



UNIVERSITÀ DEGLI STUDI DI GENOVA

CORSO DI STUDIO IN INGEGNERIA
ELETTRONICA E TECNOLOGIE
DELL'INFORMAZIONE

Tesi di laurea triennale

Marco Vercellino

Settembre 2023

**Progetto e realizzazione di un sistema embedded per la
raccolta dati di attività sportive**

**Design and implementation of an embedded system for
sports data collection.**

Relatore: Prof. Riccardo Berta

Correlatori: Dott. Matteo Fresta, Dott. Ali Dabbous

Sommario

L'obiettivo di questa tesi è quello di mostrare lo sviluppo di un sistema completo per la raccolta dei dati provenienti da sensori di una scheda elettronica, che saranno poi successivamente usati per addestrare un algoritmo di machine learning per riconoscere autonomamente alcune azioni dello sport del calcio. Nello specifico, si utilizza un Arduino Nano 33 Ble Sense per collezionare dal suo sensore IMU (Inertial Measurement Unit) le informazioni riguardanti accelerometro, giroscopio e magnetometro registrate durante i movimenti di tiro, passaggio e stop. I dati raccolti verranno inviati ad un dispositivo mobile connesso alla scheda tramite Bluetooth Low Energy, grazie ad una applicazione sviluppata con Flutter e infine verranno salvati sul cloud tramite Measurify, con l'aggiunta di un tag che identifichi il tipo di movimento registrato e il nome assegnato alla misurazione. Questi dati costituiscono così un dataset utile a rendere un algoritmo di machine learning capace di riconoscere autonomamente i movimenti su cui ha effettuato l'addestramento. La motivazione alla base del mio lavoro è riuscire a coniugare l'attuale tendenza nella programmazione informatica ad usare sempre più spesso gli "oggetti intelligenti" (smart objects) connessi ad Internet con l'attività sportiva, praticata quotidianamente da molte persone.

Indice

1	Introduzione	4
1.1	Flusso di lavoro	5
2	Metodi e strumenti utilizzati	6
2.1	Programmazione della scheda Arduino	6
2.2	Applicazione Flutter	7
2.3	Controllo con Postman e Google Colab	10
3	Sperimentazione e Risultati	15
3.1	Montaggio dell'Arduino su parastinco	15
3.2	Raccolta del dataset	16
4	Contributo personale e considerazioni conclusive	19
5	Riferimenti bibliografici	19
6	Ringraziamenti	20

1 Introduzione

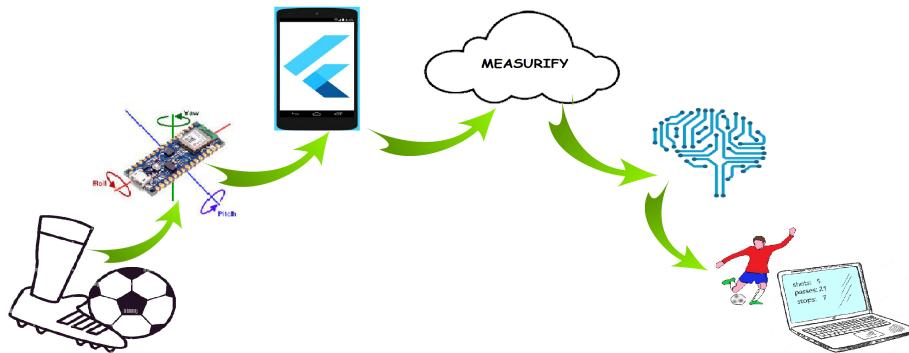
La progettazione end-to-end di sistemi basati su sensoristica ha applicazioni in qualsiasi ambito di ispirazione tecnologia e soprattutto nella nuova frontiera del "World of Things", termine che esprime il sempre più crescente coinvolgimento di oggetti nel mondo smart e interconnesso dell'Internet of Things (IoT). Nei ultimi anni però si è iniziata a porre molta attenzione sullo sviluppo del machine learning come agente di controllo nell'impiego di molti sensori. Nel contesto di questa tesi, abbiamo deciso di utilizzarlo per l'analisi e la classificazione di movimenti durante gli sport. Nello specifico, si vuole utilizzare un sensore che raccolga dati relativi ad accelerometro, magnetometro e giroscopio e da questi, tramite il machine learning, creare un modello in grado di classificare autonomamente i principali movimenti nello sport del calcio. Lo sviluppo da zero e nella sua interezza di un apparato di questo tipo richiede l'articolazione del lavoro in alcune macro-fasi, che coinvolgono diversi ambienti di sviluppo. Nel mio specifico ambito, si sono resi necessari i seguenti stadi:

- sviluppo dell'applicazione per dispositivo mobile per la ricezione, elaborazione dei dati provenienti dalla scheda elettronica ed il loro invio al cloud tramite Measurify. Il corretto invio dei dati può essere controllato con Postman e Google Colab;
- montaggio dell'Arduino su parastinco per la raccolta dei dati;
- raccolta del dataset per l'addestramento dell'algoritmo di machine learning.

