# Dispositivi utilizzati

## GY-521

Il GY-521 è un modulo che permette di includere un sensore MPU-6050 e un sensore di temperatura. MPU-6050 è un chip prodotto da Invesense [1], è un’unità di misura inerziale (IMU) basata sulla tencologia MEMS che contiene un accelerometro a tre assi e un giroscopio a tre assi all’interno di un un singolo chip. Questo sitema a 6 gradi di libertà (*6 DOF*) ci fornisce quindi 6 valori in uscita.

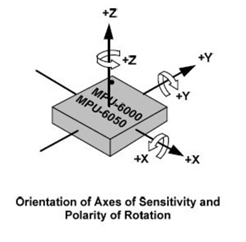
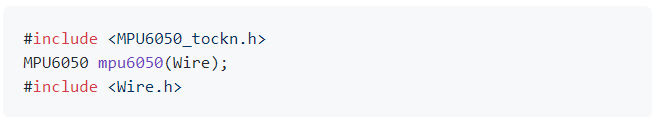


Fig. 3 – Orientazione degli assi per il sensore inerziale.

L’accelerometro è in grado di rilevare l’accelerazione di un corpo lungo il proprio asse, mentre il giroscopio è in grado di rilevare la velocità angolare di un corpo su di un proprio asse.

Il modulo GY-521 ha piccole dimensioni ma è molto preciso, contiene un chip con un convertitore analogico-digitale a 16 bit integrato per ogni canale e questo permette di catturare i canali X, Y e Z contemporaneamente. Il range di misura dell’accelerometro è +2, +4, +8, +16g, mentre il range di misura del giroscopio è e . Necessita inoltre, di una alimentazione da 3V a 5V.

Questo sensore utilizza il protocollo di comunicazione standard I2C, in particolare per la comunicazione è necessario importare due librerie una per l’I2C e una per la comunicazione con l’MPU-5060.



## Servo

Un Servomotore è un attuatore rotativo o lineare che consente un controllo preciso della posizione angolare o lineare, della velocità e dell’accelerazione.

È costituito da un opportuno motore accoppiato ad un sensore per il feedback di posizione. Richiede anche un controller relativamente sofisticato che spesso consiste in un modulo dedicato, progettato specificatamente per l’uso con i servomotori.

In questo progetto è stato usato il Microservo SG90. Può ruotare approssimativamente di 180° (90° per entrambe le direzioni) e funziona esattamente come i servomotori standard ha solo delle dimensioni ridotte.

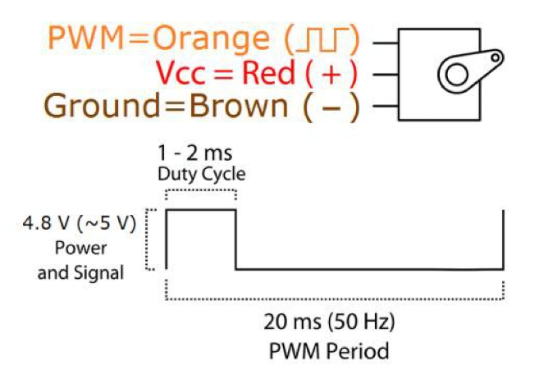
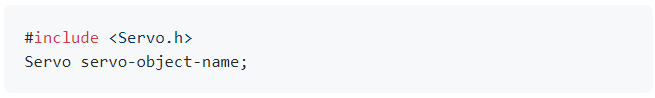


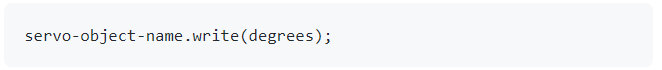
Fig. 4 – Collegamento dei pin del servo.

La posizione “0” (impulso di 1,5 ms) è al centro, “90” (impulso di 2 ms) è tutto a destra, “-90” (impulso di 1 ms) è tutto a sinistra.

La libreria Servo viene utilizzata per controllare il servomotore.



È stato inoltre utilizzato il seguente comando per controllare il servo



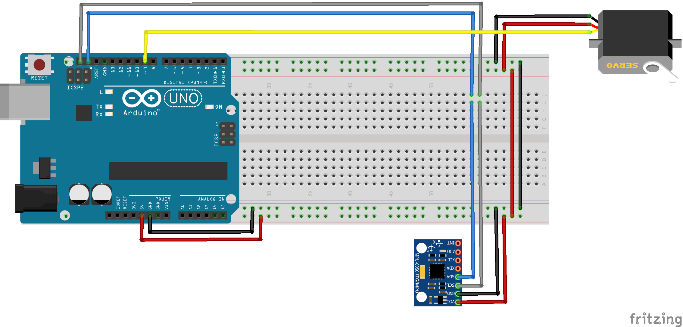


Fig. 5 – Schema di collegamento del servo e dell'IMU.

## Sensore EMG

Il sensore per l’EMG, dove EMG sta per Elettromiografia, è un sensore in grado di misurare l’attività muscolare monitorando il potenziale elettrico generato dalle cellule muscolari. L’elettromiografia nasce per la ricerca medica e per la diagnosi dei disturbi neuromuscolari. Tuttavia, i sensori per l’EMG oggi, vengono utilizzati nella robotica e in altri sistemi di controllo.

L’EMG valuta lo stato di salute dei muscoli e delle cellule (i motoneuroni) che li controllano. Queste cellule trasmettono segnali elettrici che provocano la contrazione e il rilassamento muscolare. Un EMG traduce questi segnali in grafici o dati e aiuta i medici ad eseguire una diagnosi.

Questo sensore amplifica ed elabora l’attività elettrica di un muscolo e lo converte in un segnale analogico che può essere letto da qualsiasi microcontrollore con un convertitore analogico-digitale come Arduino.

Quando il muscolo sottoposto a misurazione si flette, la tensione di uscita del sensore aumenta.

