



UNIVERSITÀ DEGLI STUDI DI GENOVA

**DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA,
ELETTRONICA E DELLE COMUNICAZIONI**

**CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E
TECNOLOGIE DELL'INFORMAZIONE**

Tesi di Laurea Triennale

Dicembre 2023

**Progetto e realizzazione di un sistema embedded applicato al
gioco del calcio-balilla**

**Design and development of an Embedded System Applied to
Table Football Game**

Candidato: Elena Molinari

Relatore: Prof. Riccardo Berta
Correlatore: Dott. Matteo Fresta

SOMMARIO

Il presente elaborato è stato redatto con l'obiettivo di progettare e realizzare un sistema embedded per il rilevamento degli urti causati da una pallina sulle stecche del tavolo da biliardino e la loro conseguente distinzione durante una partita. Infatti, il calcio-balilla è un gioco che richiede abilità, riflessi e una buona dose di concentrazione. Tuttavia, le decisioni e le azioni dei giocatori possono spesso basarsi su osservazioni soggettive e sensazioni, il che può portare a discussioni su quali colpi siano stati effettivamente realizzati durante una partita. Nello specifico, si utilizza un microcontrollore, Arduino Due, che gestisce diversi accelerometri ADXL345 posti sulle stecche di un calcio-balilla. Il codice per rilevare gli urti della pallina è stato scritto in linguaggio C++ usando il popolare IDE di Arduino.

Le prospettive future del progetto sono quelle di fornire ai giocatori uno strumento che permetta loro di registrare e monitorare gli impatti durante una partita di calcio-balilla. Questo permetterà non solo di tenere traccia delle statistiche di gioco, ma anche di analizzare le performance individuali e di squadra.

Indice

1	Introduzione	4
1.1	Flusso di lavoro	5
2	Metodi e strumenti utilizzati	6
2.1	Arduino Due	6
2.2	Accelerometro ADXL345	7
2.3	Libreria SparkFun_ADXL345	8
2.4	Arduino IDE	9
3	Sperimentazione e risultati	12
3.1	Montaggio dei sensori sulle stecche	12
3.2	Acquisizione dati dalla scheda Arduino	14
3.3	Sistema in funzione	16
4	Contributo personale e considerazione conclusive	17
5	Riferimenti biografici	18
6	Ringraziamenti	18

1. Introduzione

La presente tesi si concentra su un'integrazione di tecnologia e gioco, proponendo l'implementazione di un sistema embedded su un calcio-balilla tradizionale. Nel nostro contesto, questo significa introdurre una piattaforma tecnologica dedicata al calcio-balilla, in grado di migliorare e arricchire l'esperienza di gioco.

Un sistema embedded può essere definito come un sistema dedicato, progettato cioè per svolgere un compito preciso e determinato (a differenza di quanto succede nei sistemi general purpose).

Dunque, l'obiettivo principale di questo progetto è quello di catturare in tempo reale le interazioni tra la pallina e le stecche del calcio-balilla e attraverso l'impiego dei sensori rilevare e registrare con precisione ogni impatto, trasformando le stecche in veri e propri elementi interattivi.

Per raggiungere tale obiettivo si è pensato di utilizzare dei sensori di accelerazione (accelerometri) opportunamente posizionati sulle stecche del calcio-balilla, in comunicazione con un microcontrollore.

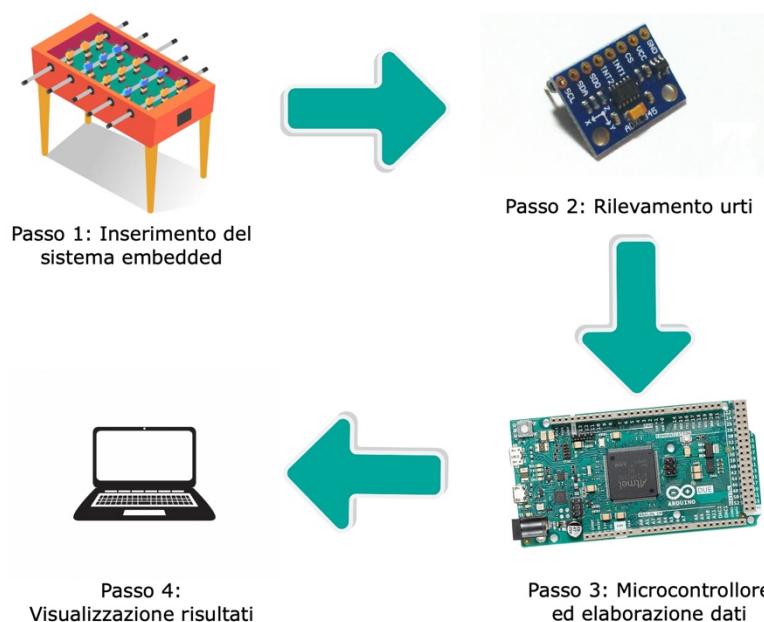


Figura 1.1: Schema funzionamento del progetto

Nella figura in alto, vengono descritti i passi che porteranno alla realizzazione del progetto. Il primo passo consiste nel montaggio dei sensori sulle stecche del calcio-balilla, più precisamente dietro la schiena degli omini (Passo 1). Dopodiché durante una partita gli accelerometri saranno completamente operativi e rileveranno gli urti per ciascuna stecca (Passo 2). Nel frattempo, il microcontrollore elaborerà gli input ricevuti (Passo 3) che verranno visualizzati sul serial monitor dell'ambiente di sviluppo di Arduino (Passo 4).