L'algoritmo sopraccitato di machine learning non fa parte del mio progetto di tesi, in quanto è argomento di corso di laurea magistrale. Verrà invece utilizzato un algoritmo sviluppato da un ricercatore del labortorio Elios Lab.

1.1 Flusso di lavoro

Per impostare il mio progetto di tesi, ho deciso di gestire le varie fasi nel seguente modo: in primo luogo ho sviluppato su Visual Studio Code l'applicazione per dispositivo mobile. In seguito, adattato il parastinco ad ospitare la scheda elettronica, ho raccolto sul cloud i dati relativi a tiri, passaggi e stop. Questi dati hanno allenato l'algoritmo rendendolo capace di riconoscere autonomamente i tre movimenti.

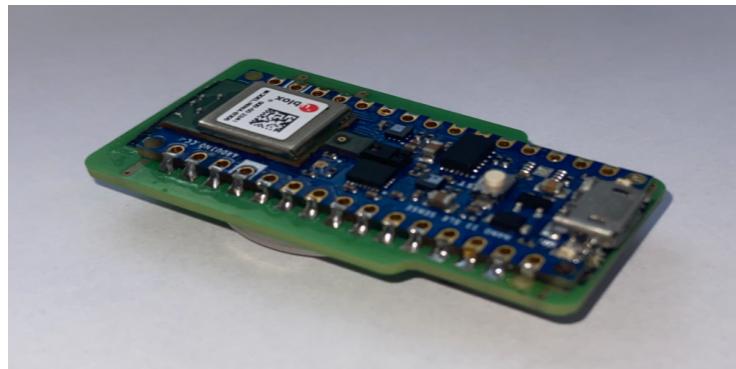
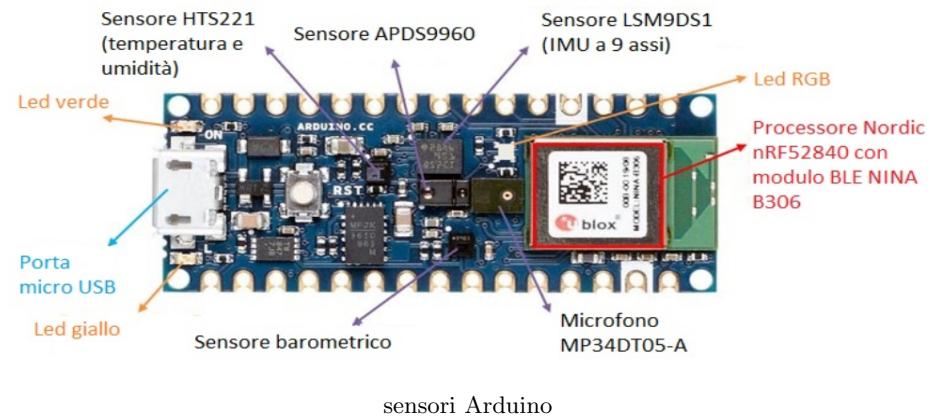


rappresentazione delle fasi del lavoro

2 Metodi e strumenti utilizzati

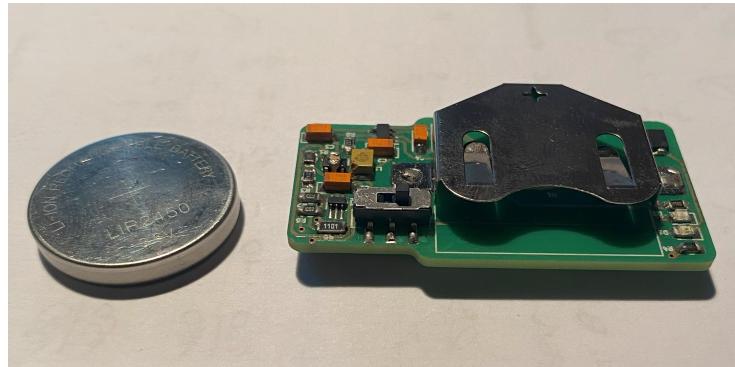
2.1 Programmazione della scheda Arduino

Per sviluppare il mio progetto di tesi sono state considerate diverse schede elettroniche. In primo luogo mi è stata fornita una Sparkfun Edge 2, ma dopo aver riscontrato alcuni problemi nelle librerie richieste per la sua programmazione, abbiamo deciso con il professore e i dottorandi di volgerci verso l'Arduino Nano 33 Ble Sense, in quanto rispecchiava meglio le nostre richieste, sia per la facilità di programmazione che per le dimensioni molto ridotte (misura infatti 18 x 45 mm).