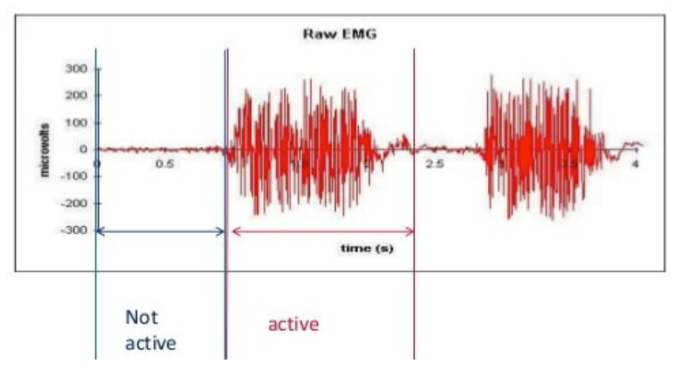


Fig. 6 – Segnale grezzo EMG prima di essere filtrato, rettificato e smooted

Collegato al sensore, tramite un connettore jack di 3,5 mm, c’è un cavo che da un lato è collegato ad Arduino, mentre dall’altro lato termina con tre connettori al clip che si attaccano a tre elettrodi.

In questo progetto è stato usato il Muscle Sensor V3. Questo sensore necessita di una tensione di riferimento positiva ed una negativa, quindi sono richieste due batterie da 9V.

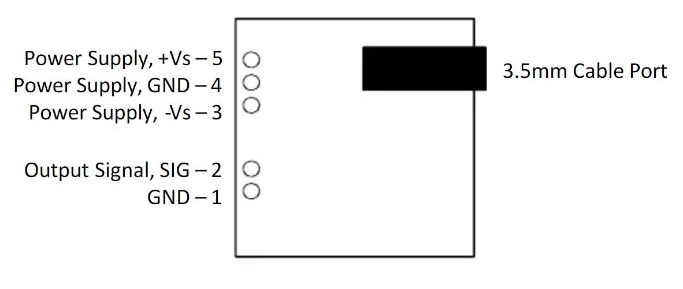


Fig. – 7 Schema di collegamento dei pin del sensore.

L’Output Signal richiede un pin analogico, in particolare è stato utilizzato il pin A0.

L’EMG di superficie può essere registrato da una coppia di elettrodi o da una matrice più complessa di elettrodi. È necessario più di un elettrodo perché le registrazioni dell’EMG mostrano la differenza di potenziale tra due elettrodi separati.

Come si può leggere nel datasheet o nello schema elettrico, il segnale dei due elettrodi entra in un amplificatore per strumentazione:

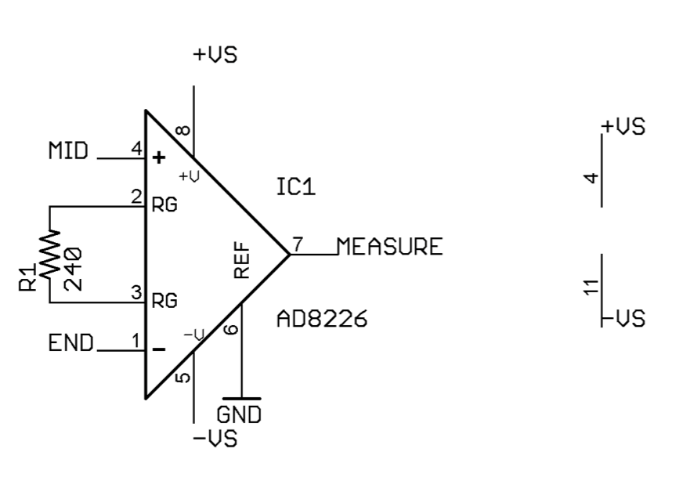


Fig. 8 – Schema dell'amplificatore per strumentazione.

Successivamente il segnale viene rettificato:

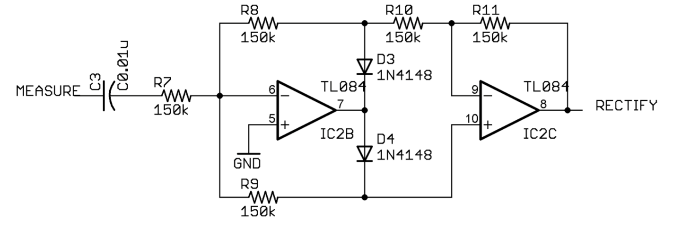


Fig. 9 – Raddrizzatore a doppia semionda che permette di rettificare il segnale amplificato.

E alla fine il segnale viene ripulito attenuando il rumore generato da artefatti ambientali, elettrici, elettronici, fisiologici ecc.

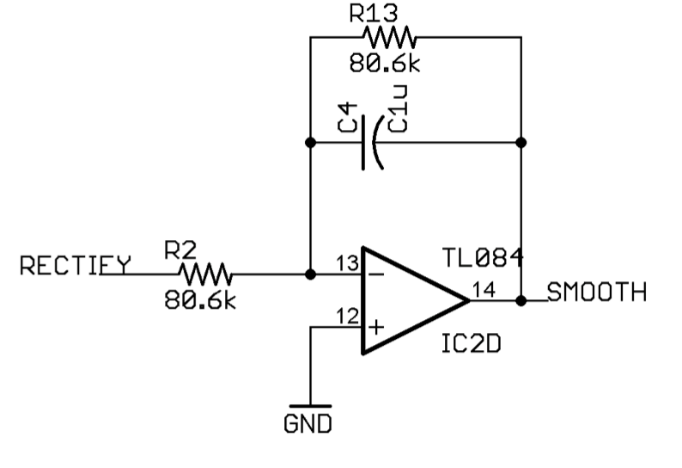


Fig. 10 – Smoohing, filtro attivo passa basso.

In sintesi, avremo un segnale variabile con la contrazione muscolare che verrà inviato al convertitore A/D di Arduino.

