Università degli Studi di Bergamo

SCUOLA DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Laboratorio di Sensori

Relazione esperienza di laboratorio del 16/11/2022

Prof.
Gianluca Traversi

Candidati
Davide Salvetti
Matricola 1057596

Martina Fanton
Matricola 1059640

Matteo Verzeroli
Matricola 1057926

Anno Accademico 2022–2023

Laboratorio 15/11/22

In questa esperienza di laboratorio, è stata presentata la board Arduino Nano 33 BLE Sense, una piattaforma open-source che integra diversi sensori e ne facilita l'utilizzo. In particolare, la piattaforma è dotata dei seguenti sensori (Fig.1.1):

- LSM9DS1: una *inertial measurement unit* (IMU) a 9 assi che integra un accelerometro, un giroscopio e un magnetometro;
- LPS22HB: integra un barometro e un sensore di temperatura;
- HTS221: un sensore di umidità relativa;
- APDS-9960: può essere usato come sensore di prossimità, luce ambientale, sensore RGB e rilevamento dei gesti;
- MP34DT05: un microfono digitale.

Inoltre, sulla board è presente un chip per la crittografia ATECC608A e un convertitore DC-DC MPM3610. Il microprocessore è integrato nel modulo NINA B306 (64 MHz Arm Cortex-M4F) che fornisce anche un modulo Bluetooth 5, con supporto multiprotocollo. La board viene alimentata con una tensione di 3.3 V. Tipicamente, l'alimentazione è fornita tramite la porta USB che viene anche utilizzata per il caricamento del firmware tramite l'Arduino IDE.

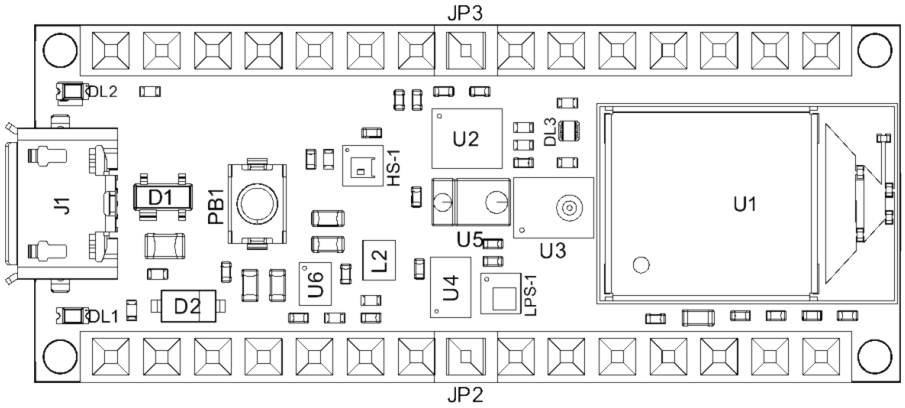

Figura 1.1: Arduino Nano 33 BLE Sense, vista top. U1: NINA-B306 Module Bluetooth® Low Energy 5.0 Module, U2: LSM9DS1TR Sensor IMU, U3: MP34DT06JTR Mems Microphone, U4: ATECC608A Crypto chip, U5: APDS-9660 Ambient Module, U6: MP2322GQH Step Down Converter, PB1: IT-1185AP1C-160G-GTR Push button, HS-1: HTS221 Humidity Sensor, DL1: Led L, DL2: Led Power.

Durante il laboratorio è stato testato il funzionamento dell'accelerometro tramite il seguente sketch che permette di ricavare le misure delle accelerazioni proprie dall'accelerometro. Di seguito si riporta il codice del firmware caricato sulla board tramite l'Arudino IDE. In particolare, le misure vengono ricavate dal buffer dell'accelerometro per poi essere scritte sulla porta seriale secondo il formato "x y z".

```
//import libreria accelerometro
#include "Nano33BLEAccelerometer.h"

//creiamo la struttura per i dati dell'accelerometro
Nano33BLEAccelerometerData accelerometerData;

//stringa di supporto per visualizzare i dati
char str[60];

void setup() {
  //apertura comunicazione seriale
  Serial.begin(115200);
  while(!Serial);

  //inizializzazione accelerometro
  Accelerometer.begin();
}

void loop() {
  //se e' disponibile una misura la estraggo
  if (Accelerometer.pop(accelerometerData)){
    //inserisci nella stringa str i valori misurati nel formato "x y z"
    sprintf(str, "%.3f %.3f %.3f", accelerometerData.x,accelerometerData.y,
    accelerometerData.z);
    //scrivi sulla porta seriale la stringa str
    Serial.println(str);
  }
  //pausa di 100ms
  delay(100);
}
```