Arduino

Tuttavia questa scheda per funzionare deve essere collegata ad una sorgente di corrente tramite cavo micro USB. Per i miei scopi, è necessario che sia dotata di alimentazione propria, quindi è stato costruito in laboratorio un circuito custom con batteria a bottone su cui montarla. In questo modo la scheda potrà funzionare all'interno del parastinco senza l'utilizzo di cavi che complicherebbero la raccolta dei dati.



batteria custom

Risolti i vincoli di alimentazione, sono passato alla programmazione su Arduino IDE. Il codice deve gestire la connessione bluetooth con il dispositivo mobile (lato scheda elettronica) e l'invio dei dati relativi ad accelerometro, giroscopio e magnetometro (tutti su tre assi). I dati da trasmettere sono quindi nove float, inseriti in un unico array. La comunicazione Bluetooth Low Energy ha dei forti limiti in termini di quantità di dati che possono essere inviati per pacchetto (20 bytes), quindi è stato scelto di convertire i valori da float32 a int16 in quanto 9 float sono 36 bytes. Per fare ciò nel codice sono state aggiunte alcune costanti moltiplicative per i nove float che consentono di ottimizzare il loro invio:

```
#define ACC_MULTIPLIER 8192
#define GYR_MULTIPLIER 16.384
#define MAG_MULTIPLIER 81.92
#define TEMP_MULTIPLIER 100
#define HUM_MULTIPLIER 100
#define PRES_MULTIPLIER 100
```

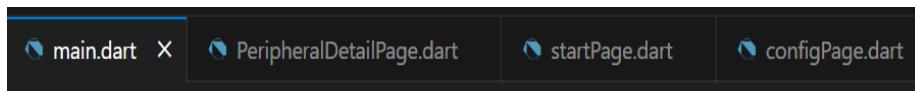
costanti moltiplicative

Il codice completo verifica lo stato della connessione bluetooth, e in caso positivo colleziona i dati provenienti dal sensore con un periodo di campionamento di 100 ms; moltiplica tali valori per le costanti di cui già discusso sopra e invia un unico array di nove int16. Per completezza, nel codice sono state aggiunte funzionalità per la gestione dei dati relativi a orientation (row,pitch and roll) e environment (come pressione e temperatura), anche se tali informazioni non sono di interesse per la mia tesi.

2.2 Applicazione Flutter

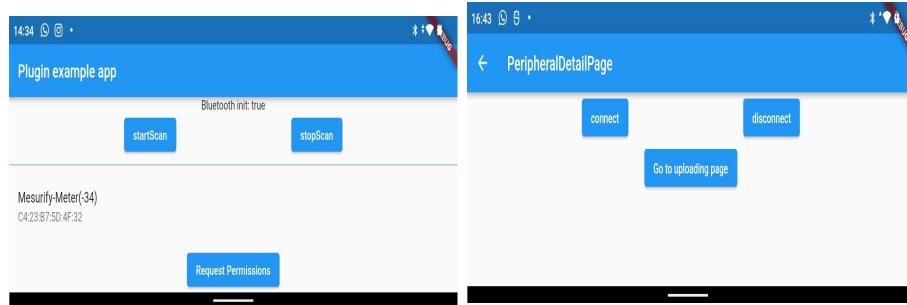
Completata la programmazione/configurazione lato Arduino, mi sono concentrato sull'applicazione che, caricata su un tablet compatibile, dopo aver verificato la connessione Bluetooth, permette l'accumulo degli array di dati provenienti

dalla scheda, l'aggiunta dei tag relativi al tipo di misurazione e il loro invio al cloud con Measurify. Questi dati ottenuti in int16 vengono riconvertiti nei valori float originali facendo la divisione per le costanti dichiarate sopra. Per una tesi di laurea triennale sviluppare l'intero codice da zero avrebbe richiesto un tempo eccessivo, perciò mi è stata fornita una base di partenza già completa di configurazione bluetooth e servizi per l'arduino, e io ho aggiunto le funzionalità per Imu, Orientation e Environment e il collegamento a Measurify. Anche se gli ultimi due non sono di alcuna utilità per l'oggetto della mia tesi in senso stretto, con il professore e i dottorandi abbiamo deciso di aggiungerli a favore di una maggiore completezza e in vista di un possibile loro utilizzo futuro in altri ambiti. Il codice è composto da più file, ma i più importanti sono main.dart, peripheralDetailPage.dart, startPage.dart e configPage.dart. L'ordine in cui li ho nominati è importante, perché è lo stesso in cui vengo aperti quando viene fatto girare il codice su dispositivo Android. Il main.dart è il file principale che chiama gli altri all'interno del suo corpo, ed è anche l'unico che può essere lanciato se si vuole caricare il codice completo. La peripheralDetailPage.dart gestisce la pagina di connessione del dispositivo con l'arduino, e la startPage.dart il conseguente pannello di invio dei dati al cloud, con tutti i tag necessari. ConfigPage.dart mantiene le informazioni necessarie al corretto invio dei dati, come 'deviceToken', 'deviceName', 'thingName'.