Un passo futuro sarà quello di permettere di visualizzare le statistiche di gioco e quindi rendere possibile un'analisi dettagliata delle performance dei giocatori per migliorare le abilità individuali e di squadra. Per esempio, i dati che si potranno avere a disposizione potrebbero essere le percentuali di possesso palla sia per squadra che per singolo giocatore oppure il numero di volte che c'è stato un passaggio tra stecche appartenenti alla stessa squadra, ecc.

1.1 Flusso di lavoro

Per completare il progetto di tesi sono state attraversate varie fasi di lavoro nel seguente modo: in primis sono stati scelti quali componenti hardware utilizzare. Per la scelta del microcontrollore sono state considerate le necessità date dal progetto, come per esempio il bisogno di molti pin per poter collegare più sensori, e le interfacce di comunicazione offerte. Quindi si è optato per la scheda Arduino Due. Mentre per gli accelerometri sono stati considerati il range di misura, l’interfaccia di comunicazione, le dimensioni e soprattutto il costo. Quindi si è optato per gli ADXL345 che offrono un range piuttosto ampio ed un costo ridotto. Successivamente si è passato alla scrittura del codice sull’IDE (Integrated Development Environment) offerto da Arduino stesso e all’assemblaggio dei componenti hardware sul calcio-balilla. Infine, sono state fatte delle prove per riuscire a trovare i giusti parametri di sensibilità degli accelerometri.

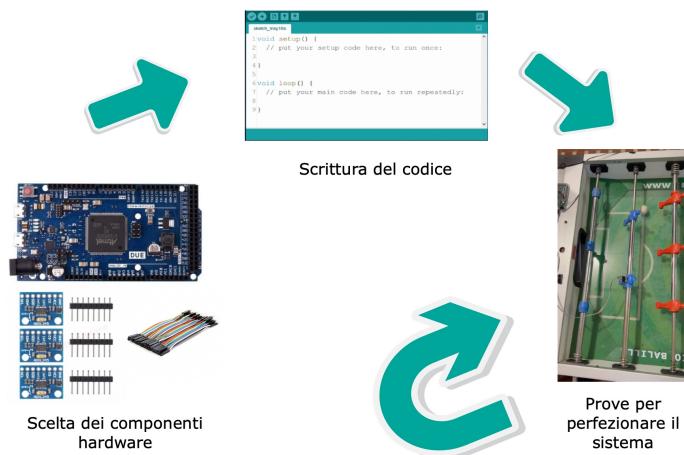


Figura 1.2: Rappresentazione delle fasi di lavoro

La tesi è articolata in tre paragrafi:

- nel primo vengono elencati e discussi i mezzi per la realizzazione del progetto, includendo, inoltre considerazioni sugli approcci utilizzati per affrontare il problema;
- nel secondo si passa alla valutazione dei risultati ottenuti dopo aver testato il sistema;
- nel terzo ed ultimo sono esposte alcune considerazioni finali e commenti.

2. Metodi e strumenti utilizzati

2.1 Arduino Due

Per sviluppare il progetto di tesi è stata utilizzata la scheda Arduino Due.

Essa è la prima scheda Arduino basata su un microcontrollore di tipo ARM a 32 bit, il SAM3X8E ARM Cortex-M3 di Atmel e come le sue simili è programmabile tramite il linguaggio C++ usando il popolare IDE di Arduino. Sulla scheda sono presenti 54 pin digitali di ingresso/uscita, tra cui di fondamentale importanza per questo progetto i pin che rendono possibile la comunicazione I²C, pin 20 (SDA) e pin 21 (SCL). Un altro motivo per cui è stata scelta questa scheda è perché ha dodici input che possono valere ciascuno come interrupt, quindi ottimo per questo progetto poiché necessari otto pin per ciascuna stecca del calcio-balilla. A differenza di altre schede Arduino, la scheda Arduino Due funziona a 3.3V.

La Arduino Due dispone, inoltre, di due porte di comunicazione micro-USB.

- la Porta di Programmazione consente l'alimentazione del microcontrollore, il caricamento dello sketch creato dallo sviluppatore e la comunicazione con l'IDE in fase di debug. Questa è la porta più adeguata per i fini del progetto e quella utilizzata;
- la Porta USB nativa è collegata direttamente al SAM3X8E ed è utilizzata quando si desidera usare Arduino Due come una qualsiasi periferica USB (così come avviene per un mouse o una tastiera), oppure quando si vuole usare la scheda come host in modo che altri dispositivi si possano collegare ad essa (mouse, tastiere o telefoni Android).