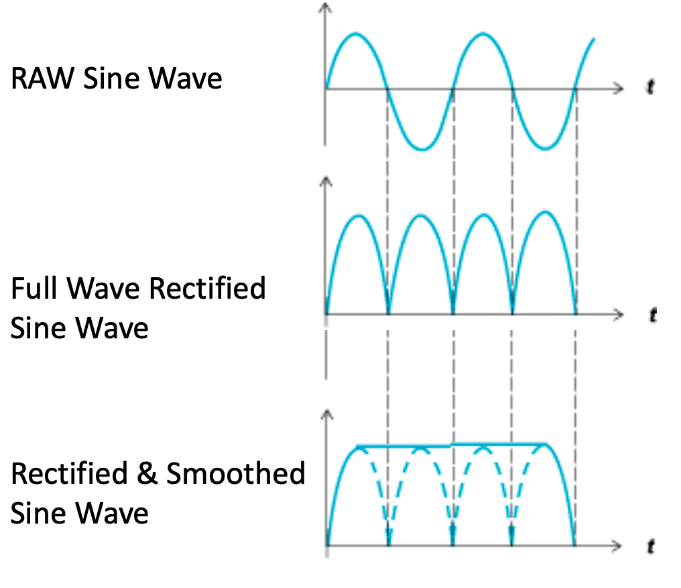


Fig. 11 – Raw EMG vs EMG dopo il raddrizzatore a doppia semionda e lo smoothing.

Il Muscle Sensor V3 è fuori produzione, quindi abbiamo acquistato una copia. Questa copia permette un notevole risparmio ma presenta alcuni difetti. Il sensore, infatti emette un segnale senza livellamento e pieno di disturbi.

Immagine che contiene testo, elettronico

Descrizione generata automaticamente

Fig. 12 – Copia del Muscle Sensor V3 utilizzata nel progetto.

## Elettrodi

L’attività elettrica di un muscolo viene rilevata con l’ausilio degli elettrodi per l’EMG. Ci sono differenti tipi di elettrodi, in questo progetto sono stati utilizzati gli elettrodi di superficie., in particolare quelli in gel. Questo tipo di elettrodi contiene un gel elettrolitico come interfaccia tra gli elettrodi e la pelle.

Gli elettrodi gelificati forniscono una tecnica non invasiva per la misurazione e il rilevamento del segnale EMG, ma sono generalmente utilizzati per i muscoli di superficie poiché vengono applicati sulla pelle.

Gli elettrodi devono essere applicati in modo appropriato, in particolare, due elettrodi devono essere posizionati tra l’unità motoria e l’inserzione tendinea del muscolo, lungo la linea longitudinale del muscolo. La distanza tra il centro dei due elettrodi dovrebbe essere di 1-2 cm.

Il connettore a clip rosso va posizionato sull’elettrodo al centro del corpo muscolare, quello verde ad un’estremità del corpo muscolare e infine, il terzo connettore (giallo) deve essere posizionato su una sezione inattiva del corpo.

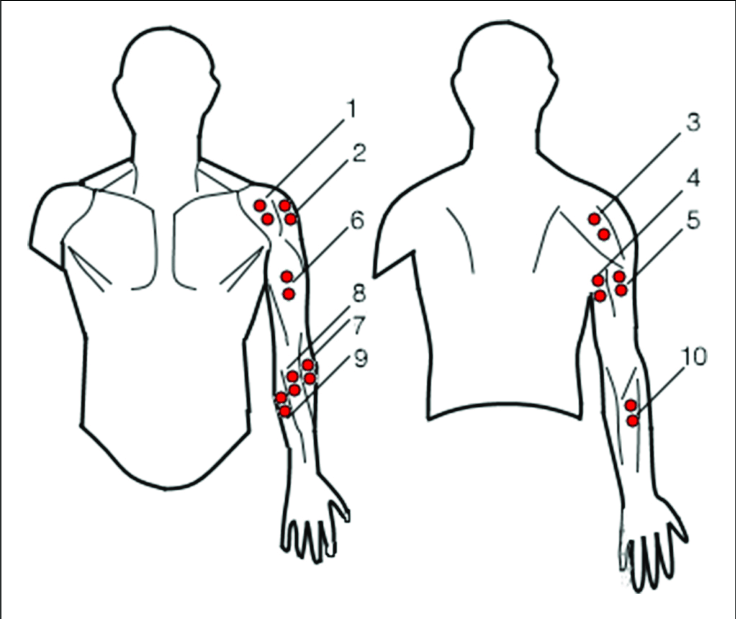


Fig. 13 – Posizionamento degli elettrodi.

È importante fare alcune considerazioni sui segnali EMG.

La fonte elettrica è il potenziale di membrana muscolare di circa -90 mV. La frequenza di ripetizione tipica dell’impulso dell’unità muscolo-motoria è di circa 7-20 Hz, a seconda della dimensione del muscolo i potenziali misurati variano tra meno di 50 µV fino a 30 mV, a seconda del muscolo sotto osservazione. Questo significa che il segnale deve essere amplificato come visto sopra.

A causa della bassa qualità del circuito EMG, il segnale analogico acquisito era pieno di rumore. La tensione è stata solo rettificata ma non molto livellata, quindi abbiamo fornito un semplice algoritmo di livellamento riportato in Fig. 14.

Quanto riportato nello sketch in Fig. 14 è lo script di smoothing di riferimento di Arduino con piccole modifiche apportate dopo un lungo periodo di test [2] (Arduino’s reference smoothing script). In Fig. 15 si può osservare il segnale nel Plotter seriale di Arduino, in cui sono stati riportati il segnale grezzo, un valore soglia e il segnale ripulito.

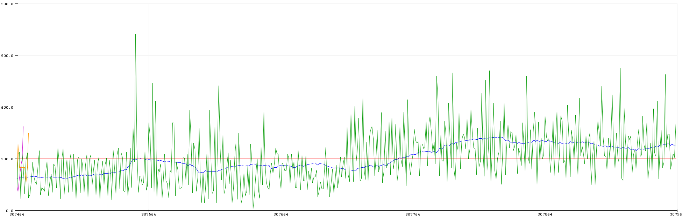


Fig. 15 – Segnale smoothed (blu), segnale originale (verde), soglia (rosso). Quando il segnale è al di sotto del valore soglia l’arto è rilassato, mentre, quando il segnale è al di sopra della soglia il muscolo è contratto.

Questa ***media*** livellata ci ha consentito di personalizzare il codice riportato in Fig. 16 per controllare il servo che permette l’apertura e la chiusura della pinza.