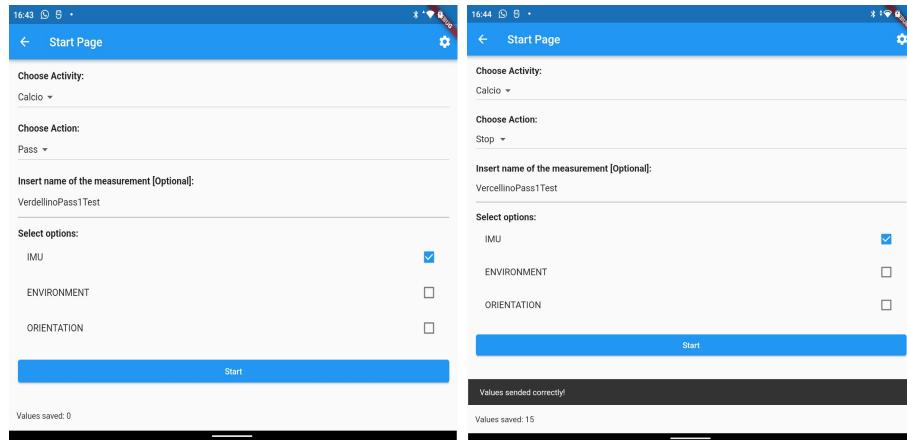
Io ho lavorato soprattutto sulla startPage, ovvero la schermata che appare una volta stabilita la connessione bluetooth tra arduino e dispositivo mobile. In questa schermata è possibile scegliere il tag relativo al tipo di misurazione che si vuole effettuare (un tiro, un passaggio, uno stop) e quali dati si vogliono inviare al cloud, spuntando una o più tra le caselle "IMU", "ORIENTATION", "ENVIRONMENT". A codice completato, collego un dispositivo mobile Android al computer, carico l'applicazione e si apre la seguente schermata per lo scanning dei dispositivi bluetooth vicini per instaurare la connessione bluetooth.

Una volta rilevato l'Arduino (che ha id 'Measurify-Meter'), lo seleziono e si apre la seconda schermata.



connessione all'Arduino tramite Bluetooth

Mi connetto alla scheda e vado alla 'uploading page'. Adesso seleziono dal primo dei due menù a tendina "calcio" e dal secondo l'azione che voglio registrare. Le opzioni selezionabili sono "shot", "pass" e "stop". Inserisco il nome della misura, ad esempio "VercellinoPass1Test", che indica che i dati che sto per raccogliere sono relativi ad uno slot di test di passaggi che intendo registrare per verificarne l'invio al cloud. Seleziono "IMU" e con il tasto "Start" sono pronto a trasmettere i dati. Una volta soddisfatto, con il pulsante "Stop and send" invio i valori a Measurify. Se l'invio è andato a buon fine, comparirà sul fondo della schermata il messaggio "Data sent correctly" e il numero di array inviati è mostrato in basso a sinistra.



raccolta valori e invio al cloud

L'applicazione risponde ai comandi in modo fluido e non da problemi durante l'invio dei dati.

2.3 Controllo con Postman e Google Colab

Measurify è un API cloud di tipo RESTful, quindi può essere interrogato con delle "chiamate" apposite per verificare i dati presenti su di esso. Per far ciò, ho a disposizione l'applicazione Postman, uno strumento per testare le chiamate API. Tutto ciò che devo fare è utilizzare i comandi che trovo nella parte sinistra dell'applicazione. Per prima cosa, effettuo il login: mi basta inviare la richiesta già predisposta dall'apposita riga di comando. Con il tasto "Send" invio tale richiesta ed effettuo il login, che dura 30 minuti.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows a tree structure with categories like TAGS, FEATURES, THINGS, DEVICES, and MEASUREMENTS. Under MEASUREMENTS, there are two items: "POST LOGIN as admin" and "POST LOGIN as Vercellino". The "POST LOGIN as Vercellino" item is highlighted with a black arrow pointing to it.
- Request Details:**
 - Method:** POST
 - URL:** {{url}}/v1/login
 - Body:** (highlighted with a red arrow)
 - Params:** none
 - Headers:** (10)
 - Tests:**
 - Settings:**
- Right Side:**
 - Send:** (highlighted with a blue arrow)
 - Cookies:**
 - Beautify:**
- Code Preview:**

```

1 <code>
2 >   "username" : "Vercellino",
3 >   "password" : "Vercellino_Password",
4 >   "tenant" : "Football-Tenant"
5 </code>