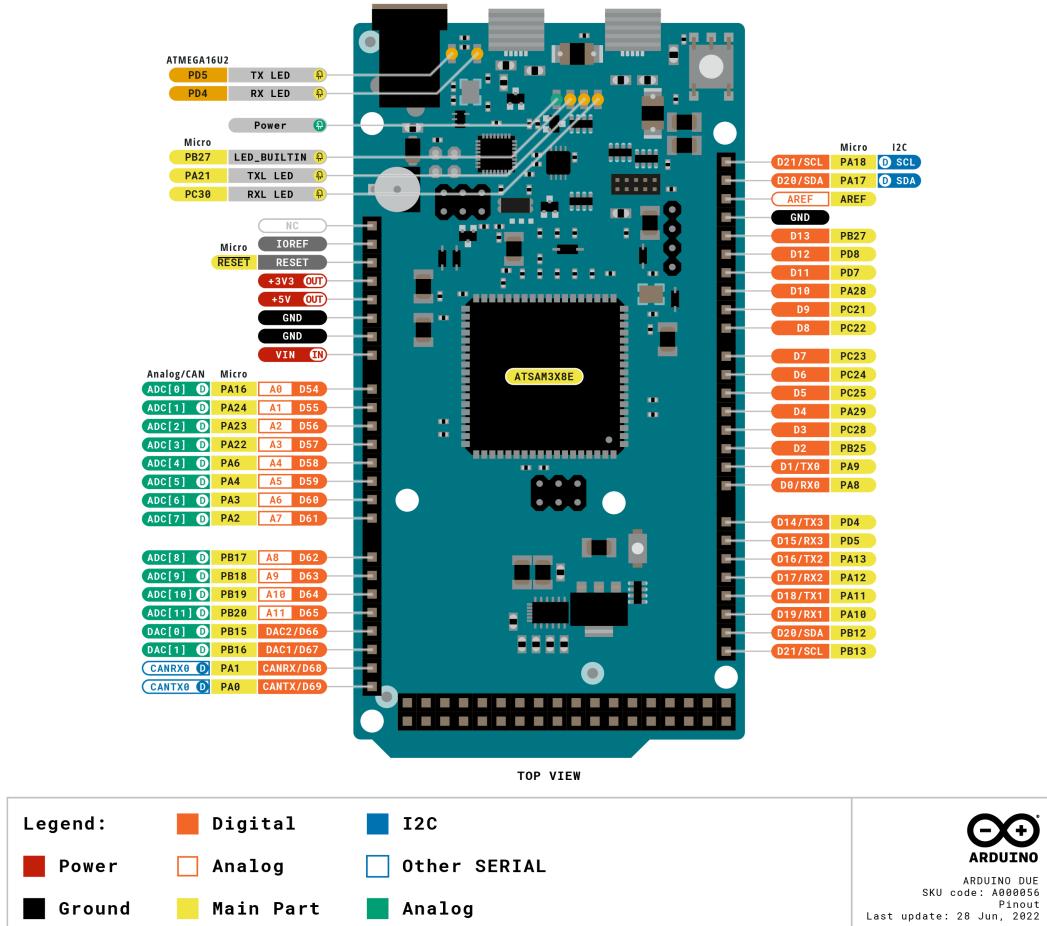


Figura 2.1: Arduino Due

2.2 Accelerometro ADXL345

L'ADXL345 è un accelerometro a 3 assi di piccole dimensioni, infatti misura $3\text{ mm} \times 5\text{ mm} \times 1\text{ mm}$, in grado di effettuare misurazioni ad alta risoluzione (13 bit) con una gamma di $\pm 16\text{ g}$.

Inoltre, il sensore mette a disposizione diverse funzioni speciali di rilevamento, come l'attività e l'inattività o la rilevazione della caduta libera. Tuttavia, la caratteristica più cruciale per il progetto è la funzione di rilevamento dei tap che rende il sensore in grado di individuare gli urti causati dalla pallina sulla stecca.

L'ADXL345 supporta due protocolli di comunicazione, SPI e I²C. Nel progetto si è optato per l'utilizzo dell'I²C poiché richiede un numero minore di cavi per l'interconnessione rispetto a SPI (due cavi contro quattro). La sua tensione di lavoro è compresa nell'intervallo tra i 2,0V e i 3,6V.

Questo sensore è dotato di otto pin, partendo dall'alto verso il basso troviamo:

- **GND:** pin di massa (ground).
- **VCC:** pin di tensione di alimentazione.
- **CS:** il pin Chip Select serve, nel nostro caso, ad abilitare il sensore cosicché possa comunicare tramite il bus I²C.
- **INT1 e INT2:** pin dedicati all'interrupt. In questo progetto verrà utilizzato un pin di interrupt per far funzionare correttamente la funzione di rilevamento tap.
- **SDO:** questo pin è utilizzato come Serial Data Output per la comunicazione SPI e come Address Select per la comunicazione I²C.
- **SDA e SCL:** pin dedicati alla comunicazione tramite il protocollo I²C, è tramite questi che il sensore riceve i comandi e trasmette le misurazioni al microcontrollore.

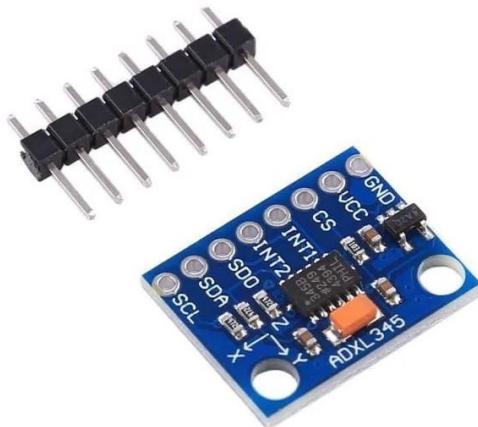


Figura 2.2: Accelerometro ADXL345

2.3 Libreria SparkFun_ADXL345

Per la scrittura del codice, necessario alla gestione dei sensori da parte del microcontrollore, è stata utilizzata la libreria ADXL345_Arduino di SparkFun. All'interno della libreria si possono trovare due file:

SparkFun_ADXL345.h e SparkFun_ADXL345.cpp.

Il file .h è l'header, nel quale è presente l'interfaccia della classe, cioè l'insieme degli attributi e tutte le dichiarazioni dei metodi con relativa visibilità. La definizione del metodo, cioè il corpo della funzione, è poi implementata nel file .cpp.