## Pulse Sensor

Il principio di questo sensore è quello di far brillare una luce tra le dita. La maggior parte della luce viene assorbita o riflessa ma una parte passerà attraverso i tessuti se questi sono abbastanza sottili. Quando il sangue viene pompato attraverso il corpo, viene schiacciato nei tessuti capillari e il volume di quei tessuti aumenta leggermente, quindi tra i battiti del cuore il volume diminuisce. La variazione del volume influisce sulla quantità di luce che verrà trasmessa. Questa fluttuazione è molto piccola ma può essere percepita con l’elettronica. Vediamo come si può fare.

Iniziamo con una sorgente e un rilevatore di luce. Si utilizzano un LED a infrarossi e un sensore a fotodiodo. È importante che il due dispositivi siano accoppiati bene, così che la lunghezza d’onda della luce emessa dal LED venga rilevata dal fotodiodo. Il fotodiodo consiste di una piccola cella solare, proprio come i pannelli sui tetti, ma molto più piccola. Genererà una piccola tensione e una corrente quando viene bombardato con i fotoni.

A questo punto ciò di cui abbiamo bisogno è un modo per amplificare il piccolo segnale che esce dal fotodiodo. Fortunatamente, la configurazione di questo circuito è nota come convertitore da corrente a tensione ed è un circuito ben noto.

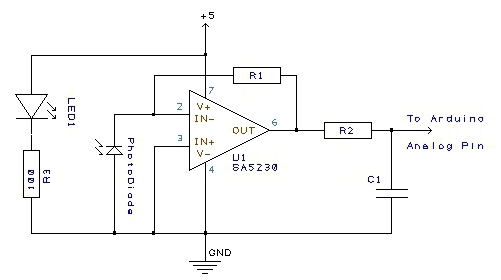


Fig. 16 – Schema circuitale del Pulse Sensor

Il fotodiodo deve essere schermato dalla luce ambientale che genera molto rumore nel segnale. Infatti, come si può vedere in Fig. 19 il segnale del sensore è distorto anche da piccoli movimenti. Inoltre, qualsiasi disallineamento o movimento del LED a infrarossi confonderà il segnale.

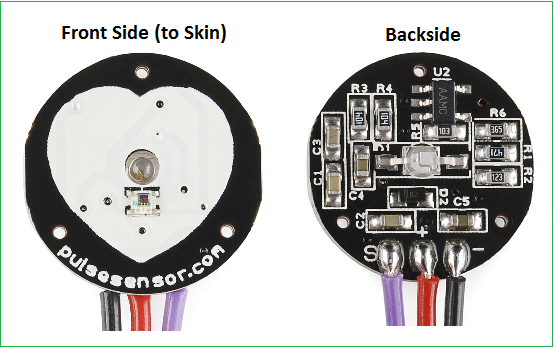


Fig. 17 – Immagine del sensore. Il cavo rosso corrisponde ai 5V, il cavo nero al GND e il cavo viola è il segnale.

Pulsesensor.com produce un sensore che è abbastanza piccolo da poter essere indossato comodamente in diverse configurazioni e immune al rumore del segnale generato dal movimento o dai cambiamenti delle condizioni di illuminazione ambientale. È costituito da un circuito integrato di dimensioni molto ridotte che ha al suo interno un fotodiodo integrato e un circuito op-amp combinati. Le dimensioni ridotte hanno come risvolto il fatto che è possibile mantenere un contatto stretto e costante con la pelle. Questo di fatto protegge il sensore dai cambiamenti della luce ambientale e riduce al minimo i rumori dovuti al movimento.

Il segnale grezzo è quello in Fig. 18

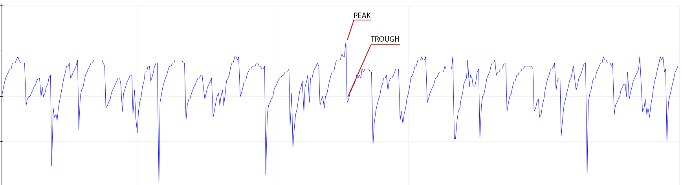


Fig. 19 – Picco e depressione del segnale grezzo.

Per le prime prove con il sensore su Arduino è stata utilizzata la libreria PulseSensor Playground, tuttavia questa non è risultata completamente compatibile con le altre funzionalità che abbiamo poi implementato. Quindi, abbiamo proceduto a scrivere un codice particolare.

La lettura è richiamata dalla funzione battiti()descritta nella libreria funsioniBPM.h.

Abbiamo messo insieme diverse idee prese da Instructables.com [3] e da pulsesensor.com [4] per ottenere un codice che si occupi di leggere il valore analogico e identificare il battito al di sopra di una certa soglia. Il codice è relativamente complesso. Oltre ad una parte deputata a stampare i simboli o una frase affermativa quando un battito viene rilevato, il Timer 2 viene utilizzato per impostare un interrupt ogni 2 ms. Un pezzo di codice più complesso si occupa di misurare il tempo trascorso dall’ultimo battito e di fare campioni appropriati per evitare il rumore. In particolare, si cerca di evitare il rumore dicrotico e di rilevare i picchi. Quando viene rilevato un battito (un picco conforme alle condizioni imposte) il valore del tempo viene aggiornato e vengono calcolati i BPM e il numero di battiti che possono verificarsi in 1 minuto (60000 ms).

Le funzioni primarie sono svolte da ISR(TIMER2\_COMPA\_vect) e battiti() e serialOutputWhenBeatHappens() si occupa di restituire il valore del battito.

## Test dei sensori

Nella Fig. 20 è riportato il circuito che mostra la configurazione del test per lo sviluppo del codice iniziale.