```

login

Nel messaggio che compare sullo schermo sono contenute tutte le informazioni relative alle mie credenziali in questo tenant:

```

"user": {
    "validityPasswordDays": 1092,
    "_id": "645e0a21739aab001efe8b64",
    "username": "Vercellino",
    "email": "a@a.it",
    "type": "provider",
    "createdPassword": "2023-05-12T09:42:57.186Z"
},
"token expiration time": "30m",

```

credenziali login

Ora posso verificare se i dati da me inviati sono arrivati al server: con il comando "Get one measurement" inserisco il nome della misura che mi interessa nella barra centrale e controllo che sia presente:

The screenshot shows the Postman interface with a sidebar containing various API endpoints under the 'MEASUREMENTS' category. A red arrow points from the sidebar to the 'Key' field in the 'Query Params' section of the main request configuration. The request URL is set to `(url)/v1/measurements/VercellinoPass1Test`. The response body is displayed in a code block:

```

1 "visibility": "public",
2 "tags": [
3     "Calcio",
4     "Stop"
5 ],
6 "_id": "VercellinoPass1Test",
7 "thing": "player1",
8 "feature": "IMU",
9 "device": "edge-football",
10 "startDate": "2023-08-29T15:08:02.085Z",
11 "endDate": "2023-08-29T15:08:02.085Z",
12 "samples": []

```

acquisizione di una misura

Se la misura è presente, come in questo caso, compariranno le informazioni relative (tags, feature, data, etc..) nel messaggio nella parte bassa dello schermo:

```

"visibility": "public",
"tags": [
    "Calcio",
    "Pass"
],
"_id": "VercellinoPass_Testing",
"thing": "player1",
"feature": "IMU",
"device": "edge-football",
"startDate": "2023-08-29T15:08:02.085Z",
"endDate": "2023-08-29T15:08:02.085Z",
"samples": []

```

visualizzazione della misura

In caso contrario, comparirebbe il messaggio "Resource not found". Ora voglio entrare nello specifico di questa misura, e controllare i dati inviati. Utilizzo il comando "Get a timeseries" e mi aspetto di ottenere 15 misurazioni:

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** {{url}}/v1/measurements/VercellinoPass1Test/timeserie?limit=1000&sort={"timestamp": "asc"}
- Params:** limit (checkbox checked, value 1000)
- Headers:** (11 items listed)
- Body:** (Pretty, Raw, Preview, Visualize, JSON, copy, delete)
- Cookies:** (empty)
- Headers (10):** (empty)
- Test Results:** (empty)
- Status:** 200 OK (111 ms, 4.2 KB)
- Actions:** Save as Example, Copy, Delete

The response body is displayed in Pretty format:

```
2 "docs": []
3 [
4   {
5     "values": [
6       -0.0286865234375,
7       -0.16333098125,
8       0.9735107421875,
9       9.82666015625,
10      5.43212890625,
11      5.55419921875,
12      -44.775390625,
13      -6.45751953125,
14      -10.11962898625]
```

An arrow points to the "GET GET a timeseries" item in the left sidebar.