Alcune funzioni importanti presenti in questa libreria sono setTapDetectionOn, setTapThreshold e setTapDuration, che servono per attivare e calibrare la rilevazione degli urti. Queste e altre funzioni saranno descritte più dettagliatamente nel prossimo capitolo.

Prima di poter scrivere il codice su Arduino IDE è necessario fare delle modifiche alla libreria. Infatti, essa non tiene conto della possibilità di poter collegare più accelerometri alla scheda Arduino, ma uno soltanto a cui è assegnato l'indirizzo di default (0x53).

Per realizzare il progetto di questa tesi è fondamentale la creazione di oggetti ADXL345 con indirizzi I²C diversi in modo da poter distinguere ogni sensore. Infatti, quando la pallina colpisce una stecca il sensore che giace su quest'ultima genererà un segnale di interrupt che viene letto dal sistema e in base a quale interrupt è avvenuto viene salvata l'informazione su quale stecca è stata colpita. Perciò sono state apportate delle modifiche al costruttore della libreria. Per poter accedere ai file da modificare ho utilizzato l'editor di codice VS Code per pura comodità dello strumento.

Un costruttore è un metodo speciale all'interno di una classe in un linguaggio di programmazione orientato agli oggetti che viene utilizzato per inizializzare gli oggetti della classe. Il costruttore ha lo stesso nome della classe (nel nostro caso la classe si chiama ADXL345) e può accettare parametri per inizializzare le variabili di istanza dell'oggetto. In poche parole, un costruttore definisce come un oggetto della classe deve essere inizializzato quando viene creato.

```
109 class ADXL345
110 {
111 public:
112     bool status;           // Set When Error Exists
113
114     byte error_code;      // Initial State
115     double gains[3];      // Counts to Gs
116     int ADXL345_DEVICE;
117
118     ➔ ADXL345(int addr);
119     void powerOn();
120     void readAccel(int* xyz);
```

Figura 2.3: Modifica al file SparkFun_ADXL345.h

La figura mostra come il costruttore ADXL345 del file .h è stato adattato in modo che accetti un parametro per l'indirizzo I²C.

Dopodiché, nel file .cpp, sono state apportate le modifiche necessarie al costruttore per utilizzare l'indirizzo I²C fornito come parametro.

```
32  ↘ ADXL345::ADXL345(int addr) {
33      ADXL345_DEVICE = addr;
34      status = ADXL345_OK;
35      error_code = ADXL345_NO_ERROR;
36
37      gains[0] = 0.00376390;      // Original gain 0.00376390
38      gains[1] = 0.00376009;      // Original gain 0.00376009
39      gains[2] = 0.00349265;      // Original gain 0.00349265
40      I2C = true;
41 }
```

Figura 2.4: Modifica al file SparkFun_ADXL345.cpp

Nel codice sopra, è stato aggiunto un parametro **addr** al costruttore e utilizzato questo parametro per impostare l'indirizzo I²C (ADXL345_DEVICE) dell'oggetto. In questo modo, è possibile specificare l'indirizzo I²C desiderato quando si crea un oggetto ADXL345.

```
1 #include <SparkFun_ADXL345.h>
2
3 ADXL345 adxl = ADXL345(0x53);
4 ADXL345 adxl2 = ADXL345(0x1D);
```

Figura 2.5: Creazione degli oggetti ADXL345 con diversi indirizzi su IDE Arduino

2.4 Arduino IDE

La stesura del codice e il caricamento sulla nostra scheda Arduino avvengono tramite l'IDE di Arduino. Un IDE è un ambiente di sviluppo integrato (in lingua inglese Integrated Development Environment), ovvero un software utile alla realizzazione di codice. In particolare, l'IDE di Arduino permette la stesura del codice (sketch) e successivo caricamento sulla memoria della scheda.

Lo sketch è diviso in due fasi, quella di Setup, che è la fase di preparazione, e quella di Loop, che è la fase di esecuzione.

La funzione di Setup() viene eseguita una sola volta all'avvio e viene utilizzata per inizializzare le variabili, configurare i pin, librerie e sensori.

La funzione di Loop(), invece, viene eseguita in modo continuo dopo la Setup(). È in questa funzione che si definisce il comportamento principale del programma, e viene ripetutamente eseguita finché la scheda è alimentata.

All'interno della funzione di Setup() vengono inizializzati i sensori utilizzando i metodi forniti dalla libreria.

```
12 void setup(){
13     Serial.begin(9600);
14     Serial.println("Start simulation:");
15     Serial.println();
16
17
18
19     adxl.powerOn();
20     adxl.setRangeSetting(16);
21     adxl.setTapDetectionOnXYZ(1, 1, 1);
22     adxl.setTapThreshold(230);
23     adxl.setTapDuration(15);
24     adxl.setInterrupt(ADXL345_INT_SINGLE_TAP_BIT, ADXL345_INT1_PIN);
25     adxl.singleTapINT(1);
26     attachInterrupt(digitalPinToInterrupt(interruptPin2), ADXL_ISR, RISING);
```

Figura 2.6: Inizializzazione ADXL345

La prima istruzione **Serial.begin(9600)** imposta la velocità di trasmissione dei dati sulla porta seriale tra il computer e la scheda Arduino a 9600 bis (bit per second). Dopodiché, a riga 19, si passa alla fase di inizializzazione dei sensori:

- **powerOn()**: questa istruzione attiva il sensore ADXL345, mettendolo in modalità di funzionamento.
- **setRangeSetting(16)**: imposta la gamma di misurazione dell'accelerometro su $\pm 16g$ ($1g = \text{accelerazione dovuta alla gravità terrestre}$). Questo significa che il sensore sarà in grado di misurare accelerazioni nell'intervallo da $-16g$ a $+16g$.
- **setTapDetectionOnXYZ(1,1,1)**: abilita la rilevazione di "tap" (urto) lungo gli assi X, Y e Z. I valori 1 indicano che il rilevamento dei colpi è abilitato su tutti e tre gli assi.
- **setTapThreshold(230)**: inizializza il valore del registro THRESH_TAP che contiene il valore di soglia per gli interrupt di tap. Il fattore di scala è di 62,5 mg/LSB. Esso viene utilizzato per convertire il valore grezzo nel registro in un'unità fisica. Nel nostro caso abbiamo che il valore da superare perché avvenga un tap è di 14375 mg; questo valore è stato trovato sperimentalmente.
- **setTapDuration(15)**: imposta la durata minima per cui un tap deve superare la soglia per essere rilevato, in questo caso, 15 ms.
- **setInterrupt(ADXL345_INT_SINGLE_TAP_BIT, ADXL345_INT1_PIN)**: configura il sensore in modo da generare un interrupt quando viene rilevato un singolo tap. L'interruzione sarà segnalata sul pin INT1 del sensore.
- **singleTapINT(1)**: abilita la generazione dell'interruzione per il rilevamento di un singolo tap.
- **attachInterrupt(digitalPinToInterrupt(InterruptPin2), ADXL_ISR, RISING)**: collega un'azione da eseguire quando si verifica un interrupt, specificata nella funzione di Interrupt Service Routine ADXL_ISR, quando il segnale di interruzione è in transizione da LOW a HIGH (RISING edge).

Per il sensore adxl2, le chiamate sono praticamente identiche a quelle per adxl, solamente il pin specificato per la funzione attachInterrupt è diverso (interruptPin invece di interruptPin2) e la funzione di Interrupt Service Routine collegata è ADXL_ISR2 invece di ADXL_ISR.

Nella funzione di loop() il codice utilizza i sensori ADXL345 per rilevare i tap su ciascuna asta e generare messaggi sulla porta seriale nel momento in cui viene rilevato un urto. Inoltre, vengono utilizzate due funzioni ISR per la gestione degli interrupt.

```

56 void loop(){
57   delay(50);
58
59   if(trigger)
60   {
61     if(adxl.getInterruptSource(ADXL345_INT_SINGLE_TAP_BIT)){
62       Serial.println("*** ASTA BLU DIFESA ***");
63     }
64     trigger = 0;
65   }
66   if(trigger2)
67   {
68     if(adxl2.getInterruptSource(ADXL345_INT_SINGLE_TAP_BIT)){
69       Serial.println("*** ASTA ROSSA ATTACCO ***");
70     }
71     trigger2 = 0;
72   }
73 }
74 void ADXL_ISR() {
75   trigger = 1;
76 }
77
78 void ADXL_ISR2() {
79   trigger2 = 1;
80 }
```

Figura 2.7: Rilevamento tap

Inizialmente, il programma verifica due condizioni principali grazie a due variabili: **trigger** e **trigger2**. Queste variabili vengono utilizzate come flag per evitare la ripetizione continua dei messaggi sulla porta seriale. La prima condizione verifica se **trigger** è vero e se il sensore **adxl** ha generato un interrupt di tap singolo, se ciò avviene verrà stampato un messaggio sulla porta seriale indicante “ASTA BLU DIFESA”. Questo messaggio viene stampato solo una volta per ciascun tap rilevato, dopodiché il flag **trigger** viene reimpostato a zero per evitare che il messaggio venga ripetuto continuamente. La stessa logica viene applicata anche per la seconda condizione che rileva i singoli tap per l’asta rossa dell’attacco, invece.

Infine, il codice utilizza due funzioni di gestione per interrupt, **ADXL_ISR** e **ADXL_ISR2**. Queste funzioni vengono chiamate quando viene rilevato un tap su ciascun sensore. Quando una di queste funzioni viene chiamata, imposta il flag corrispondente (trigger o trigger2) su 1. Questo indica al loop principale che è stato rilevato un tap sull’asta corrispondente.

In sintesi, il codice è stato progettato per rilevare i singoli tap su due asta di diverso colore utilizzando sensori ADXL345. Quando un tap viene rilevato, un messaggio appropriato viene stampato sulla porta seriale. L’uso dei flag **trigger** e **trigger2** impedisce la ripetizione continua dei messaggi. Le funzioni di interrupt gestiscono l’attivazione dei flag quando viene rilevato un tap.

3. Sperimentazione e risultati

3.1 Montaggio dei sensori sulle stecche

È stato deciso di fissare gli accelerometri sulla schiena degli omini in quanto è la posizione meno colpita dalla pallina in una partita e quindi ottimale per ridurre al minimo gli eventuali colpi diretti sui sensori. È bastato creare due fori nella plastica dura dell'omino a distanza adeguata e immobilizzare i sensori con due viti. Inoltre, tra l'omino e il sensore sono state poste delle guarnizioni di gomma per ammortizzare i colpi della pallina sulla stecca e preservare i sensori.



Figura 3.1: ADXL345 montato su stecca blu

Durante la fase di progettazione sono stati effettuati dei test per capire se fosse possibile distinguere i diversi omini su una stessa stecca. Quindi sono stati montati due sensori su una sola stecca e si è trovato che colpendo uno degli omini entrambi gli accelerometri rilevavano l'urto e inviavano un segnale di interrupt al microcontrollore. Dunque, si è cercato di capire quale segnale di interrupt si venisse rilevato per primo con l'aiuto di un oscilloscopio, il quale è uno strumento di misura elettronico utilizzato per visualizzare graficamente le variazioni di un segnale elettrico rispetto al tempo.