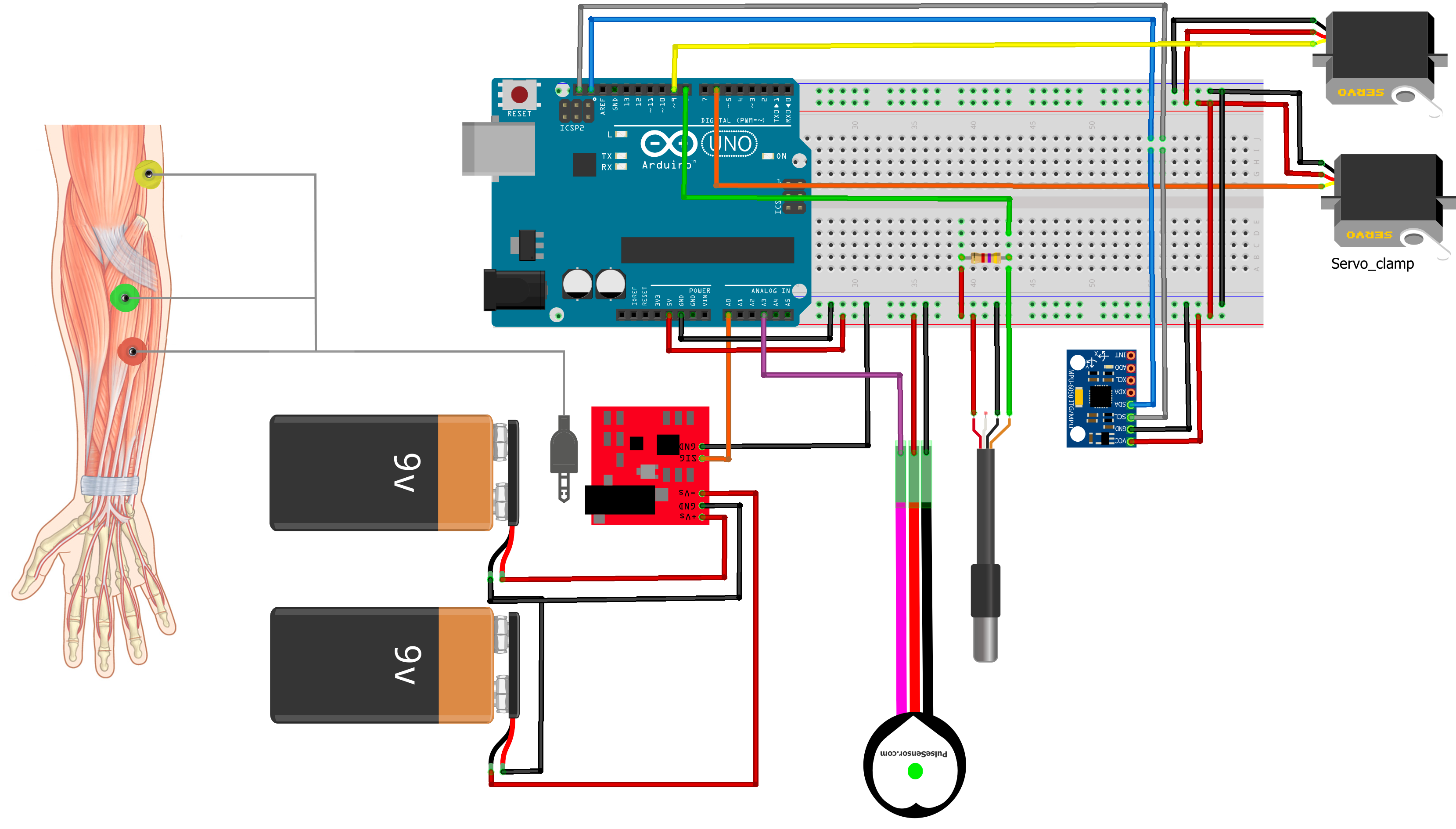


Fig. 20 – Circuito con i diversi sensori utilizzati.

Questa configurazione in Fig. 20, con due batterie da 9 V consente di avere +9 e -9 V sui terminali Vs del sensore per l’EMG. È anche possibile utilizzare un alimentatore da 18 V DC e dividerlo in +9 e -9.

Questa configurazione ci ha permesso di eseguire tutti i test dei dispositivi e del codice. Abbiamo quindi rielaborato parte del codice suddividendolo nei vari moduli in modo da avere un sistema più modulare che permettesse di aggiungere e rimuovere i dispositivi a seconda delle necessità. Da questa configurazione poi, abbiamo iniziato a suddividere i diversi dispositivi in una struttura determinata per il monitoraggio e una struttura con gli attuatori.

Alla fine, i sensori che devono arrivare al polso verranno uniti in un unico cavo dotato di connettore per collegarlo alla struttura di monitoraggio.

Tutto è stato quindi riorganizzato in modo più compatto.

# Divisione dei dispositivi

A questo punto abbiamo diviso i codici e i dispositivi. In particolare, abbiamo usato un Arduino nano per la lettura dei sensori e un Arduino uno per controllare i servomotori.

La comunicazione tra i due Arduino avviene tramite radio utilizzando due ricetrasmettitori NRF24L01 a singolo chip da 2.4 GHz.

In Fig. 21 il modulo NRF24L01;

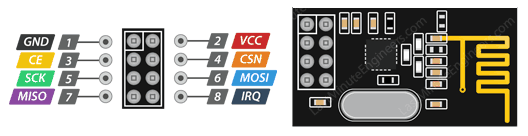
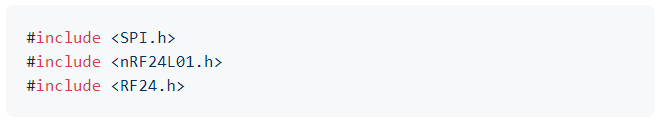


Fig. 21 – Modulo NRF24L01 e schema di collegamento dei pin.

Questo dispositivo ha una tensione operativa di 1.9V 3.6V, un’antenna di 2,4 GHz incorporata e una velocità operativa (massima) si 2Mbps. Il chip è gestito dall’interfaccia SPI. Tramite questa interfaccia è possibile accedere ai registri del chip e modificarne tutti i parametri.

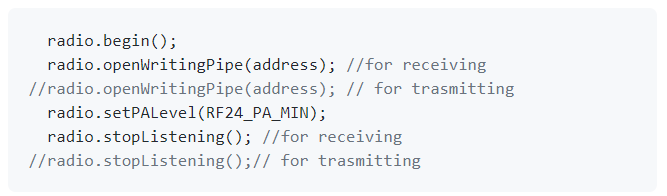
Questa operazione complessa, però, è facilitata dall’utilizzo delle librerie.