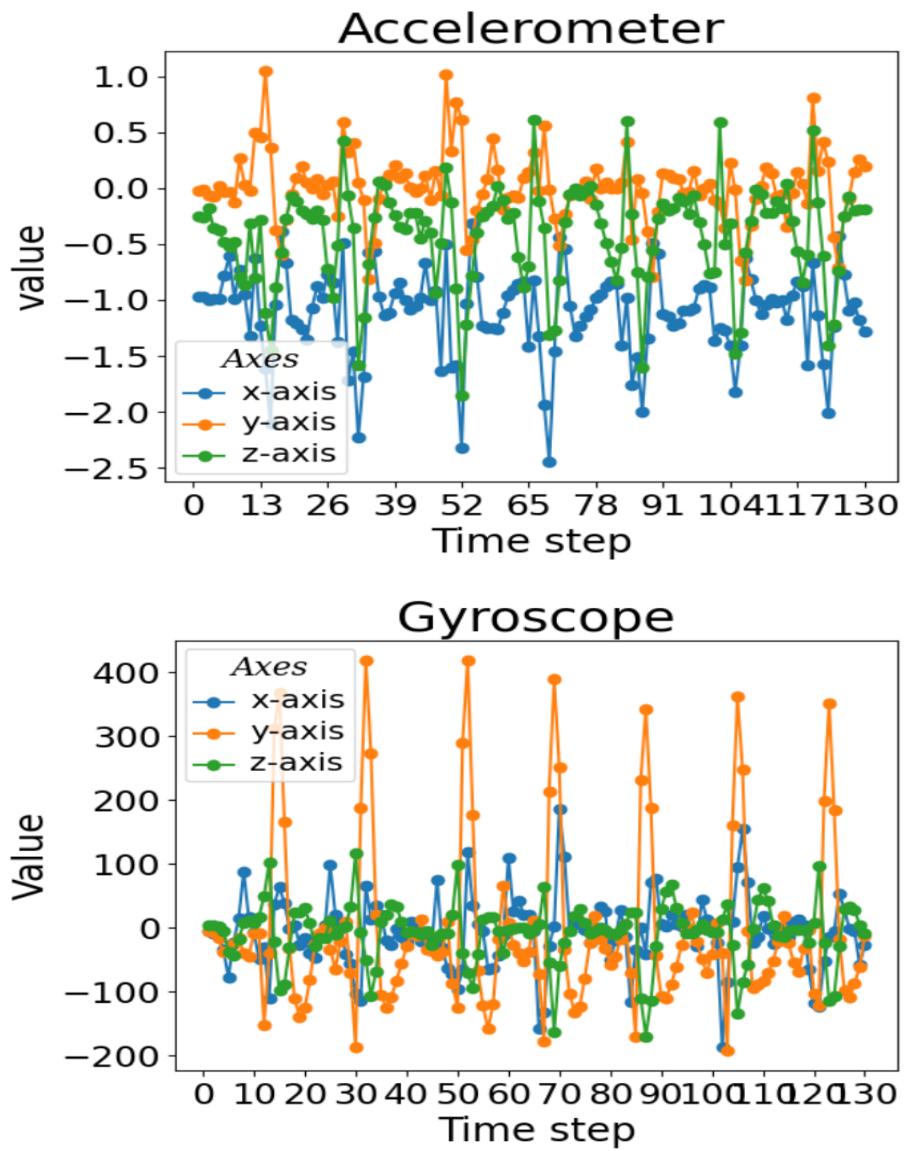
acquisizione di una timeserie

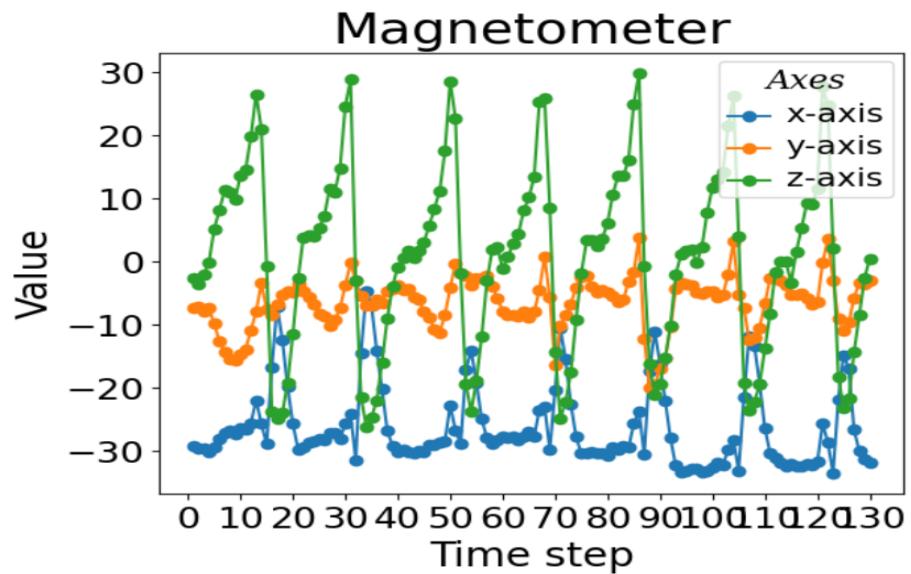
Ho ottenuto i 15 "pacchetti" contenenti i valori su accelerometro, giroscopio e magnetometro, insieme al nome della misurazione di cui fanno parte.

```
"docs": [
  {
    "values": [
      -0.99072265625,
      -0.05029296875,
      -0.3333740234375,
      1.3427734375,
      -29.052734375,
      -4.94384765625,
      -23.7548828125,
      -13.07373046875,
      23.47412109375
    ],
    "totalDocs": 280,
    "limit": 1,
    "totalPages": 280,
    "page": 1,
    "pagingCounter": 1,
    "hasPrevPage": false,
    "hasNextPage": true,
    "prevPage": null,
    "nextPage": 2
  }
]
```

pacchetto con i valori di una timesample

Ho a disposizione un’altro strumento per controllare l’andamento delle misurazioni effettuate: Google Colab. E’ una piattaforma multifunzionale molto utile ai miei scopi per visualizzare tramite grafici i valori dei sensori inviati nelle time-series. Offre una visualizzazione rapida e intuitiva dei dati raccolti, consentendo di cogliere le caratteristiche principali dei diversi movimenti osservati. Nelle foto seguenti, sono mostrati i grafici relativi a sette tiri:





grafici relativi ad alcuni tiri

Sono facilmente distinguibili le sette sequenze di valori che si ripetono simili ogni circa 2 secondi (20 valori).