Se il segnale di interrupt inviato per primo fosse corrisposto sempre all'accelerometro dell'omino colpito, sarebbe stato possibile distinguere gli urti.

Di seguito viene mostrato ciò che ha rilevato l'oscilloscopio colpendo lo stesso omino in tentativi diversi:

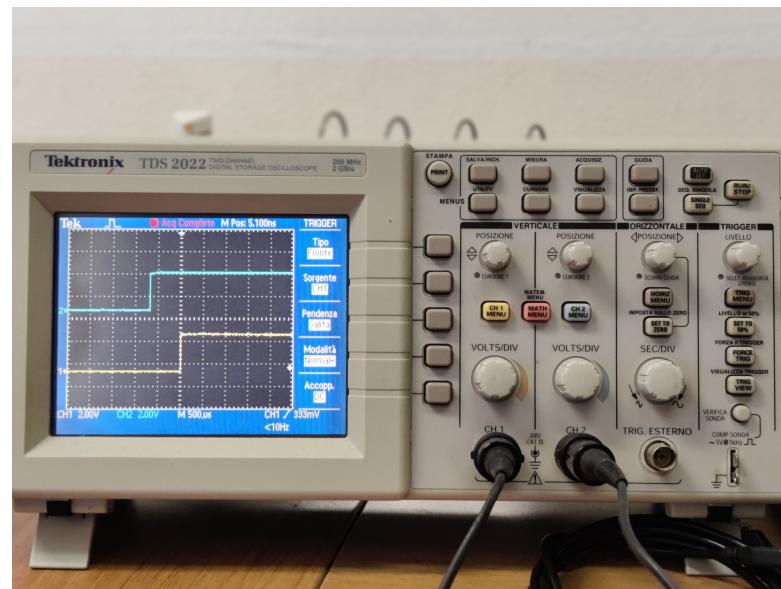
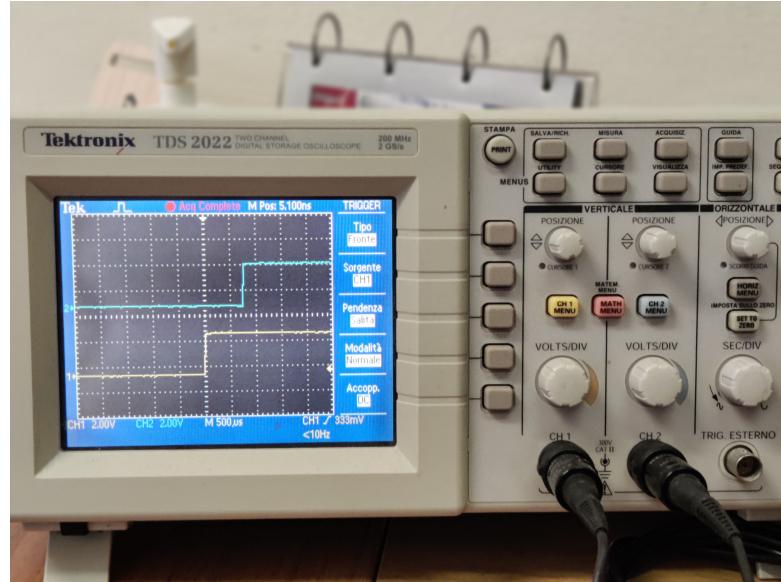


Figura 3.2: Risultati dell'oscilloscopio ottenuti colpendo lo stesso omino in momenti diversi

I segnali di interrupt sono posti su due canali diversi e vengono distinti dal colore blu o giallo. Per entrambe le figure è stato colpito l'omino su cui era montato il sensore che invia il segnale di interrupt giallo. Nelle figure vengono mostrati due risultati significativi, infatti nella prima foto è proprio l'interrupt giallo ad alzarsi per primo, ma nella seconda foto è, invece, quello blu; questo fatto rende indistinguibili gli omini colpiti sulla stessa stecca del calcio-balilla.

Questo avviene in primis a causa del materiale di cui sono fatte le stecche del calcio-balilla, che è l'acciaio. Infatti, la velocità di propagazione delle vibrazioni dipende da diversi fattori, tra cui il materiale delle stecche, la densità del materiale, la forma delle stecche stesse e la frequenza delle vibrazioni. Per le stecche di acciaio le vibrazioni solitamente si propagano molto velocemente grazie alla alta densità e alle proprietà elastiche del materiale.