Il collegamento del modulo richiede l’utilizzo di diversi pin:

* GND
* Vcc, il modulo utilizza 3.3V
* CE, Chip Enable, utilizzato per abilitare la comunicazione SPI
* CSN, Ship Select Not, un pin che deve essere sempre al livello alto altrimenti disabilita l’SPI
* SCK, Serial Clock, fornisce l’impulso di clock con il quale funziona la comunicazione SPI
* MOSI, Master Out Slave In, collegato al pin MOSI di Arduino, serve per il modulo, per ricevere dati da Arduino
* MISO, Master In Slave Out, collegato al pin MISO di Arduino, serve per il modulo, per inviare dati da Arduino.
* IRQ, è un pin basso attivo, viene utilizzato solo se è richiesto l’interrupt.

Per utilizzare questo modulo è necessario impostare la lettura o la scrittura nel **setup()** come segue:



È anche importante definire i pin CE e CSN.

Per la trasmissione abbiamo utilizzato una funzione charfortransmission()creata ad hoc che permette di concatenare una lettera che indica quali dati stiamo trasmettendo e il valore assunto dalla variabile corrispondente. Questa funzione viene ripetuta nel loop() e chiamata continuamente 7inviandole parametri diversi a seconda della variabile. Viene quindi convertito in char per consentire la corretta trasmissione con le librerie che utilizza.

Il ricevitore ha una funzione simile, radionuovo(), che innanzitutto analizza il primo carattere identificando quali informazioni vengono ricevute e inserisce i dati nelle variabili di interesse dopo aver effettuato le opportune conversioni di tipo variabile.

## Arduino nano per il rilevamento

Vediamo in Fig. 22 la configurazione dell’Arduino nano con i sensori sulla breadboard per i test.

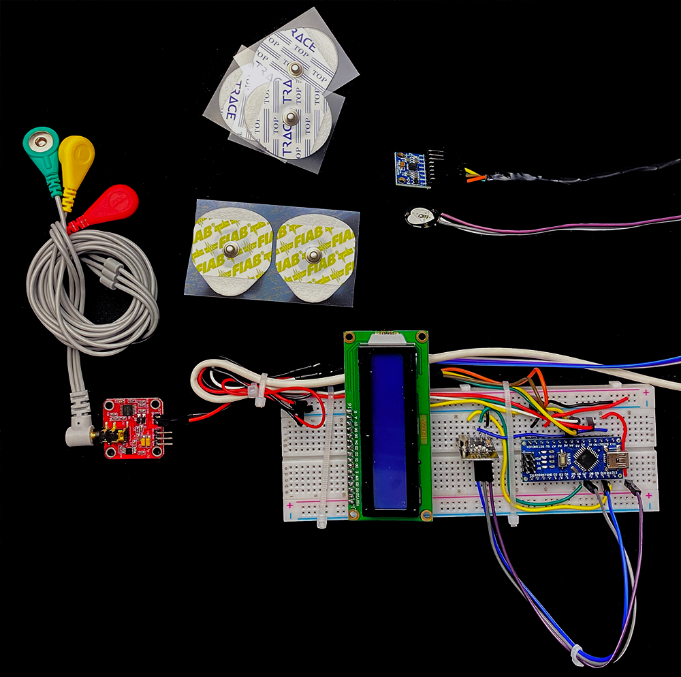


Fig. 22 – Disposizione dei sensori e dell’Arduino nano sulla breadboard.

Per l’Arduino nano dedicato ai sensori, il circuito è stato modificato aggiungendo un display LCD per monitorare i parametri oltre al modulo NRF24L01 per la trasmissione radio. Il display LCD è collegato ad un modulo di interfaccia I2C per semplificare la comunicazione e la gestione dei cavi. Si può anche fare a meno del modulo di interfaccia I2C andando a collegare il display ai rimanenti pin dell’Arduino nano, in tal caso saranno necessari 7 pin digitali, in particolare rimarranno disponibili i pin (9,10,2,3,4,5,6).

Per il resto la connessione dei sensori è rimasta la stessa.