3 Sperimentazione e Risultati

3.1 Montaggio dell'Arduino su parastinco

L'Arduino Nano 33 dispone di molteplici sensori su un'area molto ridotta, perciò è da trattare con attenzione e delicatezza. La mia tesi tratta l'esecuzione di passaggi, stop e tiri che verosimilmente si potrebbero verificare durante una partita di calcio, quindi occorre proteggere l'Arduino da colpi indesiderati. Nell'ambito del posizionamento della scheda, l'idea di inserirla nello scarpino da calcio è risultata da subito rischiosa, perché la lascerebbe sovraesposta a urti elevati, nonostante sia l'opzione migliore per rilevare i movimenti del piede nella loro completezza. La scelta finale ha dunque premiato il posizionamento dell'Arduino nel parastinco (quello della gamba che calcia), poichè rappresenta un giusto compromesso tra efficienza dei dati raccolti e protezione della scheda elettronica. Ho deciso di adattare il parastinco ritagliandone un rettangolo delle stesse dimensioni dell'Arduino per potercelo inserire; ho poi ricoperto la parte anteriore del parastinco con neoprene, per assorbire al meglio i colpi.



predisposizione del parastinco

3.2 Raccolta del dataset

A questo punto del progetto, è stato possibile iniziare a raccogliere i dati che serviranno all'addestramento dell'algoritmo di machine learning. A tal fine, i dati sono stati raccolti in un giardinetto con l'aiuto di un'altra persona. Per registrare i passaggi, ci siamo posizionati uno di fronte all'altro scambiandoci ripetutamente la palla. Per gli stop sono riuscito a fare tutto da solo, alzandomi la palla con le mani e stoppandola con il collo del piede, mentre per i tiri sono stati necessari più palloni posizionati davanti ad una rete, fermando la misurazione ogni dieci tiri per riposizionare i palloni e continuando successivamente la raccolta dei dati. Alla fine delle misurazioni, ho raccolto in totale:

- 100 passaggi;
- 100 stop;
- 110 tiri.

Per ogni movimento, che fosse tiro, passaggio o stop, ho avuto cura di riposizionare piede e gamba in posizione iniziale, per rendere possibile il riconoscimento della fine di ogni gesto. Le fasi di addestramento di un modello di machine learning sono tre (ma non essendo argomento di tesi triennale, non mi ci soffermerò): training, validation (che riguardano direttamente la fase di apprendimento del modello) e testing. Inoltre, un modo migliore di capire la validità di tale modello può essere la "confusion matrix", tabella utile per valutare la qualità delle previsioni del modello di classificazione. In particolar modo, la matrice mette in evidenza dove sbaglia il modello, in quali istanze risponde peggio e quali meglio.

		ACTUAL VALUES		
		Positive	Negative	
PREDICTED VALUES	Positive	TP	FP	Type I error : The predicted value is positive but it False
	Negative	FN	TN	Type II error : The predicted value is negative but it's positive
				The predicted value is Negative and its Negative

confusion matrix

Appena mi è stato restituito il modello allenato, lo ho caricato sull'arduino, e ho raccolto i dati per il testing: quindici per tipologia, utili a verificare se il modello sia valido o meno nel riconoscere i gesti da me prodotti. Il funzionamento del modello è piuttosto semplice: registro il movimento con l'arduino, e sul serial monitor dell'IDE vengono stampate le probabilità relative ai tre tipi di movimento (da 0 se c'è probabilità minima che sia il relativo gesto, a 1 se c'è probabilità massima).

```
Predictions (DSP: 0 ms., Classification: 8 ms., Anomaly: 0 ms.):
pass: 0.00000
stop: 0.00000
shot: 1.00000
```

```
Starting inferencing in 2 seconds...
```

```
Sampling...
```

Al termine delle misurazioni di "testing", questi sono i risultati:

- 13 su 15 tiri rilevati;
- 11 su 15 passaggi rilevati;
- 10 su 15 stop rilevati.