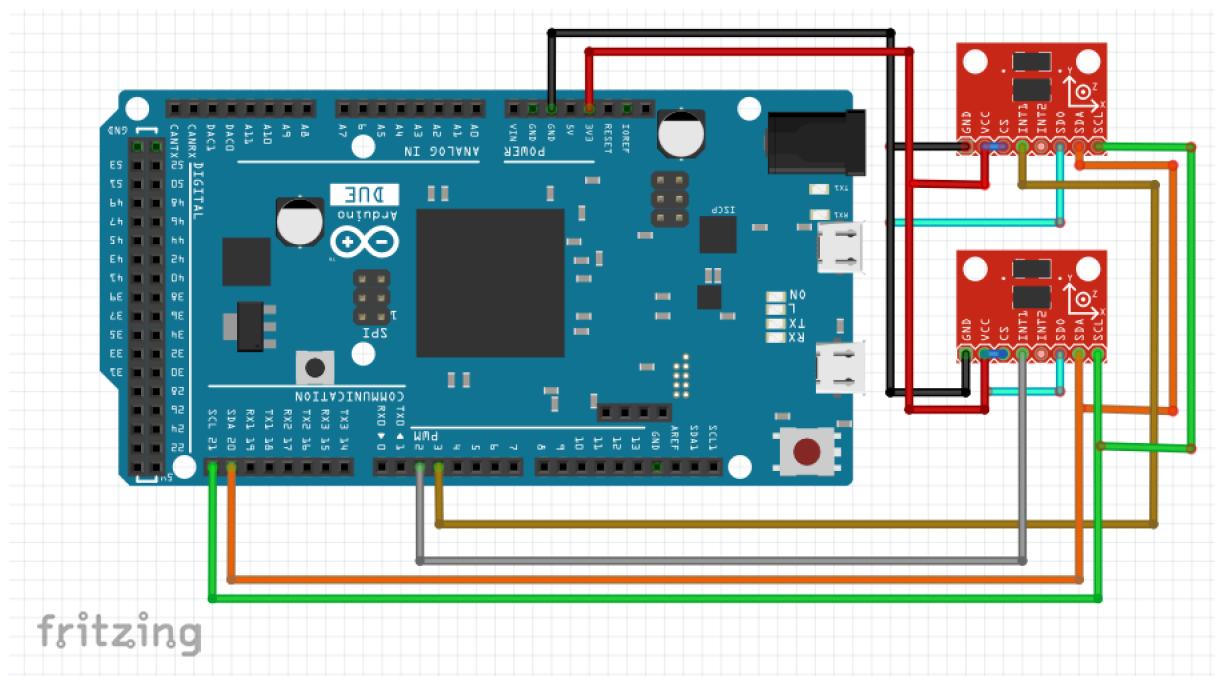
La velocità del suono in materiali come l'acciaio è significativamente più elevata rispetto a quella nell'aria (343 metri al secondo). La velocità del suono nell'acciaio può variare, ma tipicamente si aggira attorno ai 5000 metri al secondo. Ciò significa che le vibrazioni possono viaggiare attraverso l'acciaio a velocità notevoli.

Gli ADXL345 sono accelerometri molto sensibili e nonostante fossero calibrati con il valore di soglia massimo per gli interrupt di tap, l'urto viene rilevato da entrambi i sensori senza modo di distinguerli.

3.2 Acquisizione dati dalla scheda Arduino

Il sistema complessivo necessario all'acquisizione dei dati è composto da un microcontrollore, Arduino Due, al quale sono collegati gli ADXL345. Ai fini della realizzazione del progetto è necessario collegare all'Arduino Due più di un sensore. Per poter fare ciò è necessario distinguere i diversi sensori sul bus I²C usando diversi indirizzi per ognuno. L'indirizzo di default dell'ADXL345 è 0x53, ma è possibile scegliere un indirizzo I²C alternativo, 0x1D, collegando alto il pin SDO/ALT ADDRESS; questa è stata la soluzione adottata poiché per una questione di tempo sono stati collegati solamente due sensori, uno sulla stecca dell'attacco rossa e uno sulla stecca della difesa blu.

Di seguito i collegamenti tra Arduino Due e due ADXL345:



- **Cablaggio ADXL345 con indirizzo 0x53**

Il lato sinistro corrisponde ai pin dell'accelerometro, mentre il lato destro corrisponde alla scheda di Arduino.

GND → GND

VCC → VCC (3.3V)

CS → VCC (3.3V)

INT1 → PIN 2 (digitale)

SDA → PIN 20 (SDA)

SCL → PIN 21 (SCL)

SDO → GND

- **Cablaggio ADXL345 con indirizzo 0x1D**

Il lato sinistro corrisponde ai pin dell'accelerometro, mentre il lato destro corrisponde alla scheda di Arduino.

GND → GND

VCC → VCC (3.3V)

CS → VCC (3.3V)

INT1 → PIN 3 (digitale)

SDA → PIN 20 (SDA)

SCL → PIN 21 (SCL)

SDO → VCC (3,3V)

In questo modo per ogni sensore ci sono cinque fili collegati al microcontrollore.

3.2 Sistema in funzione

A questo punto si può passare al montaggio del sistema sul calcio-balilla.