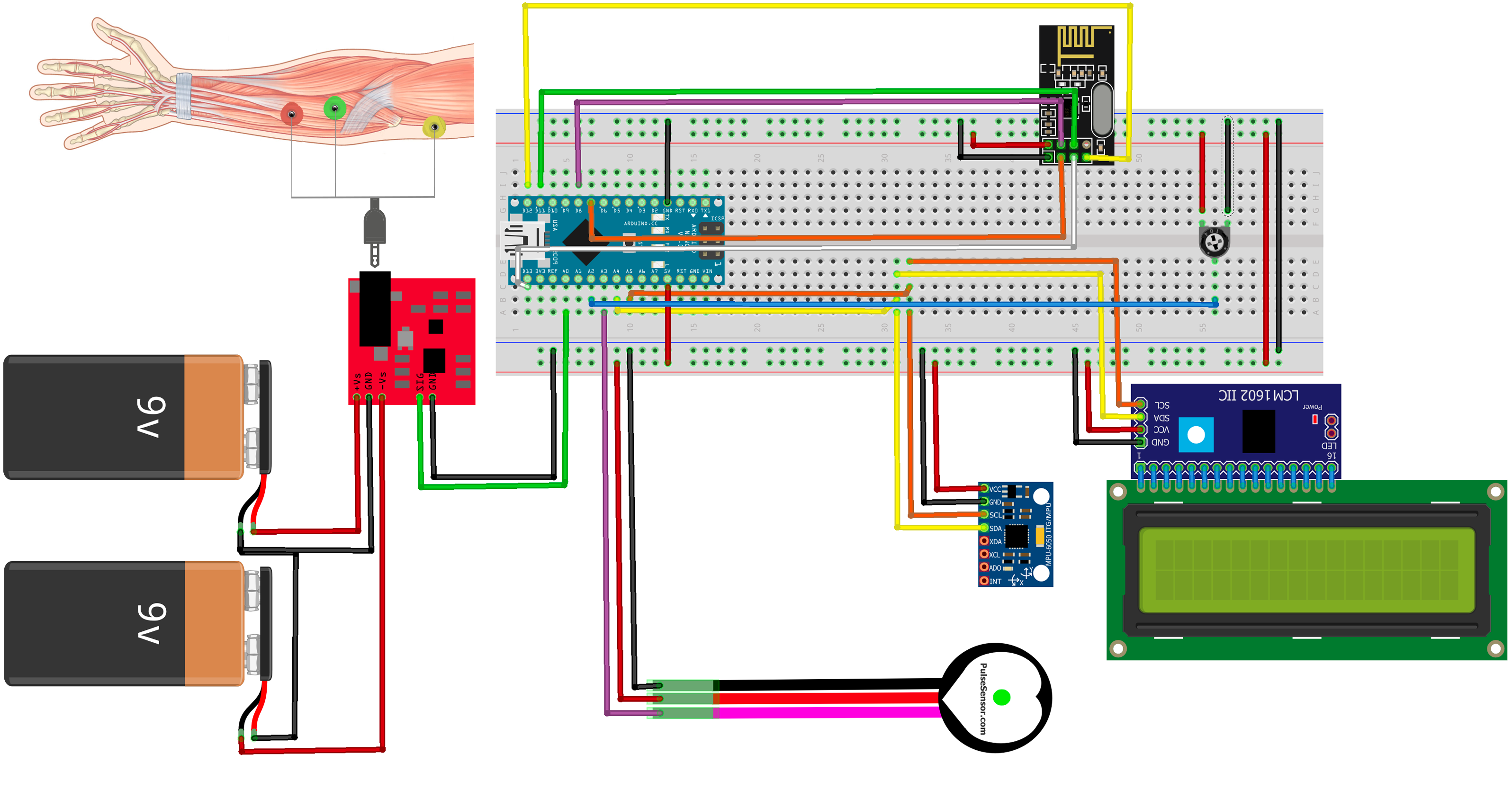


Fig. 23 – Schema completo dei sensori.

Come detto precedentemente per rendere il tutto più compatto è stato utilizzato un Arduino nano. La disposizione dei pin dei diversi sensori rimane la stessa, l’unica accortezza riguarda il fatto che non ci sono i pin dedicati SDA e SCL ma sono sui pin A4 e A5 come è possibile osservare in Fig. 24.

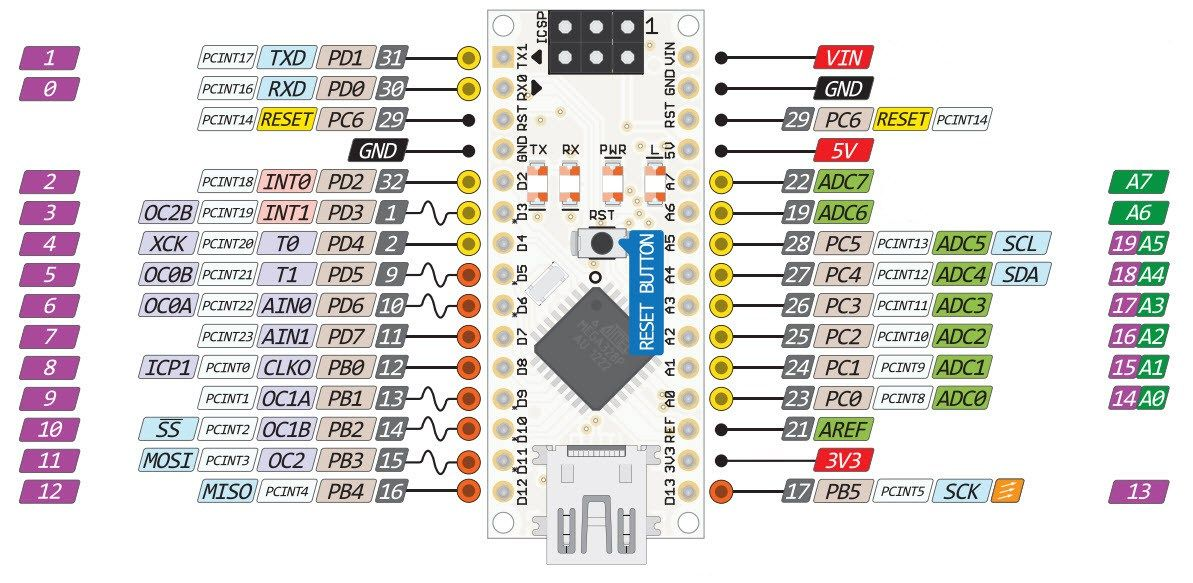


Fig. 24 – Disposizione e definizione dei pin.

Nel codice relativo a questa parte che coinvolge i diversi sensori la parte di codice relativa ai servomotori è stata rimossa e il resto è stato riorganizzato.

Il codice principale è riportato in Fig. 25 in fondo al testo.

In particolare, il codice principale è stato semplificato aggiungendo diverse funzioni che chiamano definizioni di librerie esterne:

averagecalc() è stata definita in EMGsmooth.h e viene utilizzata per calcolare la media del segnale distorto.

printatore(LOW) è una funzione speciale che può essere usata per stampare il segnale EMG per il plotter seriale in modo da eseguire il debug di eventuali problemi di rilevamento. Di solito viene utilizzato per stampare tutti i valori sul monitor seriale. È definito in printerfunzioni.h

lcdprint() viene utilizzato per stampare i diversi valori e i relativi simboli sul display LCD.

battiti() è usato per rilevare i battiti cardiaci. È stato definito nelle funzioni funzioniBPM.h

charfortransmission() per la trasmissione dei dati con le relative informazioni.

Per mantenere il codice chiaro queste funzioni sono state definite in librerie esterne. La libreria EMGsmooth.h è responsabile del livellamento del segnale EMG al fine di ottenere precisione nella determinazione della soglia. La libreria funzioniBPM.h è già stata descritta nel paragrafo del sensore di battito. Il printerfunzioni.h contiene la funzione per stampare nel monitor seriale e nel display lcd, contiene anche la definizione del simbolo freccia e cuore sul display LCD.