Il risultato non è accettabile, poiché bisognerebbe raggiungere un grado di accuratezza maggiore al 90 % per potersi considerare soddisfatti. La bassa precisione del modello è probabilmente data dal fatto che è stato allenato troppo poco rispetto a quello che necessita. D'altra parte, al primo tentativo con un algoritmo poco allenato sarebbe stato ottimistico aspettarsi valori più soddisfacenti. A questi risultati si può comunque rimediare: riallenare il modello, perfezionandolo. Avvisato il laboratorio, dopo qualche giorno mi è stato restituito il modello aggiornato. Sono tornato quindi a ad effettuare 15 misure di testing per ogni movimento; i risultati sono illustrati nella seguente tabella:

		PREDICTION		
		shot	pass	stop
REAL	shot	15		
	pass	1	14	
	stop		1	14

tabella di predizione dei risultati

Il nuovo modello ha rilevato:

- 15 tiri su 15;
- 14 passaggi su 15;
- 14 stop su 15;

L'algoritmo ha confuso solo un passaggio per un tiro e uno stop per un passaggio. Questi errori sono comprensibili, essendo il movimento del passaggio molto simile, eccetto che per l'accelerazione del piede, a quello del tiro. L'accelerazione del piede durante lo stop è invece paragonabile a quella del passaggio.

Occorre specificare che tutti i movimenti che ho effettuato costituiscono una modellizzazione dei movimenti reali, poiché durante una azione in una partita di calcio tali gesti possono essere effettuati in moltissimi modi, partendo dal fatto che nella maggior parte dei casi vengono effettuati in corsa o in movimento, e questo confonde il modello allenato dai miei dati. La mia rimane comunque una ragionevole approssimazione di tali gesti, e a scopo didattico può essere considerata accettabile.

4 Contributo personale e considerazioni conclusive

Il sistema di classificazione dei movimenti dello sport del calcio sviluppato per questa tesi si è dimostrato molto valido. Tutte le fasi del progetto in generale hanno richiesto un carico di lavoro affrontabile e tutti i problemi riscontrati durante il percorso sono stati superati pienamente. Ciò non significa che il mio progetto di tesi sia stato facile o eccessivamente semplificato: mi ci sono approcciato verso la fine di marzo e, con l'eccezione del periodo di esami di giugno, ci ho lavorato ininterrottamente. Ho riscontrato difficoltà in primis nella comprensione concettuale del lavoro da svolgere in tutte le sue parti (come tutte le fasi si legano tra loro), poi nello sviluppo della applicazione flutter (lavoro che mi è stato poi semplificato dal laboratorio). Tuttavia, con il tempo e con la pazienza, ho imparato nuove abilità e sono riuscito a portare a termine il lavoro con successo. La motivazione che mi ha spinto ad iniziare questa tesi è stata la volontà di unire una mia passione ad un ambito che costituisse una sfida per me, con la speranza che questo facilitasse tale sfida e la rendesse ancora più interessante. A progetto finito, posso confermare le mie aspettative, e sento di poter aggiungere che mi ha arricchito dal punto di vista personale. Il laboratorio ed io volevamo dimostrare che fosse possibile, con un costo ridotto e con poche conoscenze in ambito informatico/ elettronico, comprendere e utilizzare un sistema di classificazione di movimenti di uno sport, avvalendosi di alcuni strumenti messi a disposizione per tutti (come il cloud, il modello e il codice). La sfida è che in futuro ognuno possa usufruire di tutto ciò in un numero sempre crescente di attività sportive. Per ragioni di tempo, la mia tesi tratta solo movimenti modellizzati, come già detto precedentemente, ma con un dataset più completo e un modello più accurato ritengo che sarà possibile classificare più movimenti e differenziare tra loro quelli simili.

5 Riferimenti bibliografici

1. Measurify: <https://measurify.org/>
2. Arduino Nano 33 Ble Sense: <https://store.arduino.cc/products/arduino-nano-33-ble-sense>
3. Arduino IDE: <https://www.arduino.cc/en/software>
4. Postman: <https://www.postman.com/>
5. Google Colab: <https://colab.research.google.com/>
6. VS Code: <https://code.visualstudio.com/>
7. Laboratorio Elios Lab: <https://elios.diten.unige.it/>

6 Ringraziamenti

Ringrazio prima di tutto il correlatore Matteo Fresta che mi ha seguito durante tutto il percorso di tesi dimostrando una notevole pazienza, il professor Berta che mi ha dato la possibilità di confrontarmi con questa sfida unendola alla mia passione, e tutti i ragazzi del laboratorio di Elios Lab che mi hanno aiutato direttamente e indirettamente. Ringrazio inoltre la mia famiglia, che ha creduto in me anche non condividendo la mia stessa idea di "studio". Infine ringrazio i miei amici, che mi hanno aiutato a mantenere la concentrazione sugli obiettivi senza farmi vincere dalle preoccupazioni.