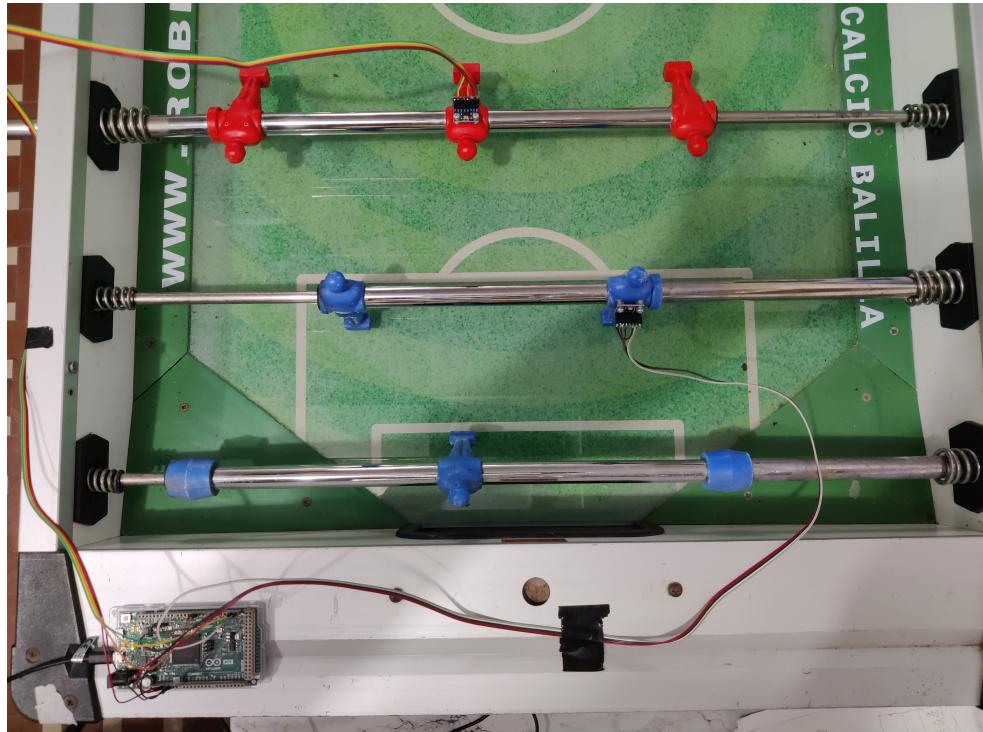


Figura 3.4: Progetto montato sul calcio-balilla

Per visualizzare i dati real-time l'ambiente di sviluppo di Arduino mette a disposizione uno strumento che consente alla scheda di comunicare con il computer tramite porta USB, il monitor seriale. È importante ricordarsi di impostare il baud rate del serial monitor uguale a quello specificato nel codice, in questo caso 9600 bps. La coerenza tra la velocità di trasmissione impostata con Serial.begin() nel codice Arduino e quella impostata nel serial monitor è fondamentale per garantire una comunicazione seriale affidabile e corretta tra Arduino e il computer. Se le velocità non corrispondessero, i dati potrebbero essere trasmessi e ricevuti in modo errato.

```
Output Serial Monitor ×
Message (Enter to send message to 'Arduino Due (Programming Port)' on '/dev/cu.usbmodem14101')
Start simulation:
0
0
tap detection on for sensor one
tap detection on for sensor two
*** ASTA BLU DIFESA ***
*** ASTA BLU DIFESA ***
*** ASTA ROSSA ATTACCO ***
```

Figura 3.5: Visualizzazione dati su serial monitor

4. Contributo personale e considerazioni conclusive

Dopo aver esposto dettagliatamente quali componenti sono stati utilizzati, come sono stati impiegati e che risultati sono stati ottenuti, è importante descrivere come è stato affrontato il problema di rilevare gli impatti ricevuti dalle stecche.

Durante la sperimentazione sono stati eseguiti test per impostare al meglio la soglia di decisione per il rilevamento o meno di un urto: scegliendo infatti una soglia troppo bassa, il rischio è di rilevare qualcosa anche quando la pallina non colpisce la stecca, mentre scegliendola troppo alta si presenta il rischio di non rilevare i colpi più deboli. Dunque, questa particolare problematica è stata risolta sperimentalmente. A causa della elevata sensibilità dei sensori, i quali se scossi con molta forza rilevano un urto anche quando non avviene nonostante la soglia di tap sia la più alta, ho scelto di privilegiare i colpi deboli in modo che il sistema funzioni bene se i giocatori giocano con più delicatezza e precisione.

È chiaro che questo progetto può rappresentare il primo passo verso qualcosa di più specifico ed elaborato, aggiungendo funzionalità: per esempio, rendendo accessibili ai giocatori i dati raccolti dal sistema grazie ad una applicazione web che presenti le statistiche più interessanti il più chiaramente possibile.

Per ragioni di tempo il progetto è stato realizzato solo su due stecche del calcio-balilla; per poter espandere il sistema a tutte le stecche del calcio-balilla devono essere fatte alcune considerazioni. Siccome l'ADXL345 può supportare due indirizzi diversi per poter collegare otto sensori alla scheda di Arduino, grazie alle interfacce di comunicazione I²C o SPI, si può parlare con un indirizzo scelto attivandolo, grazie al pin SDO, a rotazione su tutti i sensori.

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 concluso, posso confermare le mie aspettative, e sento di poter aggiungere che mi ha arricchito dal punto di vista personale.

5. Riferimenti biografici

1. Arduino Due <https://store.arduino.cc/products/arduino-due>
2. Arduino IDE <https://www.arduino.cc/en/software>
3. ADXL345 <https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf>
4. Libreria SparkFun <https://learn.sparkfun.com/tutorials/adxl345-hookup-guide/all>
5. VS Code <https://code.visualstudio.com>

6. Ringraziamenti

In conclusione dell'elaborato, desidero ringraziare tutte le persone senza le quali questo lavoro di tesi non sarebbe mai esistito.

Dunque ringrazio il mio relatore, il Prof. Riccardo Berta e il mio correlatore, il Dott. Matteo Fresta, per avermi seguito in questo percorso.

Un sentito grazie a Flavio Ansovini che mi ha aiutato a rendere questo progetto possibile.

Ringrazio la Fratellanza S. Fruttuoso per aver concesso l'uso di un calcio-balilla per la realizzazione di questo progetto e tutte le persone che la frequentano per avermi dato uno spazio in cui crescere.

Ringrazio i miei genitori che mi hanno sempre e comunque sostenuto durante il mio percorso di studi.

Infine, vorrei ringraziare i miei amici di Sardenaira gvng, per avermi dato ciò che non pensavo di meritare.

È stato un onore aver potuto condividere questi anni con voi. Spero ce ne siano molti altri.