Per creare caratteri speciali per LCD viene utilizzato LCD Custom Generator [4] che ci permette di generare un byte custom char con definizione pixel ON/OFF. Questo ci consente di usare simboli speciali oltre ai caratteri ASCII definendo individualmente i pixel di attivazione e disattivazione in una matrice 5x8 (byte\_matrix). È possibile creare fino a 8 simboli diversi con la funzione createChar() della libreria LiquidCrystal e stamparli con lcd.Write(). Ad esempio, per il simbolo del cuore Fig. 26

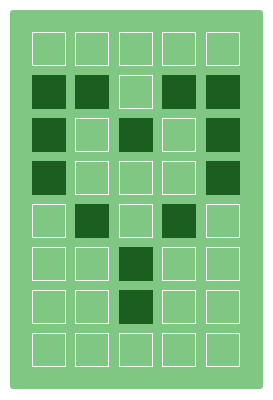




Fig. 26 – Creazione di caratteri speciali e disposizione dei valori sul display.

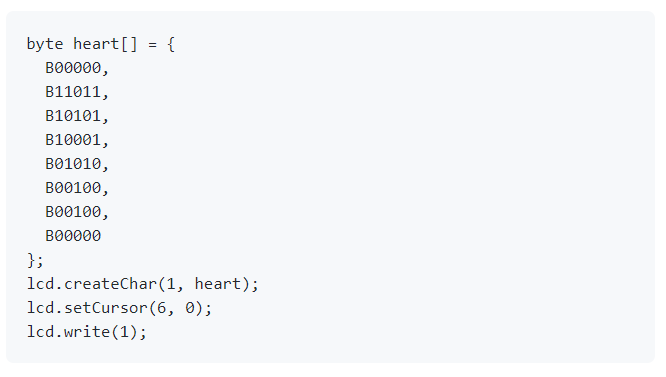
Ogni libreria contiene tutte le variabili e tutte le altre librerie necessarie al suo funzionamento.

Siamo stati, quindi, in grado di ottenere i seguenti valori che devono essere contenuti nell’Arduino con gli attuatori:

* tempmpu
* angolo
* media
* BPM
* soglia

infine, è stato aggiunto anche un trimmer per controllare manualmente la soglia sul pin analogico A2.

## Arduino uno per l’esecuzione



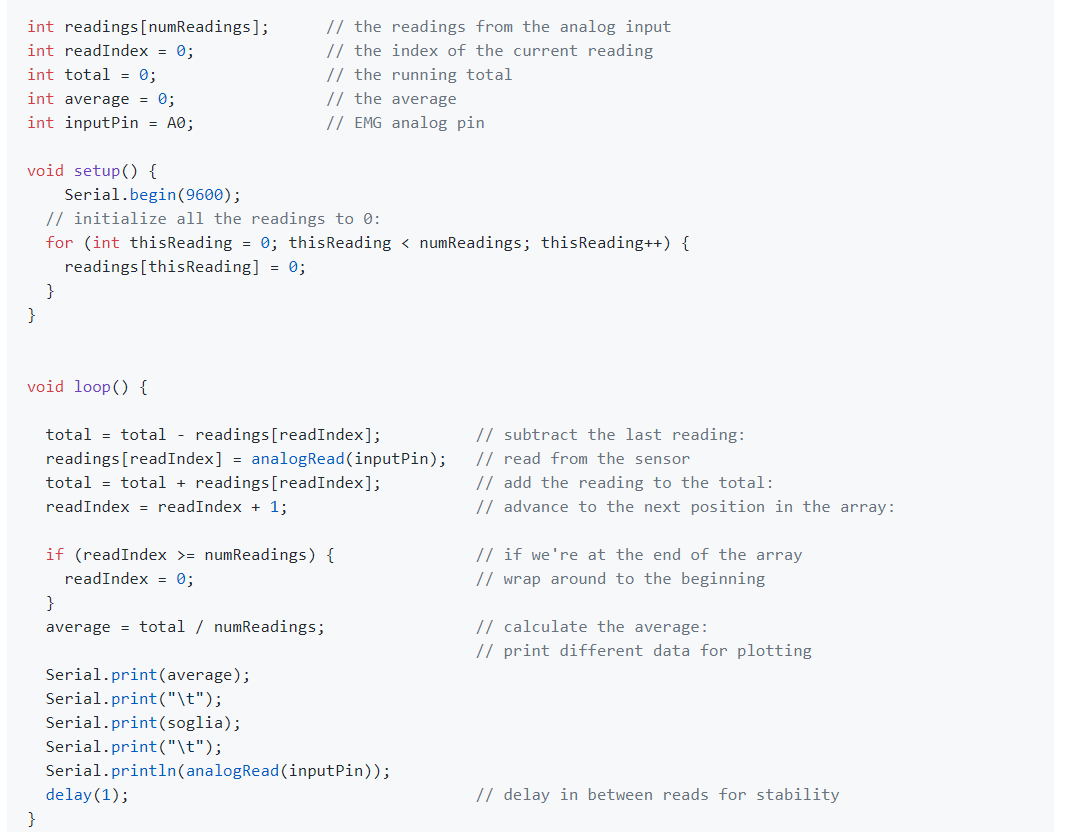


Fig. 5 Codice di livellamento del segnale



Fig. 25 – Codice principale che riguarda Arduino nano con i diversi sensori

# Indice delle figure

[Fig. 1 – Struttura di esempio della pinza per il controllo di apertura e chiusura e della rotazione. 3](file:////Users/alessandromastrofini/Documents/GitHub/progelettronica/Relazione%20progetto.docx#_Toc69841677)

[Fig. 2 – Esempio di struttura realizzata con gli ingranaggi, plastica sagomata e una base in alluminio anodizzato 3](#_Toc69841678)

[Fig. 3 – Orientazione degli assi per il sensore inerziale. 4](#_Toc69841679)

[Fig. 4 – Architettura complessiva del progetto 7](file:////Users/alessandromastrofini/Documents/GitHub/progelettronica/Relazione%20progetto.docx#_Toc69841680)