Inoltre, è possibile visualizzare graficamente i dati scritti sulla porta seriale tramite il *Serial Plotter* fornito dall'ambiente di sviluppo di Arduino. Nella figura 1.2 è possibile vedere un esempio di acquisizione dei dati, in cui l'Arduino viene tenuto fermo su un piano. In queste condizioni si noti che l'unica accelerazione misurata è l'accelerazione di gravità con un valore misurato pari a circa 1 g.

Figura 1.2: Grafici di alcune misure dell'accelerometro. Nelle immagini si può notare l'accelerazione di gravità misurata sull'asse z con segno positivo (immagine a sinistra) e segno negativo (immagine a destra).

Successivamente è stato utilizzato Matlab per analizzare le misure dell'accelerometro ottenute con Arduino. In particolare questi dati sono stati generati muovendo Arduino lungo l'asse z con un tempo di campionamento di 100 ms (Fig.1.3). L'obiettivo di questa analisi consiste nell'individuare la com-

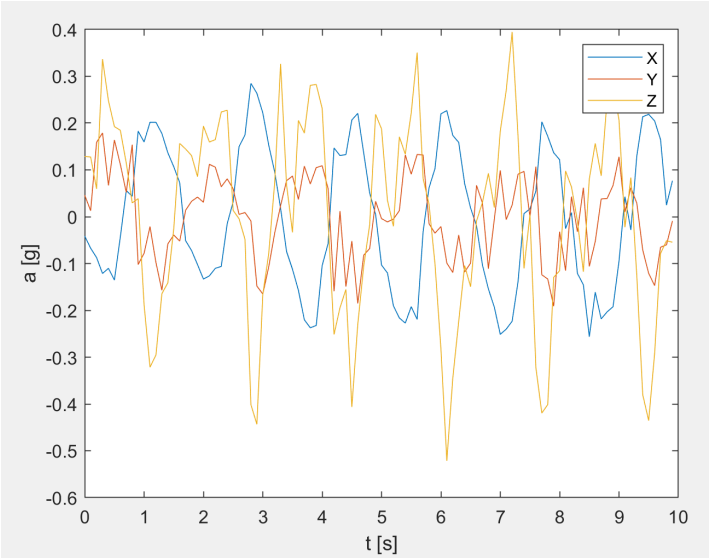

Figura 1.3: Grafico del set di dati grezzi (a cui sono stati sottratti la media).

ponente in frequenza principale del segnale ottenuto dalle misure, la quale corrisponde alla frequenza con cui l'accelerometro si muoveva. Per migliorare l'analisi in frequenza, i dati, a cui è stata sottratta la media per eliminare l'accelerazione gravitazionale, sono stati filtrati tramite un filtro digitale. Per l'implementazione del filtro digitale è stato utilizzato il comando *movmean* di Matlab, che consente di realizzare un filtro a media mobile con finestra pari a 10 campioni. I dati filtrati sono stati riportati nella figura 1.4.

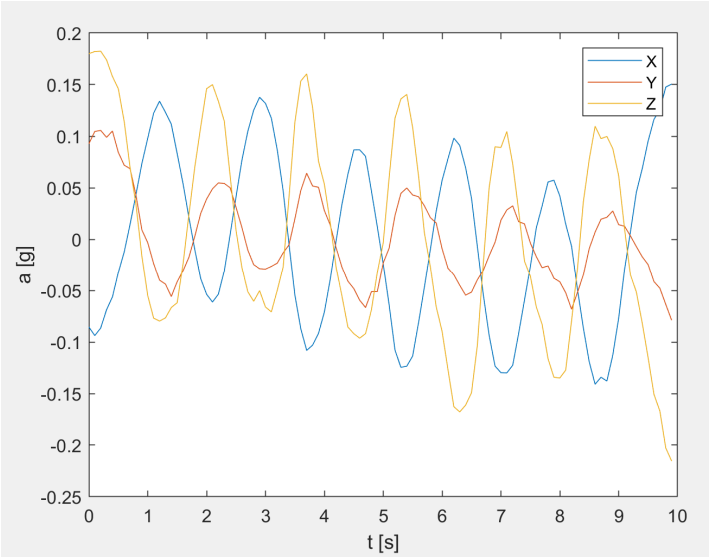

Figura 1.4: Grafico del set di dati filtrato.

