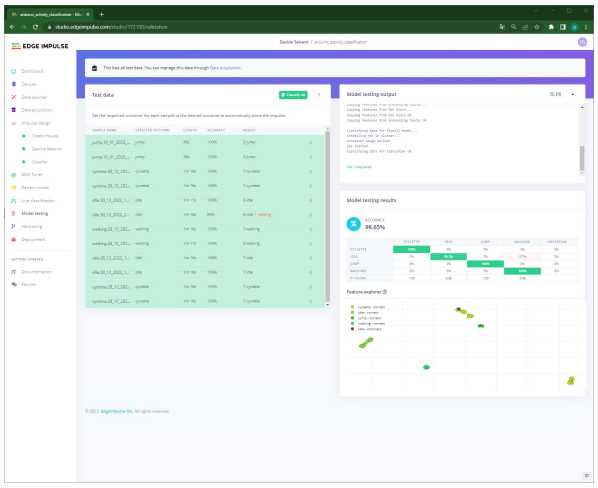

Figura 9: Schermata di Edge Impulse per il testing dei dati che mostra l'accuratezza del modello.

La fase finale è quella di deploy. Il modello sviluppato, comprensivo del blocco DSP per l'estrazione delle features, può essere scaricato dalla piattaforma Edge Impulse come una libreria di Arduino, la quale può essere caricata direttamente sulla board.

3.2 Arduino

Il funzionamento base del firmware caricato su Arduino per la classificazione dei dati è mostrato nella figura 10.

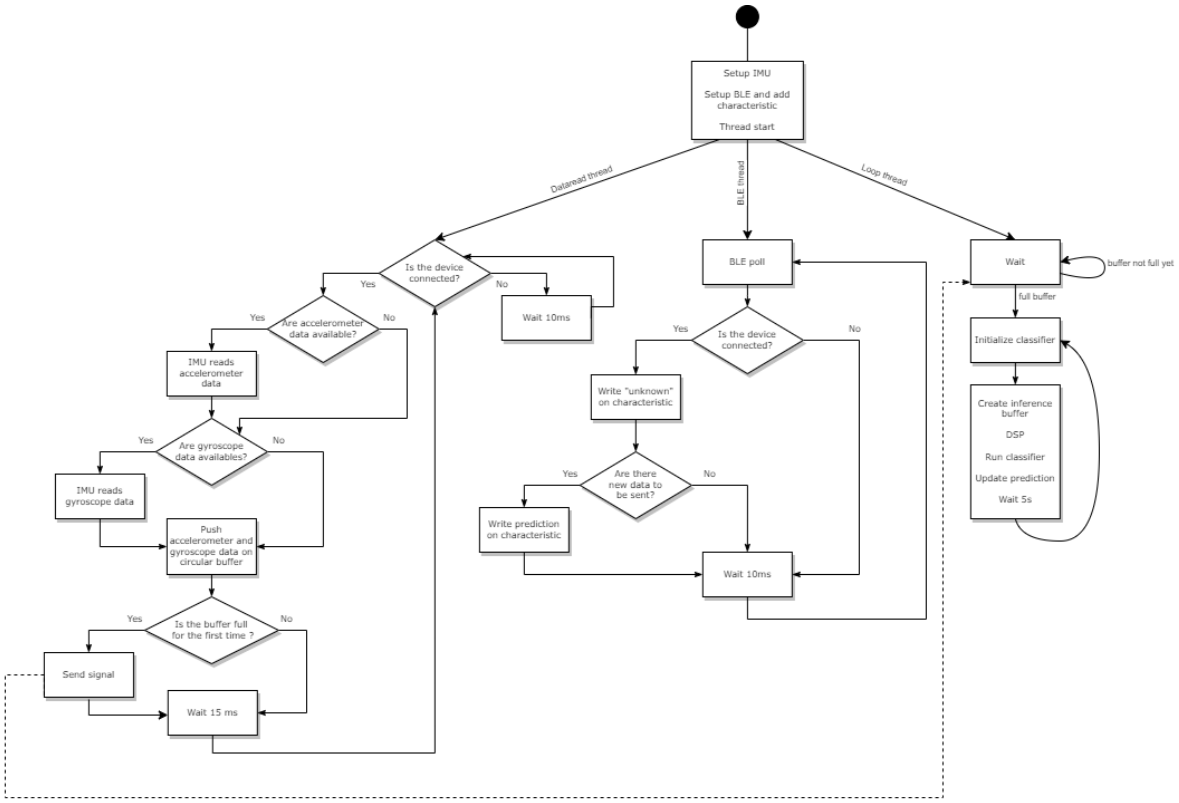

Figura 10: Macchina a stati del firmware di classificazione dei dati sensoriali.

Nel thread *loop*, la chiamata al classificatore corrisponde all'utilizzo delle librerie di Edge Impulse. Questa chiamata richiede come parametri in input il segnale, la variabile in cui salvare la previsione e un parametro che caratterizza l'inferenza. Per quanto riguarda i tempi di classificazione, il tempo necessario per passare da una classificazione alla successiva e quindi per cambiare la previsione è di circa 40-45 secondi. Questo tempo è dovuto al fatto che si deve svuotare il buffer dai dati precedenti e si deve identificare per sei volte la stessa attività prima di definirla come previsione. La previsione viene eseguita ogni 5 secondi e il buffer contiene i dati di 15 s di attività. Per diminuire il tempo necessario per classificare una attività è possibile aumentare la frequenza con cui si esegue l'inferenza.