In seguito, per determinare il valore della frequenza desiderata sono stati utilizzati due metodi. Il primo prevede di calcolare i picchi presenti nel segnale: sono stati utilizzati i comandi *gradient* e *diff* di Matlab per determinare rispettivamente il gradiente e le differenze tra i gradienti di dati adiacenti, valutando quando la derivata del segnale filtrato (denominato *filt*) cambia segno (da positivo a negativo), con lo scopo di rilevare il numero dei picchi presenti nel segnale.

```
%calcolo il gradiente sui dati filtrati (filt) e estraggo le posizioni in
%cui il gradiente e' maggiore di zero (array di 0, nelle posizioni dove il
% gradiente e' <=0, 1 nelle posizioni in cui il gradiente e' >0
grad_pos = gradient(filt)>0;

%calcolo il numero totale (sum) dei picchi vedendo i punti in cui il
%gradiente passa da un valore positivo a uno negativo (diff(grad)==-1)
number_peaks = sum(diff(grad_pos)==-1,1);

%divido il numero di picchi trovati per il tempo di misura pari a 10s
freq = number_peaks./10;
```

La frequenza risultante dall'applicazione del precedente metodo è pari a 0.6 Hz. Infatti, vengono rilevati correttamente 6 picchi nella finestra temporale di 10 s. Il secondo metodo, invece, utilizza la Fast Fourier Transform calcolata con il comando *fft* di Matlab.

```
%codice tratto da: https://it.mathworks.com/help/matlab/ref/fft.html

%frequenza di campionamento
Fs = 10;
%periodo di campionamento
T = 1/Fs;
%vettore dei tempi
tfft = (0:length(filt)-1)*T;
%FFT della componente z del segnale filtrato
Y = fft(filt(:,3));
%calcolo dello spettro a due lati (frequenze positive e negative) P2 e
%calcolo dello spettro a un lato (solo frequenza positive) P1
P2 = abs(Y/length(filt));
P1 = P2(1:length(filt)/2+1);
P1(2:end-1) = 2*P1(2:end-1);
%calcolo dello spettro delle frequenze
f = Fs*(0:(length(filt)/2))/length(filt);
%plot dello spettro in frequenza del segnale filtrato
plot(f,P1)
title("Single-Sided Amplitude Spectrum of X(t)")
xlabel("f [Hz]")
ylabel("|P1(f)|")
```

Nella figura 1.5 è possibile vedere che la componente in frequenza principale del segnale filtrato è pari a 0.6 Hz. La frequenza calcolata è quindi uguale a quella calcolata con il metodo precedente.

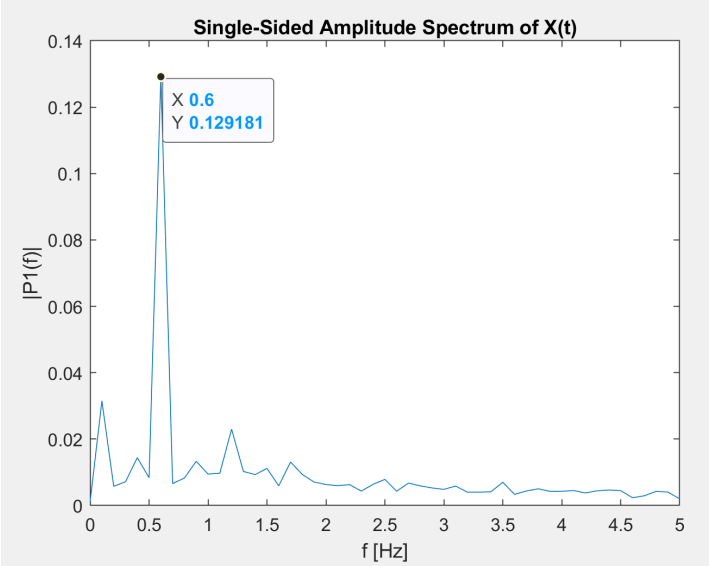

Figura 1.5: Grafico della Trasformata di Fourier del set di dati filtrato.